Zurich University
of Applied Sciences

School of
Engineering
InES Institute of
Embedded Systems

Precision Time Protocol for Spectroscope Synchronization

Armin Weiss, Tobias Kammacher

Adviser: Prof. Hans-Joachim Gelke

Tuesday 17th February, 2015

Abstract

This project was conducted by the Institue of Embedded Systems (InES) at Zurich University of Applied Sciences (ZHAW) in collaboration with Zurich Instruments AG (ZI) and the Commission for Technology and Innovation. The aim was to implement a Ethernet based time synchronization between multiple Etzel, which are advanced digital lock-in amplifiers from ZI, and validate and verify the performance. The requested time synchronization accuracy is in the order of 10 ns.

PTP was determined to be the most suitable time synchronization protocol because of the precision, the independence of other devices and the availability of an in-house PTP stack implementation of the protocol. The feasibility of a time synchronization solution using PTP was proven by evaluating multiple simulations and implementing the available stack on two Apalis evaluation hardware boards. A concept was devised for integrating the time synchronization functionality into existing Etzel devices and implemented with the help of ZI. Testing was conducted on the underlying implementation on Apalis boards as well as on the final system with multiple Etzels in order to verify the performance requirements.

During the implementation of the PTP stack, adjustments had to be made to the Ethernet controller driver. The PTP stack was adapted to work on a Linux with 3.1 kernel and the integrated interfaces for the PTP clock as well as the hardware timestamping. Hardware limitations (routing of pins) led to two variants for pin assignment. The accuracy of the time synchronization is dependent on environmental conditions and configuration of the PTP application. Time synchronization measurements between two Etzels show a standard deviation between the timestamp registers under best measurement conditions of 12.1 ns and a mean value of -1.7 ns.

The performance of the system was found to satisfy the needs of Zurich Instruments under the condition that the Ethernet connection is realized in a direct (Peer-to-Peer) manner or via a PTP-capable switch.

Contents

Glossary

ADC Analog Digital Converter

API Application Programming Interface

BC Boundary Clock

BMC Best Master Clock

BSP Board Support Package

CDF Cumulative Distribution Function

CPU Central Processing Unit

CTI Commission for Technology and Innovation

DUT Device Under Test

E2E End-to-End

FPGA Field Programmable Gate Array

GPIO General Purpose Input Output

GPS Global Positioning System

HF High Frequency

IC Integrated Circuit

InES Institute of Embedded Systems

IP Internet Protocol

L4T Linux for Tegra

LED Light Emitting Diode

LAN Local Area Network

LXDE Lightweight X11 Desktop Environment

MAC Media Access Control (layer)

nc NetCat

NMEA National Marine Electronics Association

NTP Network Time Protocol

OS Operating System

P2P Peer-to-Peer

PCB Printed Circuit Board

PCHIP Piecewise Cubic Interpolating Polynomial

PCIe Peripheral Component Interconnect Express

PHY Physical (layer)

PI Proportional-Integral

PLL Phase Locked Loop

ppm parts per million

PPS Pulse per Second

PTP Precision Time Protocol

PTPD Precision Time Protocol Daemon

pv Pipe Viewer

RNDIS Remote Network Driver Interface Specification

RMSE Root Mean Square Error

SDP Software Defined Pin

SNR Signal-to-Noise-Ratio

SNTP Simple Network Time Protocol

TAI Temps Atomique International

TC Transparent Clock

TCP Transmission Control Protocol

TSU Time Stamp Unit

TC Transparent Clock

UDP User Datagram Protocol

UTC Universal Time, Coordinated

VHDL Very High Speed Integrated Hardware Description Language

WR White Rabbit

ZHAW Zurich University of Applied Sciences

ZI Zurich Instruments AG

Chapter 1

Introduction

This chapter provides an overview of the project and states the required goals.

1.1 Overview

Current generation measurement and testing equipment most commonly provides the ability to connect to a Local Area Network (LAN) for remote control and data retrieval purposes. The recent development of thin clients, which spatially separate signal acquisition hardware from the user interface, in conjunction with widely available LAN connectivity allow for *cost-effective decentralized measurement systems*. Applications that are based on such equipment include the production of solar panels, cell-analysis in microfluidic channels and impedance spectroscopy in synchrotron particle accelerators (as displayed in figure 1.1).

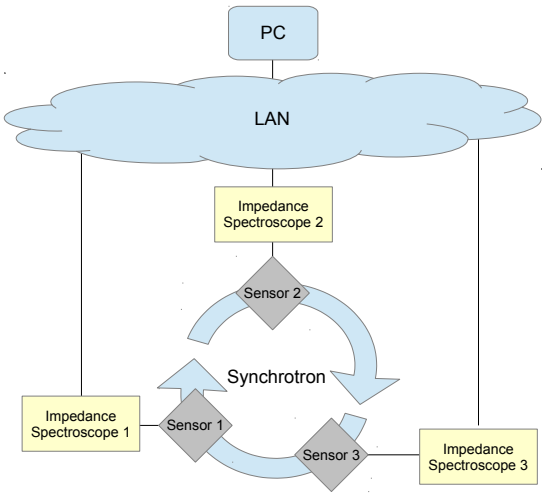


Figure 1.1: Example application of a decentralized measurement system in a synchrotron

A fundamental requirement for all these applications is a common time base. Thus allowing the identification of measurements that were taken at the same time but in different locations. One of the core competencies of the Zurich University of Applied Sciences (ZHAW) InES is time synchronization in the sub-microsecond range by means of the PTP[1]. A project

with the goal of developing a measurement device[1] with time synchronization capabilities has been conducted as part of a project cooperation between the InES and Zurich Instruments AG (ZI), supported by the federal Swiss Commission for Technology and Innovation (CTI).
The specific task described by this report (being one of the last parts of the overall CTI project) was to investigate the possibility of expanding the functionality of ZI's impedance spectroscopes by time synchronization abilities. Thus allowing them to be used in decentralized measurement setups.
The expected deviation between multiple devices' time bases is of the order of 10 ns. Furthermore the existing hardware (microprocessor and network controller) shall be used while communicating only via Ethernet over twisted-pair cables.
In order to achieve a sufficiently high degree of accuracy for the individual time bases, a synchronization method has been implemented on specific hardware, based on the precision time protocol. The main advantages of this technology are i) the usage of already *existing communication channels* (therefore not requiring to install a separate synchronization wire) and ii) the relatively *high degree of accuracy* amounting to 10-100 ns, depending on the environment. Depending on the success of this project, ZI will consider implementing the time synchronization in Etzel.

The hardware of the Etzel does already exit. Its two main parts are the main board (consisting of an analog part and a FPGA), which is responsible for the measurements, and the processor board, precisely the Apalis T30. This hardware is given and shall not be changed.

1.2 Goal

The scope of this report includes examination of different network based time synchronization protocols. Moreover, concepts for the implementation of the time synchronization and the integration into Etzel are developed. The implementation is tested in different project phases in order to verify fulfillment of the required performance. The overall workflow of the project is described in figure 1.2.

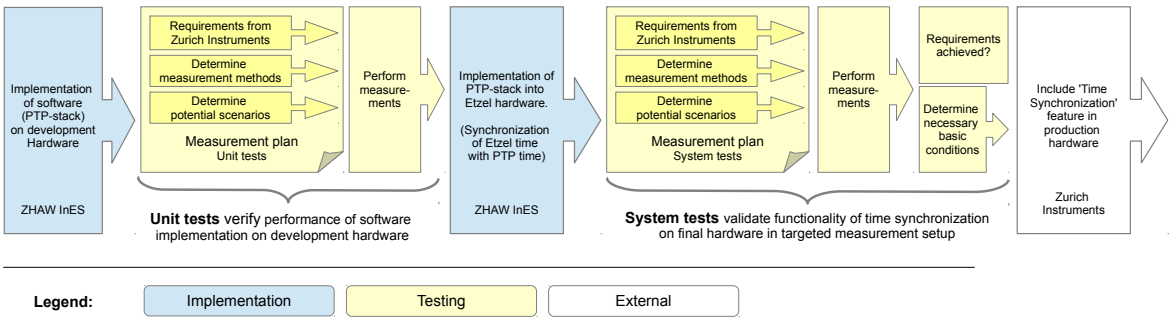


Figure 1.2: Project workflow

[1]The device by the name of 'Etzel' is a Lock-In amplifier capable of measuring signals up to 50 MHz

1.2.1 Implementation

The first goal is to choose a suitable network based time synchronization protocol for this project and give a short overview. The requested time synchronization precision between two Etzels is 10 ns.

The second goal of this project is to implement the chosen time synchronization on two Apalis evaluation boards, which contain an Apalis T30 processor board. This implementation will show the feasibility as well as the highest precision achievable.

The third goal is the integration of the time synchronization into Etzel. Therefore, a concept shall be developed and proven by simulations. The integration itself is beyond the scope of this project, since the knowledge and the source code are property of ZI.

1.2.2 Testing

In a first step, the requirements of ZI and their customers are listed. Based on these needs, the initial goal is to determine means to quantify the performance provided by our solution (the measurement concept).
Furthermore it is necessary to specify the scenarios according to which the device will be employed (measurement setup). Measurements shall be conducted according to a predefined measurement plan and their outcome assessed in order to judge the achievement of ZI's demands.

The integration of the previously developed time synchronization into ZI's measurement device is the next intermediate step with respect to testing. A second measurement phase is performed in order to validate the correct integration of the functionality into the Etzel device and finally to measure the performance of a complete system (consisting of multiple measurement devices in a decentralized measurement environment).

In summary, the goals are to:

- Confirm that the requirements were achieved
- Determine the basic conditions (in respect to temperature, supply voltages, device composition, etc.) that are necessary to perform as expected.

With these results, the developed solution is ready to be included in production hardware by ZI.

Chapter 2

Implementation Concept

The aim of this chapter is to give an overview of different time protocols, introduce different PTP stack implementations and propose concepts for the integration of the time synchronization into Etzel of ZI.

2.1 Time Synchronization Protocols

Multiple devices have to be synchronized over LAN with a precision of 10 ns. Therefore, an appropriate time synchronization protocol has to be chosen. This section provides a short overview of existing and established protocols, such as NTP, the derived SNTP and PTP.

2.1.1 Network Time Protocol (NTP) and Simple Network Time Protocol (SNTP)

NTP is a networking protocol for clock synchronization and is highly available since it is commonly used in modern computers (more information about time synchronization protocols can be found in [2]). The related user space daemon is called ntpd and is implemented in most Unix systems in order to synchronize the system clock to a NTP server.

The Architecture is composed of different levels. Such a level is called *Stratum*. The very highly accurate clocks, such as atomic or GPS clocks, are commonly located in the Stratum 0 and are connected by a RS-232 connection instead of a network. The "time servers" are located in the Stratum 1. They handle the NTP requests from the Stratum 2 servers. Theoretically, 256 Strata can be specified. However, in practice only the first 16 are used, where Stratum 15 is already considered as unsynchronized.

The precision of NTP depends on the network. Synchronization accuracy is typically in the range of 5 ms to 100 ms if synchronized on the internet and 100 μs up to several milliseconds on LAN.

SNTP is a small version NTP. The difference is that SNTP devices do not have to store state information over a long period of time and thus need less memory resources. Because of that, SNTP is commonly used in embedded systems, where resources are expensive.

The time format of NTP and SNTP is Universal Time, Coordinated (UTC).

Conclusion

The big advantage of NTP and SNTP is the high availability. However, these two protocols are not precise enough to fulfill the requirements of this project which is 10 ns. Moreover, there is always the need for an NTP server, which can not be guaranteed for all applications, in which the Etzel will be used. Therefore, neither NTP nor SNTP were chosen for the implementation.

2.1.2 Precision Time Protocol (PTP)

The PTP is a wide spread and well known time synchronization protocol. The first release of PTP was in 2002 under the name IEEE 1588-2002 (PTPv1) [3]. In 2008, a second version called IEEE 1588-2008 (PTPv2) [1] was released. Unfortunately, it is not possible to directly synchronize two PTP devices using different versions since the PTP message format has changed[4].
The time format of PTP is based on Temps Atomique International (TAI), which requires a small calculation in order to get the corresponding time in UTC format.
For new projects, it is not recommended to use the PTPv1 since this is an obsolete standard. Therefore, only the new version PTPv2 is described in more details in the following.

Precision of PTP

The precision of PTP depends on its implementation. A high resolution can be achieved by timestamping outbound packages at the very last and an inbound at the very first moment, preferably by hardware. The maximal possible resolution given by the PTP timestamp is 1 ns for PTPv1 and 2^{-16} ns (i.e. 15 femto seconds) for PTPv2. Note that these values are absolute maximum ratings and are typically not reached because of the inaccuracy of the implementations.

Best Master Clock (BMC)

PTP is basically a master-slave architecture. The master provides the clock to which all slaves synchronize. If the master has an inaccurate clock, all the slaves will adapt the inaccuracy of the master. Therefore, the best clock in the system is voted to be the master by the BMC algorithm. This algorithm takes clock quality into account as well as priority settings.

Synchronization Mechanism

As soon as the master is defined, the slaves can start synchronizing. The whole synchronization process is divided into two procedures.

The first procedure is called *syntonization*, which is responsible for adjusting the slave's clock in order to make it run at the exact same frequency as the master clock (see Figure 2.2). Therefore, so called Sync messages are continuously sent by the master.

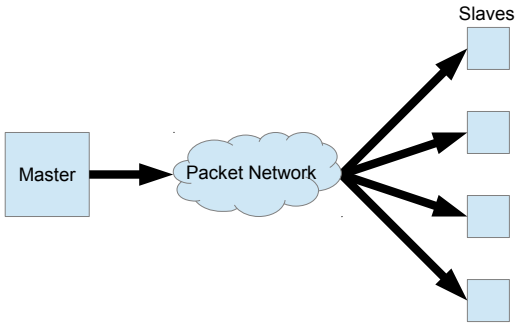


Figure 2.1: Distribution of time and frequency in PTP

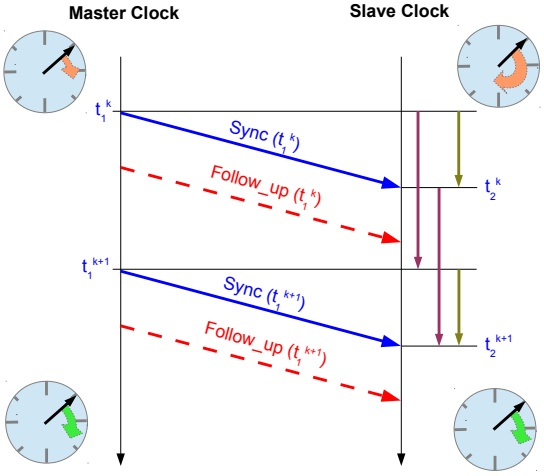


Figure 2.2: Principle of Syntonization

The Sync messages are then processed by the slaves. The calculation of the frequency deviation requires the transmit timestamp and the receive timestamp. The master has two options for sending the transmit timestamp to the slaves. One-step clocks insert the timestamp on-the-fly into the Sync message whereas two-step clocks send a Follow-up message containing the transmit timestamp of the Sync message. The receive timestamp is produced by the slaves themselves and therefore already available. With the knowledge of the sending and the receiving time, the slaves are able to calculate the interval between two Sync messages as well as the downlink delay using the following formulas:

$$Interval : t_1^{k+1} - t_1^k = t_2^{k+1} - t_2^k$$
$$DownlinkDelay : t_2^{k+1} - t_1^{k+1} = t_2^k - t_1^k$$

The second procedure is responsible for time synchronization. Until now, the clocks of the master and the slaves would run at the same speed but likely with an offset in time, since they were not started at exactly the time (see Figure 2.3). Therefore, the round trip time between master and slave clock is measured. The downlink is already known from the Sync message before and the uplink is measured with a Delay Request message sent by the slaves.

The Delay Response is used to bring the receive timestamp of the Delay Request back to the slaves.

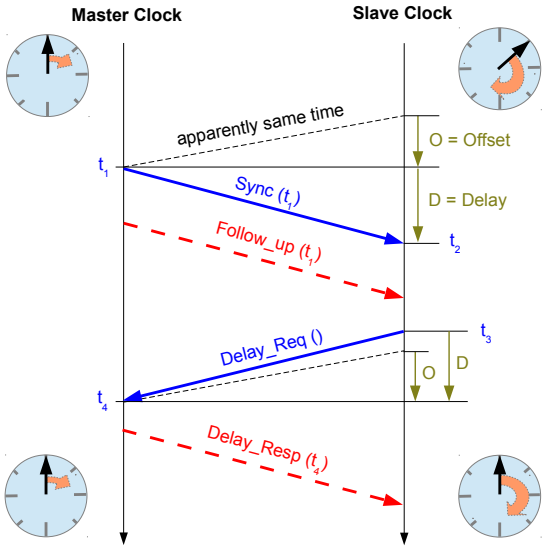


Figure 2.3: Clock adjustment mechanism

The uplink and downlink are assumed to be symmetric for the sake of simplicity. The calculation of the one-way delay and the offset are therefore performed as follows:

$$Delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}$$
$$Offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}$$

Boundary Clock

A Boundary Clock (BC) is a bridge including a PTP clock, which is synchronized over its slave port to a master clock, the so called grandmaster. On the other ports, the BC acts as a PTP master clock to which PTP slaves can synchronize. The BC is a way to synchronize PTPv1 to PTPv2 devices and vice versa, since each port represents a single segment with its own PTP protocol.

Transparent Clock

The Transparent Clock (TC) is an Ethernet bridge which is capable of measuring the time a PTP message spends in the bridge and adds this value on the fly to the correction field of the PTP message. Since this residence time is relative, it is not necessary to synchronize the TC. There are two different kinds of TC mode: End-to-End (E2E) and Peer-to-Peer (P2P).

In E2E mode, only the time spent in the TC is added to the correction field and the message exchange works as described in section 2.1.2 (Synchronization Mechanism).

In P2P mode, not only the time spent in the TC is considered, but also the path delay. Both values are added on the fly to the correction field in each TC (see Figure 2.4). The

advantage is that it is not necessary to remeasure the delay if the path of the Sync packet changes, e.g. if a connection breaks and packets have to be rerouted.

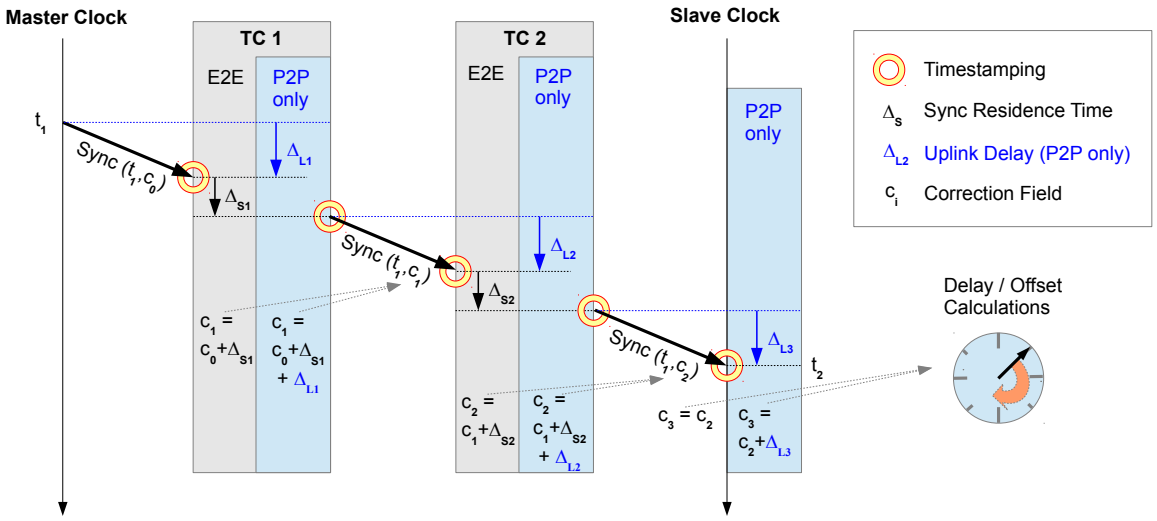


Figure 2.4: E2E and P2P mechanism

The generally used mode is the E2E since a breakdown of a connection, which could lead to the rerouting of the packages, occurs very rarely. Moreover, the used networks are typically not built up in a ring topology and therefore a rerouting is not possible.

Conclusion

PTP is an established protocol for time synchronization. The accuracy is in the range of a few nano or even pico seconds, which is exactly the required accuracy for this project. Moreover, neither dedicated time server nor specialized hardware are required. For these reasons, PTP was chosen for the implementation of this project.

2.2 PTP Stack

The aim of this section is to provide an overview of existing PTP stacks. Additionally, the chosen in-house implementation is described in detail.

2.2.1 Choice of PTP Implementation

There are several existing PTP stacks available on the internet, e.g. *Precision Time Protocol Daemon (PTPD)* and *The Linux PTP Project*[5]. Both of them are open source and available for free. There are also other implementations available which have to be payed for.

For this project, none of the existing PTP stacks was chosen because an in-house implementation was already available providing the following advantages:

- In-house know how and support
- Existing example implementations available
- Independence of third parties
- Quick response time on standard updates

2.2.2 Proprietary PTP Stack of InES

The PTP stack of InES depicted in Figure 2.5 basically consists of three layers, more precisely two interfaces and a core layer. The abstraction layer separates the protocol engine from the hardware or Operating System (OS) beneath. Functions are declared in header files which are part of the independent layer whereas the implementations are in source files in the abstraction layer.

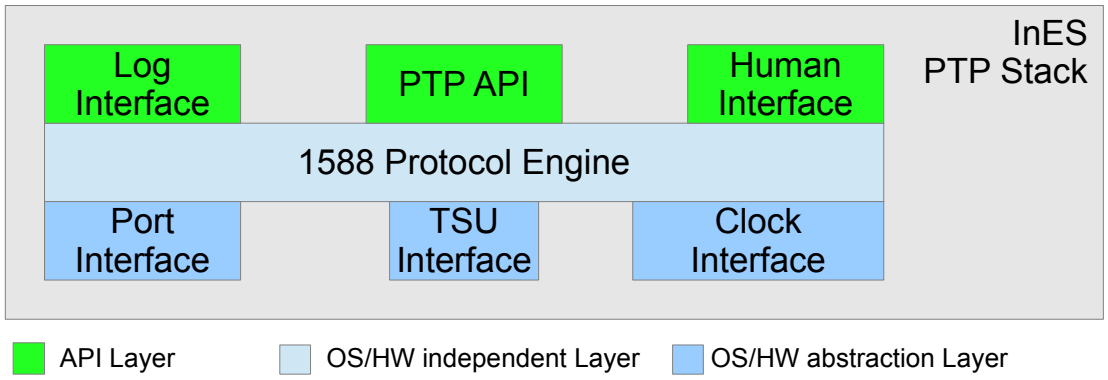


Figure 2.5: Overview of the PTP Stack implemented at InES

Interface Description

In the follwing, the interfaces of Figure 2.5 are described in more detail.

The *PTP Application Programming Interface (API)* is the interface between the user application and the 1588 Protocol Engine, containing functions to read, write or modify values. The protocol engine is started and stopped by API function calls and run by periodic timer events pushed over the API.

The *Log Interface* is responsible for printing information either on the screen or into a dedicated file.

The *Port Interface* is an OS specific implementation to send and receive PTP frames, e.g. a multicast raw socket if Linux is used.

The *Time Stamp Unit (TSU) Interface* is a hardware and an OS specific implementation to receive timestamps generated by hardware, which are forwarded to the Protocol Engine.

The *Clock Interface* is a hardware and OS specific implementation to read and write time and adjust the clock by setting the drift and offset.

The *Human Interface* is an optional application which processes user inputs from the keyboard and displays desired information on the screen.

The *GPS Interface* is also optional. It is used to receive time over National Marine Electronics Association (NMEA) messages sent by a Global Positioning System (GPS) receiver. Additionally, the PTP clock is adjusted according to the GPS time and the Pulse per Second (PPS) signal from the GPS receiver is timestamped.

2.3 Integration into Etzel

The aim of ZI is to measure synchronously using two or more Etzel boards. The measurements are processed and timestamped inside the FPGA on Etzel boards. However, the synchronized PTP time is inside the Ethernet controller. This section proposes some implementation concepts of the synchronization between the FPGA and the Ethernet controller.

Etzel consists of the processor board and a data acquisition part. The analog signal is converted by the Analog Digital Converter (ADC) with a sampling rate of 60 MS/s (See Figure 2.6). The samples are then handed to the FPGA, where they are downsampled to approx. 469 kS/s by the demodulator, i.e. a downsampling factor of 128. Afterwards, each sample is timestamped and forwarded to the user to be displayed.

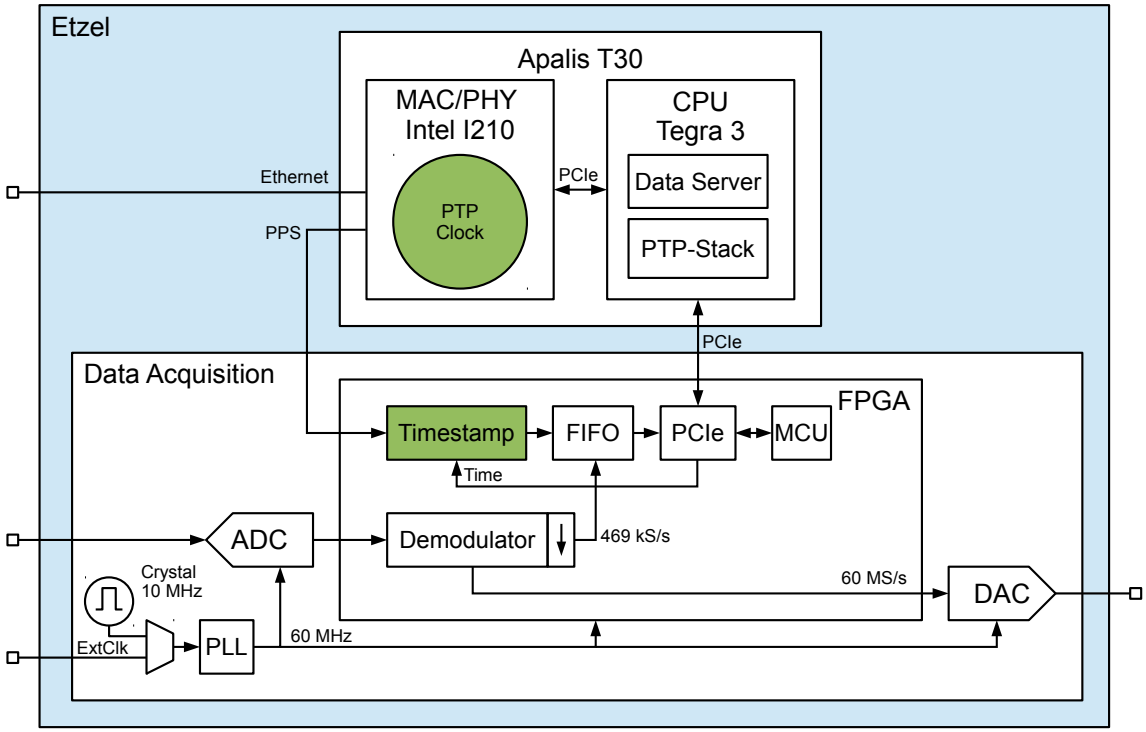


Figure 2.6: Overview of the Etzel board with the Apalis T30 processor board and the data acquisition.

2.3.1 Basic Idea

The synchronization process is divided into two parts. Firstly, the PPS signal from the Ethernet controller, which is routed to the FPGA (depicted in Figure 2.6), is used to update the synchronization every second and thus give the timestamp unit of the FPGA the ability to precisely synchronize to the full seconds. Secondly, the information about the time is missing. Therefore, the PTP application reads the time of the Ethernet controller and at the beginning of every second, the timestamp is rounded up to the next second, representing the time when the next PPS event[1] occurs. This time is then forwarded to the FPGA via the ZI application. The information about time and speed of the PTP clock enables the FPGA to fully synchronize to the PTP clock of the Ethernet controller. This basic concept is applied on Etzel in PTP master as well as slave mode, since the FPGA has to synchronize to the PTP clock in both cases.

ZI requires that timestamps are always increasing and therefore two samples never have the same timestamp. The timestamping interval is approximately 2 μs since the data from the demodulator is downsampled to 469 kS/s. Thus, the accuracy of the synchronization must always be better than ± 1 μs in order not to violate the requirement. The final synchronization accuracy shall be in the order of 10 ns.

[1]The PPS event is the rising edge on the PPS signal, which occurs at the transition of every second.

2.3.2 Synchronization of the FPGA

Three different concepts were taken into consideration for the synchronization of the PTP time of the Ethernet controller and the timestamp unit of the FPGA.

Simple Synchronization

The first idea is that the PPS signal from the Ethernet controller triggers a counter, which runs at the clock frequency of the FPGA, representing the time in nanoseconds for the timestamp unit (see Figure 2.7). After every second, the PPS signal clears the content of the counter and restarts from value 0. The next second value for the timestamp is continuously received from the processor board and buffered in the *Preset* register. Additionally, the *PTP Sync. Enable* is set. The PPS signal triggers the update of the *Timestamp* register with the value from the *Preset*. If no time is received, the *PTP Sync. Enable* is cleared and the carry from the nanosecond counter is used to increment the second value of the timestamp. The *Increment Value*, which is used for the nanoseconds counter, is constant for this implementation.

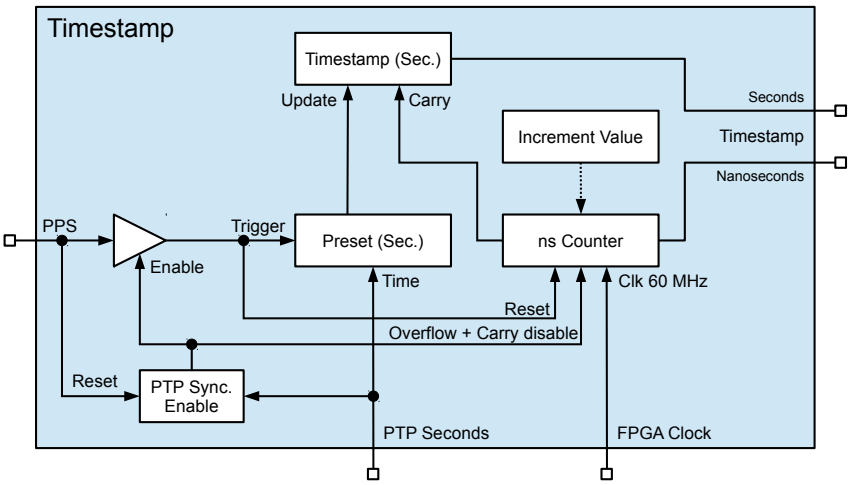


Figure 2.7: Overview of the timestamp of the FPGA

The advantage of this implementation is its simplicity. However, counter value is expected to deviate from the desired value. The offset from the deviation is getting worse during one second because of the clock drift, resulting in a jump when synchronized to the next second (see Figure 2.8). Moreover, if the FPGA clock is faster than the PTP clock, multiple measurements with the same timestamp are possible, which violates the requirements from ZI. The feasibility of this implementation directly depends on the precision of the crystal oscillator.

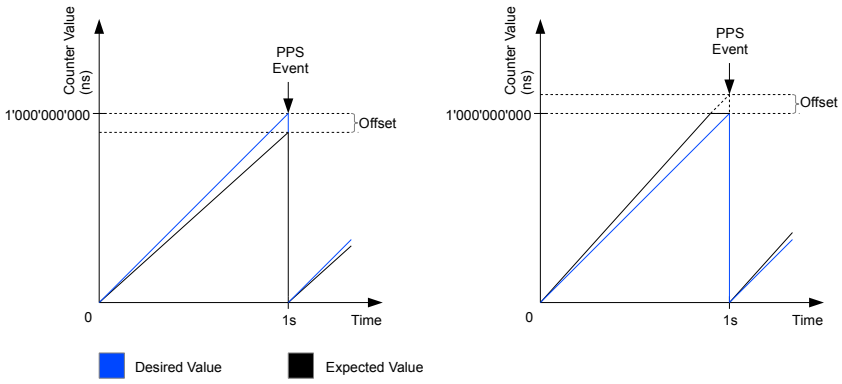


Figure 2.8: Offset with a slow (left) and fast (right) FPGA clock

The *Simple Implementation* is insufficient for the purposes of ZI since the step when synchronizing to the PPS pulse may lead to wrong results, which is unacceptable. Additionally, a 64 bit timestamp register with nanosecond resolution is already implemented and used by hardware and software blocks from the existing application and thus can not be split up into two timestamp registers.

Clock Sharing

The offset problem stems on the one hand from the fact that the clock of the Ethernet controller runs at a different speed than the clock from the FPGA and on the other hand from the jitter of the PPS signal. However, if the same clock is used for both components, the offset is stable for one second and can then easily be corrected by the new time from the Ethernet controller. Similar to the *Simple Synchronization* concept, the new time is fetched once per second into the timestamp register by the PPS signal. Since the same clock is used, there will be no drift and thus a very small correction after one second. The disadvantage is the extra routing of the PTP clock on the PCB and the usage of an extra pin on the FPGA as well as on the processor board.

Clock Sharing is also not the concept of choice because the precision of the PPS clock is insufficient and would lead to an increased phase-noise in the measurements and thus ruin the performance of Etzel.

Offset and Drift Correction

The aim of this concept is to correct the deviation of the PTP and the timestamp time of the FPGA by adjusting the offset and the drift of the timestamp counter every second.

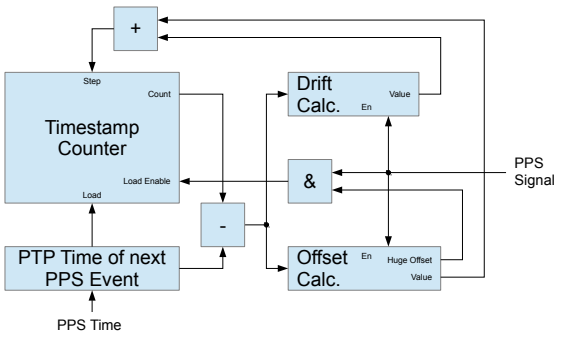


Figure 2.9: Concept of the drift and offset correction

The first idea for correcting the drift is to adjust the Phase Locked Loop (PLL) of the FPGA depicted in Figure 2.6. By doing so, the clock of the whole FPGA is affected and thus the phase-noise of the measurements is increased by the jitter of the PTP synchronization, which would again ruin the performance of Etzel. Additionally, the PLL is a simple clock multiplier and does not support the generation of arbitrary frequencies. For this reasons, the PLL in not suitable for the purposes of drift correction.

Since the PLL can not be reused, the drift and offset correction are implemented as logic blocks in the FPGA (see Figure 2.9). Basically, the timestamp counter value is compared to the PTP time after every second and the offset calculated. If the offset exceeds a threshold, the PTP time is loaded into the timestamp counter on the next PPS event instead of correcting the existing value. However, if the offset does not exceed the threshold, it is forwarded to the timestamp counter. Beside the offset calculation, the drift of the timestamp counter is determined by the drift calculation and also forwarded to the timestamp counter.

2.3.3 FPGA Implementation Concept

The main problems, which have to be tackled, are the different speed of the PTP and the FPGA clock, the jitter from the PTP synchronization and the resulting offset of the two clocks. Therefore, the concept with the offset and drift correction was chosen to be implemented, because the other concepts were all insufficient. The drift correction takes care of the clocks running at different speeds while the offset correction tackles the jitter and obviously the time offset problems. An overview of the system is given in figure 2.10.

The main problems, which have to be tackled, are the different speed of the PTP and the FPGA clock, the jitter from the PTP synchronization and the resulting offset of the two clocks. Therefore, the concept with the offset and drift correction was chosen to be implemented, because the other concepts were all insufficient. The drift correction takes care of the clocks running at different speeds while the offset correction compensates the jitter and obviously the time offset.

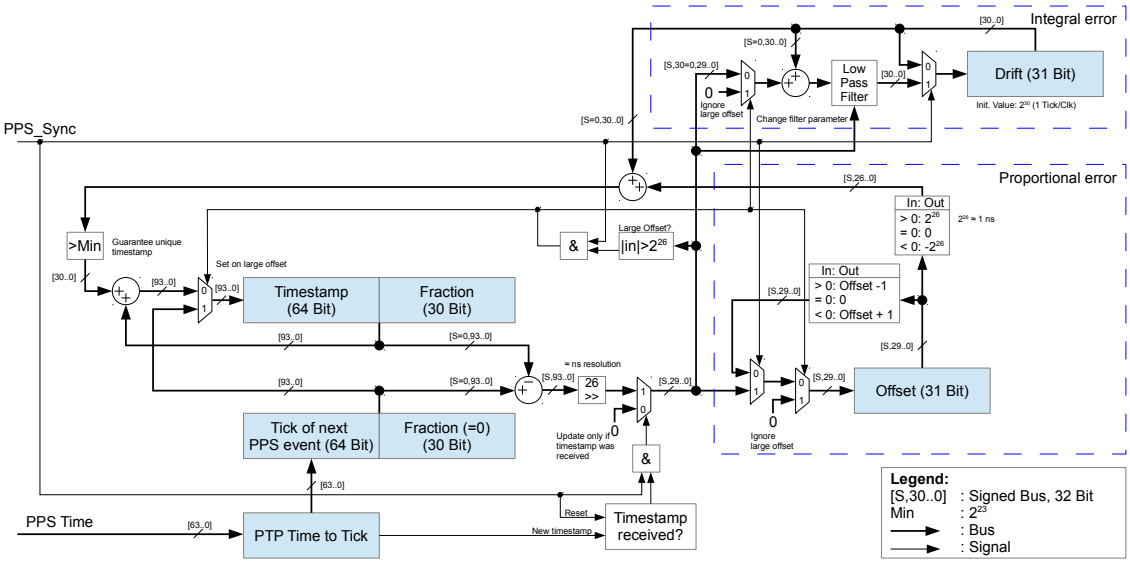


Figure 2.10: Overview of the drift and offset correction in the FPGA

The timestamp unit of the current implementation already has a 64 bit register, counting the ticks of the FPGA clock. One tick is approximately 16.7 ns[2], which is not precise enough. Since many other applications from ZI depend on this register, it must not be changed. Therefore, the timestamp is extended by a fraction register. In order to be able to correct 1 ns during one second, the fraction register has to be 30 bit wide.

$$T_{Tick} = \frac{1}{60Mhz} \approx 16.7ns$$

$$\frac{T_{Tick}}{1ns} \approx 16 (= 4bit)$$

$$60'000'000 \frac{Ticks}{s} (\approx 26bit)$$

$$26bit + 4bit = 30bit$$

Drift Correction

The speed difference of the FPGA clock and the remote clock, which is the drift, shall be equally corrected on each tick during one second (according to figure 2.10). Therefore, the Drift register holds the current amount of Fractions[3] that shall be added to the timestamp register at every clock tick. Once per second this value gets updated by the current time error (between the local time and the PPS Time).

The new drift time is low-pass filtered by an exponential moving average filter. This methods suppresses short term time error due to the noisy PTP synchronization. Further

[2]One tick is the equivalent of the FPGA clock cycle period, which is $\frac{1}{60Mhz} \approx 16.7\ ns$

[3]For higher precision, one tick is divided into 2^{30} Fractions.

details are available in section 2.3.4, Simulation of the Drift and Offset Correction.

