Software and Hardware Prototypes of the IEEE 1588 Precision Time Protocol on Wireless LAN

Juha Kannisto, Timo Vanhatupa, Marko Hännikäinen, Timo D. Hämäläinen

Abstract— **IEEE 1588 is a standard for precise clock synchronization for networked measurement and control systems in LAN environment. This paper presents the design and implementation of two IEEE 1588 prototypes for Wireless LAN (WLAN). The first one is implemented using a Linux PC platform and a standard IEEE 802.11 WLAN with modifications to the network device driver. The second prototype is implemented using an embedded WLAN development board that implements the synchronization functionality using an embedded processor with Programmable Logic Device (PLD) circuits. The measured results show that 1.1 ns average clock offset can be reached on HW based implementation, while Linux PC network driver enables 660 ns with a standard WLAN. Although WLAN is an extremely difficult environment for the synchronization, the results achieved with the prototype are fully comparable to those achieved with wired LAN implementations.**

Index Terms— time synchronization, PTP, IEEE 1588, WLAN.

I. Introduction

Clock synchronization is needed in various home, office, and industrial automation applications. Synchronization allows transactions between distributed systems to be controlled on timely basis.

Basically, synchronization accuracy can be improved using two methods. First, existing hardware can be extended with a more accurate clock. Second, clocks can be synchronized to one accurate external clock. This paper focuses on the second method, which can be considered more cost efficient and scalable.

Applications in various environments are increasingly built using Local Area Network (LAN) technologies. IEEE 1588 is a standard for precise clock synchronization for networked measurement and control systems in the LAN environment [1]. The standard defines a Precision Time Protocol (PTP) developed for synchronizing independent clocks running on separate network nodes. The standard aims for sub-microsecond accuracy, while higher accuracy is targeted by hardware implementation.

Wireless LANs (WLAN) extend wired networks with easier installation and freedom of movement. They introduce unique challenges for clock synchronization due to use of wireless error prone medium and the unpredictable operation of wireless Medium Access Control (MAC) protocol. The MAC protocol of IEEE 802.11 WLAN [2] uses contention based medium access that causes variable delays for packet transmissions. With a proper design, these inaccuracies can be avoided and the synchronization accuracy significantly improved.

The accuracy of a PTP implementation depends mainly on the accuracy of the timestamps used for synchronization. Timestamping can be done at the application layer, at an intermediate protocol, such as IP layer, in a device driver layer, MAC layer, or in hardware. Different timestamping methods are presented in Figure 1. Our earlier research [3] reports the performance of software based IEEE 1588 prototype on Windows platform. It uses only the application layer approach for timestamping the synchronization messages.

This paper extends the previous work by a device driver, and hardware based timestamping methods. The paper presents two IEEE 1588 prototypes for WLAN. The first one is implemented using a Linux PC platform and IEEE 802.11b WLAN adapter, with modifications to the network driver. The second prototype is implemented using an embedded Altera Excalibur development board. It implements the synchronization functionality using an embedded processor and Programmable Logic Device (PLD) circuits. Altera prototype is referred as the PLD prototype.

The paper is organized in the following way. Section II presents the related work on time synchronization with IEEE 1588. Section III gives an overview of the IEEE 1588 standard. Next, Section IV presents both the Linux PC driver based prototype and the PLD prototype, as well as their design and implementation. Also, the PLD clock adjustment algorithm is presented. Section V defines the measurement arrangements and gives the performance results. The final section concludes the paper.

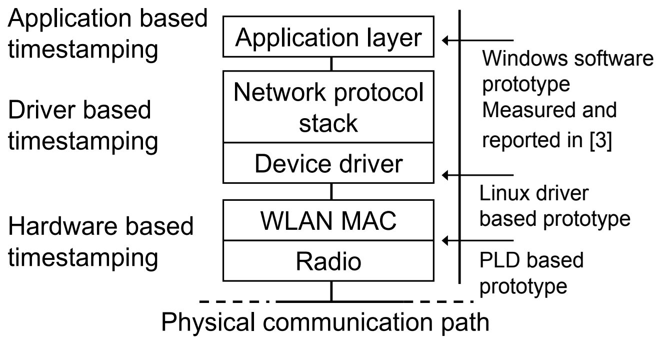
Application based timestamping
Driver based timestamping
Hardware based timestamping
Application layer
Network protocol stack
Device driver
WLAN MAC
Radio
Windows software prototype Measured and reported in [3]
Linux driver based prototype
PLD based prototype
Physical communication path

Figure 1. Timestamping methods evaluated in this research.