After filtering, it has to be guaranteed that the new drift does not fall below the minimum drift of 2^{23}. In order not to have the same timestamp twice, it is mandatory to increment the tick value after 128 ticks, because a measurement sample, which has to be uniquely timestamped, is captured every 128 ticks. Therefore, the increment value (= offset increment + drift increment) must be larger than 2^{23}(see following calculation).

The initial value of the drift register is 2^{30}, which is the equivalent of 1 tick ($\approx 16.7ns$). Because of this, the drift register is 31 bit wide.

$$1Tick = 2^{30}\ TickFractions$$

$$MinimumDrift > \frac{1Tick}{128} = \frac{2^{30}}{128} = \frac{2^{30}}{2^7} = 2^{23} = 800000_{16}$$

The maximum positive drift correction is 2 ticks - 1 Fraction per tick, which leads to a correction of approximately 2 s/s:

$$MaximumPositiveDrift = 2^{31} - 1\frac{Fractions}{Tick}$$

$$MaxCorrection = \frac{MaximumPositiveDrift * 6 * 10^7 \frac{Tick}{s}}{2^{30}\frac{Fractions}{Tick}} * \frac{10^9 ns}{6 * 10^7 \frac{Tick}{s}} \approx 2s$$

The maximum negative drift is less than one second, because the timestamps have to be unique:

$$MaximumNegativeDrift = 2^{30} - 2^{23}\frac{Fractions}{Tick}$$

$$MaxCorrection = -\frac{MaximumNegativeDrift * 6 * 10^7 \frac{Tick}{s}}{2^{30}\frac{Fractions}{Tick}} * \frac{10^9 ns}{6 * 10^7 \frac{Tick}{s}} \approx -992\ ms$$

In conclusion, the drift correction is in the range of -992 ms to 2 s.

Offset Correction

The idea is to compensate the offset after each PPS event by adding ±1 ns per tick to the timestamp register.

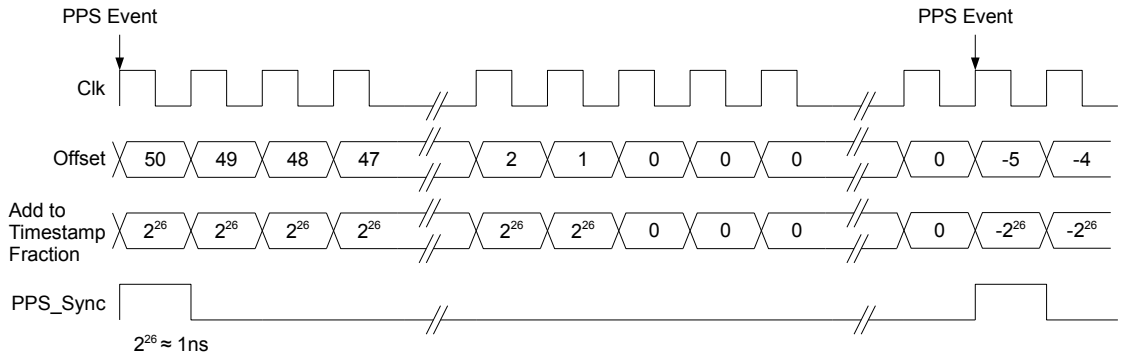


Figure 2.11: Timing diagram of the offset correction

At every tick, the timestamp register is compared to the calculated tick value of the next PPS event, received by the Ethernet controller (see Figure 2.10). The difference is then shifted by 26 bit, in order to get the difference in approximately 1 ns resolution. If a large offset is detected ($> 2^{26}$), the timestamp register is overwritten with the time of the next PPS event. This is necessary because a large error would take too long to correct, e.g. after startup. If the value is not a large offset, the difference is stored in the offset register, which is decremented to zero on every tick. As long as the offset register is nonzero, 2^{26} Fractions (equivalent to $\approx$1 ns) is added or subtracted to the timestamp register, depending on the sign of the offset (see Figure 2.11). The offset correction by itself can not cause multiple measurements with the same timestamp, because the step size of $\approx \pm 1$ ns is small enough. However, in combination with the drift, it is possible to cause multiple measurements, which is why the minimum of 2^{23} Fractions has to be guaranteed after adding the drift and the offset correction.

The size of the offset register is 31 bit, whereas the most significant bit is used as sign bit. The offset correction is in the range of $\approx \pm 60$ ms, since 1 ns or -1 ns can be corrected every tick multiplied by 60 million ticks per second. If an offset is larger, it is corrected during the next second.

Another possibility would be to overwrite the timestamp register with the PTP time instead of using an offset correction. In this case, however, the unique timestamping of the measurement frames are not guaranteed anymore if more than 127 ticks are corrected, leading to a correction range of:

$$CorrectionRange = 127Ticks * \frac{1}{6 * 10^7 \frac{Ticks}{s}} = 2.117\ \mu s$$

This correction range is sufficient since the requested synchronization precision is lower than 2 μs. However, this concept is not preferable due to the possible lack of unique timestamping. In conclusion, the concept of only correcting the offset is not used for the implementation.

2.3.4 Simulation

Before implementing one of the aforementioned concepts, simulations were made in order to verify the feasibility of the concepts and get an idea of the timestamp precision.

Simulation Overview

All simulations are executed in Matlab using double precision for all variables. The granularity of the simulation is 1 second. The timestamp error is calculated after every second and plotted in a time diagram as well as in a histogram. The simulated device is a PTP slave that synchronizes to a PTP master reference.

Figure 2.12 shows the simulation in the blue box in the center. Input vectors contain series of exact seconds that are assumed to come from the remote master. The PTP synchronization adds uncertainty to these values, which is modeled as normally distributed noise with a mean value 0 ns and a standard deviation 5 ns. The simulated synchronization then processes these values by applying offset and drift correction. It also takes into account a fixed drift of the Ethernet clock and the Etzel clock. It then generates the 'current timestamp' which is compared to the reference seconds from the input (before adding noise). The difference is finally evaluated as 'Timestamp error to master'.

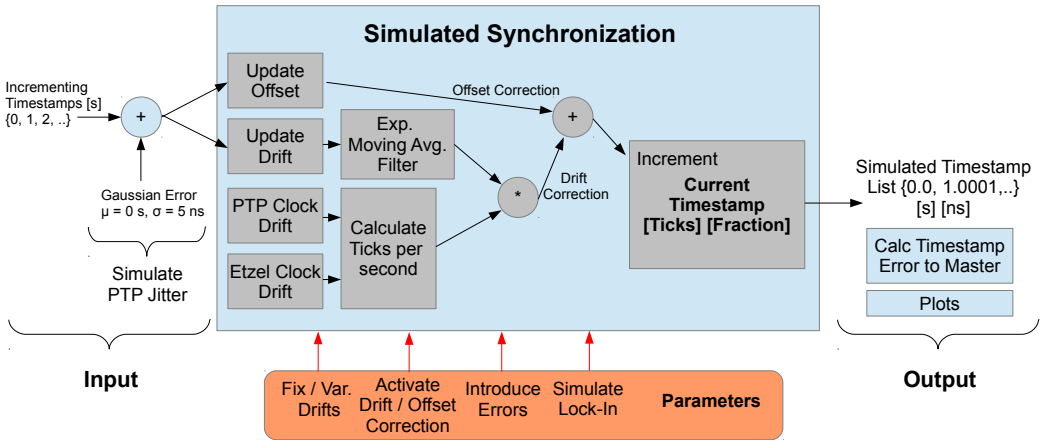


Figure 2.12: Simulation concept

The user has the option to change the following options:

- Simulation duration in seconds.
- PTP clock drift in ppm.
- FPGA clock drift in ppm.
- Standard deviation of the PPS Sync.
- Enable/Disable drift correction.
- Enable/Disable precise drift correction (divide by 60 million instead of $2^{26} \approx 67$ million).

- Skip the lock-in phase of the drift correction
- Enable/Disable offset correction.
- Enable/Disable the update of the timestamp register with the PTP time after every second.
- Define the Alpha coefficient of the exponential moving average filter, applied to drift correction.

For all simulations, the following assumptions are made:

- The master PTP clock is assumed to be perfect since it is the reference for the simulation.
- The PTP clock of the slave has an accuracy of 30 ppm.
- The accuracy of the FPGA clock is 0.5 ppm.
- The PTP synchronization noise is assumed to be normally distributed with mean 0ns and standard deviation 5 ns.

In the following, the performed simulations are described in detail and the results discussed.

Simulation of Timestamp update with PTP time

This simulation is made with neither a drift nor an offset correction. However, on every PPS event, the timestamp register is overwritten with the current PTP time. The difference just before the update is depicted as timestamp error in Figure 2.13.

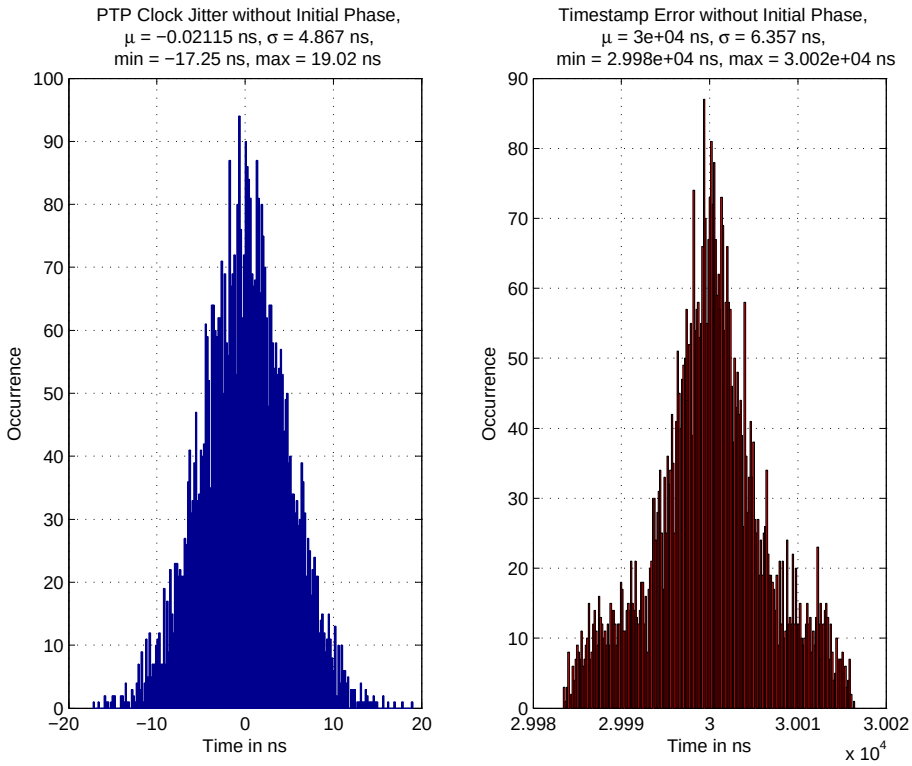


Figure 2.13: Simulation of the timestamp error. Only PTP time update enabled.

Compared to the master PTP clock, the slave has an offset with mean 30 μs due to the different clock speeds of the PTP slave and the FPGA clock. During one second, the drift is accumulated to such an error of approximately 30 μs ($= 30ppm * 1s$). The update of the timestamp register can lead to multiple measurement frames with the same timestamp if the offset is negative. This is the case if the PTP clock is slower than the FPGA clock, which is a quite possible scenario.

This concept is insufficient due to the large offset and the chance of multiple measurement frames with the same timestamps.

Simulation of the Offset Correction

This simulation was made with only the offset correction enabled. Additionally, the timestamp register is not overwritten by the PTP time on the PPS event.
The offset correction by itself leads to the same result as depicted in Figure 2.13. The problem is that the offset is calculated at the beginning of the second. Therefore, the drift of the current second can not be compensated, leading again to an offset of approximately 30 μs. However, the advantage of using the offset correction instead of updating the timestamp register with the current PTP time is that multiple measurement frames with the same timestamp are not possible. This is due to the small correction of the offset at every clock cycle instead of correcting the whole offset all at once.

This concept is also insufficient since the synchronization of the master and the slave timestamps are simulated to be in the range of ± 30 μs. This is by far too much since the precision is required to be always better than ± 1 μs, which corresponds to the timestamping rate of the measurement frames.

Simulation of the Drift Correction

This simulation was made with drift correction only, i.e. without any offset correction. Furthermore, the master and slave clock frequencies run constantly at a different speed of 30 ppm. As depicted in Figure 2.14, the system becomes unstable. This behavior is reasonable because the drift correction compensates offsets by increasing or decreasing the value added to the timestamp register. After one second, the drift correction fully compensated the clock frequency deviation. However, the timestamp error is still the same value due to the lack of an offset correction, except for the jitter caused by the PTP synchronization. In order to compensate the timestamp error, the drift correction overcompensates, resulting in a negative deviation of the clocks. Thus the system is unstable.

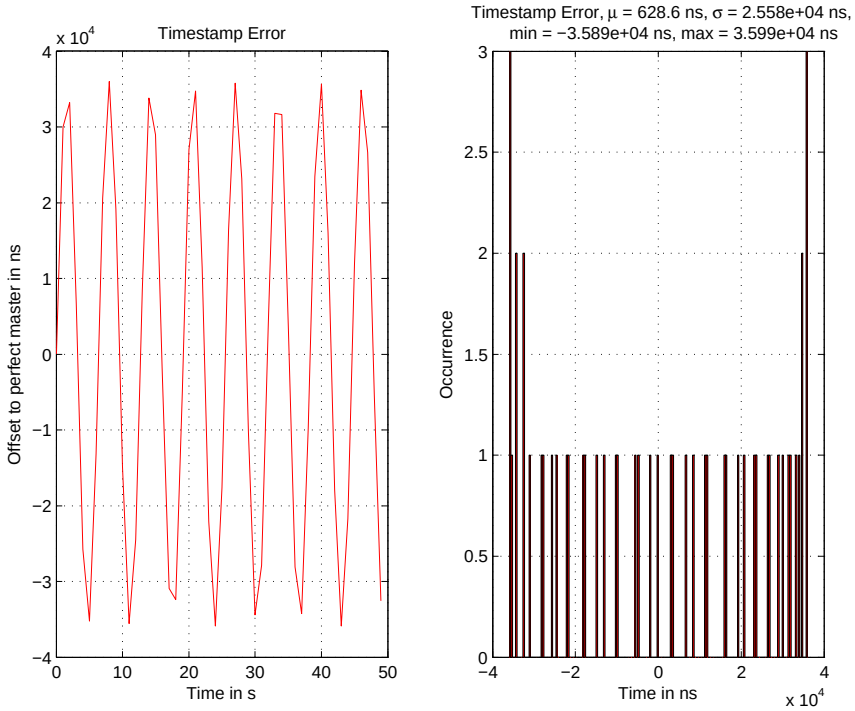


Figure 2.14: Simulation of timestamp errors using drift correction only. The system becomes unstable without offset correction.

Obviously, the drift correction can only be used in combination with an offset correction. Otherwise, the system becomes unstable.

Drift calculation The value of the *Drift* register is added to the *Timestamp* register on every tick, which means that 60 million of such additions are made during one second. In order to correct the drift precisely, the difference between the values of *Timestamp* and the

expected PTP time from the Ethernet controller would have to be divided by 60 million. However, a division in hardware is expensive in terms of time and resources. Due to that, a shift by 26, which is approximately a division by 60 million, is performed instead. The error of shifting instead of a division by 60 million is calculated in the following equation.

$$ShiftError = \frac{2^{26}}{60 * 10^6} = 1.118481067$$

The relative error of using a shift instead of a division is therefore 11.85 % of the drift error.

Simulation of the Drift and Offset Correction

This simulation determines the precision of the timestamp synchronization if offset as well as drift correction are applied. In order to get a proper distribution of the timestamp error, the lock-in phase of the drift correction is skipped, meaning that the PTP clock of the master and the slave are running at exactly the same speed. Thus, there is no difference between the precise and the approximated drift calculation.

Figure 2.15 depicts the result of the simulation of a PTP device that has been running for a while and is synchronized. Note that the timestamp error is also normally distributed.[4] However, the minimum, the maximum and the standard deviation value are higher than the jitter from the PTP synchronization. The reason is that a short shift in the mean value of the PTP synchronization jitter can make the drift correction believe that a clock drift has to be compensated. However, if the drift correction value is low-pass filtered, e.g. by using an exponential moving average filter[5] with an alpha of 0.01, the standard deviation of the timestamp errors can be reduced to 6.5 ns.

The smaller alpha is chosen, the better is the jitter of the PTP synchronization compensated. However, deviations in the clock frequency, e.g. at start up, take much longer to be compensated. The use of a variable alpha allows for the combination of a quick lock-in time with a low standard deviation, which means that above a definable threshold a larger alpha can be chosen than below the threshold.

[4]The slightly higher occurrence of large timestamp errors is due to the quantization of the FPGA clock. An early PPS event prevents the last addition of the *Drift* register to the *Timestamp*, leading to missing tick($\approx 16.7ns$). The similar behavior occurs for late PPS events.

[5]$y[n] = \alpha * x[n] + (1 - \alpha) * y[n - 1]$, current value: x[n], last value: y[n-1], new value: y[n]

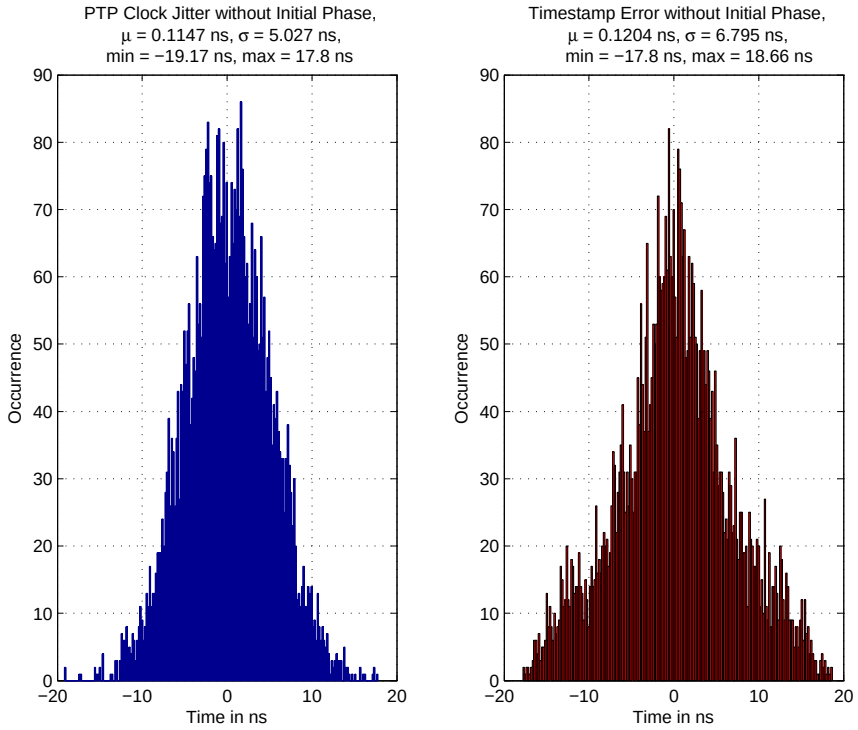
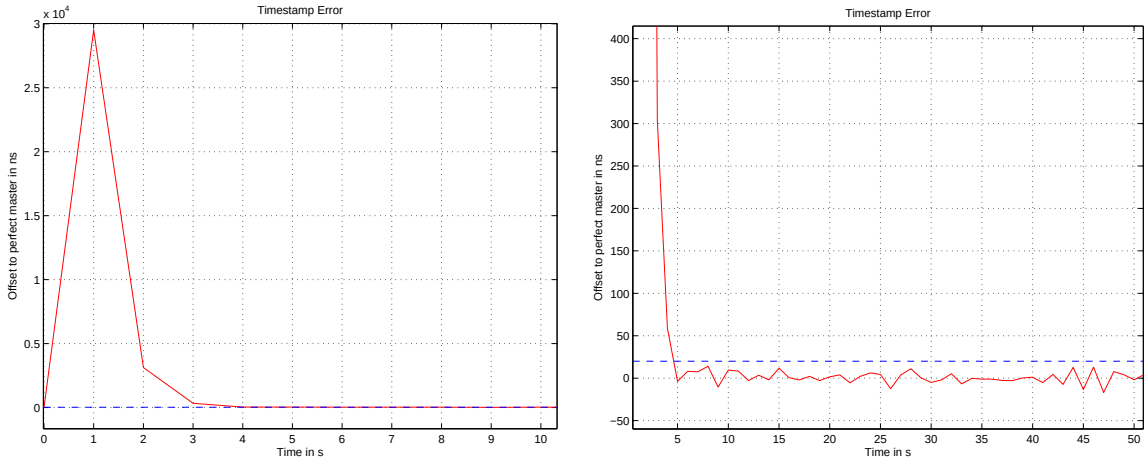


Figure 2.15: Simulation of timestamp errors using offset and drift correction.

The lock-in behavior is depicted in figure 2.16. It represents the first seconds of the system after power-up and until 'synchronization lock' has been achieved. The initial error of 30 μs, which stems from the simulated PTP clocks deviation of 30 ppm, is compensated after 5-10 seconds below a threshold of 20 ns.



(a) Overview (b) Detail

Figure 2.16: Lock-in behavior with variable alpha drift exponential moving average filter. Blue dashed line marks 20 ns.

If nothing else is mentioned, a variable alpha was used for the exponential moving average filter for drift correction. The following parameters were applied:

- Threshold for alpha switch: 50 ns
- Alpha below threshold: 1
- Alpha above threshold: 0.1

The idea of the variable alpha is to temporarily disable the moving average filter when a large offset is detected. If the offset is small, the filter is activated and helps to reduce the noise of the timestamp error.

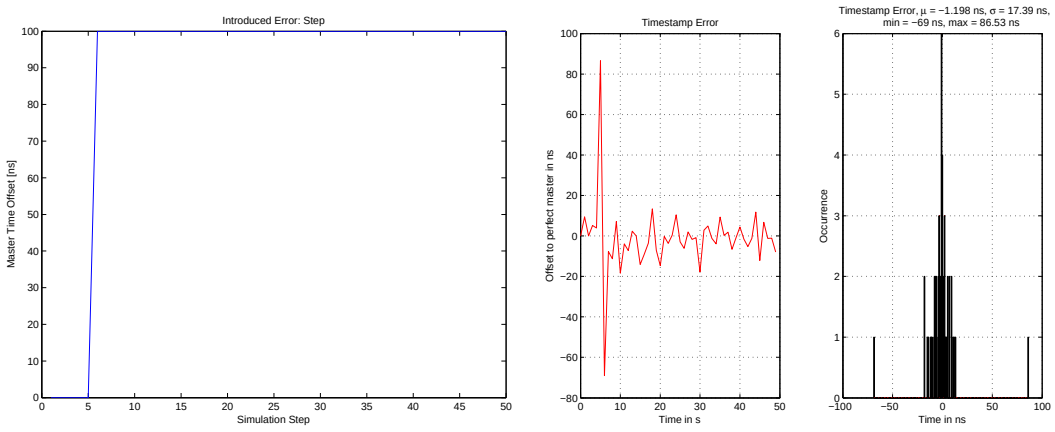
System reaction

Various errors on the master and slave side were introduced in order to observe the system reaction:

- Step in the master timestamps
- Ramp in the master timestamps
- Variable slave drift
- Missing signals are covered in section 2.3.4.

The following paragraphs describe the observed reactions of the system.

Step in the master timestamp A step of 100 ns was introduced in the master timestamps. The according input data is depicted in figure 2.17a and the system reaction in figure 2.17b. After two seconds, the error is compensated and the normally distributed noise prevails again.

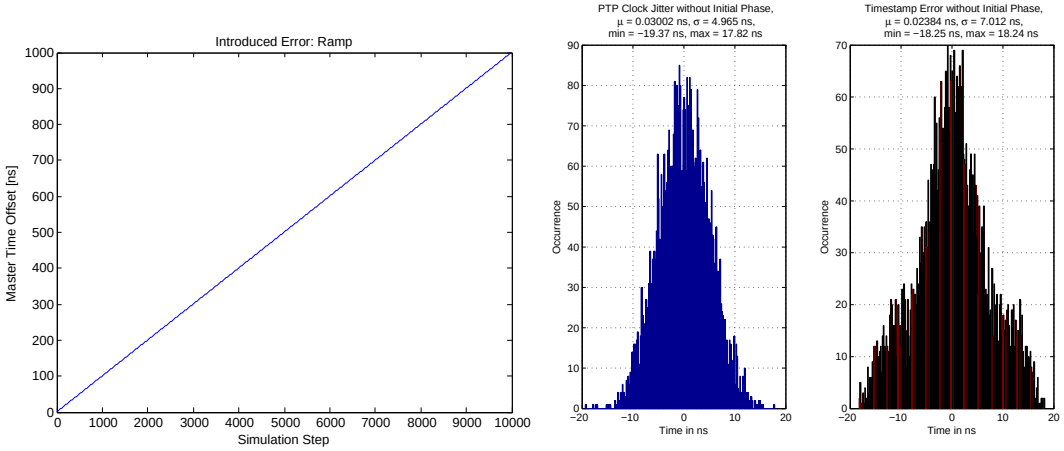


(a) Input: 100 ns step

(b) Output: Timestamp error

Figure 2.17: Reaction of the system to a manually introduced 100 ns step in the master time

Ramp in the master time A ramp of 1 μs was introduced in the master timestamps over the complete simulation duration of 10'000 seconds. This results in 0.1$\frac{ns}{step}$ of deviating master timestamps. The resulting input data and observed distribution is depicted in figure 2.18. An interesting observation is that the distribution of the timestamp error has a different form than the previously simulated output distributions. It seems more like a normal distribution and has a negligible mean value.



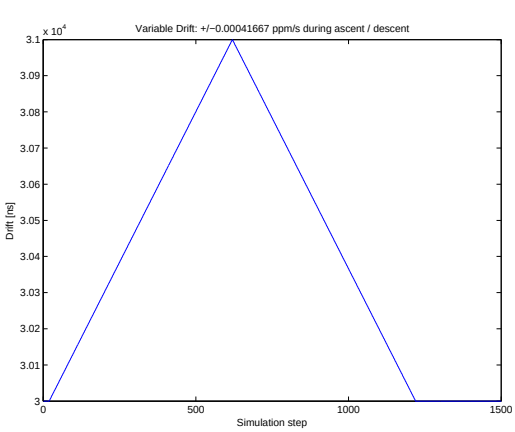
(a) Input: 1 μs ramp

(b) Comparison between distribution of the input data (left) and the timestamp error (right)

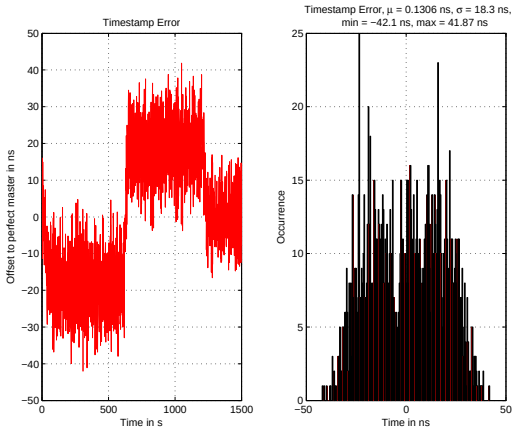
Figure 2.18: Reaction of the system to a manually introduced 1 μs ramp in the master time

Variable Drift A variable drift was introduced on the local (slave) side atop of the previously applied fixed drift. In figure 2.19 a slow drift of +1 μs and -1 μs over 10 minutes each is depicted and in figure 2.20 a fast drift over 30 seconds each. The simulated drift amounts to ± 0.25 ppm in each case which is simulated over 10 minutes or 30 seconds. The respective drift variation rates are: ± 0.00042 $\frac{ppm}{s}$ and ± 0.00833 $\frac{ppm}{s}$. It is assumed that these rates cover even worst case drift variation due to the impact of temperature variation on clock oscillators.

Due to the quick reaction of the moving average filter with variable alpha, the timestamp error stays below ± 100 ns in all tested cases.

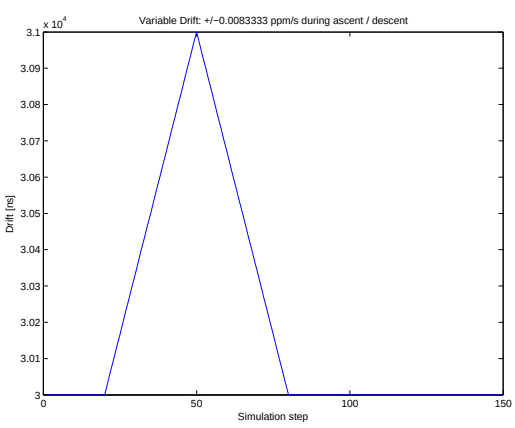


(a) Input: $\pm 1\ \mu s$ drift over 10 minutes each

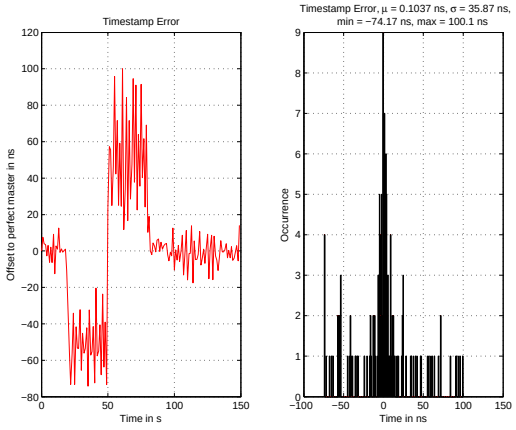


(b) Output: Timestamp error

Figure 2.19: Reaction of the system to a manually introduced slow variable drift on the slave side (local)



(a) Input: $\pm 1\ \mu s$ drift over 30 seconds each



(b) Output: Timestamp error

Figure 2.20: Reaction of the system to a manually introduced fast variable drift on the slave side (local)

Error Handling

The communication between FPGA and Central Processing Unit (CPU) takes place over the following interfaces:

- PPS via dedicated signal line
- Data exchange over PCIe
  - Timestamp Register 'Second' from the CPU to the FPGA
  - (Variable moving average Alpha from the CPU to the FPGA)
  - Readout of FPGA Registers to the CPU (for Debugging and Control purposes)

It has to be guaranteed that the FPGA still produces adequate timestamps if any of these communication methods fail. Possible scenarios include: Loss of network connection (no more active PTP synchronization), crash of the PTP application (therefore no Timestamp Updates), crash of the Ethernet controller (no PPS signal), complete communication breakdown between FPGA and CPU.

Loss of Network Connection In this case, the generation of PPS signals via the Ethernet controller and Timestamp Updates via the PTP software should still continue and therefore allow the FPGA to generate correct (although not synchronized) timestamps.

No Timestamp Updates If the timestamp does not get updated, the next PPS signal would result in the FPGA detecting a large offset. The FPGA would then reset its timestamp register to the old input value and repeat this behavior on every PPS pulse until the timestamp gets updated again.

In order to prevent the above-mentioned behavior, a block is implemented in the FPGA, which enables the overwriting of the timestamp only when a new PPS time is downloaded from the CPU. Otherwise, the timestamp unit keeps running as a free-running counter.

Another possibility to mitigate the no timestamp update problem would be a simple software solution, which introduces a check of the timestamp update value before it is sent from the CPU to the FPGA and manipulate it if it is faulty. This check should occur once per second and compare the timestamp value to the previous value. If it has not been changed, it should increment it by 1 second. The only situation where this would not work is if the PTP master correctly advises to change the time back by 1 second, resulting in the same second. This would result in an ongoing error of 1 second. If the check is performed on the last two timestamps, a failure would result in two same second timestamps before the check triggers and increments. Since this concept produces additional problems, it is not implemented.

No PPS Signal If no PPS Signal occurs for more than 1 second, the FPGA Timestamping should continue running without any interruption. The critical moment is when the PPS signal reappears. The difference between the synchronized PTP time and the 'free-running' FPGA time is the accumulated 'error from PTP master' during the time when the PPS signal was missing. As long as this difference does not exceed the large offset barrier, it can be corrected with Drift and Offset correction. Otherwise a direct update of the timestamp will occur.

No PPS Signal and no Timestamp Update The consequence of both signals missing, is the same as 'No PPS Signal'. The same behavior results: The critical moment is when the PPS signal plus Timestamp Updates reappear. The difference between the synchronized PTP time and the 'free-running' FPGA time is the accumulated error during the time when the PPS signal was missing. As long as this difference does not exceed the large offset barrier, it can be corrected with Drift and Offset correction. Otherwise a direct update of the timestamp will occur.

2.3.5 System Modeling

The offset and drift correction is similar to a Proportional-Integral (PI) controller (see Figure 2.10). The FPGA implementation can thus be described as a control loop model. The model is used to determine and optimize the performance of the implementation. Therefore, traditional control engineering tools can be used, which simplify the development process.

Control Loop Overview

The basic control loop of the FPGA implementation is depicted in Figure 2.21. The reference of the control loop is the PTP time, which is downloaded every second. The measured output is the value of the timestamp register. The measured error is fed into the controller, which basically consist of the offset and drift correction. The system in this case is a simple integrator, which adds the time from the offset and drift to the timestamp.

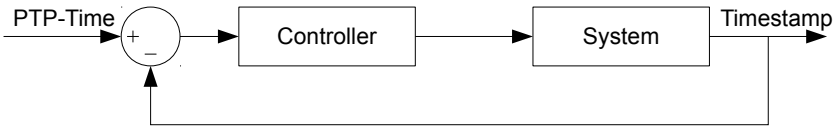


Figure 2.21: Control loop of the FPGA implementation.

Controller

The controller depicted in Figure 2.22 consists of two parts. Firstly, the offset correction, which represents the proportional part and secondly the drift correction, which represents the integral part. The introduced shift error gain is due to shifting the measured error by 26 bits instead of dividing by 60 millions. Due to this, the implementation is much simpler (see section 2.3.4, paragraph Drift calculation). This shift error factor is determined as follows:

$$k = \frac{6 * 10^7}{2^{26}} = 0.8941$$

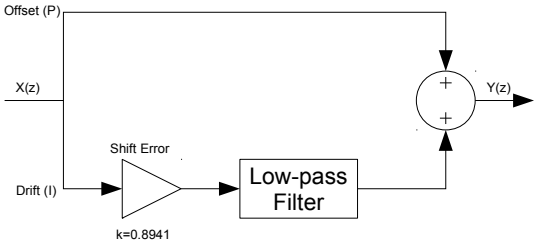
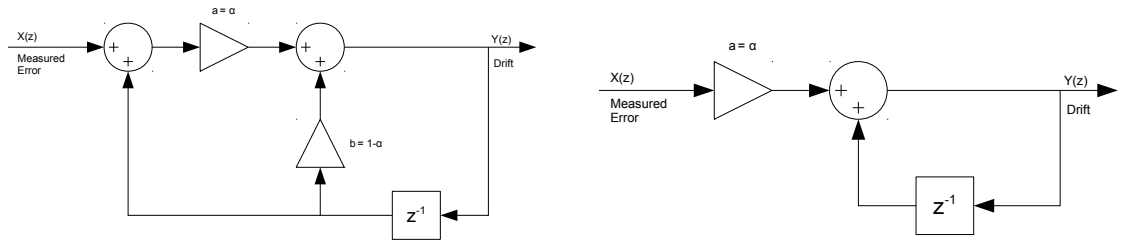


Figure 2.22: Block diagram of the controller.

Low-pass Filter

The low-pass filter introduced in the previous paragraph is depicted in detail in Figure 2.23a. First, the value of the last drift is added to the measured error. This is done in order to bias the input value. The sum is then added to the previous drift, in order to get the new value for the drift register. The weighting of the new value and the previous drift is α and $(1-\alpha)$, respectively.

The block diagram in 2.23a is the direct conversion of the FPGA concept to a model. By applying algebraic transformations, the model can be simplified to Figure 2.23b on the right hand side. The simplification calculations can be found in section 2.3.5 in paragraph Low-pass Filter.



(a) Direct model of the FPGA concept.

(b) Simplified block diagram.

Figure 2.23: Block diagram of the low-pass filter of the drift correction.

Calculation of the Transfer Functions

In this section, the transfer functions of the above mentioned model are determined. All the transfer functions are time discrete since the reference, which is the PTP time from the Ethernet controller, is updated only once per second. Moreover, the sampling time period T_s is 1 second.

Low-pass Filter In this paragraph, the transfer function of the low-pass filter is determined. The basic equation of this filter is:

$$(X(z) + Y(z) * z^{-1}) * \alpha + Y(z) * z^{-1} * (1-\alpha) = Y(z)$$

and can be simplified to:

$$Y(z) = \alpha * X(z) + Y(z) * z^{-1}$$

This equation can be rearranged to the following transfer function G_f of the low-pass filter:

$$\frac{Y(z)}{X(z)} = G_f(z) = \frac{a * z}{z-1}$$

Controller In this paragraph, the transfer function of the controller is determined. the controller consists of two paths, the proportional and the integral one. The transfer function G_c of the controller can be calculated as follows:

$$G_c(z) = (1 + k * G_f(z)) = 1 + \frac{k * \alpha * z}{z - 1} = \frac{(1 + k * \alpha) * z - 1}{z - 1}$$

The constant factor k is 0.8941 and is due to the shifting error.

System The system block consists of only an integrator, since the time is continuously added to the timestamp register. The transfer function of a time discrete integrator is:

$$G_s(z) = G_{integrator}(z) = \frac{T_s}{z - 1} = \frac{1}{z - 1}$$

Open Loop The open loop gain G_o is the transfer function of the whole implementation without a feedback path.

$$G_o(z) = G_c(z) * G_s(z) = \frac{(1 + k * \alpha) * z - 1}{z - 1} * \frac{1}{z - 1} = \frac{(1 + k * \alpha) * z - 1}{(z - 1)^2}$$

Closed Loop The closed loop gain G_{cl} is the transfer function of the whole implementation with a feedback path and thus the transfer function of the FPGA implementation.

$$G_{cl}(z) = \frac{G_o(z)}{1 + G_o(z)} = \frac{\frac{(1 + k * \alpha) * z - 1}{(z - 1)^2}}{1 + \frac{(1 + k * \alpha) * z - 1}{(z - 1)^2}} = \frac{(1 + k * \alpha) * z - 1}{z * (z + k * \alpha - 1)}$$

Model Simulations

The control model is used to simulate the behavior of the FPGA implementation. The simulation covers a step in the timestamp, a ramp in the timestamp and a clock drifting away. In the following, details about the simulations and the results are described.

Simulation Overview The simulations were conducted in Matlab. With the use of this program, the output of the above determined transfer function of the whole FPGA implementation is computed. Unfortunately, it is not possible to define an initial value, which is why only deviations from the continuously increasing timestamp are used as input data. If the increasing of the timestamp were simulated, the system would need an certain amount of time in order to count as fast as the reference. The results from the model simulation are additionally compared to the simulation results of the Very High Speed Integrated Hardware Description Language (VHDL) test bench, which is used to test the implementation of the FPGA, and the previously introduced Matlab simulation. Note that the test bench simulation was not conducted by simulating every tick but 2^{14} ticks concurrently, resulting in a precision loss. The reason is the extremely long simulation time.