II. Related Research

An IEEE 1588 prototype with hardware based timestamping has been presented in [4]. It has been implemented for Ethernet LAN environment, and measured with two clocks communicating via a repeater and a switch. According to the measurements, it has 22 ns average offset with 9800 ns^2 variance on a repeater network, and 49 ns average offset with 54000 ns^2 variance on a switched network. Although good results have been achieved with the prototype, it is aimed on wired LAN. Our goal is to reach optimum accuracy on WLAN and evaluate the affect of different implementation methods in the WLAN environment.

IEEE 1588 synchronization over IEEE 802.11b with hardware based timestamping have also been researched by Pakdaman *et al.* [8]. There was not enough information available about the experiment setups and results. Thus, comparison to our prototypes was not possible.

III. IEEE 1588 Overview

PTP divides the topology of a distributed system into network segments enabling direct communication between PTP clocks. These segments, denoted as communication paths, may contain repeaters and switches connecting same LAN technology. Devices connecting communication paths, such as routers, introduce possibly asymmetric and variable delay to the communication, and are therefore treated separately.

On each communication path a single clock is selected as a master while others are slave clocks synchronizing to it. The selection is done using the *best master clock algorithm* defined in the standard.

PTP messaging between a master clock and a slave clock is presented in Figure 2. The master clock sends a *synchronization (sync) message* once in every two seconds in a default configuration. The message contains information about the clock and an estimated timestamp t_{m1} of the message transmission time. The clock information contains the identification and the accuracy of the master clock. When a slave clock receives the sync message, it stores a timestamp t_{s1} of the reception time. As it may be difficult to timestamp a sync message with an exact transmission time, the master clock can send a *follow-up message*, which contains a more precise value for the timestamp t_{m1}.

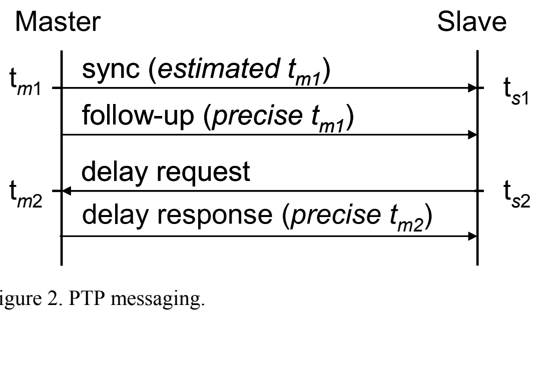
Master
Slave
t_{m1}
sync (*estimated* t_{m1})
t_{s1}
follow-up (*precise* t_{m1})
delay request
t_{m2}
t_{s2}
delay response (*precise* t_{m2})

Figure 2. PTP messaging.

A slave clock sends periodically a *delay request message* and stores its transmission time with a timestamp t_{s2}. When a master clock receives the message, it sends a *delay response message*, which contains the timestamp t_{m2} of the reception time of the corresponding request message.

The slave clock calculates the *master to slave delay* d_{ms} and the *slave to master delay* d_{sm} according to these timestamps as

$$d_{ms} = t_{s1} - t_{m1}, \tag{1}$$

$$d_{sm} = t_{m2} - t_{s2}. \tag{2}$$

The slave clock calculates the estimation of the *one way delay* d_w and the *offset from master* o_{fm} using the results from (1) and (2) as

$$d_w = \frac{d_{ms} + d_{sm}}{2}, \tag{3}$$

$$o_{fm} = d_{ms} - d_w. \tag{4}$$

The offset from the master is used to adjust the computer clock frequency and/or time. As continuous, strictly increasing time is required, the clock frequency adjustment is the only possibility.