Determination of Alpha The filter coefficient α for all simulations is 0.125 ($= \frac{1}{8}$). α was determined by the Monte Carlo simulation depicted in Figure 2.24. The x-axis represents α while the y-axis represents the standard deviation of the PTP time synchronization. α above 0.15 can not compensate the jitter from the PTP time synchronization, whereas α below 0.1 can not compensate the slow frequency aspects of the jitter. Therefore, α was chosen to be 0.125 because it is between 0.1 and 0.15 and easily to be implemented in hardware.

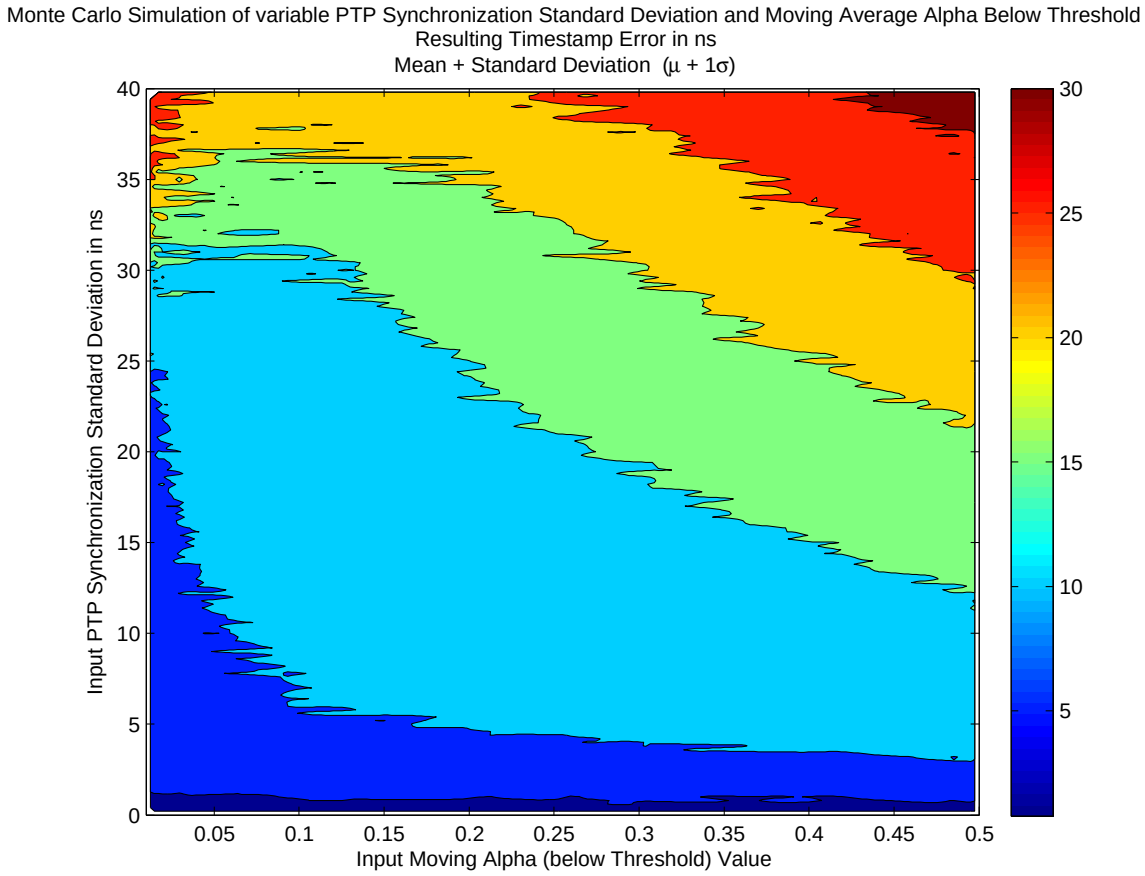


Figure 2.24: Monte Carlo simulation for determining α. Best values between 0.1 and 0.15.

Timestamp Step In this scenario, a sudden step in the reference time, which is the PTP time from the Ethernet controller, is simulated. The results are depicted in Figure 2.25. The input data, which is depicted in the upper plot, is the PTP time between two PPS events. In idle mode, the number of ticks between two PPS is 60 million. In the simulation, the values are slightly lower because of the limited resolution of the test bench.
At second four, the step in the PTP time occurs and has an amplitude of approximately 82000 ticks per seconds (equivalent to 1.37 ms). The system can react to the step for the first time one second after the step occurred. The offset correction then fully compensates the step. However, the drift correction needs around 25 seconds to recover. During this time, the measured error is negative.

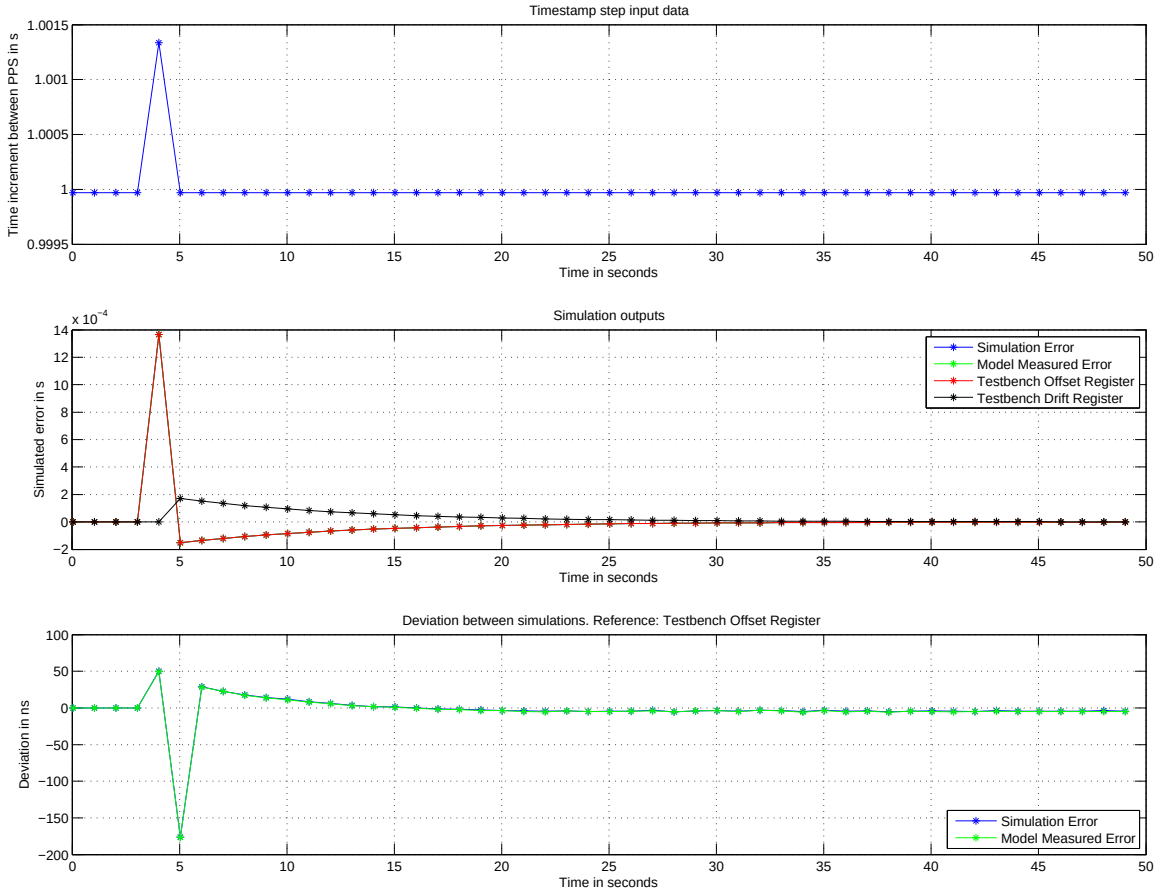


Figure 2.25: Three simulations of a step in the timestamp. The deviation between the simulations is depicted in the plot at the bottom.

The results from the model simulation, the test bench of the implementation and the simulation match very well. The difference between the three simulations is depicted in the figure at the bottom. The reference signal is the Testbench Offset Register. The maximum difference of -175 ns at second five seems to be very large at first sight, however, the introduced timestamp step of 1.37 ms is also very large and unrealistic in daily use. The reason for choosing such a huge timestamp step is the limited resolution of the test bench simulation. The deviation between the test bench and the other two simulations stem from the limited resolution of the test bench simulation.

PTP Time Ramp In this scenario, a ramp in the reference time is simulated. Such a ramp is equivalent to a step in the PTP clock frequency, meaning that e.g. the PTP master clock changes the frequency at a certain point in time. The results are depicted in Figure 2.26.
The input data are in the same format as described in the previous paragraph, representing the PTP time between two timestamps. Starting at second four, the time between two PPS events is constantly growing larger representing the ramp in the reference time. The simulated clock frequency deviation, which leads to such a ramp, is 273 ppm or an increase of 16383 ticks between two PPS events. Typical values for clock drifts are in the range of 30

ppm, however, the limited resolution of the test bench does not allow such small drifts.

Again, the system can react to the time reference ramp for the first time one second after the ramp started. The error from the last second is continuously compensated by the offset correction, preventing an accumulation. However, the drift correction needs around 45 seconds to adapt the new clock speed.

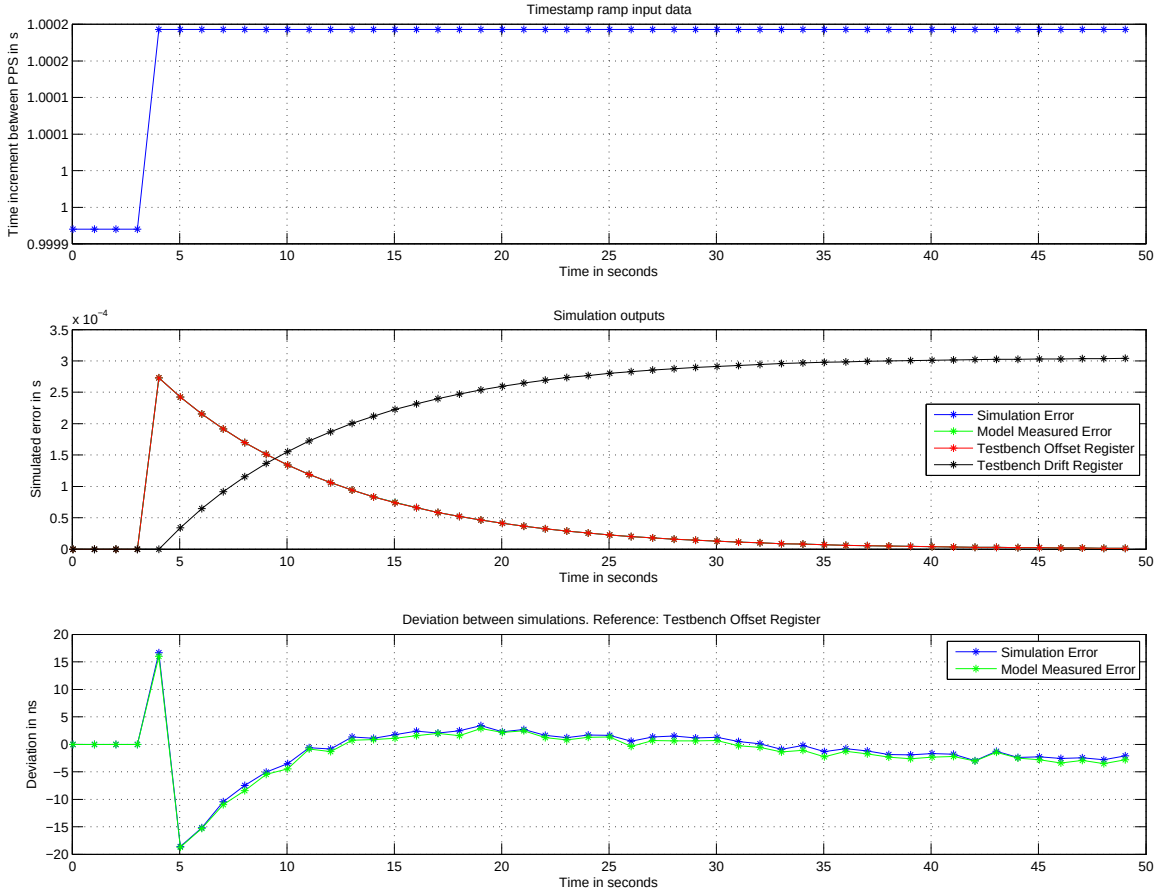


Figure 2.26: Three simulations of a ramp in the timestamp. The deviation between the simulations is depicted in the bottom plot.

Similar to the Timestamp Step simulation, the result of the Timestamp Ramp simulation is depicted. This simulation proves that all the simulations have pretty much the same output. The difference between the simulations is depicted in the figure at the bottom. The Testbench Offset Register was taken as reference signal. The deviation between the test bench and the other two simulations stems from the limited resolution of the test bench simulation.

Clock Drift In this scenario, the drift of the PTP clock frequency is simulated. Such a drift could be caused in reality by a temperature change in the master device. The results are depicted in Figure 2.27.
The input data is in the same format as described in the previous paragraphs. From second four to 26, the clock frequency of the master is linearly increasing. After that, the frequency

decreases again until the initial frequency is reached.

The error in this case is increasing at the beginning and levels off at a value of about 2.3 ms. This result is reasonable since the drift correction is not capable of correcting a continuously changing frequency. At second 27, the clock drift starts to decrease, which causes the error to become negative. In the end, the error levels off at the value of -2.3 ms.

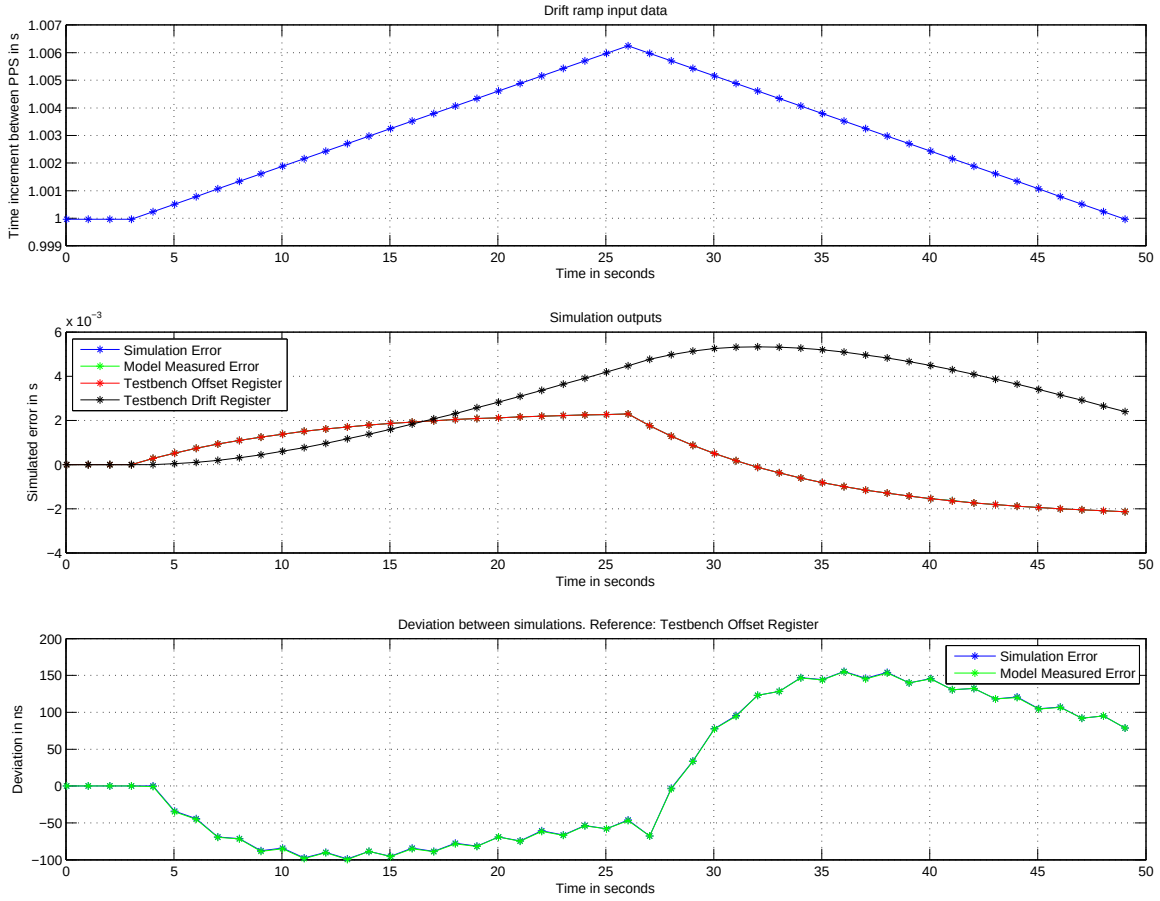


Figure 2.27: Three simulations of a ramp in the clock frequency. The deviation between the simulations is depicted in the bottom plot.

Again all the simulations produce pretty much the same output. However, this simulation reveals the limit of the drift and offset correction. Permanent changes in the clock frequency can not be fully compensated and result in a constant timestamp error as depicted in the center figure. The differences between the timestamp errors are depicted in the figure at the bottom. The Testbench Offset Register was taken as reference signal. The deviation between the errors stems from the limited resolution of the test bench simulation.

Chapter 3

Implementation

3.1 Project overview

The implementation process is divided into two steps. First, the in-house PTP stack was implemented on the Apalis Evaluation Board[6] so that the correct functionality of the time synchronization can be proved. Etzel, which is the final product of ZI on which the time synchronization will be implemented, uses the same processor board as the Apalis Evaluation Board. Second, the PTP stack will be integrated into Etzel.

3.2 PTP Stack Implementation on Apalis T30

This section describes the implementation of the InES PTP stack on the Apalis Evaluation Board. Moreover, the occurred problems of the implementation and the corresponding solutions are described.

3.2.1 Overview

The implementation, depicted in Figure 3.1, consists of four layers: PTP Application, InES PTP Stack, OS and the Ethernet Controller.

The *PTP Application* layer contains the main function, which is responsible for starting and maintaining the 1588 Protocol Engine and handling the keyboard input.

The *PTP Stack* from InES is the in-house implementation of the PTPv2 standard. For more details, see Section 2.2.2.

The *OS Linux* layer basically consists of three parts. The network socket and the PTP clock interface communicate with the IGB, which is the Intel driver of the Ethernet controller beneath. The original version of the driver is v5.1.2. Due to insufficient implementation of the offset adjustment and the lack of PPS generation, the driver was modified (see Section 3.2.5 and 3.2.8). The handling of the PTP as well as the frames of the PTP application is pretty straight forward and was already implemented and perfectly working on the original version of the IGB driver.

The I210 from Intel is the Ethernet controller mounted on the Apalis T30 processor board. It supports the PTPv2 standard and has therefore dedicated registers to adjust the offset and the drift of the PTP clock. Furthermore, the I210 supports hardware timestamping of incoming and outgoing frames.

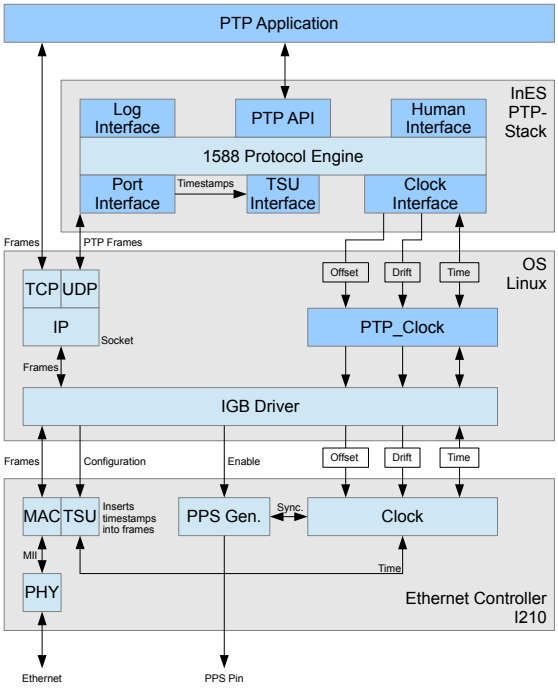


Figure 3.1: Overview of the PTP Stack integration

3.2.2 PTP Application

The PTP application is the entry point of the program and thus contains the main function which handles the passed arguments and initializes the 1588 Protocol Engine (see Figure 3.2). Additionally, three threads are started:

- The *Timer Thread* creates a tick event every 100 ms for the 1588 Protocol Engine.
- The *Listen on Keyboard Thread* receives the user input from the keyboard. If a whole line is received, it will be forwarded to the 1588 Protocol Engine by creating an event.
- The *PPS Listener Thread* prepares the time of the next PPS event at the beginning of every second and forwards it to the ZI application, which will download the time to the FPGA.

After the initialization, the *main()* function calls the dispatcher of the 1588 Protocol Engine on every tick event, until the user quits the program. The dispatcher is responsible for handling the pending events of the protocol engine.

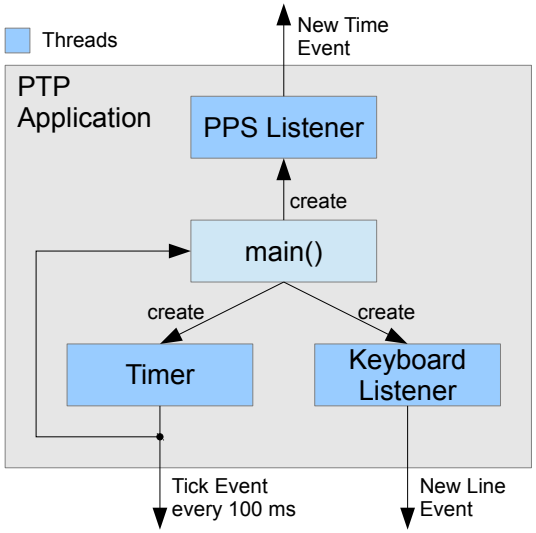


Figure 3.2: Overview of the PTP Application

3.2.3 Activation of the HW Timestamping

The Ethernet controller I210 supports software as well as hardware timestamping. The PTP protocol requires timestamps as precise as possible which is why the hardware timestamping is chosen for all of the configurations. For further readings, the Linux documentation[7] provides a description of the timestamping options.
First, the hardware of the Ethernet controller has to be enabled for transmit and receive hardware timestamping (see Listing 3.1). The timestamping of transmitting frames can either be activated or deactivated, whereas receiving frames can additionally be filtered. This provides an advantage in a scenario of high network traffic. In this case, it is possible to timestamp only incoming PTP messages instead of every frame, setting the hwconfig.rx_filter to HWTSTAMP_FILTER_PTP_V2_L2_EVENT. More details about RX timestamp filtering can be found in Section A.1 in the appendix.

```
1  // ptp2_portitf_apalis_t30.c
2
3  struct ifreq hwtstamp;
4  struct hwtstamp_config hwconfig;
5
6  ...
7  // *** Activate HW Timestamping ***
8  // Initialization
9  memset(&hwtstamp, 0, sizeof(hwtstamp));
10 strncpy(hwtstamp.ifr_name, interface_name, sizeof(hwtstamp.ifr_name));
11 hwtstamp.ifr_data = (void *)&hwconfig;
12 memset(&hwconfig, 0, sizeof(hwconfig));
13
14 // Timestamp all TX packets
15 hwconfig.tx_type = HWTSTAMP_TX_ON;
16
17 // Timestamp all RX packets.
18 hwconfig.rx_filter = HWTSTAMP_FILTER_ALL;
```

```
19
20 // Write the configuration
21 // Caution: Socket has to be open!
22 if (ioctl(port_itf.sock_id, SIOCSHWTSTAMP, &hwtstamp) < 0)
23 {
24   PTP2_MONITORING( PTP2_LEVEL_ERROR | PTP2_CAT_PORT_ITF , (PTP2_FAULT_LOG_ERROR
       , "Can't set HW Timestamping") );
25   goto clean_up;
26 }
27 ...
```

Listing 3.1: Activating the Ethernet controller for HW timestamping

Second, the socket has to be configured so that the usage of hardware timestamping is enabled in the OS. Additionally, the format of the timestamp has to be chosen (see Listing 3.2). Finally, the socket is reconfigured by calling the function *setsockopt()*.

```
1 // ptp2_portitf_apalis_t30.c
2
3 int so_timestamping_flags = 0;
4
5 ...
6 // *** Set Socket to activate HW Timestamp ***
7 // Timestamp RX and TX in hardware
8 so_timestamping_flags |= SOF_TIMESTAMPING_TX_HARDWARE |
      SOF_TIMESTAMPING_RX_HARDWARE;
9
10 // Set Socket to report timestamp in control message and its format
11 so_timestamping_flags |= SOF_TIMESTAMPING_RAW_HARDWARE;
12
13 // Write the socket options
14 // Caution: Socket has to be open!
15 if(setsockopt(port_itf.sock_id, SOL_SOCKET, SO_TIMESTAMPING, &
       so_timestamping_flags, sizeof(so_timestamping_flags)) < 0)
16 {
17   PTP2_MONITORING( PTP2_LEVEL_ERROR | PTP2_CAT_PORT_ITF , (PTP2_FAULT_LOG_ERROR
       , "Error while setting socket options\n") );
18   goto clean_up;
19 }
20 ...
```

Listing 3.2: Configure the socket for HW timestamping

The socket can be configured to timestamp receive and send packets either in software or in hardware. The corresponding flags are as follows:

- SOF_TIMESTAMPING_TX_HARDWARE: try to obtain TX time stamps in hardware
- SOF_TIMESTAMPING_TX_SOFTWARE: try to obtain TX time stamps in software
- SOF_TIMESTAMPING_RX_HARDWARE: try to obtain RX time stamps in hardware
- SOF_TIMESTAMPING_RX_SOFTWARE: try to obtain RX time stamps in software

Moreover, the socket supports three different formats for reporting timestamps in a generated control message.

- SOF_TIMESTAMPING_SOFTWARE: Report systime if available
- SOF_TIMESTAMPING_SYS_HARDWARE: Report hwtimetrans if available
- SOF_TIMESTAMPING_RAW_HARDWARE: Report hwtimeraw if available

However, only the software and the raw hardware timestamping are available from the Ethernet controller.

3.2.4 Timestamping of PTP Frames

The timestamping of ingoing and outgoing frames is performed by the TSU in the Ethernet controller[8]. The timestamps are inserted on-the-fly and the checksum of the headers are recalculated. The TSU is as close as possible to the PHY in order to keep the timestamp as deterministic as possible. The message timestamp point (depicted in Figure 3.3) is at the very beginning of the frame for the same reason.

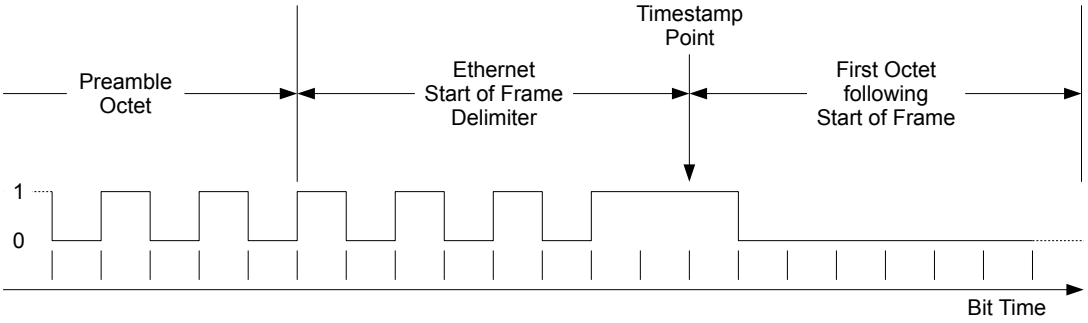


Figure 3.3: Timestamp point of in- and outgoing frames

Incoming PTP frames are received and timestamped by the MAC of the Ethernet controller. The reception timestamp is attached to the frame and via the kernel forwarded to the port interface of the PTP Stack which on the one hand provides the 1588 Protocol Engine with the received message and on the other hand moves the frame to the TSU interface of the PTP stack, where the timestamp is extracted from the frame and forwarded to the 1588 Protocol Engine.
Timestamps of outgoing frames are similarly received since the transmit PTP frame is looped back after the on-the-fly timestamping of the TSU to the message error queue of the socket. The origin field indicates that this message stems from the timestamping rather than from an error.

The RX as well as the TX messages are received from the socket by calling the *recvmsg()* function (see Listing 3.3). In order to receive messages from the error queue, the flag *MSG_ERRQUEUE* must be set.

```
1 // ptp2_portitf_apalis_t30.c
2
3 void *PTP2_PortItf_T_receiveThread( void *port_number )
```

```
{
  ...
  // Receive TX messages (looped back via the error queue)
  pkt_length = recvmsg(port_itf.sock_id, &msg, recvmsg_flags | MSG_ERRQUEUE );
  ...
  // Receive RX messages
  pkt_length = recvmsg(port_itf.sock_id, &msg, recvmsg_flags );
  ...
}
```

Listing 3.3: Receive messages including timestamps from the socket.

The timestamp is located in the *control message* of the corresponding message. In the cases of RX and TX, the message is therefore forwarded to the TSU interface in order to extract the timestamp from the message (see Listing 3.4). Afterwards, the RX message is forwarded to the 1588 Protocol Engine for further handling.

```
// ptp2_timestampitf_apalis_t30.c
PTP2Boolean PTP2_TimestampItf_T_pushTimestampToHandler( unsigned char msg_type
    , struct msghdr *msg, UInteger8* const clock_id, UInteger16 sourcePortId,
    UInteger16 sequenceId)
{
  struct cmsghdr *cmsg;
  ...
  /* get the timestamp from the socket */
  for (cmsg = CMSG_FIRSTHDR(msg);cmsg;cmsg = CMSG_NXTHDR(msg, cmsg))
  {
    switch (cmsg->cmsg_level)
    {
      case SOL_SOCKET:
        switch (cmsg->cmsg_type)
        {
          ...
          case SO_TIMESTAMPING:
          {
            // Get control message data
            struct timespec *stamp = (struct timespec *)CMSG_DATA(cmsg);

            // The cmsg data are in the format of:
            // struct scm_timestamping {
            //       struct timespec systime;
            //       struct timespec hwtimetrans;
            //       struct timespec hwtimeraw;
            // };
            // However, this struct is not defined!

            // Choose the raw time since hwtimetrans is not supported!
            stamp++;
            stamp++;

            // Copy timestamp
            timestamp2->timestamp_.seconds_field_ = stamp->tv_sec;
            timestamp2->timestamp_.nanoseconds_field_ = stamp->tv_nsec;

            // Add timestamp to 1588 Protocol Engine
            PTP2_Time_T_add( &timestamp2->timestamp_, &timestamp2->timestamp_,
    &temp_time );
            PTP2_TimestampItf_T_push( timestamp2 );
```

```
39                 break;
40               }
41               default: break;
42             }
43             break;
44           default: break;
45       }
46     }
47     ...
48 }
```

Listing 3.4: Extract the timestamp from the PTP message.

3.2.5 Generation of PPS

The Ethernet controller provides basically three different possibilities of generating a PPS signal. Only one is described in the following. The other two can be found in the datasheet of the I210[9].

The PPS generation is autonomously executed in hardware. The principle described in the following is also depicted in Figure 3.4. The SYSTIM register, which contains the current PTP time, is compared to the TRGTTIM register, which is initially set to one or two seconds in the future, in order not to miss the start of the PPS generation. As soon as the value of SYSTIM is equal or larger than TRGTTIM, the initially defined SDP is cleared. Additionally, the value from the third register, FREQOUT, is added to the TRGTTIM register. Therefore, the value of the FREQOUT register represents the half clock cycle duration. As soon as the value of SYSTIM is equal or larger than TRGTTIM the second time, the Software Defined Pin (SDP) is toggled and the value from FREQOUT is again added to TRGTTIM. From this point, this procedure will continue until it is stopped by the user.

Note that according to the datasheet of the I210 only values between 8 and 70'000'000 and 125'000'000, 250'000'000 or 500'000'000 ns are allowed for the FREQOUT register.

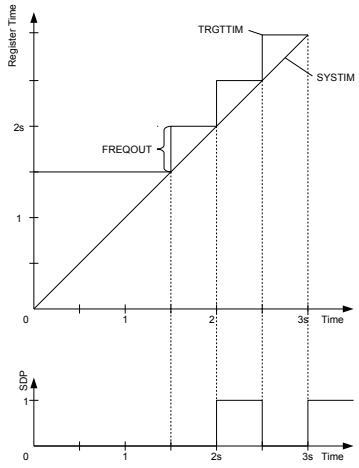


Figure 3.4: Principle of the PPS generation of the I210 Ethernet controller.

However, the IGB driver does not inherently support the generation of PPS signals. Fortunately, software from third party sources[10][11] were available on the Internet. With the use of these two examples, the IGB driver was rewritten (see Listing 3.5), in order to enable the PPS generation on the Ethernet controller I210.

```
// linux-toradex/drivers/net/igb/igb_ptp.c


static int igb_ptp_enable_i210(struct ptp_clock_info *ptp,
            struct ptp_clock_request *rq, int on)
{
  struct igb_adapter *igb = container_of(ptp, struct igb_adapter,
                ptp_caps);
  struct e1000_hw *hw = &igb->hw;
  unsigned long flags;
  struct timespec ts;
  u32 tsauxc, tsim, regval = 0;

  switch (rq->type) {
    ...
  case PTP_CLK_REQ_PPS:
    printk("%s (%d): PPS %s on SDP1\n",__FUNCTION__,__LINE__,on?"enabled":"
    disabled");

    if(on) {
      /* Ensure correct initial state of configuration registers */
      regval = E1000_READ_REG( hw, E1000_TSAUXC );
      regval &= ~TSAUXC_EN_CLK0;
      E1000_WRITE_REG( hw, E1000_TSAUXC, regval );

      regval = E1000_READ_REG( hw, E1000_TSIM );
      regval &= ~TSAUXC_EN_TT0;
      E1000_WRITE_REG( hw, E1000_TSIM, regval );

      /* Set fix half period of 500 ms */
      E1000_WRITE_REG( hw, E1000_FREQOUT0, NSEC_PER_SEC / 2 );
      E1000_WRITE_FLUSH( hw );

      /* Map SDP1 to FREQOUT0 */
      regval = E1000_READ_REG( hw, E1000_TSSDP );
      regval |= TS_SDP1_SEL_FC0 | TS_SDP1_EN;
      E1000_WRITE_REG( hw, E1000_TSSDP, regval );

      /* Set SDP1 to output */
      regval = E1000_READ_REG( hw, E1000_CTRL );
      regval |= ( E1000_CTRL_SDP1_DIR );
      E1000_WRITE_REG( hw, E1000_CTRL, regval );
      E1000_WRITE_FLUSH( hw );

      /* Align the first rising edge to the start of the
       * next second.
       * As soon as SYSTIM reaches the TRGTTIM the first time,
       * the output will be set to 0. Therefore, we start at
       * the next second + 500ms (Half Period).
       */
      ptp->gettime( ptp,&ts );
      ts.tv_nsec = NSEC_PER_SEC / 2;
```

```
52          ts.tv_sec += 1;
53          E1000_WRITE_REG( hw, E1000_TRGTTIML0, ts.tv_nsec );
54          E1000_WRITE_REG( hw, E1000_TRGTTIMH0, ts.tv_sec );
55          E1000_WRITE_FLUSH( hw ;
56
57          /* Enable interrupts:
58           * An interrupt is generated at the beginning of
59           * every second.
60           */
61          regval = E1000_READ_REG( hw, E1000_TSIM );
62          regval |= ( TSINTR_SYS_WRAP );
63          E1000_WRITE_REG( hw, E1000_TSIM, regval );
64          E1000_WRITE_FLUSH( hw );
65
66          /* Enable FREQOUT0 */
67          regval = E1000_READ_REG(hw, E1000_TSAUXC );
68          regval |= ( TSAUXC_EN_CLK0 | TSAUXC_ST0 );
69          E1000_WRITE_REG( hw, E1000_TSAUXC, regval );
70          E1000_WRITE_FLUSH( hw );
71      }
72      else
73      {
74          /* Deactivate the PPS generation */
75          regval = E1000_READ_REG( hw, E1000_TSAUXC );
76          regval &= ~TSAUXC_EN_CLK0;
77          E1000_WRITE_REG( hw, E1000_TSAUXC, regval );
78
79          regval = E1000_READ_REG( hw, E1000_TSIM );
80          regval &= ~TSAUXC_EN_TT0;
81          E1000_WRITE_REG( hw, E1000_TSIM, regval );
82
83          regval = E1000_READ_REG( hw, E1000_TSSDP );
84          regval &= ~TS_SDP1_EN;
85          E1000_WRITE_REG( hw, E1000_TSSDP, regval );
86          E1000_WRITE_FLUSH( hw );
87      }
88      ...
89
90      return 0;
91  }
92
93  return -EOPNOTSUPP;
94 }
```

Listing 3.5: Activation/Deactivation of PPS pulses on the Ethernet controller I210.

3.2.6 PTP Clock Drift Control

The PTP clock of the master and the slave run at a different speed, which is why the drift of the slave clock has to be controlled. Therefore, a PI (Proportional-Integral) controller is implemented, which updates the desired drift value. The control loop function is called after receiving a Follow-up message in Two-step mode or after receiving a Sync message in One-step mode, respectively. The weighting of the P and the I value (denoted in the Listing 3.6) is based on empirical values, developed by InES. If offsets larger than 400 μs are detected, the whole drift control loop will be skipped and the new time directly set by the software.

```
// ptp2_clockitf_apalis_t30.c

PTP2Boolean PTP2_ClockItf_T_loopControl(
    PTP2_Time_T* const out,
    const PTP2_Time_T* offset,
    const PTP2_Time_T* received_timestamp_new,
    const PTP2_Time_T* received_timestamp_old,
    const PTP2_Time_T* sync_receipt_new,
    const PTP2_Time_T* sync_receipt_old )
{
    static long long accu_drift = 0;
    static long long old_drift = 0;
    ...
    // Add new difference to accumulated drift
    accu_drift = accu_drift + diff_ns_per_s;

  // Copy initial values
    I = accu_drift;
    p = diff_ns_per_s;

  // Weighting the values
    p = (p+2)*3/4;
    I = (I+8)*3/16;

  // Calculate the controlled variable
    drift = old_drift + p + I;
    ...
}
```

Listing 3.6: Drift control loop.

3.2.7 PTP Clock Offset Correction

The above-mentioned control loop does also calculate the offset between the master and the slave clock. The *PTP2_ClockItf_T_setOffset()* function (also in *ptp2_clockitf_apalis_t30.c*) then calls the driver to set the new offset. For large offsets of 400 μs or more, the offset is added to the actual time and directly set from the application. Otherwise, the offset is handed to the driver for more accurate, incremental correction. In order to get the highest accuracy, the hardware correction, which is not implemented in the standard IGB driver v5.1.2, was activated (see Offset Adjust Implementation in Section 3.2.8).

3.2.8 Problems during the Implementation

Although a working PTP stack with example implementations and a Linux driver, which supports hardware timestamping and clock adjustments, was available, some problems occurred during the implementation.

SDP Pin not Routed

PPS signals are very important in order to compare the synchronization of two or more PTP devices. In the Ethernet controller I210, there are no dedicated pins for PPS output, but SDP are used for this purpose.
The problem of using a SDP is the fact that none of them are routed on the processor board

and thus not accessible from the outside. However, the Ethernet controller provides a function to internally route a SDP to either the active or the link Light Emitting Diode (LED) of the Ethernet connector.
In the case of this project, the PPS is routed to the active LED. It requires a hardware modification, since there is a low pass filter on the connection between Ethernet controller and LED. Moreover, the LED has to be removed from the connection since it would disturb the measurements. By removing the resistor R5 and the capacity C109 on the evaluation board[12], the connection can be used for measuring purposes.
Unfortunately, this workaround leads to other problems. First, it is obviously impossible to concurrently use the LED for Ethernet signaling purposes and for PPS output. In some applications, this may be a problem. Second, the slew rate, which is the change of the output in volts per seconds, is much smaller in case of the LED pin compared to the SDP since the pin is only supposed to drive a LED. As depicted in Figure 3.5, a level change on the SDP takes one or two nanoseconds whereas on the LED pin it takes around 45 ns. The slew rate of the LED pin has to be increased on the final Printed Circuit Board (PCB) by an appropriate termination of the signal line. Nevertheless, using the LED pin is the only option to get the signal out of the processor board without modifying its hardware. This is why this pin is going be used by ZI.

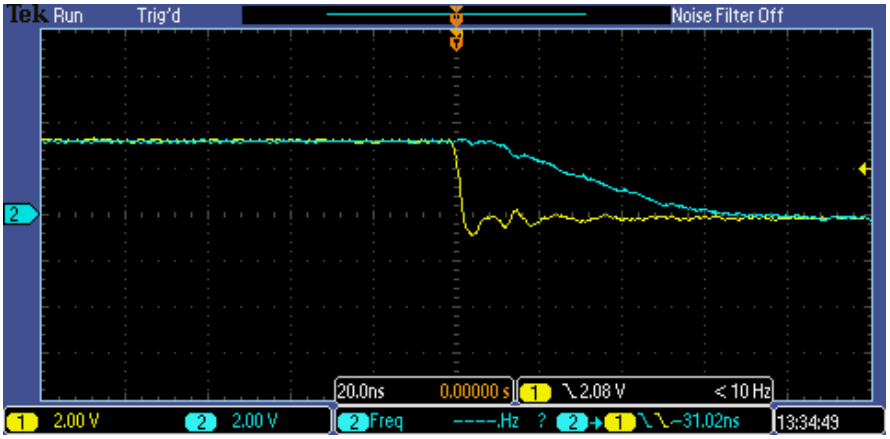


Figure 3.5: Slew rate difference between the LED pin (blue) and the SPD (yellow)

Additionally to the aforementioned problems, another problem concerning the LED pin occurred during the implementation, namely the pulse duration varies from second to second due to an unpredictable offset on the positive edge. While the negative slope occurs precisely with a period of one second, the period of the positive slope is in a range of about 33 ms (see Figure 3.6). The positive slope of the SDP does not exhibit such variable offset behavior.

Using the LED pin has a lot of disadvantages which could be avoided if the SDP pin is routed to the connector of the processor board instead of the LED pin. This would require that the hardware of the Apalis processor board has to be modified for every Etzel produced, which is not acceptable for ZI.

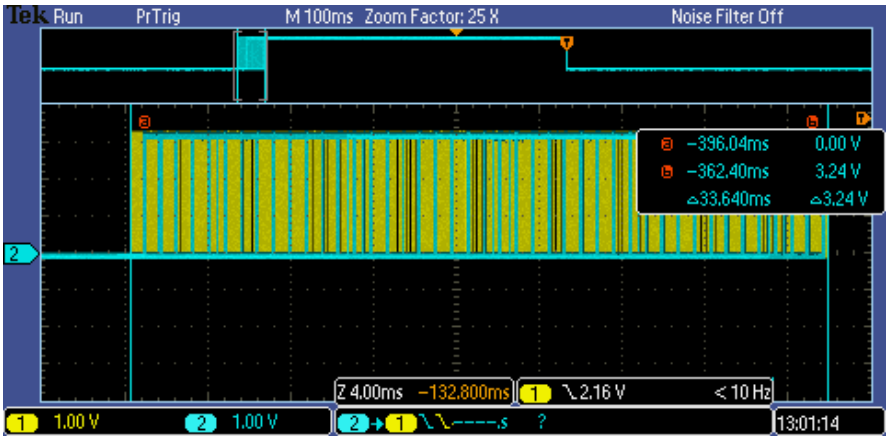


Figure 3.6: Range of the positive slope occurrence on the LED pin with infinite persistence. Blue is the master, yellow the slave PPS.

Offset Adjust Implementation

After getting the PTP stack to work on the evaluation board, the comparison of the PPS signal of the master and the slave revealed a synchronization of approximately ±10 μs, which is much higher than the expectation of a synchronization in the range of nanoseconds. The reason for this inaccuracy was the badly implemented driver function (*igb_ptp_adjtime_i210()*) responsible for correcting the PTP clock offset. Instead of using the dedicated hardware register TIMADJ of the Ethernet controller, the correction in the driver was done in software by reading the current time, adding the offset and writing back the corrected time. This method in unusable in the case of nanosecond offsets since an unpredictable time passes between the reading and the writing, leading to an imprecise offset correction.
In order to get the best synchronization, the driver function was rewritten to use the dedicated TIMADJ register if the offset is smaller than 10 μs (see Listing 3.7). For larger offsets, the previously implemented code is reused since the TIMADJ corrects the offset by adding or subtracting 1 ns at each clock cycle (8 ns) to the PTP time, until the offset is corrected. Therefore, correcting a large offset (i.e. hours or days) would take a lot of time[1] whereas correcting the time by adding the offset to the current time only takes one execution of the time adjustment function.

```
1  // linux-toradex/drivers/net/igb/igb_ptp.c
2
3  ...
4  static int igb_ptp_adjtime_i210(struct ptp_clock_info *ptp, s64 delta)
5  {
6    struct igb_adapter *igb = container_of(ptp, struct igb_adapter,
7                                           ptp_caps);
8    unsigned long flags;
9    struct e1000_hw *hw = &igb->hw;
10   u32 abs_delta = 0;
11   const s64 limit = 10000;    // Set limit to +/- 10 us
12   struct timespec now, then = ns_to_timespec(delta);
13
```

[1]The TIMADJ is capable of correcting a maximum offset of ±125 ms during one second (125'000'000 clock cycles times ±1 ns).

```
14    // If the offset is within the limit, adjust by hardware, else by software
15    if( delta < limit && delta > (-1)*limit )
16    {
17      if( delta != 0 )
18      {
19        if( delta < 0 )
20        {
21          abs_delta = ( u32 ) ( (-1) * delta );
22          abs_delta |= ISGN;
23        }
24        else
25        {
26          abs_delta = ( u32 )delta;
27        }
28
29        spin_lock_irqsave(&igb->tmreg_lock, flags);
30
31        E1000_WRITE_REG(hw, E1000_TIMADJL, abs_delta);
32
33        spin_unlock_irqrestore(&igb->tmreg_lock, flags);
34      }
35    }
36    else
37    {
38      // *** Original Code ***
39      spin_lock_irqsave(&igb->tmreg_lock, flags);
40
41      igb_ptp_read_i210(igb, &now);
42
43      now = timespec_add(now, then);
44
45      igb_ptp_write_i210(igb, (const struct timespec *)&now);
46
47      spin_unlock_irqrestore(&igb->tmreg_lock, flags);
48
49      // *** End of Original Code ***
50    }
51
52    return 0;
53  }
54  ...
```

Listing 3.7: Implementation of the hardware offset correction enable in the IGB driver.

3.2.9 Precision of the PTP Synchronization

Before integrating the PTP stack implementation into Etzel, the synchronization accuracy between two evaluation boards and thus the feasibility to fulfill the synchronization requirement of 10 ns was determined. Measurements showed a standard deviation of 4.18 ns and a mean value of -0.5 ns, measured between the PPS pulses (see *Results U1 - Time Offset (PPS)* in section 7.1.2).

3.3 Integration into Etzel

The integration of the PTP implementation is mainly done by ZI itself since they have the know-how of the Etzel board and its software, i.e. the VHDL code of the FPGA and the software of the data and web server. The work covered by this project is described in this section.

3.3.1 FPGA Interface

The PPS time, which is the PTP time of the next PPS event, has to be delivered to the FPGA at the beginning of every second. This ensures that there will be enough time for the delivery.

The beginning of a second is detected by the dedicated thread *PPS Listener* (see Section 3.2.2), polling the PTP time every 10 ms. Another possible way would be to use the interrupt generated every second by the Ethernet controller. However, the implementation of the handling or even the signaling of interrupts to the user space is an expensive task to implement, which is why the polling method was eventually used.
Whenever the second field of the PTP time increments, the time of the next PPS event is generated by rounding up the actual PTP time and then forwarded to the ZI application, which will write the time to the FPGA. The interface between PTP application and ZI software is depicted in Figure 3.7.

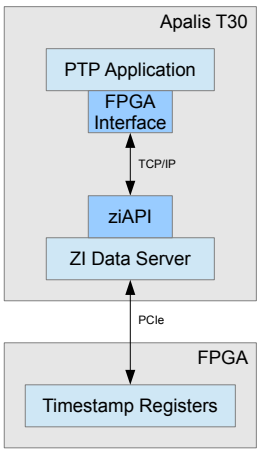


Figure 3.7: Block diagram of the connection between PTP application and FPGA.

The PTP application sends and receives data from the FPGA via the *ZI Data Server*. Both programs run concurrently on the *Apalis T30* processor board and are internally connected by Transmission Control Protocol (TCP)/Internet Protocol (IP).

In order to read and write data from and to the FPGA, the *PTP Application* has to open a connection to the *ZI Data Server*. Therefore, the *FPGA Interface* provides the function

```
PTP2Boolean PTP2_FPGAItf_T_Init( );
```

which initializes and establishes a connection to the data server. Whether the connection is established can be checked by the function

```
PTP2Boolean PTP2_FPGAItf_T_IsConnected( void );
```

If the connection is no longer used anymore, the following function can be called:

```
void PTP2_FPGAItf_T_Disconnect( void );
```

Inside the FPGA, registers to control the behavior of the implementation are implemented and can be externally accessed using the ZI Data Server. The registers are listed in Table A.1 in Appendix A.2. Functions to set and get values from the registers are implemented in the FPGA interface.

3.3.2 PPS Activation

For testing purposes, the PPS signal generated by the Ethernet controller is routed on the PCB to the High Frequency (HF) trigger input of the FPGA. Therefore, the HF trigger input has to be activated on the ZI web interface, in order to be able to synchronize the FPGA to the PTP time of the Ethernet controller.

3.3.3 FPGA PPS Generation

The measurements of the timastamp register deviation of two Etzel devices desired for a PPS signal generated by the FPGA. Thus, the 25^{th} bit of the timestamp register, which toggles with approximately 1 Hz[2], is routed via the HF trigger output to the outside of the Etzel device. In order to be able to use the PPS of the FPGA, the HF trigger output has to be enabled in the web interface of the Etzel device.

3.3.4 ZI Server Integration Problem

ZI requires that timestamps are always increasing. The concept of the FPGA considers this requirement. However, when two Etzel devices are synchronized the first time after start-up, the slave has to set the PTP time to the PTP time of the master. If the master Etzel device was started after the slave, the slave has to turn back the local PTP time in order to get synchronous to the master. In this situation, the requirement of an always increasing timestamp is violated and leads to a crash of the ZI data server. Unfortunately, the ZI data server has to be running because the trigger input of the PPS signal of the Ethernet controller has to be activated. A missing PPS signal means that the timestamp register of the FPGA is never overwritten by the PTP time of the Ethernet controller.

Mitigation Strategies A concept to solve the above-mentioned problem is to force the Etzel with the highest timestamp to be the master of the PTP synchronization. This means that Etzel devices have to check which of the devices has the highest timestamp, before the PTP application is started. The drawback of this concept is the development of the extra software.

Another concept is to enable the trigger input of the PPS signal by default. In this case, the PTP application can be started before the ZI data server. The ZI data server is started when the Etzel devices are synchronous. The advantage of this concept is the relatively small development effort.

[2]The frequency of the least significant bit of the timestamp register is 30 MHz. Thus, the 25^{th} bit toggles with a frequency of $f_{25} = \frac{30Mhz}{2^{25}} = 0.894Hz$, which is approximately a PPS signal.

Chapter 4

Measurement Concept

This chapter provides an insight into the requirements for a decentralized measurement system in respect to critical timing issues. The functionality required by ZI and their customers is specified. 'Test points' of the present implementation are identified in order to relate requirements to the actual performance. In chapter 6 tests are developed based on these fundamentals.

Problem Description

Decentralized measurement systems are based on the idea of using a local clock (pulse generator) and a local time base in each device to perform spatially distributed measurements. This concept bears two main problems: Two physically separate clocks never run at the exact same speed, thus resulting in a drift over time. Even if drift can be compensated perfectly, one device will be started earlier or later than the other, resulting in a constant offset. Both these problems are resolved in the PTP specification[1] by defining one device as master, which all other devices (slaves) synchronize to. Hardware limitations introduce a limit to the achievable time synchronization performance as calculated and simulated in section 2.3.4.

The different components of such a structure are depicted in figure 4.1 and described subsequently.

Unit Marked by a blue dotted line are the basic components that perform the time synchronization over Ethernet and are tested in the first part. Individual development hardware boards[13] are referred to as units, the corresponding tests are called unit tests.

System The complete setup is marked by a red dotted line and includes per-device synchronization of the local PTP time with the time used for timestamping incoming measurement events. Thus defined is a complete measurement system.

4.1 Performance Key Figures

Measurements are split up into two parts as described in section 1.2.2. Firstly the performance of the general PTP time synchronization is evaluated. Secondly the functionality of the final product is verified in order to demonstrate fulfillment of all requirements. Figure 4.1 represents an overview of a generic test setup in its complete form.

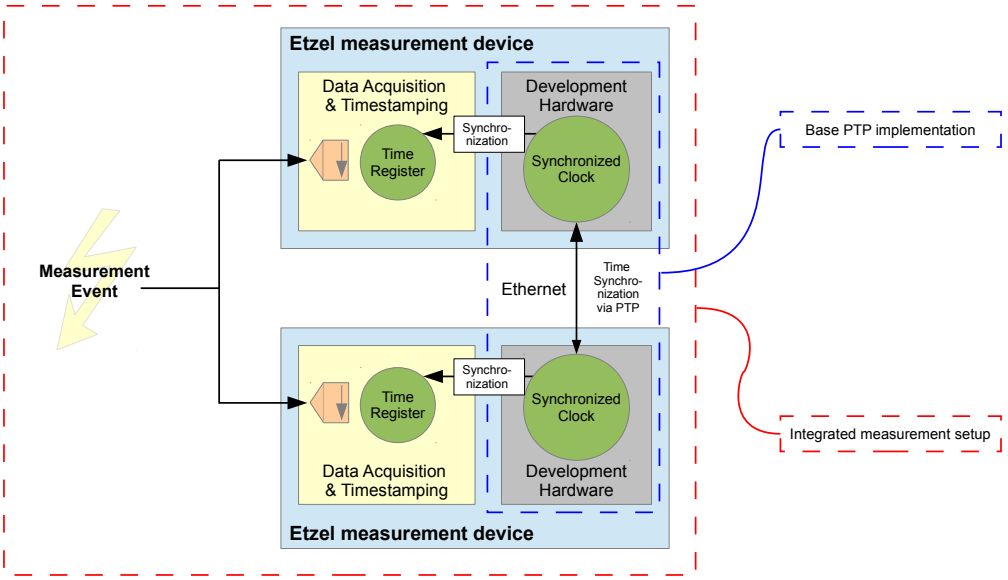


Figure 4.1: Measurement concept: Base implementation and integration into overall setup

The basic principles on which measurements are performed in each of these two parts, are described by the following performance key figures.

4.1.1 Base PTP Implementation (Unit)

The base PTP implementation consists of two or more development hardware boards which are connected via Ethernet and are running a PTP software application on top of a Linux OS.

Time-to-Lock A PTP-enabled device requires a certain amount of time after it has been installed into a PTP network (or switched on) until it can be considered to be synchronized to the other PTP devices.

Time Synchronization Accuracy The core feature of a PTP-enabled device is its ability to synchronize its time register with all other PTP devices. Thus defining synchronization accuracy. The measurable equivalent is the *time offset of a device from the PTP master*.

4.1.2 Integrated Measurement Setup (System)

The integrated measurement setup consists of two or more Etzel measurement devices, which each contain the equivalent of one development hardware board each. Furthermore they perform data acquisition and allow a user to access the data.

Time Offset Between Locally Separated Measurements The most fundamental requirement for a decentralized measurement setup is the time offset between two or more devices. The worst offset defines the synchronization accuracy of the whole system. Due to the nature of the synchronization process, this value is fluctuating and has to be observed continually.

Quality of Time Synchronization The Etzel measurement device, as it is depicted in a simplified way in figure 4.1, contains two individual clocks[1]. Since only the clock of the ethernet controller is synchronized via the precision time protocol, the time that is kept by the timestamping unit has to be updated continually. The quality of this process is of interest, since it is performed individually per device and directly influences the accuracy mentioned in the previous paragraph.

4.2 Requirements

The requirements for each of these indicators have mostly been defined by ZI. Where no target was given, reasonable values were assumed.

Synchronization Accuracy The goal is to achieve a time offset between two modules in the order of 10 ns. This accuracy is only necessary for the synchronization between the individual time bases of the devices. With respect to actual measurements taken by the Etzel instrument it is required to synchronize timestamps of measurement samples better than 1 μs (half the sample rate).[2] Furthermore it is necessary to verify the long-term stability of the synchronization over 24 hours.
It is not necessary to synchronize to an external absolute 'world time' but rather ensure that time differences between involved devices are minimal.

Time-to-Lock Since no specific requirements are given, it is assumed that Time-to-Lock should amount to less than five minutes from a cold boot.

Quality of Time Synchronization It is imperative that timestamps are increased at all time, in order to enforce that no multiple measurements with the same timestamp may exist. Additionally it is beneficial to the synchronization accuracy if consecutive time increments are as similar as possible.[3] Thus preventing sudden spikes in the time scale and enabling smooth and steady behaviour.

4.3 Implementation Testpoints

In order to measure the performance of the PTP implementation, the following methods have been devised. They are used for either Unit or System Tests (as described in chapter 6 'Measurement Plan').

4.3.1 Pulse-Per-Second (for Unit Tests)

The measurement of the time offset between multiple devices has to be done continually and needs to be presented in a way that is easy to compare. The standard method for PTP

[1] Clock is meant as 'time keeping device', not as 'pulse generator'.

[2] Although samples are taken at a sampling rate of 60 MS/s, they are initially downsampled to 469kS/s and then timestamped. This corresponds to a sample time of 2.132 μs, resulting in a range of ± 1 μs.

[3] The internal clock consists of a time register that is updated at each clock cycle. This means that the evolution of the time register is inherently discontinuous. This increment value can be varied in order to compensate for time synchronization mismatches.

devices is by implementing a signal that is triggered every second (aptly called PPS). This signal features a highly precise flank that represents the beginning of a new second. Thus it displays the current time offset and can easily be compared between multiple devices. This method applies to all measurements for the base PTP implementation directly on development hardware and describes the *synchronization accuracy*.

Implementation At the core of each Ethernet controller is a time register which is updated every 8 ns (at each 125 MHz clock cycle) by a variable increment value. If time synchronization via PTP is activated, the increment value of a slave device is adjusted slightly by the PTP application in order to keep the same time as reported by the master. As soon as the nanosecond register exceeds 10^9 nanoseconds, three actions are performed: the nanoseconds are set to zero, the seconds are incremented and a pulse is generated on a General Purpose Input Output (GPIO) pin. This pin can be connected to a measurement device like an oscilloscope. The process is depicted in figure 4.2 and described in detail in section 3.2.5.

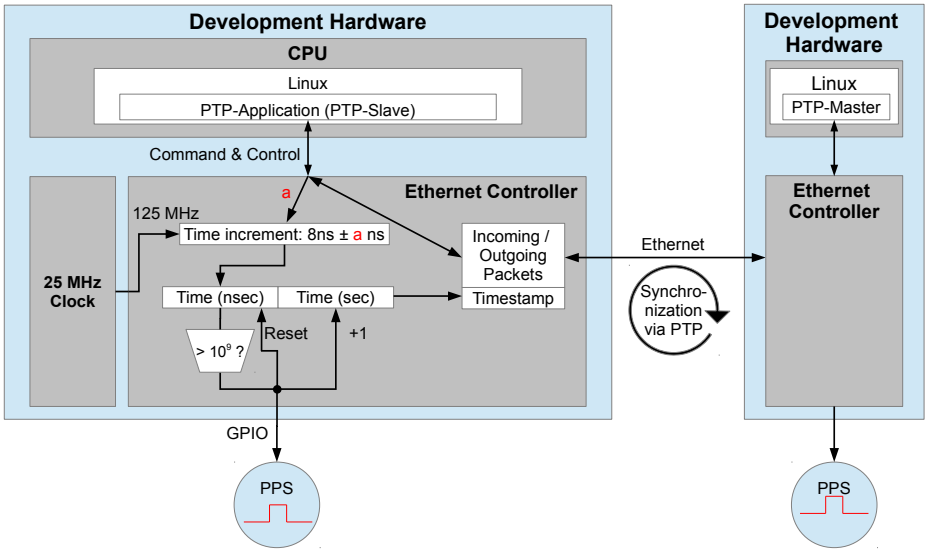


Figure 4.2: PPS signal generation

The PPS signals (generated as described above) exhibit timing behaviour as displayed in figure 4.3.

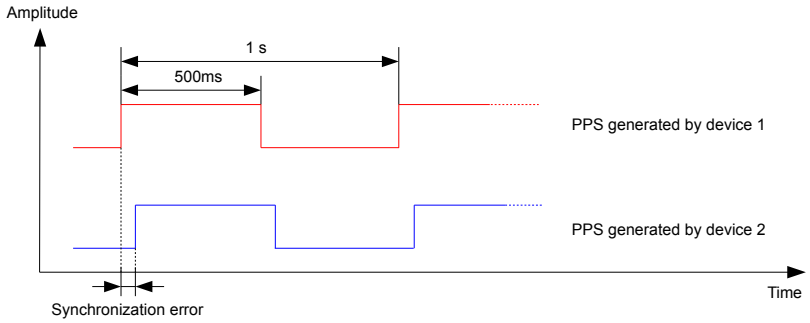


Figure 4.3: PPS time diagram

4.3.2 Pulse-Per-Second (for System Tests)

A similar method to measure the offset between PPS signals (as described in chapter 4.3.1) is used for evaluating the synchronization accuracy of a synchronized FPGA time unit. The implementation provides an additional 'FPGA PPS' signal that indicates the updates of the timestamps that are used for tagging the actual measurements of the Etzel device.

Implementation The 'FPGA PPS' is generated by the FPGA time unit each time the bit representing 1 second of the timestsamp gets set[4]. The exact update rate of this register is handled by a loop controller which calculates a timestamp increment for each clock tick. This value is based on information about the current FPGA time and the time of the Ethernet controller at the next PPS. The Ethernet controller is constantly adjusting its time in order to synchronize to the PTP master. Figure 4.4 depicts the major involved components in a PTP environment with two Etzel devices. The timing of the 'FPGA PPS' signals is the same as described in figure 4.3.
The 'FPGA PPS' signal represents the current rate of the FPGA time and includes uncertainties that were introduced by the additional layer of synchronization from the Ethernet controller to the FPGA. Therefore it is expected that the time synchronization accuracy of the FPGA is slightly worse than the Ethernet controller.

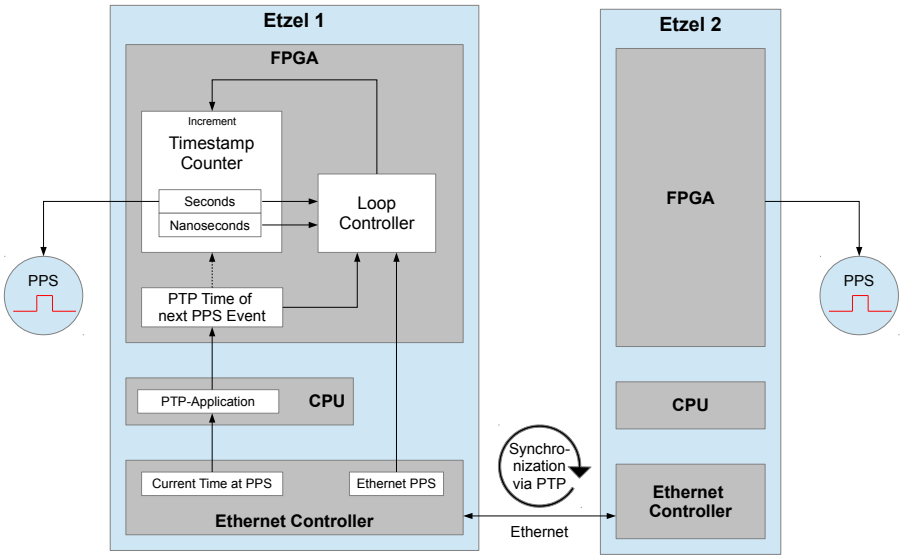


Figure 4.4: FPGA PPS signal generation

4.3.3 Timestamp of Measurement Event (for System Tests)

A different measurement method is required to evaluate the performance of the complete system (i.e. after inclusion of the time synchronization feature into the Etzel device). The data acquisition unit of the device is set to take measurement samples at a rate of 469 kS/s. Each of these samples is associated with a unique timestamp, which are compared in order

[4]Since this can be compared to bit 25 of a counter based on a 60 MHz clock (as described in section 3.3.3) the period is not 1 second but rather about 0.894 s.

to evaluate the system performance. The following methods apply for measurements in the second phase (evaluation of the integrated measurement setup) aimed at the two performance key figures from section 4.1.2.

- *Time offset between locally separated measurements:* This method is targeted at a complete measurement system with multiple Etzel devices as 'devices under test'. A suitable signal is generated and used as input to create a clearly identifiable measurement event. This signal is simultaneously fed into all devices under test, where it is timestamped. The difference between a timestamp from any device to the associated timestamp of the PTP master device defines the time synchronization accuracy. Edges in the input signal are used to find related timestamps.

- *Quality of time synchronization:* This method requires a complete measurement system to function, but targets one single Etzel as device under test. The time difference between two or more consecutive timestamps can be evaluated in order to describe the quality of the time synchronization process. Figure 4.5 illustrates a few possible scenarios[5]. In a) the local device is perfectly synchronized with the master device and therefore performs equal increments throughout the two displayed reference seconds. In situation b) and c) the first local second takes longer than the reference (i.e. due to the local clock being slower). Whereas in b) the additional increment values are spread equally over the time left until the next second, the time difference is initially compensated in c).
  Further undesired behaviour might be sudden large increases (interrupting continuity), the lack of time increments (hurting the 'no multiple timestamps' requirement) or extremely varying time increments.

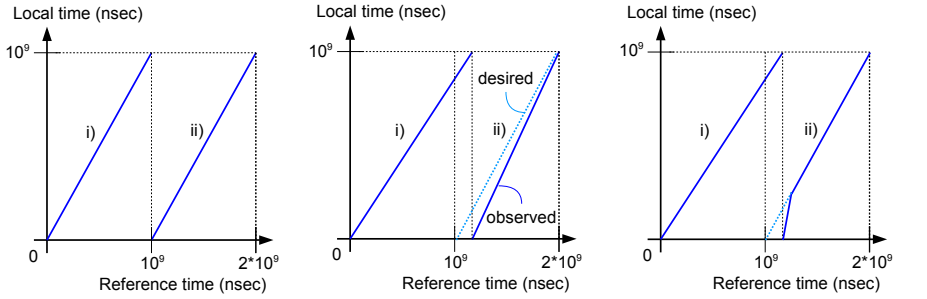


Figure 4.5: Subsecond performance of time synchronization

[5]Deviations from the standard 'synchronized' case are exaggerated in order to clearly show the differences

Chapter 5

Measurement Conditions

This chapter's aim is to devise potential use cases for decentralized measurement setups. All factors that may affect timing performance shall be evaluated. The influential ones will be used to develop individual test scenarios for the *measurement plan* in chapter 6.
A certain amount of inaccuracy is inherent to the system, firstly because the frequency stability of each clock is not perfect, but only finitely accurate (this applies to master and slave devices independently) and secondly because the synchronization process takes time. The most critical influences are the actual network topology, the applied software parameters and any environmental impacts on the behaviour of the hardware clocks.

5.1 Network Topologies

Different use cases for the final product ask for certain network topologies. The simplest method to connect two devices is a direct (P2P) connection. The majority of applications ask for the deployment of more than two measurement devices, thus requiring a method for connecting many Ethernet devices (hereafter referred to as a E2E connection).

End-to-End The interconnection of multiple Ethernet hosts requires at least one central switching point. For this device the following possibilities exist:

- Switch[1]
- Switch (PTP-capable)
- Hub[2] (not supported by the current PTP core application)

When using any kind of multiport ethernet connector, the main problem is the introduction of a variable time delay (jitter). This jitter can depend on the specific implementation of the multiport device or even on the amount of network traffic. Asymmetry between the sending and receiving path may also be introduced. All these factors lead to a reduction of the time synchronization accuracy, since the PTP can not predict and compensate this varying behaviour of a switch. The variable delay introduced is mainly dependent on the forwarding method of the switch. The following methods are ordered according to the potential variable delay they may introduce, best first.

[1] A network bridge operating at the data link layer
[2] A network hub or repeater operating at the physical layer

1. *Fragment free*: This type of switch reads the first 64 bytes of every frame before forwarding it, without performing any kind of error checking. This method should provide the most stable and therefore most predictable behaviour.

2. *Cut through*: Reads frames only up to the hardware address before forwarding them. Because this type may fall back to a 'store and forward' type, it is considered less predictable than variant 1.

3. *Store and forward*: Each frame is buffered and verified before it is forwarded. This method depends on the actual size of the frame which varies on the message type and therefore renders this switch relatively unpredictable.

4. *Adaptive switching*: Dynamically changes between 1. and 2., thus being the least predictable (in a worst case scenario).

Hubs can not be utilized because they operate in half-duplex mode only, which is not supported by the PTP core implementation.

PTP-Capable Switches For most use cases it can be assumed that a potential customer is willing to use network hardware that supports additional PTP features. These 'PTP-capable' devices provide the best possible synchronization accuracy for connecting more than two PTP devices. They can act as either a 'TC' or 'BC'. A transparent clock measures the time that passes between a PTP packet is recognized and forwarded by the switch. This so-called residence time is written into a correction field available in a 'IEEE1588-2008' message and will be used to calculate the correct path delay. A boundary clock on the other hand acts as PTP slave on its incoming port and as PTP master on the outgoing port. Thus creating additional PTP segments and preventing the introduction of variable path delay (see section 2.1.2 for details).

Peer-to-Peer An alternative connection method is to link two devices directly via a single Ethernet cable, thus eliminating the need for a switch or hub. This method is called P2P. The drawbacks include the restriction to only two measurement devices and since only one Ethernet port is available per device, the need for a different method to connect a host computer to the Device Under Test (DUT). For the purposes of these measurements, a host computer was connected via the "Ethernet over USB" mode that is supported by default in the current Linux kernel[3].

5.2 Software Settings

The software implementation of the PTP stack provides several parameters that may influence time synchronization performance. The two most important parameters are:

- *Sync interval*: The rate at which 'Sync' PTP packets are transmitted by the PTP master device. These messages intend to readjuste the clock rate of all PTP slave

[3]The Linux for Tegra (L4T) kernel which is used by default in the Toradex v2.x Board Support Package (BSP) contains the Android Remote Network Driver Interface Specification (RNDIS) Ethernet gadget driver. Since the BSP automatically assigns IP addresses to the host and the target device, this setup allows for a plug and play use.

devices according to the rate of the PTP master. Thus they aim at minimizing clock drift between a local and a reference clock. Since the clock of each network device is subject to local environmental influences, this adjustment has to be executed fairly often (e.g. at least once per second).

- *Delay request interval*: In order to continually evaluate the offset caused by the network path, the 'PTP offset correction method' measures the time taken between sending a packet and receiving a response (the so-called path delay). If the network topology is not subject to change, the rate at which this is executed should be fairly constant (although it may be influenced by switches as described in section 5.1).

Both of these parameters shall be evaluated, but the main focus lies on finding the optimal sync interval with maximized time synchronization accuracy and minimized generation of network traffic.

5.3 Oscillator Stability

Time synchronization accuracy is based on the accuracy of the pulse generator that is used to drive time increments. The stability of oscillators is usually specified by the manufacturer[14] in terms of:

- Short term stability (phase-noise)
- Long term stability (over the duration of many 100'000 of clock cycles or more, including aging but excluding environmentally induced drift)
- Temperature stability (the most prominent example of environmentally induced drift)
- Aging (e.g. stability after the first year)

Phase noise describes the frequency spectrum of the oscillator and can be measured using a spectrum analyzer.[4] It is usually measured as relative power ratio compared to the carrier at the desired clock frequency. Therefore it is specified in -db_c at 10 kHz offset from the carrier. Long-term stability on the other hand describes the difference between the observed clock signal and a reference clock (Δt) that builds up during a certain period of time ($t_{measurement}$). This observation is called drift and is usually specified in units of parts per million (ppm) as represented in equation 5.1 and is subject to change due to aging[15].

$$long\_term\_stability = \frac{\Delta t}{t_{measurement}} * 10^6 \ \ [ppm] \tag{5.1}$$

$$\text{E.g.: impact of drift after 1s @ 25 ppm:} \quad \Delta t = 1s * \frac{25}{1'000'000} = 25\mu s \tag{5.2}$$

[4] An alternative perspective is the time domain, where the short term stability is referred to as jitter. In order to create a meaningful data set, a large amount of clock periods (in the order of 1000) have to be measured and statistically analyzed.

Ethernet Controller Oscillator In regard to the clocks used in the final product, two elements have to be differentiated. Firstly for basic time synchronization via the precision time protocol, the ethernet controller is driven by a clock signal from a relatively inaccurate crystal oscillator at 25 MHz. It is specified at a long term stability of 25 ppm, while no details are known about short-term stability (phase noise). Temperature stability is an additional important attribute that is not documented. Aging properties are specified at ±3 ppm in the first year and are supposed to decrease logarithmically with time[16].
Errors introduced by clock drift (or more precisely the deviation between the exact clock of the PTP master and the PTP slave device) are accounted for by the drift correction method of the PTP stack. Since no value is given for temperature stability, a typical 30 ppm/° is assumed. A wind gust might cause a sudden cooling down of the oscillator in the order of a few degrees. An exemplary calculation for a cooling of 5° results in a drift after 1 second of $\Delta t = 1s * \frac{30}{10^6} * 5° = 150\mu s$.

Etzel Oscillator Secondly, the timestamping unit in the FPGA is driven by a relatively accurate oscillator which is specified according to table B.22 in the appendix at section B.4. In order to synchronize the time registers of two (or more) Etzel devices with the PTP time, the respective clock drift has to be considered. In a worst case scenario, a drift of up to $\Delta t = 1s * 0.5\text{ppm} = 500ns$ can occur on each device between two PPS events (during 1 second). This error should be compensated and the quality of this method has to be tested.

5.4 Environmental Conditions

Various environmental influences (concerning the Ethernet controller and its clock, the processor or the network) may influence the time synchronization accuracy. The device has to operate under the same limited environmental conditions as the Etzel (+5°C to +40°C).
The following influences were identified:

- Temperature differences and changes
- High 'CPU' load
- Network traffic
- Asymmetric paths
- Hardware differences

Temperature Stability

All electrical parts feature characteristics that depend on the ambient temperature to a certain degree. In a scenario where two Etzel devices are located in different rooms, the temperatures may be different as well. Thus causing differing effects depending on the location. The parts that are most affected by temperature are oscillators (as described in detail in section 5.3). Other parts like resistors, capacitances, etc. exhibit certain dependence effects as well. They are optimized to minimize these effects in the observed temperature range of +5°C to +40°C and only influence time synchronization accuracy to a small degree. The highly integrated circuit chips are designed to operate at least at commercial temperatures (0°C to +70°C) and should therefore not exhibit any noticable temperature dependence.

Network Traffic

Any network traffic present on the ethernet link could potentially interfere with the timely delivery of PTP packets and therefore lead to variable delays between sending and receiving PTP packets. Since the Etzel devices continually send their measurement results to a client via ethernet, relatively high peak rates of external traffic have to be considered.
The synchronization of PTP devices is implemented using multicast connections (User Datagram Protocol (UDP) / IPv4) in order to minimize the generated traffic.

Asymmetric Paths

The precision time protocol periodically performs measurements of the delay between two devices. It is then assumed that this so called transit delay is the same for both directions. This is not necessarily true since asymmetric paths might be caused by network cables, switches or even the Physical (layer) (PHY) hardware. The resulting delays may vary on the direction of traffic. Since it is assumed that potential customers mostly use Etzel devices as PTP master and slaves, the exact same PHY hardware will be present in the PTP signal path. Therefore asymmetry will be neglected in further considerations.

Hardware Differences

Because all PTP devices in a network rely on one single master, this particular device should work as precise as possible to allow for high overall system accuracy. The precise time protocol uses a 'best master algorithm' to determine the optimal master / slave configuration. This theoretically allows to use a highly precise atomic or GPS driven clock to act as PTP master. It is assumed that ZI customers would not want to invest in additional hardware, but only use multiple Etzel devices as master and slaves.
The custom PTP software implementation allows to configure values for the clock class[5], clock accuracy[6] and clock variance[7].

[5] In a range of 0 to 255, where lowest is best. 0 shall only temporarily be used to force a device to be master and 255 is used for slave-only devices

[6] Defines a list of ranges, e.g. 25-100 ns

[7] This is a logarithmically scaled statistical value which describes the time offset that may happen over the duration of a sync message interval. More precisely: 2^8 multiplied by the logarithm of the actual PTP standard deviation in seconds

Chapter 6

Measurement Plan

The measurement plan describes the course of action taken to describe the characteristics and measure the performance of the time synchronisation implementation. The goal is to create a list of tests that covers all potential situations that may be encountered in daily use. Thus allowing the verification of the correct behaviour of the device under test. Each test shall be defined with regard to the concept as described in chapter 4, the setup as described in chapter 5 and environmental conditions.

6.1 Schedule

In a first stage, the performance of the software implementation of the PTP-stack on the underlying hardware will be examined. Development hardware was used for these tests. After having successfully integrated the synchronization functionality into ZI's impedance spectroscopes, the second stage of measurements is to verify the performance of a complete decentralized measurement setup on production ready hardware.
Accordingly the tests are split up into:

- Unit tests: The measurements in the first stage, concerning the time synchronisation feature as modular component of the Etzel device (the system).
- System tests: Measurements occurring in the second stage and concerning the performance of the Etzel device in a decentralized measurement setup.

A combination of a letter and a number serves as reference to each measurement ('U' for unit tests and 'S' for system tests)[1]. A complete specification exists for each test, containing the description of the DUT and its setup, the tested device characteristics. Furthermore information about the measured indicator and the measurement device used as well as the duration of the test are defined.
The following sections offer only an overview of the tests, whereas the complete test descriptions can be found in the appendix.

[1] According to the V-Model, which represents a framework for development and validation of soft- and hardware.

6.1.1 Unit Tests and Performance Measurement

In order to verify the correct operation of the core implementation of PTP, a series of tests has been devised. The goal is to emulate situations of high stress in order to find the constraints under which the device still works as intended. Thus verifying that the DUT will function correctly under working conditions.
For all these tests, two development boards are connected via ethernet and the PTP-stack application is executed to enable time synchronization. Depending on the individual test, different device characteristics are assessed. These characteristics consist of synchronization accuracy and time to synchronization lock.
The common indicator that can be measured and by which the device characteristics are determined, is the time difference of the PPS signal between (at least) two devices. This value is statistically evaluated in order to portray the dynamic behaviour of the synchronization process.