The accuracy of the PTP system is affected by the variation of the latencies in PTP messaging. These latencies are presented in Figure 3. The delay between the timestamping point and the sending of the first message bit to the medium is called the *outbound latency* while the *inbound latency* is a corresponding delay for received messages. Inbound and outbound latencies are affected by the operation of the PTP clock host computer, mainly by the delay fluctuation in the protocol stack. The closer the timestamp is taken from the transmission of the message, the smaller the latency is, and consequently the accuracy is also better. In the ideal case, the timestamps are taken in hardware when the message is actually sent and received, and the inbound and outbound latencies become zero. Regardless of the timestamping location, the implementation is required to correct the values of reported timestamps with the estimation of the corresponding latencies. The estimation method is outside the standard scope, but averaging is advised to be used. The approach that was used in our prototypes is described in the Section IV.

Transmission latency is the delay in the physical communication path. It depends on the used transmission technology and involved devices. For example using Access Point (AP) or a switch between the PTP clocks may cause variable transmission latency. This variation should also be taken into account by the PTP implementation. Practically this requires averaging of the timestamp values used for the calculations.

IV. Prototype Implementations

The main differentiating factor between the prototypes is the method how the timestamping is implemented. The Linux PC prototype timestamping uses the device driver while the PLD prototype does the timestamping using hardware. However, the both prototypes use the following methods for implementing the synchronization.

The standard defines a follow-up message for sending a more precise estimate of the transmission time of the sync message. The follow-up message is used in both prototypes. The outbound and inbound latencies are estimated to be constant. This is a preferred method especially in the PLD prototype, since the latencies are practically zero.

Both client clocks use the timestamps for calculating how the local clock has to be adjusted to achieve the synchronization with the master clock. The clock adjustment is implemented using an algorithm that is similar to the original algorithm used with the Windows SW prototype. It is a control algorithm based on the last measured o_{fm} value and the previous value using the first derivate. Algorithm follows the o_{fm} values and distributes the adjustment over a time period to provide continuous time. The algorithm used in the Windows SW prototype is described in [3].

The performance of the prototypes is evaluated using an external reference pulse generator. It is connected to a serial port on both master and slave clocks. This is possible also in the PLD prototype since the Altera Excalibur development board contains a serial port with the same functionality as in a common PC serial port.

When the implementation receives a rising edge on a serial port Clear-To-Send (CTS) signal, it generates a reference timestamp. Reference timestamps are values of the local clock on the trigger event. Thus, they can be used for calculating the clock offsets. The generator creates approximately one pulse per second independently from the synchronization process.

A. Linux PC Prototype

Our first prototype is an implementation of the IEEE 1588 standard for Linux PC environment. Figure 4 presents the prototype architecture. It contains the PTP implementation module that has been implemented as a user mode application. The PTP implementation communicates with the peer clock using the sync, follow-up, delay request, and delay response messages. The messages are encoded and decoded in the PTP implementation, and transmitted using UDP/IP packets via the Sockets Application Programming Interface (API).

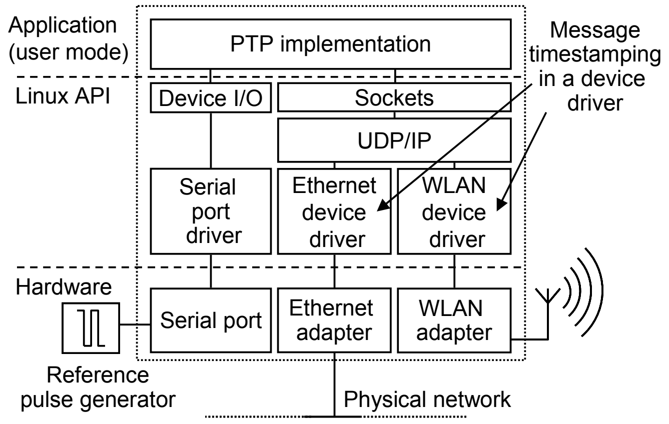

Figure 4. Architecture of the Linux PC based prototype.