Test Setup An example setup for two development boards is shown in figure 6.1. When developing and adjusting parameters, it is very convenient to have the PPS output on a oscilloscope for getting a quick impression of the system behaviour. In order to produce significant results, it is necessary to measure the statistical distribution of the time offset.

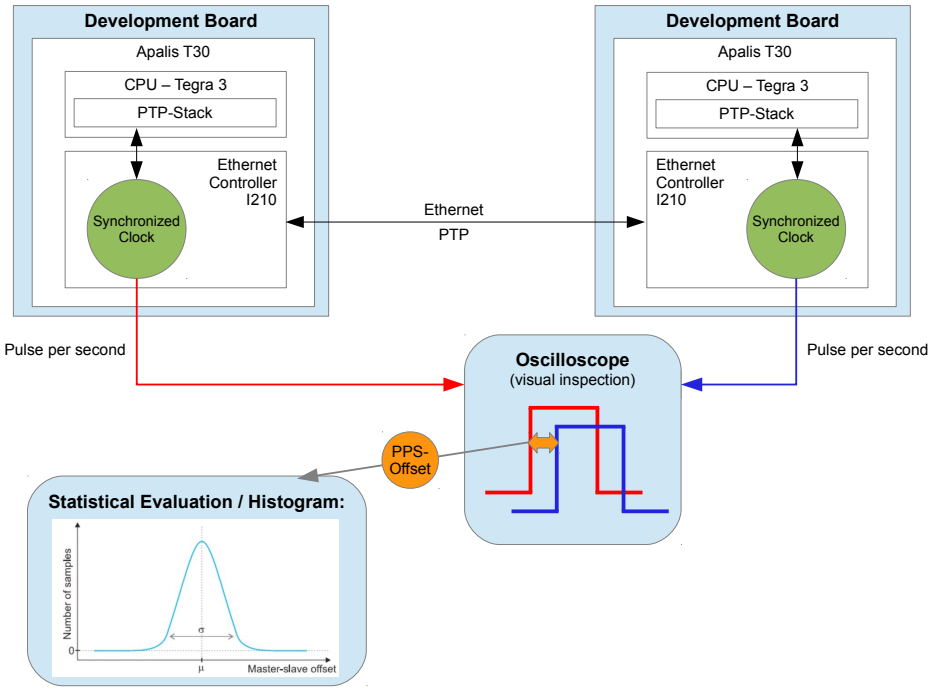


Figure 6.1: Generic time offset (PPS) measurement setup

Measurement Plan for Unit Tests The following table 6.1 contains an overview of the scheduled tests. Complete test specifications can be found in the appendix under B.1.

Reference	Description
U0	Time-to-Lock
U1	Time offset (PPS)
U2	Time offset (PPS) long-term
U3	Bandwidth for synchronization packets vs. accuracy
U4	Network load vs. accuracy
U5	CPU load vs. accuracy
U6	Temperature vs. accuracy
U7	Cable length vs. accuracy
U8	More than two devices (with and without a PTP-capable switch)

Table 6.1: Short measurement descriptions for unit tests of development boards

6.1.2 System Tests Etzel

In order to establish a decentralized measurement setup, all connected measurement devices have to operate on a common time base. Although the time bases of the Ethernet controllers of all devices are synchronized, the data acquisition unit of each device runs on an individual time. Therefore it is necessary to continually readjust these time registers to the respective PTP master clock. The implementation details concerning integration of the PTP synchronization functionality into the measurement devices of ZI (by the name of Etzel) can be found in section 2.3 'Integration into Etzel'.

Since Etzel uses the same hardware as the previously presented PTP implementation (i.e. the same CPU, Media Access Control (layer) (MAC)/PHY controller and communication via Peripheral Component Interconnect Express (PCIe) bus), it is assumed that the same limitations apply as they were examined in chapter 6.1.1. Therefore considerations concerning behaviour under heavy computational load (CPU), high network traffic or environmental conditions still apply.

The goal of the following measurements is to verify the performance of a model decentralized measurement setup with multiple interconnected Etzel devices. According to the V-Model, these tests can be categorized as system testing.

Test Setup The main use of Etzel devices is to perform impedance spectroscopy[17], thus measuring amplitude and phase of multiple input frequencies. In order to generate unambiguous output events at each Etzel, a sine wave is fed into the input signal port of each device simultaneously. It is then periodically enabled and disabled, which results in a recorded event on each toggle. These events are associated with the current time by means of a timestamp. By comparing the event timestamps between all involved Etzels, the deviation from the common time base can be shown for each device. The setup of this verification test is shown in figure 6.2.

Potential Problems A fundamental difficulty that arises when synchronizing two physically separate devices is the granularity of synchronization steps. A transfer of the complete time register is expensive in terms of communication cost and can therefore not be executed

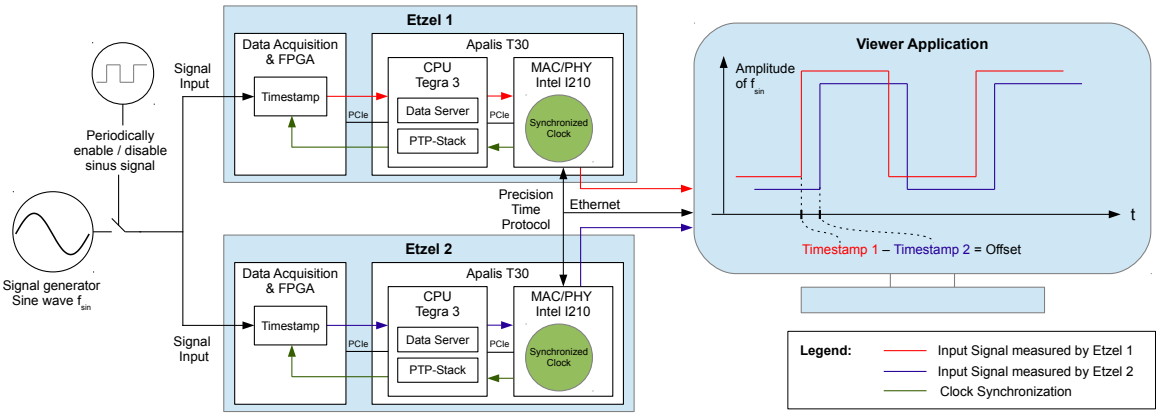


Figure 6.2: Final product verification setup

at every time update (i.e. every clock cycle). Instead the applied method is to rely on highly accurate periodic pulses that are generated every second and indicate the validity of a previously transmitted timestamp. In-between these accurate seconds, the task is to adjust the granular time update steps as evenly as possible. The desired output of such a time updating mechanism is a steadily increasing time, where the increment of each time update is optimally the same. Furthermore it is essential that the time shall never be decreased, therefore ensuring that no multiple samples can be measured at the same time.

In order to verify the quality of the synchronized time update mechanism, it is important to generate measurement events that occur in very short intervals. Additionally testing should occur over a long period of time to ensure that no inconsistencies occur.

Measurement Plan for System Tests

The following tables describe the tests that have been employed to measure the performance of the final Etzel hard- and software. The complete descriptions can be found in the appendix under B.2.

Reference	Description
S0	Etzel time synchronization
S1	Verification of time increments
S2	Synchronized measurement events
S3	Long-term consistency of time increments
S4	Long-term evaluation of synchronized measurement events

Table 6.2: Short measurement descriptions for system tests of Etzel devices

Chapter 7

Results

The results of the measurements are listed and interpreted in this chapter. They have been performed according to the *measurement plan* in chapter 6.

Most tests consisted of multiple measurements of one setup with two or more DUT, which were analyzed with respect to certain synchronization accuracy parameters. These parameters were measured during 30 minutes up to 24 hours while subjecting the DUT to specific configurations and environmental conditions. The exact test specifications and details to the specific setup can be found in the appendix under B.1 (unit tests) and B.2 (system tests). The used measurement devices and their configuration are described in detail in the appendix under B.3. The results of all measurements are listed below and an interpretation with respect to our expectation is given. Certain less significant secondary measurements have been omitted in this chapter but can be found in the appendix next to the respective test specification.

7.1 Unit Tests

Most unit test measurements are based on comparing the time offset of a PTP slave device from the PTP master and therefore only generate one measurement event per second[1]. Therefore the sample size mentioned in U1 to U8 corresponds to the time in seconds that each measurement has taken. In order to provide a visual aid for the reader, result plots contain a dashed line that marks the 0 ns synchronization accuracy (thus perfect sync. between master and slave). Certain figures also include a dotted line for ±5 ns and a dash-dot line for ±10 ns.

7.1.1 U0 - Time-to-Lock

In order to give an overview of the behaviour of a PTP device upon (re)boot, a single scenario from power-on to 'synchronization lock' is described first. In a second stage, a test series over 100 reboots was executed to statistically evaluate the time needed to achieve synchronization lock. This series resulted i) in an evaluation of the time to synchronization lock and ii) in a representation of the accuracy of the initial PPS pulses.

[1] This interval originates from the PPS signal, which is generated once per second and acts as common time denominator

Overview The behaviour of the time offset from power-on until synchronization lock is described in figure 7.1. After about 30 seconds, the operating system has initialized the Ethernet driver and started the PTP application (which furthermore continually executes methods to verify and regulate the synchronization to the PTP master). The PTP activity is indicated by the appearance of the PPS signal in figure 7.1a. In the initial unsynchronized state, the time offset from master amounts to ca. 220 ms.[2] Further important events are the drift compensation at second 46 (red circle), which improves the accuracy to about 620 ns. The offset compensation at second 53 (green circle), which leads to an accuracy of less than ±10 ns.

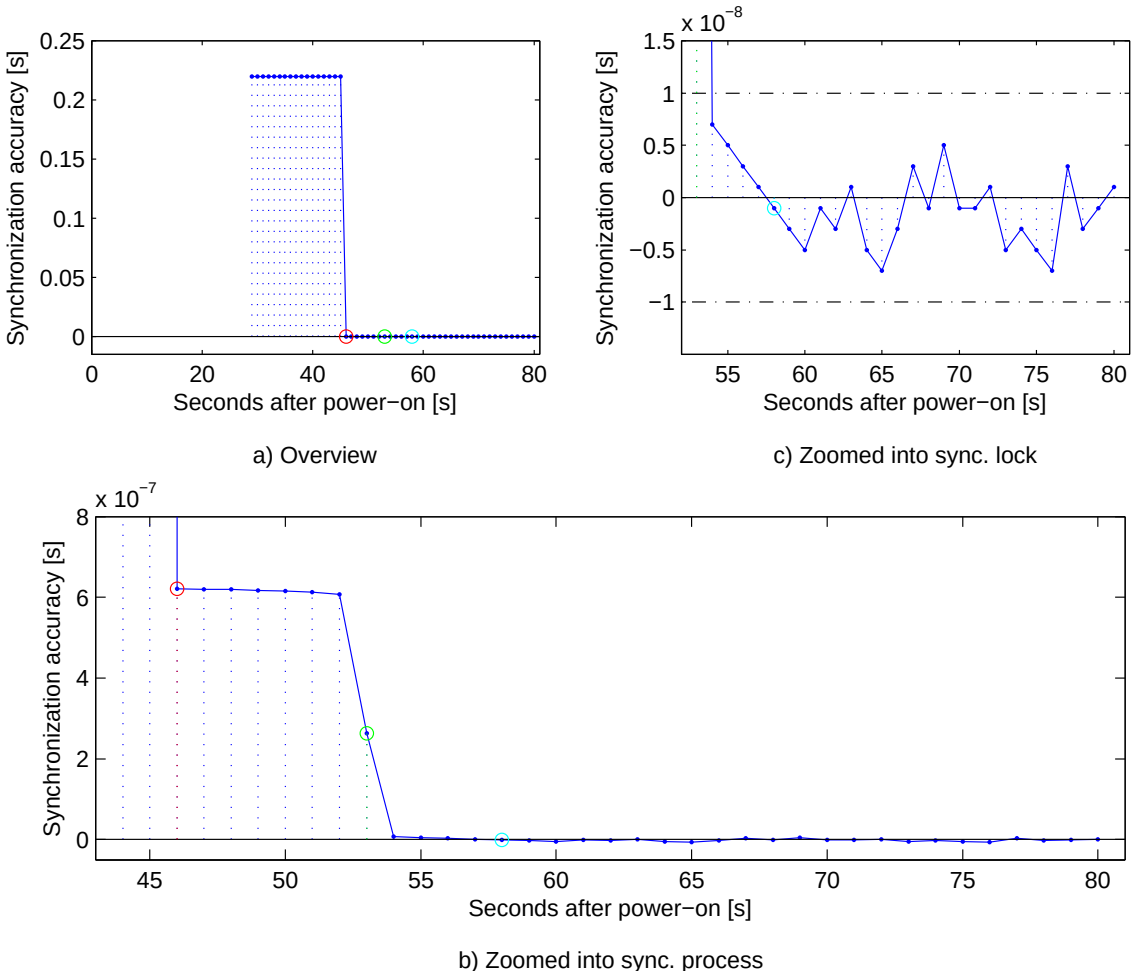


Figure 7.1: Initial time offset measurement

Time to Synchronization Lock Synchronization lock has been defined as the first occurence of five subsequent measurements which are more accurate than ±10 ns.[3] The synchro-

[2]This value was present in the measurement at hand, but might be any time between 0 and 1 second. It represents the time offset between the driver initialization of the two compared DUT modulo 1 second. Since the devices are powered on without exact synchronization, this value is random.

[3]This is exemplified in figure 7.1 by the cyan circle at second 58.

nization lock time of the system was measured during 100 reboots. The observed statistical distribution of the synchronization lock time is given in table 7.1 and depicted in figure 7.2.

Time-to-Lock [s]				**Sample size**
Mean	**Std. deviation**	**Min**	**Max**	
60.56	4.243	52	75	100

Table 7.1: Measurement results - Time-to-Lock

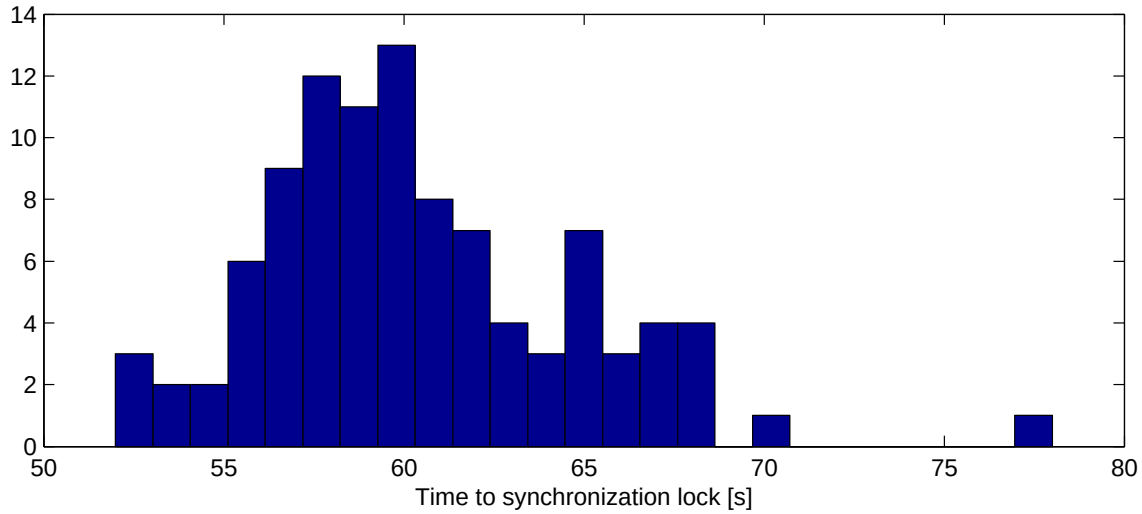


Figure 7.2: Results histogram - Time-to-Lock

Interpretation

The Time-to-Lock is mainly determined by the Linux booting process and the time it takes until the first rough offset correction takes place. The most amount of time is spent during booting (30 seconds) which we assume to be unchangeable. The second most amount of time passes while waiting for the first 'DelayReq' packet[4], which provides offset correction. This interval could be modified to allow quicker lock in times but would result in additional network traffic.

7.1.2 U1 - Time Offset (PPS)

The following measurements (U1 to U8) all describe the settled state of the system (i.e. at least 2 minutes after power-up, thus synchronization lock has been achieved).
In test U1 the synchronization accuracy is measured over two hours in a configuration with two PTP devices connected via P2P. The DUT were placed in a room without any particular temperature stabilization.

[4]see also section 5.2 and the measurement 'U3 - Bandwidth for Synchronization Packets vs. Accuracy' at section 7.1.4

Synchronization accuracy [ns]				Sample size
Mean	**Std. deviation**	**Min**	**Max**	
-0.499	4.181	-13.527	13.527	7214

Table 7.2: Measurement results - Time offset (PPS)

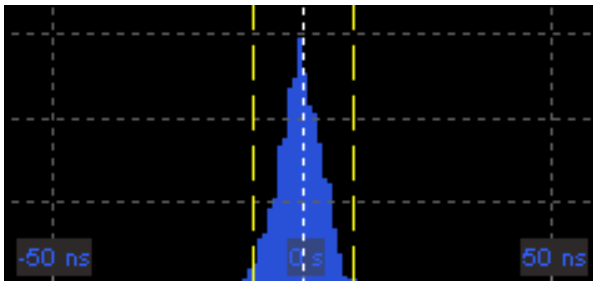


Figure 7.3: Results histogram - Distribution PPS pulses over 2 hours

Interpretation

The measurement results as described in table 7.2 and depicted in figure 7.3 appear to be normally distributed around a mean value close to 0 ns. The yellow markers represent ±10ns. In order to confirm that the measured data are truly normally distributed, the probability for a synchronization error of 13.527 ns under the assumption of a normal distribution with a mean value and standard deviation according to table 7.2 calculates to $\Phi(X > 13.527) = 3.43{*}10^{-4}$. The probability of the maximum occurring in the actual measurement is $\frac{1}{SampleSize} = 1.39^{-4}$. The observed probability differs from the calculated probability by a factor of 2.47. This large error of magnitude is presumably due to the low resolution of bins in the histogram values and the calculation not significant.

7.1.3 U2 - Time Offset (PPS) Long-Term

The synchronization accuracy has been measured over a period of 24 hours in order to verify that there are no outliers in long-term operation. The measurement conditions were equal to U1, with the exception of the duration.

Synchronization accuracy [ns]				Sample size
Mean	**Std. deviation**	**Min**	**Max**	
1.396	3.688	-11.523	14.529	86749

Table 7.3: Measurement results - Time offset (PPS) Long-Term

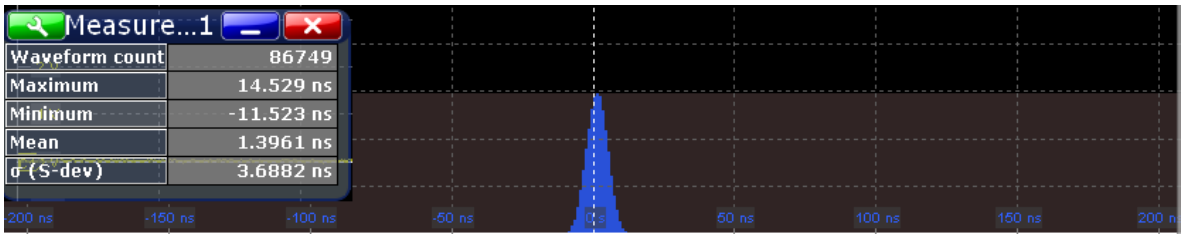


Figure 7.4: Results histogram - Distribution PPS pulses over 24 hours

Interpretation

The measurement results as described in table 7.3 and depicted in figure 7.4 appear to be normally distributed. Compared to the results of the two hour measurement in U1, the mean value differs by 1.9 ns, whereas the standard deviation varies only by about 10 percent. Most importantly the maximum values are almost exactly the same and below an absolute value of 15 ns. No outlier was detected in the 24 hours measurement period.

7.1.4 U3 - Bandwidth for Synchronization Packets vs. Accuracy

The IEEE1588-2008 standard[1] allows for a variety of different data rates for synchronization packets (so called 'Sync' and 'DelayReq' packets as described in section 5.2). These rates have an influence on the synchronization accuracy since they define the frequency with which time errors can be corrected.
The PTP application allows the adjustment of the different data rates over a wide range, but it has been discovered that only a limited range yields a measurable difference in traffic. Traffic measurements are listed right next to the software settings in table 7.4.

Software settings		**Meas. traffic**		**Synchronization accuracy**				**Sample size**
Sync [pkt/s]	**DelayReq [pkt/s]**	**Mean[5] [pkt/s]**	**Mean [B/s]**	**Mean [ns]**	**Std. dev. [ns]**	**Min [ns]**	**Max [ns]**	
1/2	1/16	0.5	120	-28.335	16.787	-78.657	82.665	1877
1	1/16	2.5	190	-10.258	8.202	-34.569	17.535	1818
2	1/16	4.5	330	-3.362	5.090	-18.537	11.523	1828
4	1/16	10	800	0.835	3.665	-8.517	12.525	2110
8	1/16	20	1500	2.493	3.254	-6.513	11.523	1809
32	1/16	20	1500	2.780	3.412	-5.511	13.527	2110
64	1/16	20	1500	2.373	3.275	-8.517	11.523	1874
128	1/16	20	1500	2.895	3.566	-5.511	11.523	1820
256	1/16	20	1500	3.011	3.488	-6.513	11.523	1834
1024	1/16	21	1510	5.072	3.192	-3.507	12.525	1814
1024	1/2	21	1550	4.854	3.617	-5.511	13.527	1867
1024	1	22	1600	4.896	3.525	-4.509	15.531	2061
1024	1024	31	2200	4.3426	3.377	-7.515	13.527	1933

Table 7.4: Measurement results - Bandwidth for synchronization packets vs. accuracy

The top 7 data sets from table 7.4 (marked in yellow) are plotted in figure 7.5 since these settings appear to influence the synchronization error the most. They display the achievable synchronization accuracy when varying the rate of Sync packets, while maintaining a fixed rate of DelayReq packets at $\frac{1}{16}$ packets/s.

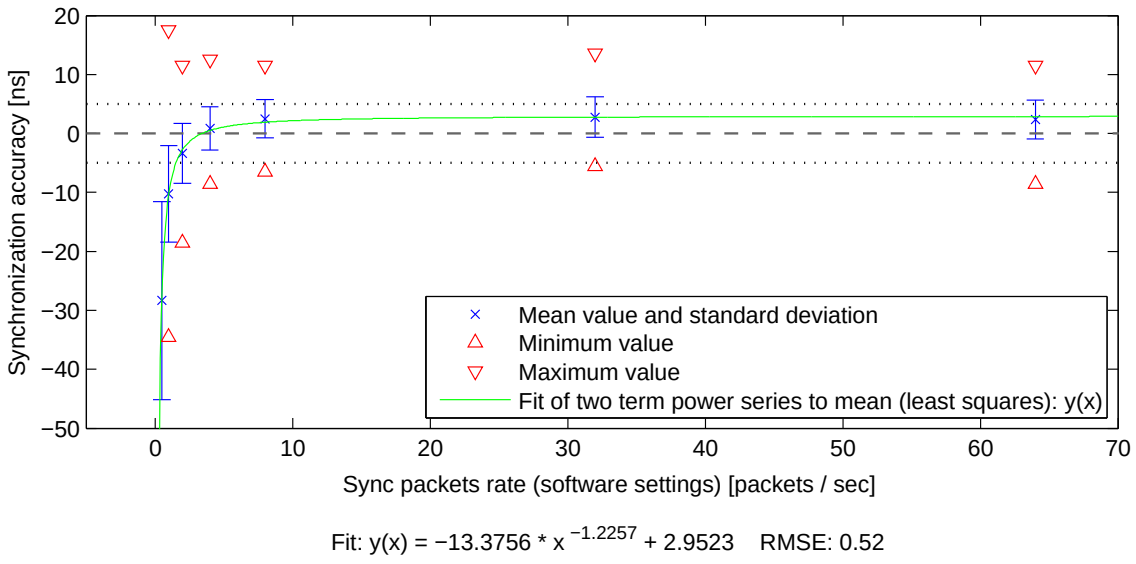


Figure 7.5: Results plot - Bandwidth for synchronization packets vs. accuracy

The method used for PTP message transport was 'PTP over UDP over IEEE802.3 Ethernet'. For comparison the packet sizes of the PTP packets are given in table 7.5.

Packet type	**Size [B]**
Announce	106
Sync	86
Follow-Up	86
Delay Request	86
Delay Response	96

Table 7.5: PTP packet sizes

Interpretation During standard operation (after lock-in and without any changes in the network topology) the synchronization accuracy appears to mostly depend on the rate of Sync packets. The optimum synchronization accuracy was found to be achieved when using a rate of 4 Sync packets per second and was then set as default for the following measurements. The rate of DelayReq packets does not noticeably influence the accuracy as expected and is thus set to 1/16 packets per second which is sufficient to react to changes in the network topology

[5]Due to the synchronization method being set to the 'two-step' mechanism, a Follow-up packet is automatically sent shortly after each Sync packet. Every DelayReq packet is answered by a DelayResp. Announce packets are sent at a fixed rate of 1 packet per second and are used to coordinate the synchronization hierarchy.

at a comparable rate as the lock-in time at boot. The bandwidth occupied in this setting is as low as 800 B/s.
It has been discovered that the maximum rate of synchronization packets seems to be limited at 8 packets per second. This might be due to a limitation of the PTP stack application which restricts the maximum amount of events that can be handled per second.

7.1.5 U4 - Network Load vs. Accuracy

The influence of externally produced traffic, which takes place on the same network as the PTP communication, has been measured. The results are listed in table 7.6 and are plotted in figure 7.6. For comparison the maximum achievable data rate has been measured between 35 and 39 MiB/s (in a 'P2P' configuration, depending on the device).

Network traffic [MiB/s]	**Synchronization accuracy [ns]**				**Sample size**
	Mean	**Std. deviation**	**Min**	**Max**	
0	-0.246	3.555	-9.519	8.517	1955
1	1.108	3.420	-8.517	12.525	1909
5	1.5536	3.816	-11.523	13.527	1920
10	1.795	4.446	-36.573	13.527	1851
15	0.522	7.474	-98.697	29.559	1820
20	1.297	8.946	-73.647	46.593	2555
25	-1.5689	16.343	-128.76	132.77	2208
30	2.6218	31.259	-215.93	240.98	1932

Table 7.6: Measurement results - Network load vs. accuracy

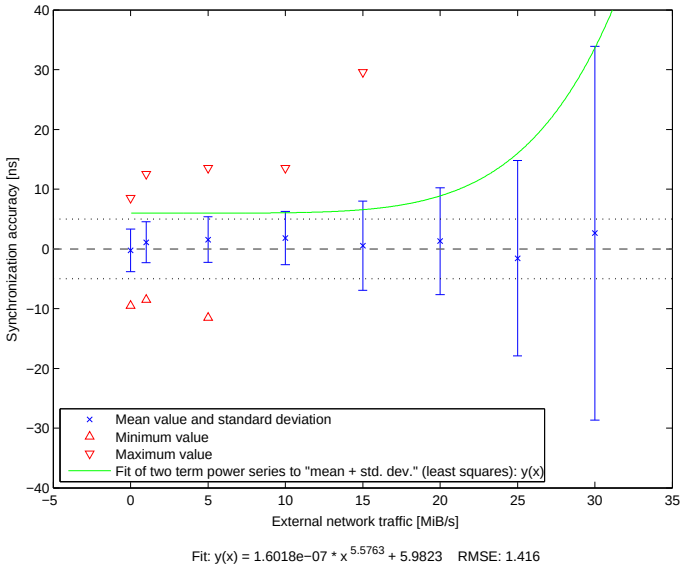


Figure 7.6: Results plot - Network load vs. accuracy

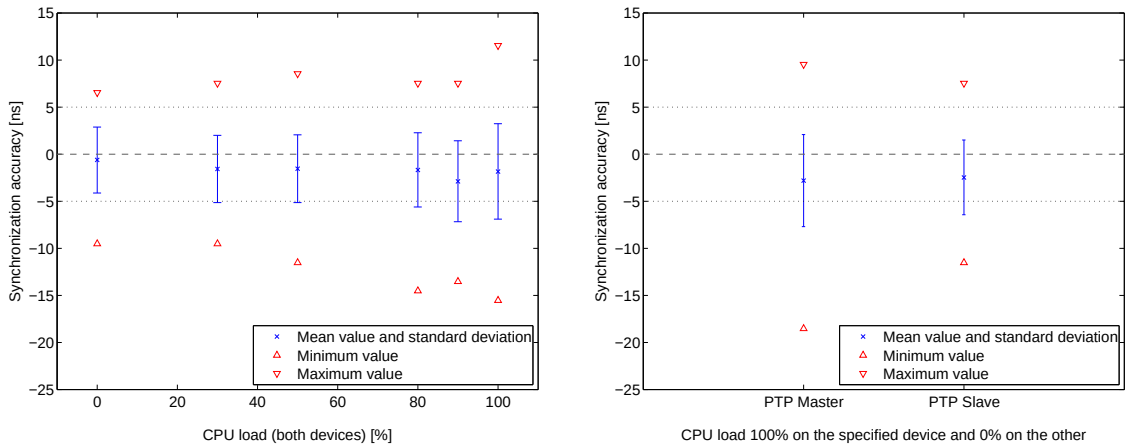
Interpretation Network traffic below 10 MiB/s barely shows any influence on the synchronization accuracy, whereas the standard deviation doubles from 20 to 25 MiB/s and again from 25 to 30 MiB/s of traffic. A fit to the power of 5.58 has been calculated to the 'mean + standard deviation' and resulted in a Root Mean Square Error (RMSE) of 1.416.
In conclusion the influence of external network traffic on the synchronization accuracy is assumed to be negligible up to 10 MiB/s, since the mean plus standard deviation does not exceed 5 ns.

7.1.6 U5 - CPU Load vs. Accuracy

The influence of limited CPU resources due to the processor being utilized by ZI applications has been simulated. The results are listed in table 7.7. The plots for both DUT under stress are depicted in figure 7.7a and for only one DUT being stressed in figure 7.7b.

CPU load [%]		Synchronization accuracy [ns]				Sample size
PTP slave	PTP master	Mean	Std. deviation	Min	Max	
0	0	-0.622	3.505	-9.519	6.513	959
30	30	-1.569	3.569	-9.519	7.515	1112
50	50	-1.560	3.603	-11.523	8.517	1502
80	80	-1.681	3.936	-14.529	7.515	1009
90	90	-2.886	4.307	-13.527	7.515	1017
100	100	-1.842	5.071	-15.531	11.523	1217
0	100	-2.807	4.893	-18.537	9.519	1133
100	0	-2.473	3.959	-11.523	7.515	1025

Table 7.7: Measurement results - CPU load vs. accuracy



(a) Both devices under stress

(b) Single device under stress

Figure 7.7: Results plot - CPU load vs. accuracy

Interpretation The worst performance was measured when subjecting the PTP master to 100% CPU load, resulting in a deterioration of the synchronization accuracy mean of -2.2

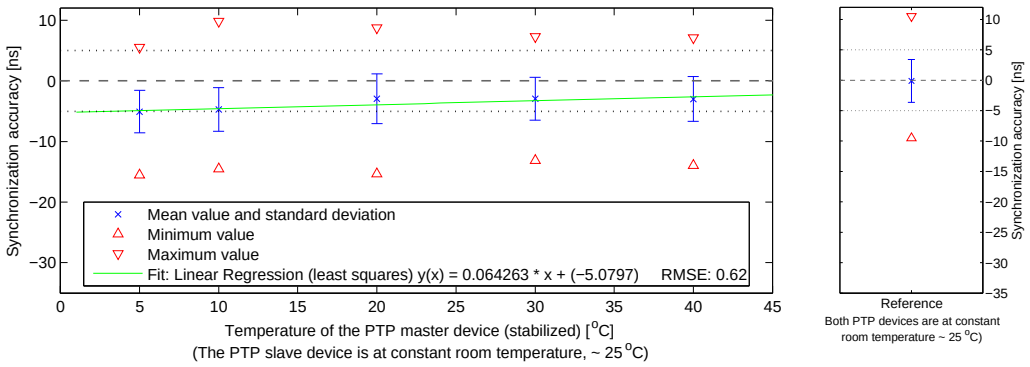
ns and the standard deviation of 1.4 ns. A slightly better performance was achieved when stressing only the PTP slave device, or both devices simultaneously. The application of 30% to 80% CPU load on both devices leads to a unchanged standard deviation and slightly elevated mean synchronization accuracy by about 1 ns.
Since it is assumed that generally 20% or more CPU ressources are available, and the CPU overhead generated by the PTP application is less than 1%, the influence of the CPU load is deemed negligible.

7.1.7 U6 - Temperature vs. Accuracy

Changes in the surrounding temperature may affect the synchronization accuracy of the DUT noticeably. Tests have been conducted for i) a single DUT under stabilized temperature conditions and ii) both devices stabilized. The reference measurement at the top of table 7.8 and the following five measurements, where the PTP master device was kept at a stable temperature, are depicted in figure 7.8. Due to additional noise caused by the climatic chamber, the setup of the oscilloscope had to be altered for this test. Details to the configuration can be found in the appendix under B.3.

Temperature		**Synchronization accuracy [ns]**				**Sample size**
Master [°C]	**Slave [°C]**	**Mean**	**Std. deviation**	**Min**	**Max**	
25* (Reference)		-0.117	3.522	-9.519	10.521	2169
5 (stabilized)	24*	-5.059	3.532	-15.531	5.511	2920
10 (stabilized)	24*	-4.720	3.586	-14.482	9.799	3345
20 (stabilized)	22*	-2.947	4.109	-15.331	8.717	2088
30 (stabilized)	24*	-2.946	3.514	-13.126	7.315	2205
40 (stabilized)	24*	-2.979	3.704	-13.928	7.114	1810
5 (stabilized)	5 (stabilized)	-5.206	3.636	-14.529	4.509	2501
25 (stabilized)	25 (stabilized)	-4.314	3.388	-13.527	3.507	1815
40 (stabilized)	40 (stabilized)	-4.477	3.523	-13.527	4.509	1919

Table 7.8: Measurement results - Temperature vs. accuracy. Temperatures marked with * indicate approximate room temperatures without the influence of a climatic chamber.



(a) Temperature of the PTP master device stabilized only (b) Reference

Figure 7.8: Results plot - Temperature vs. accuracy

A linear regression was applied to the measured mean synchronization accuracy in order to determine the temperature coefficient of the regulated system. It was calculated using the least squares method and resulted in $0.064\frac{ns}{^\circ C} = \frac{0.064\frac{ns}{^\circ C}}{1s} * 10^6 = 64 * 10^{-6}\frac{ppm}{^\circ C}$.

Interpretation Subjecting both of the DUT to static ambient temperatures did not have a noticeable effect on the standard deviation of the synchronization accuracy. When considering the measurements with only the master device stabilized, it can be observed that the mean value differs by -3 to -5 ns to the reference depending on the temperature setting. This observation is reflected in a positive temperature coefficient of the system with respect to the synchronization accuracy.
In contrast the measurements with both devices stabilized feature a consistently low mean synchronization accuracy of -4.3 to -5.2 ns. Since the clocks on both devices are now subjected to the same conditions, it would be expected that both exhibit the same drift behaviour and thus the accuracy mean would be around ±0.2 ns. Since this is not the case, it is assumed that other characteristics of the measurement setup have a noticeable effect on the mean value of the synchronization accuracy. Possible causes include variations in the temperature due to the control loop of the climatic chamber and electronic interference generated by the climatic chamber when switching relays.
In summary, the observed deviations of the synchronization accuracy for fixed temperatures are below 5 ns and thus relatively small.

Temperature Ramp An additional measurement was performed to test the effect of temperature changes over time on the synchronization accuracy. The master device was subjected to a temperature ramp from 5 to 40 °C over 396 seconds, while the slave was kept at room temperature. The observed temperature rise rate amounts to $5.3\frac{^\circ C}{min}$, which is close to the specified[6] $4.5\frac{^\circ C}{min}$. The results described in table 7.9 show a minimal deviation from the reference synchronization accuracy.

[6]As described in the specifications of the climatic chamber in the appendix B.3.2

Temperature		Synchronization accuracy [ns]				Sample size
Master [°C]	Slave [°C]	Mean	Std. deviation	Min	Max	
25* (Reference)		-0.117	3.522	-9.519	10.521	2169
5 to 40 ramp	24*	-1.840	3.448	-11.523	6.5133	396

Table 7.9: Measurement results - Variable Temperature vs. accuracy.

7.1.8 U7 - Cable Length vs. Accuracy

A critical element on the signal path between PTP master and slave is the Ethernet cable. The influence of the utilization of different types and lengths of cables has been evaluated in table 7.10 and figure 7.9.

Cable type	Synchronization accuracy [ns]				Sample size
	Mean	Std. deviation	Min	Max	
1 m Cat5e	2.721	4.166	-10.521	14.529	4631
1.5 m Cat5 ref[7]	-0.246	3.555	-9.519	8.517	1955
1.5 m Cat5	-2.056	3.561	-11.523	7.515	1926
10 m Cat5	-0.376	4.142	-14.529	10.521	2304
40 m Cat6	-3.1125	4.418	-28.557	24.549	1865

Table 7.10: Measurement results - Cable length & type vs. accuracy

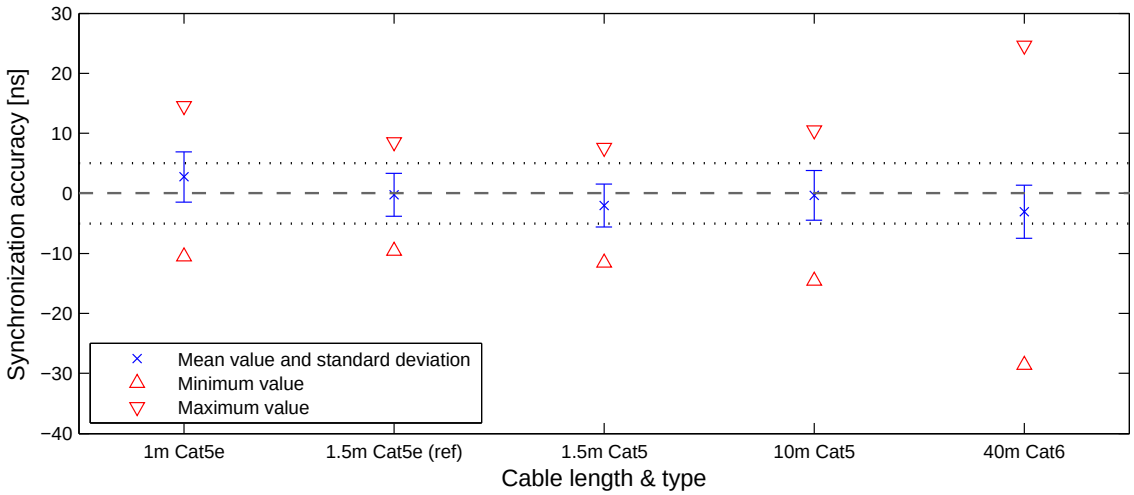


Figure 7.9: Results plot - Cable length & type vs. accuracy

Interpretation Observed are variations of the synchronization accuracy mean of -2.9 to +3.0 ns and the standard deviation of 0 to 0.9 ns compared to a reference measurement. No clear trend is recognizable in respect to cable length or cable category.

[7]The cable described as '1.5m Cat5 ref' was used for all other unit tests.

7.1.9 U8 - More than Two Devices (with and without a PTP-Capable Switch)

In order to identify the influence of a non-PTP-capable switch on a setup with multiple DUT, initial reference measurements were performed with multiple devices connected to a non-PTP-capable switch. The setup was then altered by replacing the default Ethernet switch[8] by a PTP-capable Hirschmann RSP35 switch[9]. Since the RSP35 switch supports both BC and TC modes for structuring the PTP network (see section 5.1 for details), both options were used as individual testcases. The results are listed in table 7.11 and plotted for each testcase in figure 7.10. The network setup in all measurements can be categorized as E2E with one single switching point.

Switch	**Network Topology**	**Synchronization accuracy [ns]**				**Sample size**
		Mean	**Std. deviation**	**Min**	**Max**	
a) Zyxel GS-108B *No PTP*	2 Apalis + PC[10]	-28.717	78.552	-289.58	235.47	1824
	2 Apalis	-28.181	70.586	-245.49	229.46	1989
	3 Apalis	-40.757	72.807	-269.54	227.45	1806
	4 Apalis	-42.356	70.373	-307.62	193.39	1874
b) Hirschmann RSP35 *PTP: TC*	2 Apalis + PC[11]	-5.801	28.212	-79.158	89.178	2528
	2 Apalis	-1.595	25.12	-69.138	69.138	2046
	3 Apalis	-9.965	32.089	-101.2	77.154	2412
	4 Apalis	-16.663	27.706	-167.33	69.138	1951
c) Hirschmann RSP35 *PTP: BC*	2 Apalis + PC[12]	1.7982	42.124	-127.25	159.32	1870
	2 Apalis	15.939	30.601	-65.13	105.21	3361
	3 Apalis	21.347	34.206	-91.182	121.24	1810
	4 Apalis	18.686	32.623	-81.162	119.24	2016

Table 7.11: Measurement results - More than two devices (with and without a PTP-capable switch)

[8] Zyxel GS-108B. Specifications are listed in the appendix B.4.3. Transmission method: Store-and-Forward
[9] Hirschmann RSP35. Specifications are listed in the appendix B.4.4
[10] Actively capturing with wireshark
[11] Connected to the Web-GUI of the switch
[12] Connected to the Web-GUI of the switch

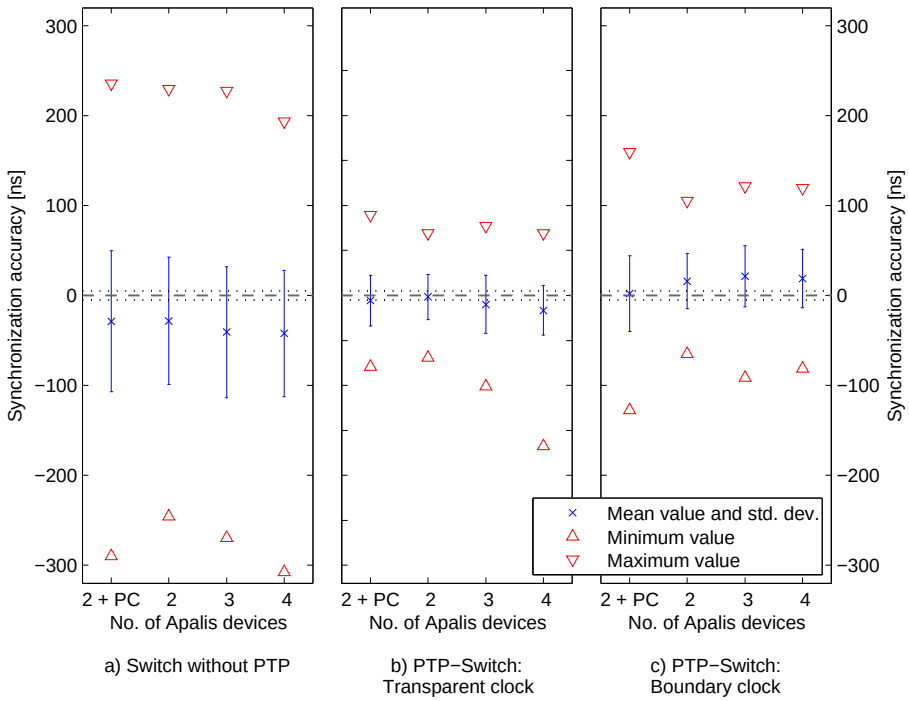


Figure 7.10: Results plot - More than two devices (with and without a PTP-capable switch)

Interpretation Connecting a PC for control and evaluation purposes did not significantly deteriorate the synchronization accuracy, as can be seen when comparing the measurements of '2 Apalis + PC' against '2 Apalis' for each of the network configurations.
Using a PTP-switch over a generic Ethernet switch improves the synchronization accuracy by approximately a factor 2 in respect to mean value and standard deviation. The choice of PTP-mode (TC or BC) does not noticeably influence synchronization accuracy, although TC mode appears to perform slightly better. The introduction of 3 or 4 Apalis devices seems to not significantly influence the performance.

7.2 System Tests

In order to validate the performance of the time synchronization on the final hardware and verify compliance with specific requirements from ZI (e.g. monotonically increasing timestamps), additional indicators were introduced. In addition to the PPS signal from the Ethernet controller, a PPS signal is available from the synchronized FPGA time unit. Moreover, the ZI data server software allows access to the recorded data including timestamps for measurements.
Tests have been performed to verify the quality of the time synchronization process and its influence on the accuracy of synchronous measurements. All tests were conducted under nominal environmental conditions. The gathered data is listed in tables and partially plotted. In order to provide a visual aid for the reader, result plots contain a dashed line that marks the 0 ns synchronization accuracy and a dotted line for $\pm T_S$[13].

7.2.1 S0 - Etzel Time Synchronization

In order to evaluate the overall time synchronization performance of a distributed Etzel setup, the first step was to measure the characteristics of the timestamps on a single device. Furthermore the PTP synchronization accuracy between two Etzels was measured in a similar fashion to the PPS measurements that were used in most unit tests.

Sub-Second Time Update Performance

The quality of the progression of timestamps on a single Etzel device was evaluated in a setup with two devices connected and synchronized via Ethernet. The data generated by a single device includes one timestamp per sampling period. The difference between two consecutive timestamps is referred to as 'time increment'. The evaluation of the time increments is given in table 7.12. A single measurement run consists of two rows in the table, describing the two involved Etzel devices. In all cases 'dev107' acted as PTP master and 'dev77' as PTP slave.

[13] T_S is the sampling period of the Etzel ADC and amounts to $T_S = \frac{1}{469kS} = 2.132\mu s$

Network configuration	Etzel device[14]	Time increments [ns]				Sample size
		Mean	Std. dev.	Min	Max	
P2P	dev107 (M)	8533.5	1.7	8500.0	8550.0	2262448
	dev77 (S)	8533.5	1.6	8500.0	8550.0	2262448
P2P	dev107 (M)	8533.5	1.7	8500.0	8550.0	1810490
	dev77 (S)	8533.5	1.6	8500.0	8550.0	1810490
P2P	dev107 (M)	8533.5	1.7	8516.7	8550.0	1623528
	dev77 (S)	8533.5	1.6	8516.7	8566.7	1623528
P2P	dev107 (M)	8533.5	1.7	8516.7	8550.0	2185665
	dev77 (S)	8533.5	1.6	8516.7	8566.7	2185665
P2P	dev107 (M)	8533.5	1.7	8516.7	8550.0	970473
	dev77 (S)	8533.5	1.6	8516.7	8550.0	970473
E2E PTP	dev107 (M)	8533.5	1.8	8500.0	8550.0	3935390
	dev77 (S)	8533.5	1.6	8433.3	8633.3	3935390
E2E	dev107 (M)	8533.5	1.7	8516.7	8550.0	2522846
	dev77 (S)	9043.0	38405.0	8250.0	27222000.0	2522846
E2E	dev107 (M)	8533.5	1.7	8516.7	8550.0	3187771
	dev77 (S)	8865.9	29004.0	8133.3	13705000.0	3187771
E2E	dev107 (M)	8533.5	1.7	8516.7	8550.0	1250585
	dev77 (S)	10381.0	75450.0	8383.3	16973000.0	1250585
E2E	dev107 (M)	8533.5	1.7	8516.7	8550.0	745292
	dev77 (S)	10032.0	66608.0	8483.3	9199100.0	745292
E2E	dev107 (M)	8533.5	1.7	8516.7	8550.0	1443623
	dev77 (S)	8961.8	30715.0	8216.7	7654500.0	1443623

Table 7.12: Measurement results - Sub-second time increments

The behaviour of the time increments of one specific measurement in a 'E2E PTP' environment with a PTP switch is depicted in figure 7.11. The chosen measurement is marked in orange in table 7.12. The two plots on the top describe the Etzel master (dev107) and the two plots on the bottom the slave (dev77). On the left the progress of the time increments is depicted in the time domain (time increments vs. timestamps). On the right the distribution of time increment values is given. The y-axis is logarithmically scaled. Similar plots of a 'P2P' and 'E2E PTP' measurement (marked in yellow in table B.5) can be found in the appendix B.2.1.

[14]M = PTP master, S = PTP slave.

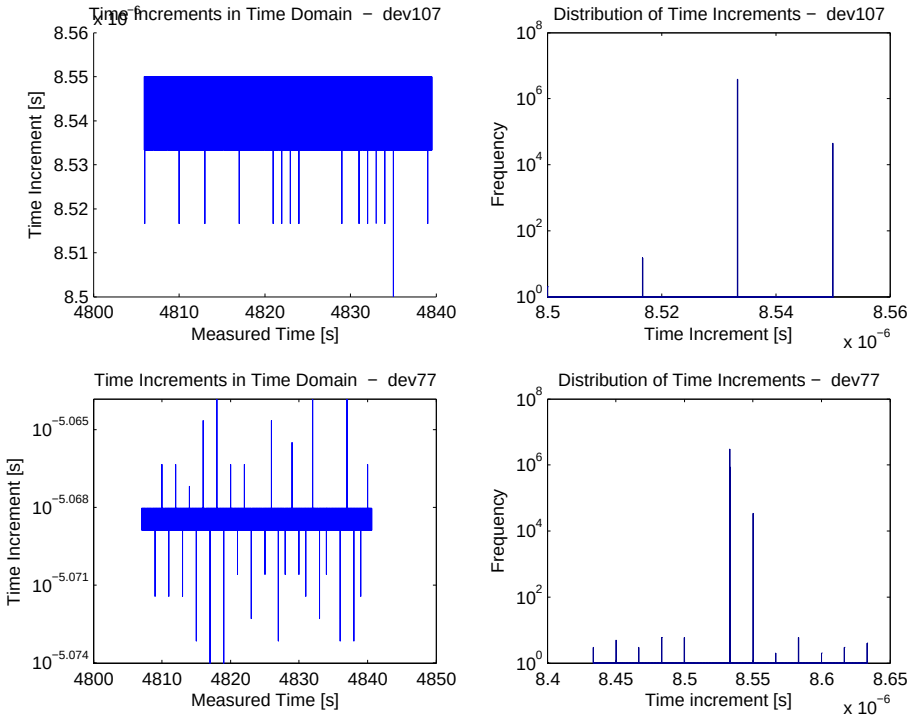


Figure 7.11: Results plot - Sub-second time increments for a chosen E2E PTP measurement (top: PTP master, bottom: PTP slave)

Interpretation The results depicted in figure 7.11 show the influence of the time synchronization method on the time increments, as the time increments are not constant. The PTP master (top plots) shows the occurrence of three discrete time increment values since the FPGA has to synchronize to the Ethernet controller. The PTP slave (bottom plots) on the other hand shows a higher degree of variability of discrete increment values. The reason for this behaviour is the additional noise to the time synchronization, that stems from the communications channel.
The following considerations regard the complete set of measurements with respect to time increments as given in table 7.12. The measurements in the **P2P network configuration** show the best possible performance with very consistent time increments of 8.533 μs[15]. The fact that the standard deviation of the time increments is smaller for the PTP slave than for the PTP master might seem wrong at first glance. Since the PTP master shows a standard deviation of 1.7 ns, this is a base uncertainty for the time synchronization which only gets worse when synchronizing to the PTP slave due to the varying nature of the communications channel. To reduce the standard deviation, the slave performs moving average filtering on the time error and thus is able to suppress high frequency noise. This observation is also confirmed by simulations (see chapter 2.3.4).

[15]The time increment period of 8.533 μs for the measurement data is generated due to limited bandwidth of the internal data server on an Etzel device. The internal rate of $\frac{60MHz}{128} = 469kSps$ is decimated by 4 and thus results in 117 kHz and a period of 8.533 μs. Ultimately the period of 8.533 μs can only be observed in the output data, but in order to consider possible future devices without the bandwidth restriction, the original sampling rate of 2.132 μs has to be taken into account.

The **'E2E' configuration using a PTP switch** exhibits a standard deviation of 1.6 to 1.8 ns which is as expected. The performance of time increments is comparable to the 'P2P' configuration, whereas the extreme values are slightly worse ($< 1\%$).
When using a generic **'E2E' (Non-PTP) switch**, the master device exhibits very similar behaviour as described above, but the slave Etzel performs much worse. The large maximum time increments stem from missing fragments of measurement data from the slave device. This behaviour was only observed when using a Non-PTP switch and is described in the appendix B.2.3.

Synchronization Accuracy

In order to expand the evaluation of time synchronization accuracy from one Etzel device to two synchronized devices, the PPS pulse generated by the FPGA was measured. These measurements were performed during the previous 'sub-second time update performance' tests and the results are listed in table 7.13. They are furthermore depicted in figure 7.12 for the cases 'P2P' and 'E2E PTP', which are the most important for potential use cases. The measurement of PPS signals is independent from the data server output and provides a means of evaluating the synchronization accuracy of the time stamp unit in the FPGAs.
In unit tests 'U1' to 'U8', the synchronization accuracy was measured by comparing the 'Ethernet PPS' signal from the Ethernet controller of the PTP slave to that of the PTP master. A similar PPS signal is generated by the timestamp unit in the FPGA (as defined in section 4.3); it is referred to as 'FPGA PPS'.
The two measurements 'FPGA PPS' and Ethernet PPS' were performed under comparable conditions but not concurrently. Therefore they do not represent exactly the same situation.

Network Configuration	**Input PPS**	**Synchronization accuracy (PPS) [ns]**				**Sample size**
		Mean	**Std. dev.**	**Min**	**Max**	
P2P	Ethernet Ctl.	-5.703	6.049	-21.343	9.719	393
	FPGA	-1.690	12.104	–42.685	55.11	3751
E2E PTP	Ethernet Ctl.	38.989	33.136	-259.52	331.66	4836
	FPGA	-3.669	39.179	-213.93	198.9	3138

Table 7.13: Measurement results - Synchronization accuracy depending on network configuration

Interpretation In a P2P network configuration, the 'Ethernet PPS' shows a time synchronization behaviour as expected from unit test 'U1 - Time Offset (PPS)' with two Apalis boards in the same configuration. The only difference is the higher mean value in the measurement at hand, which might be due to the relatively low sample size of 393 samples. Comparing the 'FPGA PPS' in the same configuration, the standard deviation is worse by an additional 6 ns.
In a 'E2E PTP' configuration the measurement of the 'Ethernet Controller PPS' shows a mean value that is unexpectedly far-off zero. From the results of 'U8' we would expect a mean value of less than 2 ns, instead we observe 39 ns. It is assumed that this divergence is due to a fault in the measurement setup. Asymmetric paths may also be contributors to

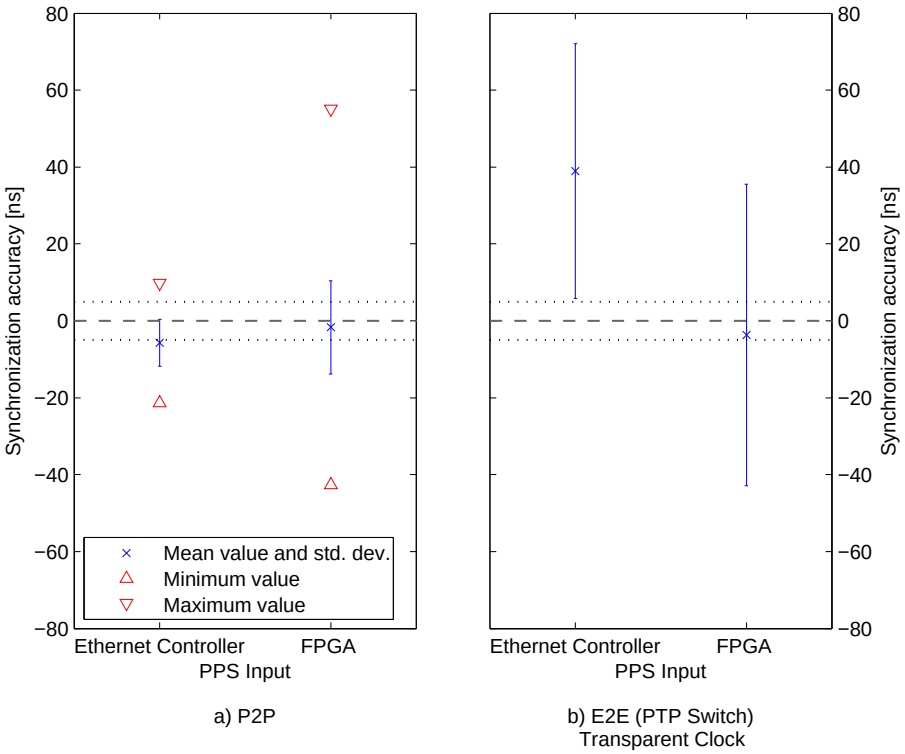


Figure 7.12: Results plot - Synchronization accuracy depending on network configuration

shifted mean values. When neglecting the mean value, the difference in standard deviation between 'FPGA PPS' and 'Ethernet Controller PPS' is again about 6 ns.

7.2.2 S1 - Verification of Time Increments

The verification of the time increments in respect to ZIs requirements is based on the previous results in table 7.12. Due to the fact that the minimum time increment is positive in all measurements, the requirement for monotonically increasing timestamps is fulfilled. Therefore the uniqueness of the timestamp for each sample on one device is given as well. It has to be noted that these measurements only describe the locked-in state of the measurement setup. Therefore possible negative increments during startup are not considered here.

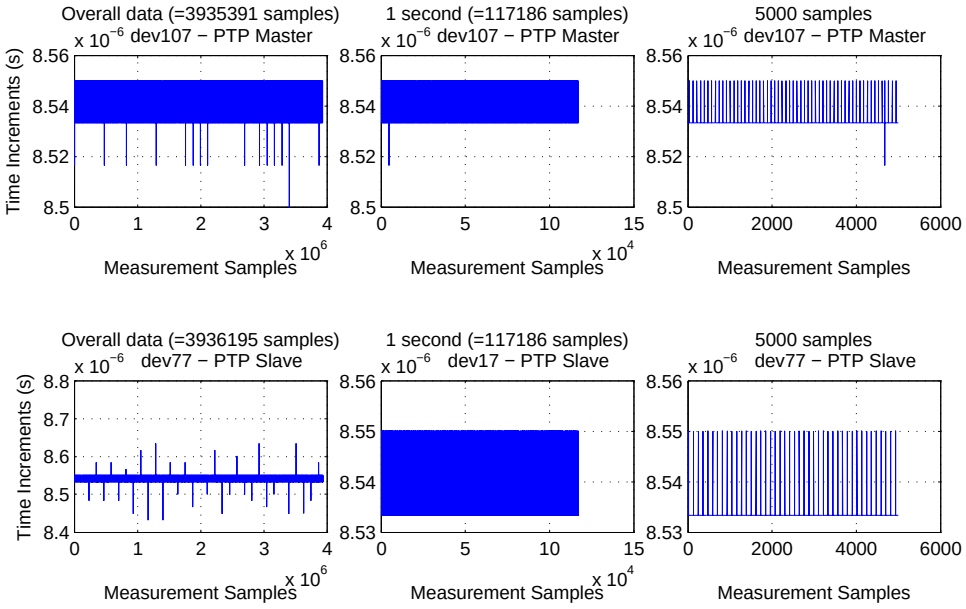


Figure 7.13: Results plot - Quality of time increments in a chosen 'E2E PTP' measurement. Top: PTP Master (dev107). Bottom: PTP Slave (dev77)

A visual inspection of time increment quality[16] for a 'E2E PTP' measurement was performed based on figure 7.13. The top row represents the time increments observed on the PTP master device, whereas the bottom row corresponds to the PTP slave device. The 'overview plot' on the very left shows the occurrence of the discrete time increment values and the effect of the loop controller. In the middle the time increment behaviour during 1 second is depicted. Most of the time the increments toggle between two discrete values. This is also represented in the figure to the very right during the first 5000 samples. The difference between the possible time increment values is 16.67 ns which corresponds to the smallest possible difference of increments between two samples. It stems from the clock period of the Etzel device of $\frac{1}{60MHz} = 16.67ns$.
These observations satisfy the demand for 'evenly distributed' time increments and confirm the expected function of the loop controller according to the implementation.

7.2.3 S2 - Synchronized Measurement Events

The core feature of the synchronization of Etzel devices is the ability to perform synchronous measurements. Measurement events are generated periodically and cause a rising or falling edge in the measurement data over approximately 5 timestamps. In order to discern such events as clearly as possible, the recorded data is subjected to signal processing, as described in the appendix B.2.3. The difference between the timestamps generated by two Etzel devices, measuring the same event, is referred to as 'measurement event error'. The same measurements as in section 7.2.1 (S0 - Etzel time synchronization) were evaluated and the results are listed in table 7.14. The measurement event error of these measurements is depicted in figure 7.14 for the three network configurations.

[16] A representation of the same data is also given in figure 7.11 including a histogram.

Network config.	Measurement event error [ns]				Sample size[17]	No. of edges	Duration [s]	Error $> T_S$[18]
	Mean	Std. dev.	Min	Max				
P2P	111.1	625.6	-1449.2	1934.4	10122212	723	19.307	0
P2P	124.5	667.3	-1919.0	1978.0	8100146	582	15.450	0
P2P	59.0	658.5	-1683.2	1848.0	7263673	521	13.854	0
P2P	96.7	631.4	-1634.3	1859.0	9778685	699	18.651	0
P2P	92.8	632.5	-1391.3	1709.5	4341899	308	8.282	0
E2E PTP	92.7	693.6	-1810.6	2035.7	4026593	291	7.680	0
E2E PTP	92.8	654.3	-2300.8	2078.4	4026597	289	7.680	1
E2E PTP	99.3	1349.8	-2605.2	2519.3	1003188	72	1.913	0
E2E	143.0	109326	-936913	958514	11961097	851	22.814	150
E2E	-4131.4	85691	-976670	675905	14817591	1058	28.252	185
E2E	-9828.8	181502	-907245	908574	6806554	460	12.983	135
E2E	6000.8	149280	-888975	877568	3919821	273	7.477	64
E2E	754.4	77882	-885172	872504	6782986	487	12.938	76

Table 7.14: Measurement results - Synchronization error between two Etzel devices in different network configurations

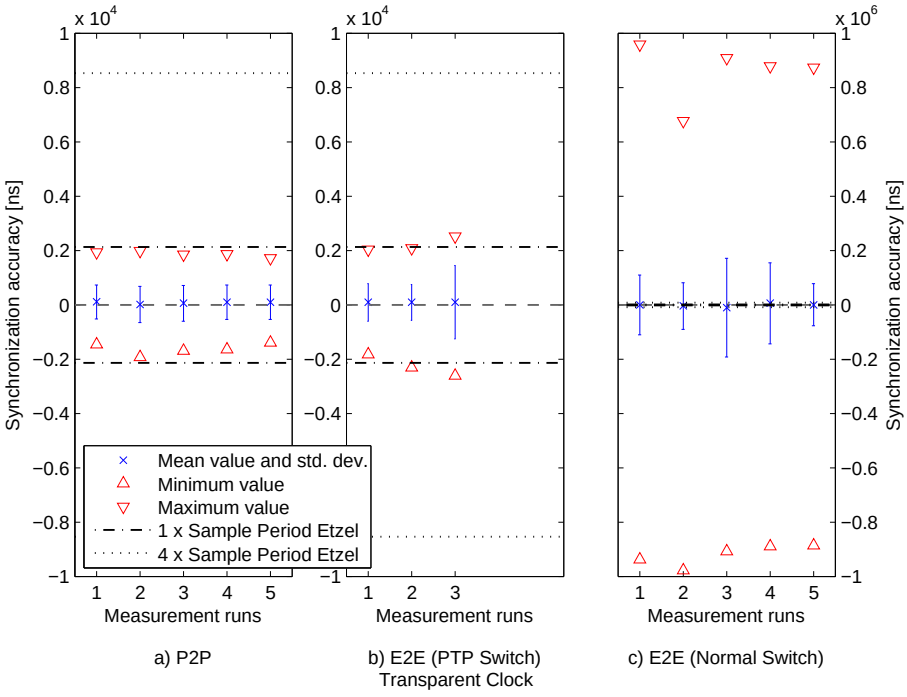
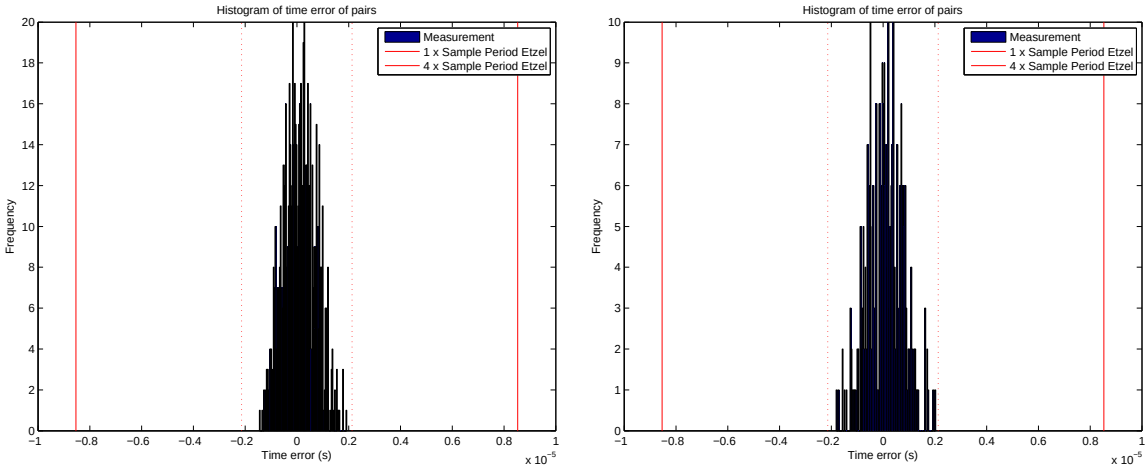


Figure 7.14: Results plot - Measurement event error in different network configurations

[17]Sample size of the interpolated data. Interpolation Method: Piecewise Cubic Interpolating Polynomial (PCHIP) adding 2 interpolation points per sample

[18]T_S is the sampling period of the Etzel ADC and amounts to $T_S = \frac{1}{469kS} = 2.132\mu s$

The distribution of the detected measurement event errors for one chosen measurement run in each network configuration is given in figure 7.15 and 7.16. The selected runs are marked in yellow in table 7.14 and serve to illustrate the characteristic distribution of each network configuration.
The sampling of data on the two Etzel devices is driven by two independent 60 MHz clocks. Because the two clock frequency is specified only to about ± 0.5 ppm each, a shifted frequency between 1 and 0.1 Hz is observed between the two clocks. With respect to the sampling of data this creates a systematic error on the 'measurement event error' varying between 0 and $\frac{1}{2} * T_S = 1066ns$ over the course of a few seconds.



(a) 'P2P' (b) 'E2E PTP'

Figure 7.15: Results plot - Distribution of measurement event errors for chosen measurement

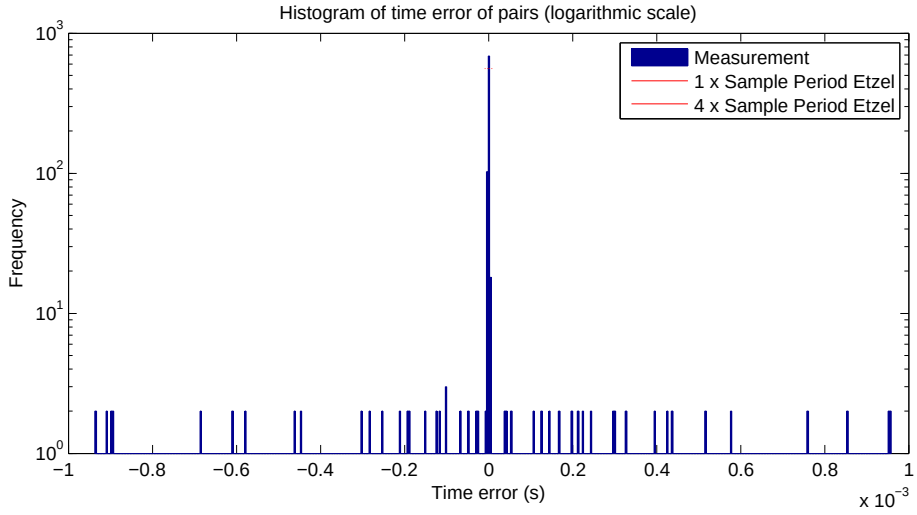


Figure 7.16: Results plot - Distribution of measurement event errors for chosen measurement 'E2E' (logarithmic y-scale)

The measurement event error in in 'P2P' and E2E PTP' mode appears to be Gaussian distributed and is in between the ±sample period marker. The 'E2E' result however shows a different distribution. The accuracy of the measured error is about a factor 100 worse, which seems to be typical for all measurements with non-PTP switches. A closer inspection of the data shows that a lot of the data from the PTP device is missing. These losses are irregularly spread and may have been caused by packet loss. An example is given in the appendix B.2.3.

It has to be stressed that the meaningfulness of the results for the 'measurement event error' is limited. On the one hand the sampling period is restricted by internal bandwidth limitations of the Etzel device ($4 * T_S = 8.533\mu s$), which results in only one sampling point on a measurement event (= a rising or falling flank). On the other hand the method used to determine the measurement event error is strongly depending on the exact implementation of the algorithm and is not perfectly well defined. Furthermore the sampling on two Etzel devices is not synchronized and therefore varies between ± 1 μs. The measurement event error results are thus a subjective examination of the available data. It is nonetheless assumed that the following observations are valid for potential use cases. The applied method is described in detail in the appendix B.2.3.
In summary the results for the 'measurement event error' is considered to be useful to show that the underlying time synchronization is applied to measurements. It may also be used for performance indication while keeping in mind that the results are a pessimistic estimation of the real performance.

Interpretation

For the two network configurations 'P2P' and 'E2E PTP' the mean value ± standard deviation of the the measurement event error is below the limit of the Etzel sampling period T_S. The recorded maximum and minimum values rarely exceed the 1000 us limit (1 out of hundreds) and never exceed the 8533 us limit. The 'E2E' configuration using a generic switch produces measurement event errors up to 0.95 ms and thus severely violates the timing requirement. Nonetheless it is assumed that this performance depends on the hardware implementation of the switch and therefore does not necessarily have to be as bad as present.
When taking a closer look at the extreme errors that were detected in a 'P2P' and 'E2E PTP' setting, it becomes clear that the exact implementation of the algorithm to detect the measurement event error is crucial. The conditions that are present in the one case that violates the limit in the 'E2E PTP' setting are described below. The situation shown in figure 7.17 shows a falling edge input signal that has been detected by two Etzel devices (dev107 acting as PTP master and dev77 acting as PTP slave).

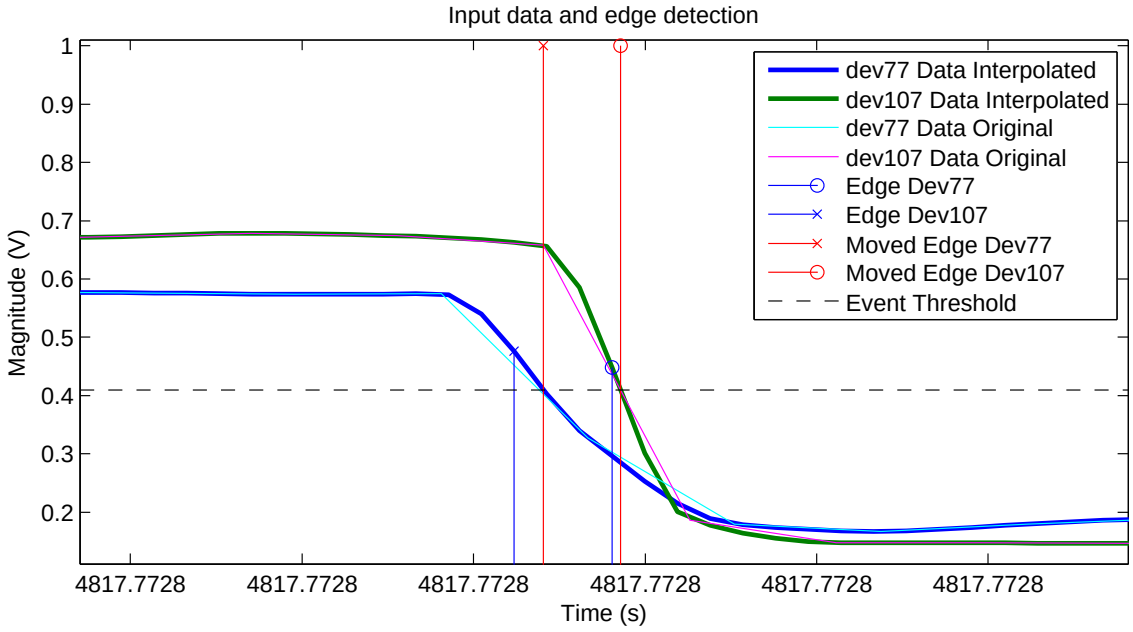


Figure 7.17: Results plot - Single measurement event that results in a large measurement event error (time difference between the two red lines)

The originally sampled data is shown in magenta for dev107 and in cyan for dev77. This data is interpolated (green and blue thick line) by the detection algorithm and acts as base for calculating the difference between two events (an event is defined as a flank crossing the threshold value). This example clearly shows the influence of the relatively low sample rate on the detection of the exact timestamp of a flank. For the original data of dev77 (cyan) there is exactly one timestamp assigned to the flank, the data for dev107 contains only timestamps before and after the flank. Therefore a nearly linear flank is assumed between the two 'surrounding' timestamps. The position of the sampling points on two Etzel devices is dependent on the FPGA clocks. Both clocks are subject to drift and jitter and their relative position is thus variable. After all, the measured time difference between the two flanks is probably not too bad, considering the available data. An improvement to the applied method might be to consider a fixed flank rise / fall time instead of interpolating the data which generates variable rise / fall times.

Significance of Measurement Data

In order to answer the question whether the required synchronization accuracy was achieved, an evaluation of the measurement data has been conducted under the consideration of the statistical significance of the data.[18]

$$\text{Is the measured } \textit{mean} + \textit{standard deviation} \leq 1000 \text{ ns ?} \tag{7.1}$$

For the purpose of answering the question 7.1, the following null hypothesis H_0 was tested. (μ = true mean value, σ = true standard deviation, σ_0 = nominal standard deviation)

$$H_0 : |\mu + \sigma| \leq \sigma_0$$

An upper-tailed test is performed to calculate type I errors and prove the claim. H_0 is rejected only if the test statistic χ^2 is larger than the critical value x_p.

$$\chi^2 > x_p$$

The test statistic χ^2 is calculated as Chi-square statistic. (N = Sample size, s = measured (sample) mean + standard deviation)

$$\chi^2 = \frac{(N-1)\cdot s^2}{\sigma_0^2}$$

For a level of significance of $\alpha = 0.01$ the critical value x_p is calculated by means of the *Chi-Square Inverse Cumulative Distribution Function (CDF)* $F^{-1}(p|v)$.[19]

$$x_p = F^{-1}(p|v)$$

$$p = F(x|v) = \int_0^x \frac{t^{(v-2)/2}\cdot e^{-t/2}}{2^{v/2}\cdot\Gamma(v/2)}dt$$

$$v = N - 1$$

$$p = 1 - \alpha$$

The gamma function Γ is defined by interpolating the factorial function $\Gamma(x+1) = x!$.

$$\Gamma(x) = \int_0^\infty e^{-t}\cdot t^{x-1}dt$$

For each measurement the calculated test statistic χ^2 and the critical value x_p is given in table 7.15.

Network config.	**Meas. event error [ns] | Mean + Std. Dev. |**	**No. of edges N**	**Test statistic χ^2**	**Critical value x_p**	**Reject H_0**
P2P	736.7	726	393.5	816.5	No
P2P	791.8	582	364.3	663.2	No
P2P	717.5	521	267.7	597.9	No
P2P	728.1	699	370.0	787.8	No
P2P	725.3	308	161.5	367.6	No
E2E PTP	786.3	291	179.3	349.9	No
E2E PTP	747.1	289	160.7	346.8	No
E2E PTP	1449.1	72	149.1	101.6	Yes
E2E	109469.0	851	10185943	948.8	Yes
E2E	89822.4	1058	8527943	1167	Yes
E2E	191330.8	460	16802831	532.4	Yes
E2E	155280.8	273	6558498	329.2	Yes
E2E	78636.4	487	3005270	561.5	Yes

Table 7.15: Measurement results - Evaluation of significance of measurement event error results

[19] Details concerning the execution of the calculations in Matlab are listed in appendix B.2.3

The results of the evaluation of significance of the measurement event error results leads to the conclusion that the question 7.1 can only be answered with yes in all 'P2P' measurements and all 'E2E PTP' measurements except one. In this specific measurement the sample size was very low which presumably lead to a high standard deviation of the measurement event error. This result is omitted due to the low sample size.
The performance is adequate for the actual Etzel sample rate of 2.132 μs. It has to be considered that due to bandwidth limitations samples are only collected at a rate of 8.533 ns and therefore require a certain minimum sample size to produce meaningful results. The conclusion is thus that the 'P2P' and 'E2E PTP' setting is suitable to deliver the required performance. Since all five 'E2E' measurements result in rejection of H_0, it is assumed that the 'E2E' setting is not suitable to deliver the required performance.
Measurements in 'P2P' and 'E2E PTP' configuration are deemed to perform sufficiently with a significance of $1 - 0.01 = 99\%$.

Chapter 8

Conclusion

This chapter describes conclusions that were drawn by the authors after finishing the project. Achieved goals are listed as well as necessary conditions for operating a synchronized distributed measurement network. Ultimately future development opportunities are given.
The chapter is split up into *implementation* and *testing* sections.

8.1 Implementation

8.1.1 Achieved Goals

Feasibility of the PTP Synchronization

The first aim of this project was to perform a feasibility study. Therefore, multiple network based time synchronization protocols were analyzed. Thereof, PTP was evaluated to be the most suitable protocol for this project. Furthermore, the in-house PTP stack was migrated to an Apalis evaluation board, in order to prove that a synchronization is possible between two evaluation boards. Moreover, a concept for the synchronization between the FPGA and the Ethernet controller was elaborated and simulated by three different simulations. The difference between the simulations never exceeded 150 ns. The main reason for the differences were the limitations of the simulations.