In addition to the user mode application, both Ethernet and WLAN drivers have been slightly modified to generate timestamps on transmission and reception of PTP synchronization messages.

The modified network drivers store a timestamp to a temporary variable when the network adapter raises an interrupt. The timestamp is stored when the raised interrupt is *rx_frame* or *tx_done*, and the frame contains a sync, a delay request, or response message. When the PTP implementation receives the message through UDP/IP protocol stack, it reads the timestamp from the device driver. Respectively, when the application needs the transmission time of the last PTP message, it reads the stored timestamp from the device driver.

The Linux PC prototype measurement setup is presented in Figure 5. The prototype consists of a master and slave clock, a reference pulse generator, and a connecting LAN or WLAN technology.

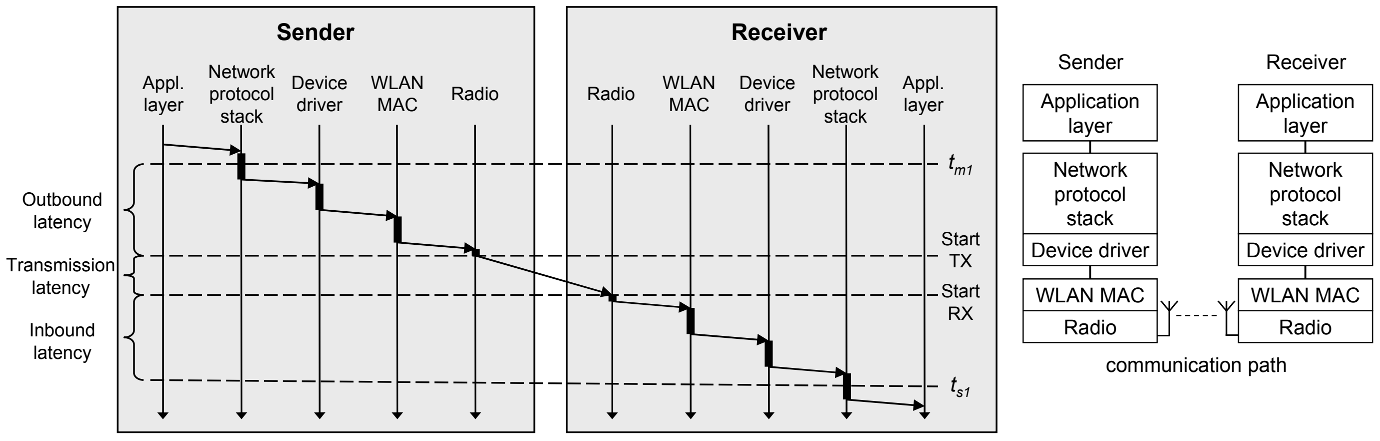

Figure 3. Latencies involved with the PTP messaging.

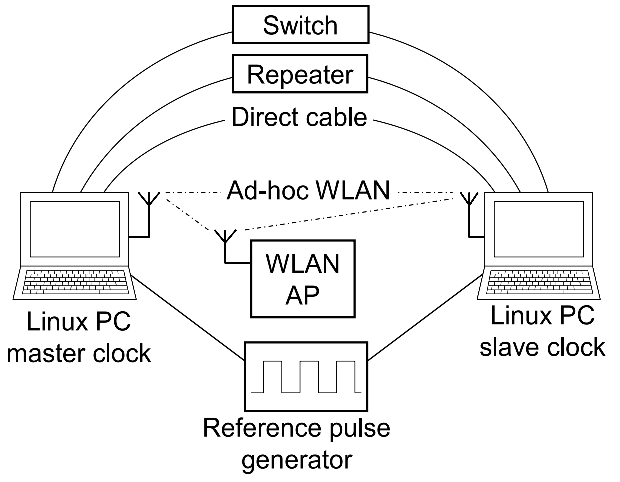

Figure 5. Linux PC prototype topology and measurement arrangements.