Since the migration of the PTP stack was successful and the simulations of the FPGA implementation matched and did not reveal problems for the implementation on Etzel, the feasibility of this project was considered to be given.

Implementation on Etzel

The implementation of the PTP synchronization was mainly done by ZI, i.e. the implementation of the FPGA concept. In the scope of this project was the development of the interface between PTP application and ZI data server. The implementation on Etzel was successful. Multiple Etzel devices were synchronized via the Ethernet network at ZI.

8.1.2 Improvements on the Implementation

Although the implementation was successful, such that Etzel devices could be synchronized, limitations of the implementation were discovered during measurements, which are described

in the following.

Synchronization Precision A remaining problem is that it is impossible to determine the PTP synchronization precision as a user. Although, the PTP stack calculates the time difference between the local and the master time, which could be used as feedback to the user, the time difference is not accessible by the PTP application. Note that the difference between local and master time is dependent on the path delay measurements of the PTP. A wrong path delay can give the impression that the PTP synchronization is locked in and synchronous, although it is not.

PPS Signal Improvement The signal form of the PPS signal on the LED pin of the Ethernet controller I210 shows a slow flank of approximately 40 ns. The reason for that is a missing termination and the fact that this pin is supposed to drive a LED. In order to make the flank steeper, the signal line of the PPS should be terminated. Furthermore, the input pin of the FPGA should be well synchronized, i.e. at least 4 flip-flops in a row.

8.1.3 Further Development on the Implementation

The precision of the time synchronization is limited by the hardware of the Ethernet controller as well as the PTP standard, e.g. the asymmetry of the up- and downlink is considered to be equal and the drift and offset are determined only on each clock cycle. *White Rabbit (WR)*[19] is a project, which takes care of the aforementioned problems. It is a fully deterministic Ethernet-based network for general purpose data transfer and synchronization. It can synchronize over 1000 nodes with sub-nanosecond accuracy. Since WR extends the PTPv2, it is perfectly compatible with other PTPv2 devices. However, WR would require extra hardware in order go get the full precision.

8.2 Testing

The results that were achieved in chapter 7 are compared against the initially established requirements in section 4.2. Additionally the basic conditions that are necessary for the expected performance of the system are determined.

8.2.1 Achieved Goals

The requirements from ZI have originally been defined in terms of 'Synchronisation accuracy', 'Time-to-Lock' and 'Quality of time synchronization'.

Synchronization Accuracy The initial goal concerning synchronization accuracy of the order of 10 ns has been confirmed by measurements of the PPS between Apalis devices. The distributed time bases showed time errors with mean values plus standard deviation of 15 to 45 ns, depending on the network configuration. These values are considered as lower bound for the error.
System tests have shown that two Etzel devices can be operated synchronously and maintain sufficient synchronization accuracy. The question "Is the measured mean value plus standard deviation (of the measurement event error) less or equal than 1000 ns?" has been answered with yes for both 'P2P' and 'E2E PTP' network configurations.

For these two configurations, extreme values of the error are always better than ±2000 ns, which represents an upper bound for the synchronization error and is sufficient below the sample period of 8533 ns. Therefore measurements can be performed synchronously with a distributed Etzel measurement setup under these conditions.

Time-to-Lock The Time-to-Lock for the PTP synchronization between two devices was measured at a mean value of 60.6 s and a worst case of 75 s meets the requirement of five minutes.

Quality of Time Synchronization The measurements prove that timestamps are incremented at each sample in a sufficiently uniform way. No negative time increments were recorded, however only the locked-in state was considered. Therefore the conclusion is that the required quality is achieved during locked-in operation. The time increments in 'P2P' and 'E2E PTP' configurations show a small standard deviation (around 1.7 ns) and thus fit the desired behavior very well.

8.2.2 Necessary Basic Conditions

In order to ensure the expected performance, the following requirements have to be fulfilled in respect to the device setup and environmental conditions.

Network The Etzel devices need to be connected directly (P2P) or with a PTP-capable switch to guarantee sufficient performance. Generic (Non-PTP) switches may also provide the required performance but are dependent on the exact implementation of the switch.
External network traffic below 30 % of the maximum achievable data rate does not influence the synchronization accuracy at all. The effect of external traffic on the accuracy got noticeable only for data rates above 75 % of the maximum achievable rate of about 40 MiB/s.

PTP Configuration The PTP application needs to be set to a rate of 'Sync' packets of 8 $\frac{packets}{s}$ and the 'DelayReq' to $\frac{1}{16}\frac{packets}{s}$ for highest accuracy and lowest traffic overhead. These values have been set as default configuration. The required bandwidth for PTP Ethernet packets in this configuration is 0.8 KB/s.

CPU Load The influence of limited CPU resources is negligible even for the case of all involved CPUs undergoing 100 % load.

Temperature Static ambient temperature differences between two devices influence the synchronization accuracy below 10 ns and are thus negligible. Variable temperature changes at the maximum measurable rate of $5.3\frac{^\circ C}{min}$ are negligible as well.

Multiple Devices Up to four Apalis devices have been tested in a synchronized network setup. While the standard deviation of the synchronization accuracy was unaffected, the mean value differed by about 15 ns (when comparing four to two devices).

8.2.3 Discovered Limitations

The measurements led to the discovery of unexpected limitations to the operation of time synchronized Etzel devices in an Ethernet network.

Maximum Rate of PTP Sync Packets The maximum rate of PTP Sync packets was measured to be 8 packets per second. The reason for this limitation is assumed to be a restriction defined in the core part of the PTP stack. In order to limit the CPU load generated by the processing of PTP events, new events are only handled at a rate of at most 10 events per second. The corresponding measurement is 'U3 - Bandwidth for Synchronization Packets vs. Accuracy'.

Insufficient Time Synchronization When Using a Generic Switch In an 'E2E' configuration of two Etzel devices, the quality of time increments was restricted due to missing samples and timestamps at the slave device. It is assumed that this loss was caused by the switch not being able to timely route the PTP packets. Therefore it is advised to only use PTP-capable switches or direct (P2P) connections for synchronized measurements.
Measurement results are listed in 'S0 - Etzel time synchronization'.

Negative Time Increments The ZI data server that distributes the measurement data is not capable of handling non-monotonous timestamps. The only time this behaviour has been observed is during the startup phase of an Etzel. The problem occurs only when the PTP application connects to a PTP master that runs a clock which is in the past (compared to the local clock). Another requirement for this problem to occur is that the time difference between the two devices must be larger than $\frac{1}{60MHz} * \frac{1}{16} * 2^{26} = 69.905ms$. Possible mitigation strategies are described in section 3.3.4.

8.2.4 Future Measurements

It is advised that the following additional system tests with complete Etzel measurement setups are performed in the future to ensure reliability in the long run.

- S3 - Long-Term Consistency of Time Increments
- S4 - Long-Term Evaluation of Synchronized Measurement Events

8.2.5 Further Development Opportunities

Possibilities to improve the performance of synchronized Etzel measurement setups are discussed below.

Variations in the Mean Value of Synchronization Accuracy Although measurement results under the same conditions were fairly consistent and repeatable, certain outliers have been observed.[1] In order to determine the cause of such unexpected behaviour additional examinations would be necessary. Since the resulting PPS mean plus standard deviation stayed below 50 ns the current state of the implementation was deemed acceptable.

[1] For example the Ethernet PPS result for 'E2E PTP' configuration in section 7.2.1 with a mean synchronization accuracy (PPS) of 39 ns instead of expected 2 ns.

For further development it might be interesting to try to move away from the dedicated Intel I210 controller and move the Ethernet MAC into the FPGA. With more control over the hardware implementation, the accuracy might be improved.

Synchronize Sampling Clocks Due to the sampling clocks of two Etzel devices being unsynchronized, a systematic clock drift is observed. The maximum error between two clocks is half the sampling rate ($\frac{1}{2} \cdot \frac{1}{60MHz} \cdot 128 = 1066ns$).
In order to mitigate the effects of unequal sampling clocks on the measurement event error as described in the results of 'S2', the sampling clock might instead be locked to the synchronized time in the FPGA.

Consider Variable Sampling Clock at Evaluation Instead of correcting the unsynchronized sampling clocks at runtime, their difference might (partially) be corrected when evaluating the measurement data. If all involved devices are exposed to the same environmental conditions, the difference in oscillator frequency should be fairly constant and can therefore be estimated and corrected. With this measure, the systematic error (of up to 1066 ns) could be corrected as well.

Appendix Implementation

A.1 Configuration of the Hardware Timestamping

In Listing A.1, the possible definitions for RX hardware timestamp filtering are listed.

```
// /usr/include/linux/net_tstamp.h

...
/**
 * struct hwtstamp_config - %SIOCSHWTSTAMP parameter
 *
 * @flags:  no flags defined right now, must be zero
 * @tx_type:  one of HWTSTAMP_TX_*
 * @rx_type:  one of one of HWTSTAMP_FILTER_*
 *
 * %SIOCSHWTSTAMP expects a &struct ifreq with a ifr_data pointer to
 * this structure. dev_ifsioc() in the kernel takes care of the
 * translation between 32 bit userspace and 64 bit kernel. The
 * structure is intentionally chosen so that it has the same layout on
 * 32 and 64 bit systems, don't break this!
 */
struct hwtstamp_config {
  int flags;
  int tx_type;
  int rx_filter;
};

/* possible values for hwtstamp_config->tx_type */
enum {
  /*
   * No outgoing packet will need hardware time stamping;
   * should a packet arrive which asks for it, no hardware
   * time stamping will be done.
   */
  HWTSTAMP_TX_OFF,

  /*
   * Enables hardware time stamping for outgoing packets;
   * the sender of the packet decides which are to be
   * time stamped by setting %SOF_TIMESTAMPING_TX_SOFTWARE
   * before sending the packet.
   */
  HWTSTAMP_TX_ON,
};

/* possible values for hwtstamp_config->rx_filter */
```

```
42 enum {
43   /* time stamp no incoming packet at all */
44   HWTSTAMP_FILTER_NONE,
45
46   /* time stamp any incoming packet */
47   HWTSTAMP_FILTER_ALL,
48
49   /* return value: time stamp all packets requested plus some others */
50   HWTSTAMP_FILTER_SOME,
51
52   /* PTP v1, UDP, any kind of event packet */
53   HWTSTAMP_FILTER_PTP_V1_L4_EVENT,
54   /* PTP v1, UDP, Sync packet */
55   HWTSTAMP_FILTER_PTP_V1_L4_SYNC,
56   /* PTP v1, UDP, Delay_req packet */
57   HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ,
58   /* PTP v2, UDP, any kind of event packet */
59   HWTSTAMP_FILTER_PTP_V2_L4_EVENT,
60   /* PTP v2, UDP, Sync packet */
61   HWTSTAMP_FILTER_PTP_V2_L4_SYNC,
62   /* PTP v2, UDP, Delay_req packet */
63   HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ,
64
65   /* 802.AS1, Ethernet, any kind of event packet */
66   HWTSTAMP_FILTER_PTP_V2_L2_EVENT,
67   /* 802.AS1, Ethernet, Sync packet */
68   HWTSTAMP_FILTER_PTP_V2_L2_SYNC,
69   /* 802.AS1, Ethernet, Delay_req packet */
70   HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ,
71
72   /* PTP v2/802.AS1, any layer, any kind of event packet */
73   HWTSTAMP_FILTER_PTP_V2_EVENT,
74   /* PTP v2/802.AS1, any layer, Sync packet */
75   HWTSTAMP_FILTER_PTP_V2_SYNC,
76   /* PTP v2/802.AS1, any layer, Delay_req packet */
77   HWTSTAMP_FILTER_PTP_V2_DELAY_REQ,
78 };
```

Listing A.1: Possible values for fimestamp filtering

A.2 FPGA Registers

In Table A.1, the implemented registers of the FPGA are listed, which are externally accessible lateexternallyusing the ZI Data Server.

Register	R/W	Size	Description
Timestamp	R	64 bit	*Timestamp* is the current timestamp.
Next Timestamp	R/W	64 bit	*Next Timestamp* defines the time of the next PPS.
Control Enable	R/W	1 bit	If *Control Enable* is set, the FPGA synchronizes to the above defined time. Otherwise, the timestamp unit acts as a free-running counter.
Alpha Low	R/W	4 bit	*Alpha Low* is the low-pass filter parameter of the drift correction, if the offset is below the *Threshold*. The filter parameter α is defined as follows: $\alpha = \frac{1}{2^{AlphaLow}}$.
Alpha High	R/W	4 bit	*Alpha High* is the low-pass filter parameter of the drift correction, if the offset is above the *Threshold*. The filter parameter α is defined as follows: $\alpha = \frac{1}{2^{AlphaHigh}}$.
Threshold	R/W	6 bit	The *Threshold* defines the switching between *Alpha Low* and *Alpha High*. The threshold th is defined as follows: $th = 2^{Threshold+1} * 16.67ns$.
Drift	R	signed 32 bit	The *Drift* represents the drift register value. The value is updated with every *Next Timestamp* write.
Offset	R	signed 32 bit	The *Drift* represents the offset register value. The value is updated with every *Next Timestamp* write.

Table A.1: Implemented registers which can be accessed by the FPGA interface.

Appendix Testing

B.1 Measurement Plan for Unit Tests

On the following pages all unit tests are described in detail. The important parameters for each test setup are characterized in a table. Further details are given in text form after each table, describing how exactly the measurement was performed and how problems were solved.

Test Setup Details Since (almost) all unit tests rely on the measurement of the offset between the PPS signal of multiple devices, the general setup was the same for all tests. It is displayed in figure B.1. The measurement setup consists of two or more development boards with a Apalis processor module on each device, these are the DUT. The PPS signal is generated by the Ethernet controller, where a small cable is soldered onto the corresponding SDP[2]. An oscilloscope is used to sample these signals and evaluate the time difference between the rising edges. The oscilloscope always triggers on the device which acts as PTP master and is connected to channel 1 (yellow). The second device (which acts as PTP slave and which shall be compared to the master) is connected to channel 2 (blue). More details to the configuration of the oscilloscope are given in section B.3.

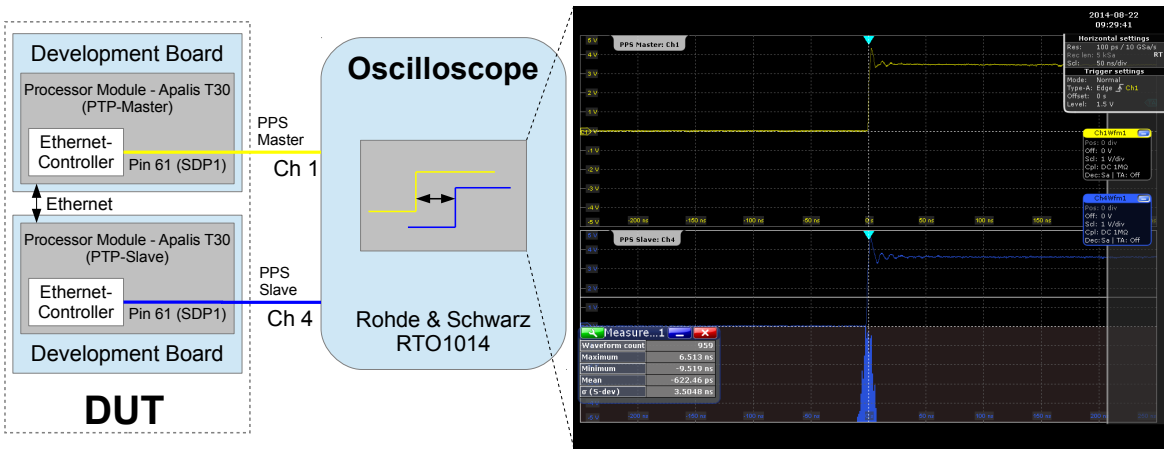


Figure B.1: Test setup for PPS time offset measurement

The right side of figure B.1 contains a screen shot of the oscilloscope used for PPS measurements. The yellow waveform on top represents the previously triggered rising edge of the PTP master PPS signal and the blue waveform in the middle the corresponding signal

[2]On the Intel I210 Integrated Circuit (IC): Pin 61 (SDP1)

from the PTP slave. On the bottom a histogram describes the distribution of the slave PPS relative to the master PPS. Thus characterizing the synchronization accuracy.

Default settings and conditions Where nothing else is specified, the following default settings / conditions were applied: Two or more DUT were placed on a table and connected via P2P Ethernet connection[3]. Both boards were exposed to room temperature (approximately 24 °C). One development board was always used as master and the other always as slave. The PTP message rates were set at $4\frac{packets}{s}$ for the 'Sync' packets and at $\frac{1}{16}\frac{packets}{s}$ for the 'DelayReq' packets.

The following subsections contain information about each test in detail.

B.1.1 U0 - Time-to-Lock

U0	Verification of the synchronized time keeping accuracy of two development boards. **Time-to-Lock.**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Time to synchronization lock.
Measured indicator	Time difference between the two PPS signals is used to indicate when a sufficient synchronization is achieved.
Measurement device	The PPS signals of two development boards are compared and a signal is generated as soon as a predefined accuracy is reached. This signal is used to reset the PTP clock and restart the synchronization process on both boards. The Time-to-Lock is logged.
Additional Material	Ethernet connection between the two development boards (P2P).
Duration	100 synchronization cycles
Input Signal	-
Comment	-

Table B.2: Measurement description for Time-to-Lock measurement of two development boards

Detailed Description Two different measurements shall be performed. Firstly the development of the time synchronization accuracy shall be examined on one device by plotting the time error of every PPS pulse from power-on to synchroniation lock. This measurement was a single event. Secondly the statistical distribution of the complete time-to-lock (from power-on to synchronization lock) shall be evaluated over 100 reboots. The main evaluation of this measurement is located in section 7.1.1, although secondary measurement results are presented after this measurement description in paragraph 'Distribution of Initial PPS Pulses'.

[3]A 1.5 m Cat5 Twisted-Pair cable was used

Initial Time Offset Development For the measurement of the initial time offset from master (from power-on to synchronization lock), the oscilloscope had to be set to a reduced time resolution. An initial coarse time resolution allowed the sampling of ±500 ms time errors. In a second phase, with a minimum resolution of 2 ns, offsets of up to ±1 ms could be detected. Due to the slow signal period of 1 second, settings were changed on the fly.

Time-to-Lock In order to get comparable data in a worst case scenario (i.e. start from a powered off state), the time from the device booting to achieving synchronization lock has been measured. After each test, the master and slave device were rebooted. The moment when the screen went black was considered to be the starting time for a new measurement. The end time of a test was defined by the first occurrence of five consecutive PPS pulses with an accuracy of less than ±10 ns. These 5 samples have been included in each measurement of 'time to synchronization lock' in table B.3. The start and end conditions were checked visually with the aid of the previously used oscilloscope setup.
The time between reboot and the Linux Lightweight X11 Desktop Environment (LXDE) [20] being ready to accept user input has been measured to 28 seconds. The PTP application was then automatically started as service by systemd with the standard configuration (4 synchronization packets per second and 1/16 delay request packets per second). The service that was started at each boot (listing B.2), calls a shell script which first detects the local IP address and then runs the PTP application with the IP address as parameter.

```
1 [Unit]
2 Description = PTP-Stack for time synchronization
3
4 [Service]
5 ExecStart = /home/root/run-ptp.sh
6
7 [Install]
8 WantedBy = multi-user.target
```

Listing B.2: Service definition file for ptp-application

The following table B.3 contains all measurement results for time-to-lock in seconds.

Time to synchronization lock [s][4]									
57	65	56	63	66	58	57	57	60	63
64	63	58	57	58	67	62	57	57	65
60	62	67	63	68	59	55	55	61	59
62	67	62	56	66	60	61	58	58	60
61	65	58	53	59	52	60	56	59	60
62	60	68	54	62	60	65	60	58	61
59	68	64	65	59	57	58	56	59	61
70	68	65	52	60	57	59	59	56	62
58	60	67	56	57	59	59	54	58	61
78	66	64	65	61	58	60	58	60	61

Table B.3: Measurement results for Time-to-Lock measurement

Distribution of Initial PPS Pulses While performing the time-to-lock measurements, the statistical distribution of the initial time synchronization pulses that fall within synchro-

nization lock was measured as well. The first 10 PPS pulses with an accuracy better than ±10 ns have been captured[5] in figure B.2 and are evaluated in table B.4. The yellow marker lines in figure B.2 represent ±10 ns.

Synchronization accuracy [ns]				**Sample size**
Mean	**Std. deviation**	**Min**	**Max**	
-2.403	13.28	-250	184.87	5870

Table B.4: Measurement results - Distribution of initial PPS pulses

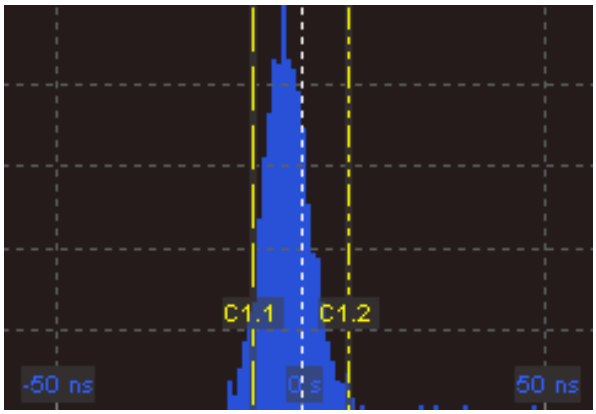


Figure B.2: Results histogram - Distribution of initial 10 'accurate' PPS pulses

[4]The 5 seconds from the first consecutive samples that were accurate enough to qualify as 'synchronized' are included

[5]The preceding pulses (before synchronization lock was achieved) were less accurate than ±250 ns, so that they have not been captured by the scope of this measurement. Some of them happened to randomly fall into the capturing range, which led to the extreme maximum / minimum values

B.1.2 U1 - Time Offset (PPS)

U1	Verification of the synchronized time keeping accuracy of two development boards. **Time offset (PPS).**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Synchronization accuracy.
Measured indicator	Time difference between the two PPS signals.
Measurement device	The network controllers on each Apalis module generate a pulse per second on a GPIO pin. This pin is probed with a digital oscilloscope. By triggering on the PPS signal of the PTP master device, the time offset of the PTP slave is displayed. The offset is evaluated statistically.
Additional Material	Ethernet connection between the two development boards (P2P).
Duration	2 hours
Input Signal	-
Comment	Measurement setup according to figure B.1

Table B.5: Measurement description for time offset measurement of two development boards

Detailed Description Two Apalis boards were exposed to default environmental conditions (room temperature, default software settings, etc.) to measure the reference time synchronization accuracy via the PPS signal. The measurement was performed at the 'SDP1' pin of the Ethernet controller, which will not directly be available to the FPGA for implementation of the Etzel synchronization. Instead a LED pin is available[6], which routes the same functionality as the SDP pin but has a large rise time of about 40 ns. For the purpose of the unit tests we use the SDP1 pin, which provides sharp edges and very little jitter. Details concerning the distinction between these two pins are described in 3.2.8 'Problems during the Implementation'.
The measurement results of the LED pin are described in table B.6 and depicted in figure B.3 for comparison with the SDP pin. The results mostly match the previous SDP results, except for the maximum value of 224.95 ns.

Synchronization accuracy [ns]				**Sample size**
Mean	**Std. deviation**	**Min**	**Max**	
-6.946	5.808	-22.545	224.95	7499

Table B.6: Measurement results - Time offset (PPS) measured at LED-Pin

[6]The PPS signal can internally be routed to the 'Ethernet Active' LED pin but requires small hardware modifications in order to allow the measurement of a clean signal.

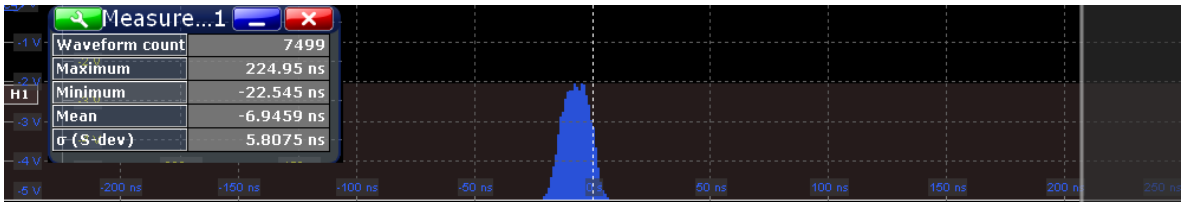


Figure B.3: Results histogram - Time offset (PPS) measured at LED-Pin

B.1.3 U2 - Time Offset (PPS) long-term

U2	Verification of the synchronized time keeping accuracy of two development boards. **Time offset (PPS) long-term.**
	All parameters except the duration are the same as in U1
Duration	24 hours
Input Signal	-
Comment	Measurement setup according to figure B.1

Table B.7: Measurement description for long-term time offset measurement of two development boards

Detailed Description The exact same setup was used as in B.1.2, only the duration was changed to 24 hours (= 86400 samples).

B.1.4 U3 - Bandwidth for Synchronization Packets vs. Accuracy

U3	Verification of the synchronized time keeping accuracy of two development boards. **Bandwidth for synchronization packets vs. accuracy**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Synchronization accuracy.
Measured indicator	Time difference between the two PPS signals. Standard deviation.
Measurement device	The network controllers on each Apalis module generate a pulse per second on a GPIO pin. This pin is probed with a digital oscilloscope. By triggering on the PPS signal of the PTP master device, the time offset of the PTP slave is displayed. The offset is evaluated statistically.
Additional Material	Ethernet connection between the two development boards (P2P).
Duration	30 minutes per interval setting
Input Signal	-
Comment	Choose bandwidth settings from nearly 0 B/s to the highest possible value.

Table B.8: Measurement description for synchronization packet bandwidth measurement of two development boards

Detailed Description The bandwidth used by synchronization packets is determined by the interval of synchronization and delay request packets. The PTP application core software only permits rates of synchronization or delay request packets of 2^x where x can be chosen. Per default synchronization packets are sent every second (2^0) and delay request packets every 16 seconds (2^4). Sync intervals values between 2^1 and 2^{-10} (= 0.5 to 1024 packets per second) were used as setting values with a focus on optimizing accuracy (with the least possible bandwidth used). Delay request intervals are secondary but shall be tested as well. A range of 2^4 to 2^{-10} was considered.
The scripts that were used to measure the network traffic are listed below in B.3 and B.4.

```
#!/bin/bash

INTERVAL="1"  # update interval in seconds

if [ -z "$1" ]; then
        echo
        echo usage: $0 [network-interface]
        echo
        echo e.g. $0 eth0
        echo
        exit
```

```
fi

IF=$1

while true
do
        R1=`cat /sys/class/net/$1/statistics/rx_bytes`
        T1=`cat /sys/class/net/$1/statistics/tx_bytes`
        sleep $INTERVAL
        R2=`cat /sys/class/net/$1/statistics/rx_bytes`
        T2=`cat /sys/class/net/$1/statistics/tx_bytes`
        TBPS=`expr $T2 - $T1`
        RBPS=`expr $R2 - $R1`
        TKBPS=`expr $TBPS`
        RKBPS=`expr $RBPS`
        echo "TX $1: $TKBPS B/s RX $1: $RKBPS B/s"
done
```

Listing B.3: U3: Traffic measurement: Bandwidth

```
#!/bin/bash

INTERVAL="1"  # update interval in seconds

if [ -z "$1" ]; then
        echo
        echo usage: $0 [network-interface]
        echo
        echo e.g. $0 eth0
        echo
        echo shows packets-per-second
        exit
fi

IF=$1

while true
do
        R1=`cat /sys/class/net/$1/statistics/rx_packets`
        T1=`cat /sys/class/net/$1/statistics/tx_packets`
        sleep $INTERVAL
        R2=`cat /sys/class/net/$1/statistics/rx_packets`
        T2=`cat /sys/class/net/$1/statistics/tx_packets`
        TXPPS=`expr $T2 - $T1`
        RXPPS=`expr $R2 - $R1`
        echo "TX $1: $TXPPS pkts/s RX $1: $RXPPS pkts/s"
done
```

Listing B.4: U3: Traffic measurement: Packets per seocnd

B.1.5 U4 - Network Load vs. Accuracy

U4	Verification of the synchronized time keeping accuracy of two development boards. **Network load vs. accuracy**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Synchronization accuracy.
Measured indicator	Time difference between the two PPS signals. Standard deviation.
Measurement device	The network controllers on each Apalis module generate a pulse per second on a GPIO pin. This pin is probed with a digital oscilloscope. By triggering on the PPS signal of the PTP master device, the time offset of the PTP slave is displayed. The offset is evaluated statistically.
Additional Material	Ethernet connection between the two development boards (P2P). The additional network traffic shall be generated by the PTP slave development board.
Duration	30 minutes per load setting
Input Signal	-
Comment	The network load is defined by bandwidth used. Rates up to the maximum achievable data rates shall be tested.

Table B.9: Measurement description for network load impact on accuracy measurement of two development boards

Detailed Description Traffic is generated by sending zeros from the PTP slave device to the PTP master via TCP. Transmitting is done via 'NetCat (nc)' and receiving is handled by 'socat'. In order to limit the traffic rate and display the current rate, all traffic is piped through 'Pipe Viewer (pv)'. The commands used to execute the transmit and receive functionality are listed in B.5 and B.6.

```
1 $ dd if=/dev/zero | pv -ar --rate-limit 1M | nc 192.168.1.150 42
```

Listing B.5: U4: PTP Slave as Transmitter

```
1 $ socat - TCP-LISTEN:42 | pv -ar > /dev/null
```

Listing B.6: U4: PTP Master as Receiver

First of all the local maximum data rate was measured to 1.92 GiB/s. This determines the upper bound of data transfer rates achievable by the CPU.

```
1 $ pv /dev/zero > /dev/null
```

Listing B.7: Local maximum rate

Subsequently the maximum data rate achievable when sending the same data over the network was determined[7]. The results define the upper bound of data rates achievable over Ethernet. The results were taken with and without the PTP application running in the background, according to listing B.8.

Average rates	Client	Server
With PTP	32.2 MiB/s	31.2 MiB/s
Without PTP	38.6 MiB/s	34.7 MiB/s

Table B.10: Maximum transfer rates achievable over Ethernet using 'netcat'

```
#Server:
$ socat - TCP-LISTEN:42 | pv -ar > /dev/null

#Client:
$ dd if=/dev/zero bs=1M count=500 | pv -ar | nc 192.168.1.150 42
```

Listing B.8: TCP maximum rate using netcat

A different method to determine the maximum transfer rates achievable over Ethernet is using the program 'iperf' as mentioned in B.9. This method resulted in a bandwidth of 46 MB/s. Presumably this value is higher than in table B.10 because this application is optimized for network bandwidth, whereas the aforementioned method (based on netcat) is very versatile but probably less efficient.

```
#Server:
$ iperf -s

#Client:
$ iperf -c 192.168.1.150
```

Listing B.9: TCP maximum rate using iperf

[7]The rates measured are lower due to the server program requiring to be started earlier and therefore measuring a rate of 0 MiB/s for a certain time.

B.1.6 U5 - CPU Load vs. Accuracy

U5	Verification of the synchronized time keeping accuracy of two development boards. **CPU load vs. accuracy**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Synchronization accuracy.
Measured indicator	Time difference between the two PPS signals. Standard deviation.
Measurement device	The network controllers on each Apalis module generate a pulse per second on a GPIO pin. This pin is probed with a digital oscilloscope. By triggering on the PPS signal of the PTP master device, the time offset of the PTP slave is displayed. The offset is evaluated statistically.
Additional Material	Ethernet connection between the two development boards (P2P).
Duration	15 minutes per load setting
Input Signal	-
Comment	PTP Master: 0 / 50 / 80 / 90 / 100 % and PTP Slave: 0 / 50 / 80 / 90 / 100 %

Table B.11: Measurement description for CPU load impact on accuracy measurement of two development boards

Detailed Description Generic CPU load can be emulated by running the program 'stress'. The option '-c 4' spawns four workers, which spin on sqrt(). This results in all four CPU cores being fully stressed. The fact that all four cores are utilized was verified running 'ps -L -o pid,tid,psr,pcpu'. The keyword 'PSR' specifies the CPU core that is assigned to a process. In order to limit the generated load, the program 'cpulimit' was applied after the workers have been spawned. The load can be limited per worker between 0 and 100% with the variable 'LOAD_PERC' in listing B.10. Since an equal load shall be simulated, the same 'LOAD_PERC' limit was applied to all four workers per test.

```
1 $ stress -c 4
2 $ pidof stress                          #returns {PID_parent,PID0,PID1,PID2,PID3}
3 $ cpulimit -p $PIDx -l $LOAD_PERC       #for $PIDx = {PID0,PID1,PID2,PID3}
```

Listing B.10: CPU load testing

B.1.7 U6 - Temperature vs. Accuracy

U6	Verification of the synchronized time keeping accuracy of two development boards. **Temperature vs. accuracy**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Synchronization accuracy.
Measured indicator	Time difference between the two PPS signals. Standard deviation.
Measurement device	The network controllers on each Apalis module generate a pulse per second on a GPIO pin. This pin is probed with a digital oscilloscope. By triggering on the PPS signal of the PTP master device, the time offset of the PTP slave is displayed. The offset is evaluated statistically.
Additional Material	Ethernet connection between the two development boards (P2P).
Duration	30 minutes per load setting
Input Signal	-
Comment	The PTP slave is kept at room temperature while the PTP master is subjected to the following temperatures: 5°C / 20°C / 30°C / 40°C

Table B.12: Measurement description for temperature impact on accuracy measurement of two development boards

Detailed Description The usecase of 'decentralized synchronization' contains the placement of Etzel devices in separate rooms. Environmental conditions may differ amongst different rooms, thus measurements were performed between 5°C and 40 °C in the following configurations:

- The master DUT was subjected to a certain fixed temperature, while the slave was kept at unstabilized room temperature
- Both devices were subjected to the same fixed temperature
- The master DUT was subjected to a ramping up temperature, while the slave was kept at unstabilized room temperature

A climatic chamber of the type 'Heraeus Vötsch VMT 04/16' (details in appendix B.3.2) was used to stabilize the temperature of one or two DUT. The chamber provides two operating modes: i) setting up a fixed temperature, which is then held by switching on or off the heating or cooling unit and ii) toggling between two preset temperatures in order to perform a linear temperature ramp.

B.1.8 U7 - Cable Length vs. Accuracy

U7	Verification of the synchronized time keeping accuracy of two development boards. **Cable length vs. accuracy**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Synchronization accuracy.
Measured indicator	Time difference between the two PPS signals. Standard deviation.
Measurement device	The network controllers on each Apalis module generate a pulse per second on a GPIO pin. This pin is probed with a digital oscilloscope. By triggering on the PPS signal of the PTP master device, the time offset of the PTP slave is displayed. The offset is evaluated statistically.
Additional Material	Ethernet connection between the two development boards (P2P).
Duration	30 minutes per cable length
Input Signal	-
Comment	Compare twisted pair Ethernet cables with different lengths (1 m to 40 m) and cable categories (5 to 6)

Table B.13: Measurement description for impact of the cable length on accuracy of two development boards

Detailed Description The influence of the Ethernet cable used to connect two DUT is assumed to be relatively small, which shall be proved by measurements. It is assumed that any delays introduced by the cable would be rather fixed than variable. Thus they can be detected by the PTP and corrected. Potentially influential cable parameters are the length and the cable category[21].

B.1.9 U8 - More Than Two Devices (With and Without a PTP-Capable Switch)

U8	Verification of the synchronized time keeping accuracy of two development boards. **More than two devices (with and without a PTP-capable switch)**
DUT	Development board with Apalis T30 module and PTP-stack application
No. of DUT and Setup	Two development boards are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Synchronization accuracy for more than two devices. Either connected over a PTP-capable switch or a ordinary switch.
Measured indicator	Time difference between two PPS signals. Standard deviation.
Measurement device	The network controllers on each Apalis module generate a pulse per second on a GPIO pin. This pin is probed with a digital oscilloscope. By triggering on the PPS signal of the PTP master device, the time offset of the PTP slave is displayed. The offset is evaluated statistically.
Additional Material	Ethernet connection between the development boards (via PTP-aware switch).
Duration	30 minutes per setup
Input Signal	-
Comment	Similar test to U1, except for the connection method.

Table B.14: Measurement description for impact of a PTP-switch on the accuracy of multiple development boards

Detailed Description It is central to the usecase of 'decentralized synchronization' to employ more than two DUT, thus requiring a central multiport Ethernet connector (switch). It is expected that the following parameters of the network topology influence the synchronization accuracy:

- The type of Ethernet switch (mainly whether it is PTP-capable or not)
- The amount of DUT that attempts to synchronize via PTP

In order to examine synchronization accuracy in these various network topologies, the PPS offset between a fixed slave and the master was measured.
The non-PTP-capable switch used for the tests was a Zyxel GS-108B (details in B.4.3), which is a 8-port Gigabit Ethernet switch. It supports only the 'Store-and-Forward' transmission mode.
The PTP-capable switch that was available for tests, was a Hirschmann RSP 35 managed switch (details in B.4.4). Although it is intended to be used as high reliability industrial Ethernet switch, it also provides configurable PTP capabilities. In particular both boundary clock and transparent clock modes as described in section 5.1. The settings that were chosen for each mode are listed below:

General PTP settings

- Operation IEEE 1588/PTP: On
- PTP Mode: v2-transparent-clock OR v2-boundary-clock
- Sync Lower Bound [ns]: 30
- Sync Upper Bound [ns]: 5000
- Enable PTP Management: No

Transparent clock

- Delay Mechanism: E2E
- Primary Domain: 0
- Network Protocol: UDP/IPv4
- Multi Domain Mode: No
- VLAN: none
- VLAN Priority: 4
- Syntonize: Yes
- Synchronize local clock: Yes

Boundary clock

- Priority 1: 128
- Priority 2: 128
- Domain Number: 0
- Two Step: Yes
- Network Protocol (under Port Tab): UDP/IPv4

Slave-to-Slave Synchronization Accuracy Aditional measurements were performed to verify the time synchronization functionality between two PTP slave devices. The setup consisted of three Apalis boards connected to the PTP-enabled switch. The oscilloscope was connected to both slaves instead of the master and a slave. The measured offset therefore is not a 'PPS offset from master', but a 'PPS offset between slaves'. The results are described in table B.15 and the histogram is displayed in figure B.4.

Synchronization accuracy [ns]				**Sample size**
Mean	**Std. deviation**	**Min**	**Max**	
4.851	33.317	-84.669	99.699	2610

Table B.15: Measurement results - Slave to slave PPS offset

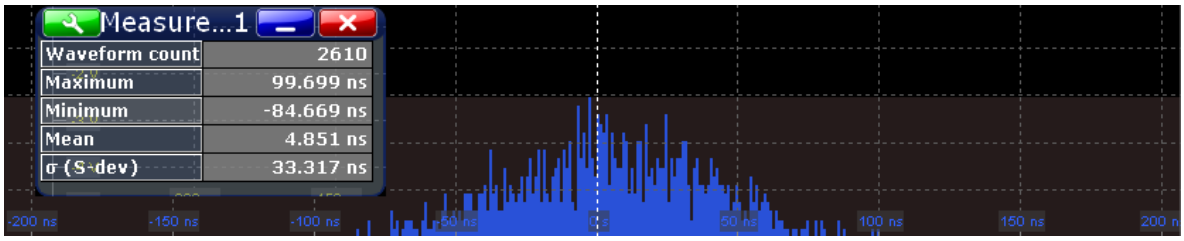


Figure B.4: Results histogram - Slave to slave PPS offset

B.2 Measurement Plan for System Tests

Detail Description The execution of system tests S0 to S2 was split up into two parts: measurements at the ZI facilities and subsequent analysis of the data. The measurement required two Etzel devices, a PC and Ethernet connectivity plus a Rhode & Schwarz RTO1024 oscilloscope. Analysis of the measurement data was done in Matlab.