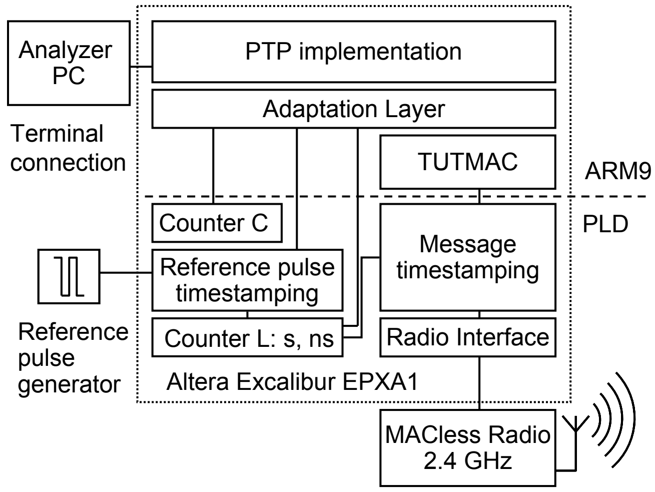

Figure 7. Architecture of the PLD prototype including the reference pulse generator and the analyser PC used for measurements.

B. *PLD prototype*

Our second prototype is implemented using an Altera Excalibur EPXA1 [5] embedded development board. The prototype is presented in Figure 6. The board contains an ARM9 processor and PLD that are connected by AMBA Bus (AHB) and dual-port memory. A 2.4 GHz MAC'less Intersil HW1151-EVAL radio transceiver [6] is connected to the development board with an expansion header. The transceiver contains only the radio implementation and on the radio interface it is fully compatible with the IEEE 802.11b standard. This corresponds to a situation where the WLAN adapter manufacturer would implement the PTP functionality in a standard WLAN adapter.

On PLD, custom hardware accelerators and interfaces to access external devices have been implemented using VHDL.

Figure 7 shows the PLD prototype functional architecture. An adaptation layer has been implemented below the PTP implementation module for necessary changes for porting the implementation to the ARM9 environment.

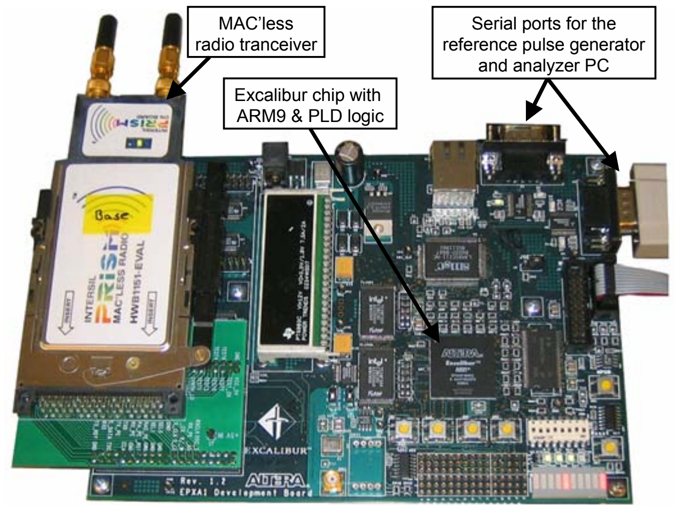
MAC'less radio tranceiver
Serial ports for the reference pulse generator and analyzer PC
Excalibur chip with ARM9 & PLD logic

Figure 6. PLD prototype implemented with Altera Excalibur EPXA1 development board.

The WLAN MAC protocol runs also in the ARM9 processor. The used MAC protocol is a test version of TUTMAC that is a research based custom WLAN protocol [7]. Although it is not an IEEE 802.11 MAC, it does not affect the results since the timestamping is done below the MAC layer. This corresponds to a situation where Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism in IEEE 802.11 MAC waits until the medium is free and sends the packet. Because timestamping is done after the MAC layer, the packet has a correct timestamp regardless of the medium access waiting time. Moreover, even a packet collision and retransmission does not cause inaccuracy to the results with the PLD prototype. The retransmitted packet is simply timestamped again instead of using the old timestamp.