Measurement Two Etzel devices were connected to the same switch (depending on the chosen network configuration) or directly P2P and operated for at least 1 minute to ensure that the PTP synchronization is locked in. Subsequently the Etzels were configured by means of the ZI web interface. Due to memory and bandwidth limitations in the Etzel architecture a consistent stream of data could only be recorded for less than 30 seconds. The bandwidth of the Etzel frontend was selected as large as possible to achieve a high resolution (at the time of the measurement it was restricted to maximum 51.1 kHz). However this resulted in a low Signal-to-Noise-Ratio (SNR) and the need for relatively extensive post-processing in order to find corresponding data for each measurement event.
During operation each Etzel produces a constant stream of measurement data which is sent from the local[8] ZI data server to the local web server. For the purpose of analyzing the data after the actual measurement, a special instance of the web server was set up on a PC that was connected to the DUT. This allowed the collection of the measurement data in a '.mat' file. Multiple measurements with a duration of a few seconds up to a minute each were taken per network configuration.

Setup The setup for all measurements varied in terms of the network configuration and the connection method to the host PC. The following overall steps were followed to perform a measurement:

- Connect the Etzels via Ethernet: direct (P2P) or via a switch (E2E) or a PTP-enabled switch (E2E PTP)
- Connect a host PC to the same network or establish a direct connection to each Etzel via USB
- Start the ZI data server on both Etzels
- Start the ZI web server for both Etzels on the host PC:

[8] i.e. on the actual Etzel device

```
$ ziWebServer -r ./WebServer/html --server-port <RemotePort, default:8004> --
    api-level 4 --server-ip <RemoteIp_1>
$ ziWebServer -r ./WebServer/html --server-port <RemotePort, default:8004> --
    api-level 4 --server-ip <RemoteIp_2> --port <LocalPort, instead of 8006>
```

Listing B.11: Start ZI web server on the host PC for 2 remote Etzel devices

- Enable the PTP synchronisation in the web server of each Etzel:
  - Tab 'Etzel' ->Enable HF Trigger Input and HF Trigger Output
  - Tab 'Etzel' ->Disable Gain Amplification
  - Tab 'LockIn' ->Choose an input channel. Set filter and ADC bandwidth to the maximum
- Start the PTP application manually. Specify the IP address of the connected network interface and the device number[9].

```
$ ./ptp2_linux_Apalis_I210_ZI <LocalIP> <DevNo>
```

Listing B.12: Start PTP application on Etzel

- Wait at least 60 seconds to ensure synchronization lock-in
- Define one Etzel as function generator. Connect the Output of this device to the desired input of both Etzel devices.
- Enable the function generator output. E.g. via the program 'ziLink'.
- Enable the recording on both devices on the web server (preferably simultaneously) for the desired time: Tab 'Lock-In' ->'Record'

This method enables the generation of measurement data in the matlab '.mat' file format. These files contain data in a complex format, timestamps for each data sample and additional information on the device setup. For further processing the timestamps were scaled to nanoseconds and data samples were converted to absolute values.
For measurements that required the 'FPGA PPS' output, an oscilloscope was connected to the 'HF Trigger Output' of both Etzel devices.
Since the Etzel devices only have a single Ethernet port, it is not possible to connect the host PC directly when applying a 'P2P' Ethernet setup. The Linux kernel that is used on the Etzel devices supports 'Ethernet-over-USB' which acts as a bridge protocol using the RNDIS protocol. In this setup, the host PC was connected via USB to each Etzel device which acted as USB host. After configuring IPs in different subnets for the two Etzels, the connection from the host PC over an emulated Ethernet port was possible without any hassle.

[9]The device number is defined by the ZI data server. Eg. 'dev3005'

Analysis Unfortunately the measurement could not be started on both devices synchronously but had to be activated by hand on the web server of each device. The first step in evaluating the measurement data was to extract the samples that were taken during the same time on the two devices. Processing of the data was programmed in the following Matlab script files:

- Concatenate multiple data streams of one Etzel to form a single continuous measurement: 'data_concatenate.m'
- Find overlapping samples from two Etzels and cut out the data that has been measured synchronously: 'data_pre.m'

These scripts read measurement data from .mat files that exist per Etzel device and ultimately generate one single .mat file per measurement that includes the data of both Etzel devices. The subsequent analysis steps differ depending on the actual test and are therefore described in the following subsections.

B.2.1 S0 - Etzel Time Synchronization

S0	Validation of the performance of the synchronized time keeping accuracy of a single and multiple Etzel devices. **Etzel time synchronization.**
DUT	'Etzel' Lock-In Amplifier from Zurich Instruments
No. of DUT and Setup	Two Etzel devices are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Time synchronization performance in sub-second time span. Especially the linearity of the granular time increments between two precise 'second' pulses.
Measured indicator	Time stamps that are generated for each sample. The difference between two consecutive timestamps is referred to as 'time increment'. The behaviour and distribution of time increments is examined, depending on the network configuration.
Measurement device	The data server software running on each Etzel is used to gain access to the measurement timestamps. A computer connected to the same network as the DUT is used to display and analyze the data.
Additional Material	Ethernet connection between Etzels and the computer according to one of these configurations: P2P, E2E with a generic switch and E2E with a PTP switch.
Duration	At least multiple seconds
Input Signal	Sinus wave at measurable frequency that is toggled periodically
Comment	For measurement setup see figure 6.2

Table B.16: Measurement description for sub-second time update performance of two Etzel devices

Analysis The actual analysis of the measurement data in respect to sub-second time update performance was programmed in the file 'incrementEval.m' and contained the following steps:

- Select matching measurement files from the two Etzels

- Calculate the time increments by applying the matlab function diff[10] to the timestamps of each Etzel (as described in listing B.13)

```
tmp = load(filename);
timeDiffs107 = diff(tmp.dev107.ts);
timeDiffs77 = diff(tmp.dev77.ts);
```

Listing B.13: Calculate time increments in Matlab

The variable 'timeDiffs107' and 'timeDiffs77' are furthermore described as time increments and are used to describe the sub-second time update performance of the two involved Etzel devices. The data is evaluated visually with a plot in the time domain and a histogram to show the distribution of the time increments.

Additional Measurement Results Figure B.5 depicts the time increments as they were measured in a 'E2E' network configuration with the behaviour of the PTP master on the top and the PTP slave on the bottom. The distribution of both PTP master and slave reflects the results of table 7.12 in section 7.2.1. The time increment behaviour shown in figure B.5 reflects the worst case scenario for time synchronization with respect to the network topology because a switch was used that is not aware of the PTP requirements and the presence of many network devices. While the time increments of the master device differ by less than 35 ns, the slave device has a large standard deviation of 38 μs due to the fluctuating timing properties of the network path.
It is assumed that these errors are caused by the existing network traffic on the generic Ethernet switch.

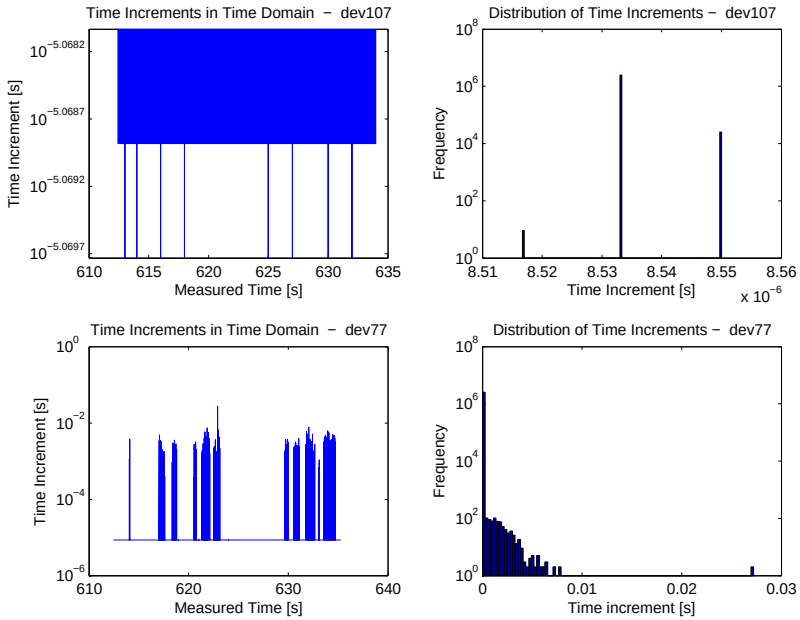


Figure B.5: Results plot - Sub-second time increments for a chosen E2E measurement

[10] Y = diff(X) calculates differences between consecutive elements of the vector X

The results of a 'P2P' measurement are depicted in figure B.6. The behaviour of the PTP master is similar to what has been observed for 'E2E' measurements in table 7.12 in section 7.2.1. In contrast to the 'E2E' measurements, the PTP slave shows very good synchronization behaviour and performs very similar to the master. Thus confirming that the synchronization works as expected.

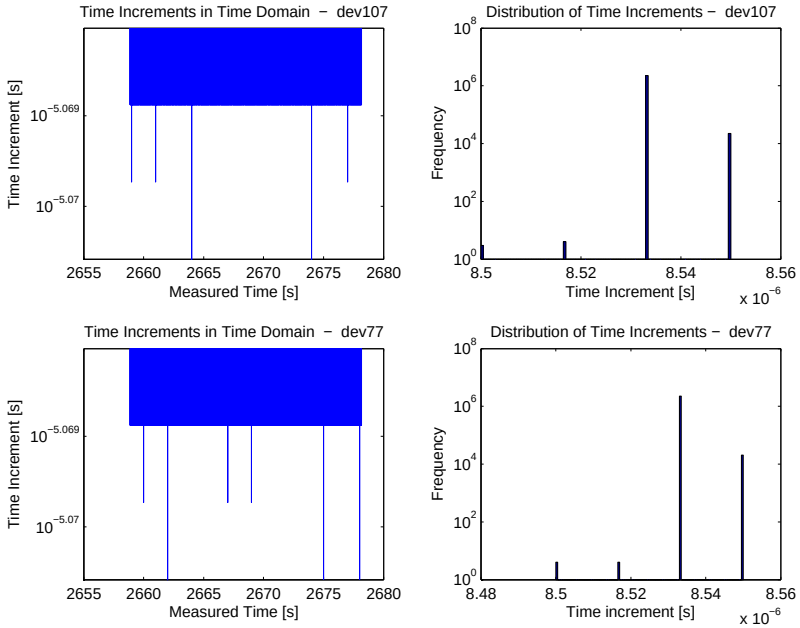


Figure B.6: Results plot - Sub-second time increments for a chosen P2P measurement

Analysis of Synchronization Accuracy

The synchronization accuracy was measured by comparing the PPS pulse from the slave device to the master device. While this pulse is already known from the unit tests, a new signal has been implemented to reflect the state of the 'second' bit in the timestamp register of the FPGA. The meaning therefore is that on each toggle the FPGA time unit has determined that (depending on its current information) one second (actually rather 0.9 s) has passed.

Variable Drift Filter Coefficient

The time unit implementation in the Etzel FPGA performs drift correction in a similar method to the PTP application. For the FPGA implementation, an exponential moving average filter has been implemented. The current output value of the filter is calculated according to: $Y_t = \alpha * X_t + (1 - \alpha) * Y_{t-1}$. ($X_t$: Input value at time t, Y_t: Output value at time t) If the 'Error from Master' is too large, the coefficient is set to 1 in order to bypass the moving average filter. In standard operation the coefficient is fixed at $\frac{1}{2^3} = 0.125$. The value for the coefficient α was chosen according to simulation results 2.3.5 and this choice shall be verified with measurements. The resulting 'FPGA PPS' depending on various coefficients (in case the offset is not too large) is described in table B.17. The test setup consisted of two Etzels in a 'P2P' configuration.

Drift filter coefficient α	Synchronization accuracy [ns]				Sample size
	Mean	Std. deviation	Min	Max	
0.125 (Reference)	-1.690	12.104	-42.685	55.11	3751
0.5	-19.967	12.758	-66.633	20.541	1803
0.25	-20.738	12.172	-56.613	13.527	1807
0.0625	-20.536	12.880	-59.619	27.555	1949
0.03125	-20.482	12.516	-66.633	13.527	1812

Table B.17: Measurement results - Variable drift filter coefficients vs. FPGA PPS

The results in table B.17 appear to contain a shift of the mean value of 20 ns. More importantly the standard deviation is very consistent for all measurements. It is therefore assumed that the influence of the actual value of the α coefficient is very small.

B.2.2 S1 - Verification of Time Increments

S1	Verification of the synchronized time keeping accuracy of two Etzel devices. **Verification of time increments.**
DUT	'Etzel' Lock-In Amplifier from Zurich Instruments
No. of DUT and Setup	Two Etzel devices are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Verification of time increments.
Measured indicator	The growth of the current time is observed. It is expected that no inconsistencies are observed (multiple occurrences of the same timestamp or decrease of the time).
Measurement device	The data server software running on each Etzel is used to gain access to the measurement timestamps. A computer connected to the same network as the DUT is used to display and analyze the data.
Additional Material	Ethernet connection between Etzels and the computer is established via a PTP-Switch.
Duration	24 hours
Input Signal	Sinus wave at measurable frequency that is toggled periodically
Comment	For measurement setup see figure 6.2

Table B.18: Measurement description for long-term consistency of time updates of two Etzel devices

Setup The setup for all measurements varied in terms of the network configuration and the connection method to the host PC. The variants are described in detail in B.2.1.

Analysis The measurement data was processed in order to calculate the time increments, as described in section B.2.1. These time increments were then evaluated to find inconsistencies in the timestamps. In order for no timestamp to appear more than once and the timestamp being monotonically increasing, the minimum time increment must be larger than 0.

```
% Find minimum time increment
minInc = min([min(timeDiffs77) min(timeDiffs107)]);
if minInc <= 0
  disp('Timestamp inconsistency found!');
else
  disp('No timestamp inconsistency found!');
end
```

Listing B.14: Verification of time increments in Matlab

B.2.3 S2 - Synchronized Measurement Events

S2	Verification of the synchronized time keeping accuracy of two Etzel devices. **Synchronized measurement events.**
DUT	'Etzel' Lock-In Amplifier from Zurich Instruments
No. of DUT and Setup	Two Etzel devices are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Accuracy of the timestamps that are generated by two separate Etzel devices that receive the same input pulse.
Measured indicator	Offset from common time base. The offset is specified as the difference between the timestamp of a PTP-Slave device to the timestamp of the PTP-Master for the same signal input event.
Measurement device	The data server software running on each Etzel is used to gain access to the measurement timestamps. A computer connected to the same network as the DUT is used to display and analyze the data.
Additional Material	Ethernet connection between Etzels and the computer is established via a PTP-Switch.
Duration	At least multiple seconds
Input Signal	Sinus wave at measurable frequency that is toggled periodically. Possibly an Etzel device could be used as signal source, thus generating exact time stamps for the signal generation.
Comment	For measurement setup see figure 6.2

Table B.19: Measurement description for the evaluation of synchronized measurement events of two Etzel devices

Setup The setup for all measurements varied in terms of the network configuration and the connection method to the host PC. The variants are described in detail in B.2.1.

Analysis In order to correlate measurement events in the data stream of two Etzels, a post-processing algorithm was implemented in Matlab. Because the measurement results strongly depend on the exact implementation of this algorithm, it is described in detail here. The main functionality of the script ('measEventPostProcessing.m') is described hereafter.

- Select matching measurement files from the two Etzels
- **Interpolation:** Since the two Etzels operate with individual clocks, the sample periods are slightly different. In order to artificially raise and align the sample rates the data

is interpolated using the PCHIP method with 2 additional interpolation steps. The following listing B.15 contains the main calculations.

```
% Interpolation of absolute data on dev107 - shortened (the same is done for
    dev77 accordingly)
step107 = tmp.dev107.ts(2)-tmp.dev107.ts(1);
step77 = tmp.dev77.ts(2)-tmp.dev77.ts(1);
if(step107 < step77)
    tsStep = step107;
else
    tsStep = step77;
end
additionalInterpSteps = 2;
tsStep = 1/2^ceil(log(1/tsStep)/log(2)+additionalInterpSteps);
if tmp.dev107.ts(end) < tmp.dev77.ts(end)
    tsInterp = tmp.dev107.ts(1):tsStep:tmp.dev107.ts(end-1);
else
    tsInterp = tmp.dev77.ts(1):tsStep:tmp.dev77.ts(end-1);
end
tsInterp107 = tsInterp';
data107Interp = interp1(tmp.dev107.ts, abs(tmp.dev107.s{1}),...
                tsInterp, 'pchip')';
% 1D data interpolation is calculated at the timestamps defined in 'tsInterp'.
    The interpolation method 'pchip' returns a piecewise cubic hermite
    interpolating polynomial. This method preserves the shape of the original
    data and maintains monotonicity.
```

Listing B.15: Interpolation of data in Matlab

- **Lowpass Filter (deprecated):** In order to suppress noise and ease the detection of measurement events, a zero-phase digital lowpass filter has been implemented. The disadvantage of this method is the decrease of flank steepness. Since we are interested in having a flank that is defined as precisely as possible, this approach has been discarded. The idea is described in listing B.16.

```
origDataAbs107 = abs(data107Interp);
dLow = designfilt('lowpassfir','DesignMethod','equiripple', ...
                  'PassbandFrequency',0.01,'StopbandFrequency',0.05, ...
                  'PassbandRipple',1,'StopbandAttenuation',60);
data107Filtered = filtfilt(dLow,origDataAbs107);
```

Listing B.16: Zero-phase lowpass filtering in Matlab (discarded)

- **Edge Detection:** An edge detection method was devised to detect the measurement events. It consists of the following steps and is described in listing B.17.
  - Define a threshold at the mean value of the data
  - Detect crossings of the data with the threshold, each crossing is assumed to be a measurement event
  - Search pairs of events (= crossings on the data of the PTP master and of the slave that were measured less than $\frac{T_S}{2} = 1$ ms apart)

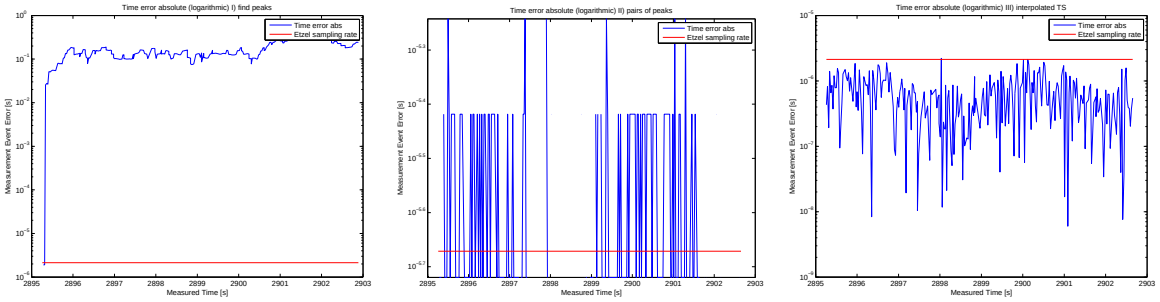
– For each pair: Try to find the exact time when the two devices amplitude crosses the threshold. The time difference between these events corresponds to the final 'measurement event error'. A linear interpolation is used to calculate the estimated timestamp. $ts_{77,adjusted} = ts_{77} + \frac{|threshold - data_{77}|}{|\frac{\partial data_{77}}{\partial ts_{77}}|}$

```
%Edge Detection
dataIntThreshold = 1/2*(mean(data107Interp) + mean(data107Interp));
k=1;
for i=1:length(data107Interp)-1
    if( data107Interp(i) < dataIntThreshold && data107Interp(i+1) >
    dataIntThreshold || data107Interp(i) > dataIntThreshold && data107Interp(i
    +1) < dataIntThreshold )
        idxThreshold107(k) = i;
        k = k+1;
    end
end
idxThreshold107 = idxThreshold107(1:k-1);
k=1;
for i=1:length(data77Interp)-1
    if( data77Interp(i) < dataIntThreshold && data77Interp(i+1) >
    dataIntThreshold || data77Interp(i) > dataIntThreshold && data77Interp(i+1)
    < dataIntThreshold )
        idxThreshold77(k) = i;
        k = k+1;
    end
end
idxThreshold77 = idxThreshold77(1:k-1);

% Search pairs of events
MAX_EVENT_DISTANCE = 0.001;
pairLocs107 = zeros(1, edgeSize);
pairLocs77 = zeros(1, edgeSize);
pairIdx = 1;
timeDiffLen = 0;
l = 1;
for k = 1:edgeSize
    % diff = a - b
    difference = tsInterp107(idxThreshold107(l)) - tsInterp77(idxThreshold77(k)
    );
    while abs(difference) > MAX_EVENT_DISTANCE && difference < 0 && l <
    edgeSize
        % Discard single peak of a
        l = l + 1;
        % Recalculate the diff
        difference = tsInterp107(idxThreshold107(l)) - tsInterp77(
    idxThreshold77(k));
    end
    if abs(difference) < MAX_EVENT_DISTANCE
        % Save pair of peaks
    pairLocs107(pairIdx) = idxThreshold107(l);
        pairLocs77(pairIdx) = idxThreshold77(k);
        pairIdx = pairIdx + 1;

        % Increment l for next round.
        l = l+1;
    else
```

```
45          % Discard single peak of b
46      end
47  end
48
49  % Timestamp interpolation
50  for i = 1:pairIdx-1
51      idx77 = pairLocs77(i);
52      idx107 = pairLocs107(i);
53      tsEdgeInterp77(i) = tsInterp77(pairLocs77(i)) + abs(dataIntThreshold -
        data77Interp(pairLocs77(i)))/abs(diff77(pairLocs77(i)));
54      tsEdgeInterp107(i) = tsInterp77(pairLocs107(i)) + abs(dataIntThreshold -
        data107Interp(pairLocs107(i)))/abs(diff107(pairLocs107(i)));
55      tsEdges(i) = tsInterp107(pairLocs107(i));
56  end
```

Listing B.17: Edge detection in Matlab

The improvement of observed measurement event error while progressing through the algorithm is shown in the following figures compared to a threshold of the Etzel sampling period at $\frac{1}{469kSps}$. Whereas the observed error is huge after the initial peak search in figure B.7a, it gets progressively better after including pair detection (figure B.7b) and ultimately timestamp interpolation in figure B.7c.



(a) Initial peak search (b) Pair detection (c) Timestamp interpolation

Figure B.7: Incremental improvement of observed time error within edge detection algorithm on the example of a 'E2E PTP' measurement

The final measurement event error as depicted in B.7c still contains some periodic frequency components. A superimposed sinusoidal waveform is easily recognized. It can be attributed to the frequency shift of the doppler effect that occurs due to the slight frequency difference of the clocks of the two Etzel devices. The observed waveform shows a period of about 4 seconds.
The effect of the timestamp interpolation is depicted in figure B.8. The plot is taken from a 'E2E PTP' measurement and represents a falling edge of a measurement event. The interpolated data is displayed with the thick blue (dev107: PTP master) and green (dev77: PTP slave) line. A measurement is detected as crossing of the data with the threshold and the last sample (before the crossing) is marked with a blue circle and blue cross. E.g. the blue cross (dev77) needs to be moved to the right in order to intersect the data at the required threshold level. The time difference that ultimately makes up the Measurement Error is the time difference between the moved edges (red circle and red cross).

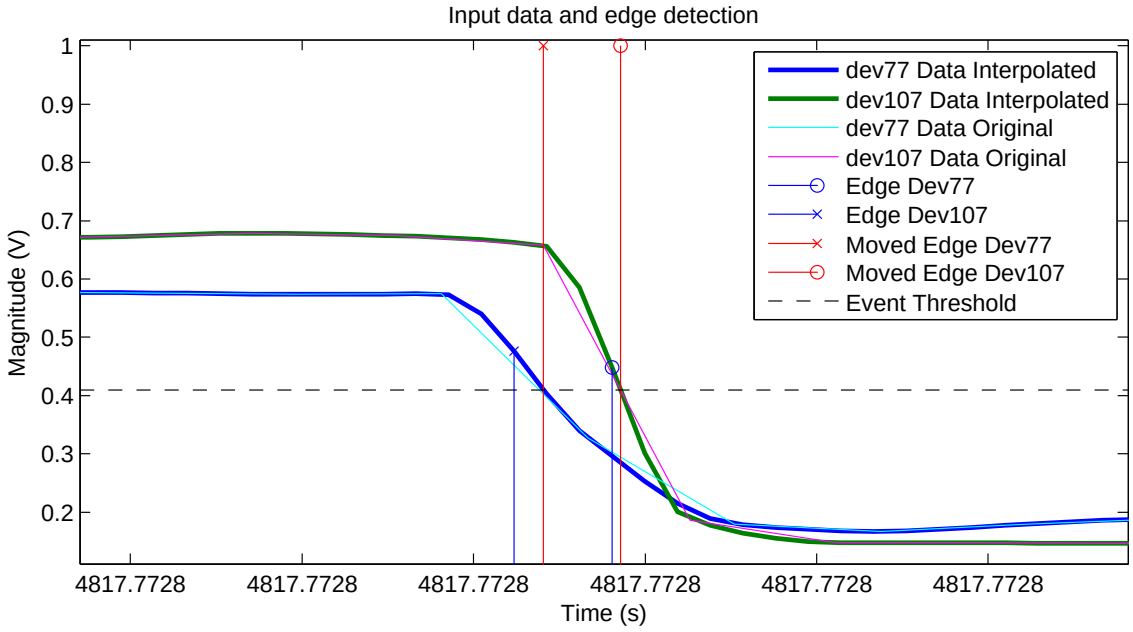


Figure B.8: Effect of timestamp interpolation on Measurement Event Error

After seeing the significant improvements to the measurement event error after each step, it becomes apparent that the edge detection algorithm is not perfectly accurate and still provides space for improvements.

Probability Calculation The evaluation of the measurement data under consideration of the statistical significance in order to answer question 7.1 has been computed in Matlab using the script 'probCalc.m'. An example for the calculation of a 'P2P' measurement is given in listing B.18. The results are described in section 7.2.3.

```
% Measurement 1: P2P
sig0 = 1000;                          % ns Target Std. Dev.
a = 0.01;                             % Allowed Probability for Error of Type I
p = 1-a;
s = 795.3;                            % ns Measured Std. Dev.
N = 318;                              % Sample Size
chi2 = ( (N-1) * s^2 ) / ( sig0^2 );  % Chi Square Statistic
x = chi2inv(1-a, N-1);                % Chi-Square Inverse CDF
```

Listing B.18: Probability calculation in Matlab

Example for Data Loss in 'E2E' Situation The effect of missing measurement data on the edge detection is shown in figure B.9. The missing rising flank is approximated with a very flat curve (blue), thus causing a large measurement event error.

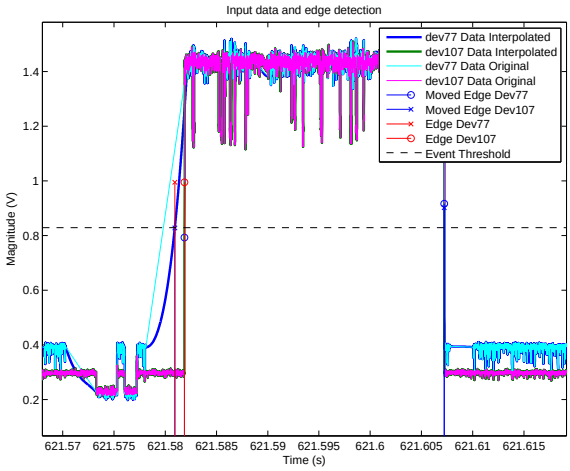


Figure B.9: Results plot - Edge detection failing due to missing measurement data of the PTP slave device (dev77, cyan and blue)

B.2.4 S3 - Long-Term Consistency of Time Increments

S1	Verification of the synchronized time keeping accuracy of two Etzel devices. **Long-term consistency of time increments.**
DUT	'Etzel' Lock-In Amplifier from Zurich Instruments
No. of DUT and Setup	Two Etzel devices are connected via Ethernet. Time synchronization via PTP has to be enabled.
Tested device characteristics	Consistency of time increments over a long time period.
Measured indicator	The growth of the current time is observed. It is expected that no inconsistencies are observed (multiple occurrences of the same timestamp or decrease of the time).
Measurement device	The data server software running on each Etzel is used to gain access to the measurement timestamps. A computer connected to the same network as the DUT is used to display and analyze the data.
Additional Material	Ethernet connection between Etzels and the computer is established via a PTP-Switch.
Duration	24 hours
Input Signal	Sinus wave at measurable frequency that is toggled periodically
Comment	For measurement setup see figure 6.2

Table B.20: Measurement description for long-term consistency of time updates of two Etzel devices

This measurement has not yet been performed.

B.2.5 S4 - Long-Term Evaluation of Synchronized Measurement Events

S3	Verification of the synchronized time keeping accuracy of two Etzel devices. **Long-term evaluation of synchronized measurement events.**
	All parameters except the duration are the same as in S2
Duration	24 hours
Comment	For measurement setup see figure 6.2

Table B.21: Measurement description for the long-term evaluation of synchronized measurement events of two Etzel devices

This measurement has not yet been performed.

B.3 Configuration of Measurement Devices

In order to correctly interpret the results, it is important to understand the configuration and setup of the devices that were used to perform the tests.

B.3.1 Oscilloscope

The oscilloscope that was utilized for unit tests is a Rhode & Schwarz RTO1014[22], which offers a bandwidth of 1 GHz on 4 channels, statistical evaluation including histogram plotting and advanced triggering options. A model RTO1024 with a bandwidth of 2 GHz was used for the system tests. Two channels were used to connect to the PPS pin of two DUT, in order to measure the time difference between the rising edge of the two signals. Thus measuring the time synchronization accuracy of these devices and performing statistical evaluation.

Acquisition For most unit tests the oscilloscope was set to the highest resolution (100 ps / 10 GSa/s), which results in a sample memory of 5 kSa. The displayed waveforms from screenshots are taken with 50 ns/div. The resulting range of time offsets that can be detected amounts to ±250 ns.
In order to be able to acquire even relatively inaccurate signals, the range had to be extended to ±500ns for U8.

Trigger Mode For most tests, a standard trigger on the rising edge over a level of 1.5 V was applied to detect a PPS pulse. The trigger source was always set to the channel that connected to the PPS pin of the PTP master device.
Test U6 required one (or two) DUT to be placed inside a climatic chamber with three cables (power, Ethernet, oscilloscope probe) being run through a small service hole on top of the chamber. The temperature control unit of the chamber causes the internal cooling unit to be switched on or off upon the detection of certain temperatures. This switching causes undesired noise signals to couple into the measurement setup, which results in the detection of erroneous PPS signals. In order to avoid this incorrect detection, the trigger mode was changed to the slew rate mode. In this mode only signals that change their amplitude from 0.5 V to 3 V within a window of 1.2 ns (±0.1 ns) are recognized as trigger events.

Statistical Functions Since the oscilloscope is set to trigger on the PPS of the first DUT, the measurement of the PPS of the second DUT automatically results in the wanted time difference. Thus the statistical function was applied to the second channel. The statistical evaluation allows for a restriction of the detected signals in relation to the horizontal and vertical domain of the current capture buffer. For most tests the horizontal restriction was set to allow 100% of the range and the vertical restriction was set to the same 1.5 V as the trigger was set. For U6 the vertical restriction was set to 3 V in order to more closely represent the slew rate trigger mode. Additionally for certain subtests[11] in U6 the horizontal restriction was set to 10% before and 10% after the 0 ns offset to reduce the influence of any noise signal.

[11] The tests requiring the use of the cooling unit (5°C and 10°C) were especially susceptible to noise generated by switching.

B.3.2 Climatic Chamber

The climatic chamber used to generate various ambient temperatures is of the type 'Heraeus Vötsch VMT 04/16'. It spans a range of -40°C to 130°C and is specified to heat up at 4.5°C/min and cool down at 3.0°C/min.[12]

B.4 Hardware Specifications

B.4.1 Etzel FPGA Crystal Oscillator

Description	**Value**
Reference frequency (nominal)	10 MHz
Reference initial accuracy	±0.5 ppm
Reference short term stability (over 30s)	<0.00005 ppm
Reference long term stability (aging)	+/- 0.4 ppm / year
Reference temperature stability at 23° +/-5° C	+/-0.03 ppm/°
Reference phase noise @ 100 Hz	-130 dBc/Hz
Reference phase noise @ 1 kHz	-140 dBc/Hz
Time to reach specification @ 25° C	60 s

Table B.22: Specifications of the high stability quartz oscillator used in the time stamping unit of Etzel

B.4.2 Ethernet Controller Crystal Oscillator

As defined in the datasheet for the 'Horizon Enterprises' HEOC33-SMD Crystal Oscillator[13].

Description	**Value**
Reference frequency (nominal)	25 MHz
Reference initial accuracy	±25 ppm
Reference short term stability (over 30s)	not specified
Reference long term stability (aging)	+/- 3 ppm / year (first year)
Reference temperature stability at 23° +/-5° C	not specified
Reference phase noise @ 100 Hz	not specified
Reference phase noise @ 1 kHz	not specified
Time to reach specification	10 ms max.

Table B.23: Specifications of the high stability quartz oscillator used for the Ethernet controller of both the Apalis development board and the Etzel

[12] In the required range of 5°C to 40°C.

[13] Available online at http://www.horizonxtal.com/02pdf/heoc33.pdf. Last visit: 21.11.2014

B.4.3 Non PTP-Capable Ethernet Switch

Description	Value
Manufacturer and Type	Zyxel GS-108B
Device Description	8-Port Desktop Gigabit Ethernet Switch
Ports	Eight RJ-45 10/100/1000 Mbps Ethernet ports with auto MDI/MDIX support. Two high-priority QoS ports and two medium-priority QoS ports.
Transmission method	Store-and-forward architecture
Operating Voltage	9 V DC, 0.85 A
Time Synchronisation	N/A

Table B.24: Specifications of the generic Ethernet switch 'Zyxel GS-108B'

B.4.4 PTP-Capable Ethernet Switch

Description	Value
Manufacturer and Type	Hirschmann RSP35
Device Description	Managed, Industrial Switch DIN Rail, fanless design. Fast Ethernet, Gigabit Uplink type. Enhanced Redundancy (PRP, Fast MRP, HSR).
Ports	11 Ports in total, thereof Fast Ethernet ports: 8 x 10/100BASE TX / RJ45; therof Uplink ports: 3 x Gigabit Ethernet SFP slots (1000 MBit).
Operating Voltage	1 x 60/110/125/220/250 VDC (48 V-320 VDC) and 110/120/220/230 VAC (88-265 VAC)
Time Synchronisation	PTPv2 TC two-step, SNTP server and client, Buffered RTC

Table B.25: Specifications of the PTP Ethernet switch 'Hirschmann RSP35'

B.5 Software Tools

- Apalis BSP v2.2 (Kernel 3.1.10-g277321a-dirty) and BSP v2.3
- Cross compilation toolchain: arm-linux-gnueabihf-g++ (crosstool-NG linaro-1.13.1-2012.09-20120921 - Linaro GCC 2012.09) 4.7.2 20120910 (prerelease)
- ZI LabOne version Linux64-14.08.26352 containing ziWebServer
- Wireshark 1.6.7
- Libreoffice Draw 3.5.7.2
- Latex: texlive, TeXstudio
- Matlab R2014a
- pdfcrop

Bibliography

[1] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2008.

[2] Sascha Montellese. Use case and technology study for synchronisation of video and audio streams. Report, 2012.

[3] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2002*, pages i–144, 2002.

[4] Hans Weibel. The Second Edition of the High Precision Clock Synchronization Protocol. *Zurich University of Applied Sciences, Institute of Embedded Systems*, 2009.

[5] List of PTP Implementations. Available online at `http://en.wikipedia.org/wiki/List_of_PTP_implementations#cite_ref-25`. Last visit: August 4th 2014.

[6] Apalis Evaluation Board. Available online at `http://developer.toradex.com/product-selector/apalis-evaluation-board`. Last visit: August 6th 2014.

[7] Linux Documentation of Timestamping. Available online at `http://lxr.free-electrons.com/source/Documentation/networking/timestamping.txt`. Last visit: August 15th 2014.

[8] Intel Ethernet Controller I210-AT. Available online at `http://ark.intel.com/products/64400/Intel-Ethernet-Controller-I210-AT`. Last visit: August 12th 2014.

[9] Intel. Datasheet Intel I210. Available online at `http://www.intel.com/content/dam/www/public/us/en/documents/datasheets/i210-ethernet-controller-datasheet.pdf`. Last visit: September 3rd 2014.

[10] Patch: Enable Auxiliary PHC Functions for the I210. Available online at `http://patchwork.ozlabs.org/patch/246552/`. Last visit: August 18th 2014.

[11] Balint Ferencz. Igb Driver with Pulse Per Second I/O for Intel I350 and I210 Series. Online available at `https://bitbucket.org/fernya/igb-pps/overview`. Last visit: August 18th 2014.

[12] Apalis Evaluation Board Schematic. Online available at `docs.toradex.com/100908-apalis-evaluation-board-v1-0-schematics.zip`. Last visit: September 3rd 2014.

[13] Apalis Evaluation Board. Available online at: http://developer.toradex.com/product-selector/apalis-evaluation-board. Last visit: 21.11.2014.

[14] Cardinal Components Inc., Applications Brief A.N.1006 - Clock Oscillator Stability - Measuring Clock Oscillator Frequency Stability.

[15] Hui Zhou, Charles Nicholls, Thomas Kunz, and Howard Schwartz. Frequency Accuracy and Stability Dependencies of Crystal Oscillators. Report, Carleton University, Systems and Computer Engineering, 2008.

[16] John R. Vig and T. R. Meeker. The aging of bulk acoustic wave resonators, filters and oscillators. In *Frequency Control, 1991., Proceedings of the 45th Annual Symposium on*, pages 77–101.

[17] S. C. Burgel, Z. Zhu, N. Haandbak, O. Frey, and A. Hierlemann. Dynamic and static impedance spectroscopy for single particle characterization in microfluidic chips. In *Micro Electro Mechanical Systems (MEMS), 2012 IEEE 25th International Conference on*, pages 1033–1036.

[18] NIST Data Significance Website. Available online at `http://www.itl.nist.gov/div898/handbook/prc/section2/prc23.htm`. Last visit: 21.11.2014.

[19] White Rabbit. Available online at `http://www.whiterabbitsolution.com/`. Last visit: September 5th 2014.

[20] LXDE project website. Available online at `http://lxde.org/`. Last visit: 21.11.2014.

[21] TIA/EIA. Commercial Building Telecommunications Cabling Standard. *TIA/EIA-568-B.1 (Revision of TIA/EIA-568-A)*, page 94, 2001.

[22] Oscilloscope RTO1014 manufacturer Website. Available online at `http://www.rohde-schwarz.com/en/product/rto-productstartpage_63493-10790.html`. Last visit: 21.11.2014.

List of Tables

List of Figures

Listings