The timestamping is implemented on PLD with three modules: the message timestamping, counter L, and reference pulse timestamping. Additionally, a counter C is used when adjusting the local clock frequency.

The message timestamping has a direct read access to the interface signals from the radio. The radio gives a signal about transmitted and received messages between the last preamble bit of the frame header and the first data bit using the TX_RDY signal on transmission and the RX_RDY signal on reception. When TX_RDY or RX_RDY signal is set, the module stores the value of the counter as a timestamp. This is precisely the point where timestamp should be taken according to the standard.

The counter L is the local clock of the prototype. It consists of two counters, one 32 bit counter for seconds and another 32 bit counter for nanoseconds.

Reference pulse timestamping is used for measuring the prototype performance. It receives pulses from the external reference pulse generator, and creates timestamps by reading the counter value. Timestamps are given to the PTP implementation module that sends them to an analyzer PC using a RS232 terminal connection. Reference pulses are not used by the PTP implementation for synchronization in any way. They are simply used for measuring the accuracy of the prototype implementation.

The prototype topology and test arrangements are shown in Figure 8. The reference pulse generator triggers the PLD implementation to create a reference timestamp. The analyzer PC is connected to both clocks via RS232 connection. It collects the reference timestamps and clock values to calculate the accuracy of the prototype.

V. Measurement Results

Two laptop PCs and two embedded development boards were used in the measurements. Test equipment details are presented in Table I. Both laptop PCs were equipped with 10/100 Mbit/s Ethernet NICs and 11 Mbit/s 802.11b WLAN adapters. In Linux measurements, the 500 MHz laptop was used as a master clock and the 700 MHz laptop as a slave clock. The development boards were equipped with 11 Mbit/s WLAN transceiver radio cards that do not include the MAC protocol.

In each test run, the clocks were first set to different times. The length of each test was about 15 minutes on Linux and ten minutes on PLD. At first, about 5 minutes was needed before the slave clock is stabilized, depending on the original clock offset. The reference timestamps generated before the slave clock was stabilized were ignored. The rest of the corresponding timestamps were compared and the difference of each timestamp pair was calculated. The clock offset for a single test run is defined as the average of the timestamp differences. Each test run was repeated 10 times. The average offset presented in the results for a set of test runs is the average value of the 10 test run offsets, and the variance is the corresponding variance of the test run offsets.

The long stabilization time for the slave clock is caused by the averaging used for minimizing the affect of variation in inbound, outbound, and transmission latencies, as well as the fact that the clock adjustment algorithm is implemented to adjust the clock slowly. This way the adjustment algorithm is more stable during the normal operation. Generally, the required stabilization time depends mainly on the amount of variation in the latencies. Thus, the required time is shorter for the PLD implementation. It would also be possible to shorten the stabilization time to a less than a minute. The limiting factor is the learning of the master clock frequency because it requires several sync and delay messages to be transmitted between the clocks.

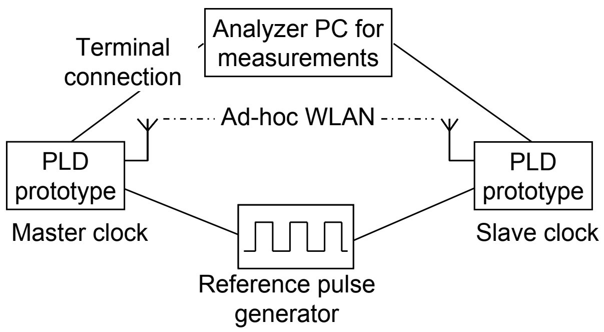

Figure 8. PLD prototype topology and measurement arrangements.

Table I. Test equipment.

Equipment	Description
Laptop PC 1	Intel Pentium III 500 MHz WinXP
Laptop PC 2	Intel Pentium III 700 MHz WinXP
Direct cable	10 Mbit/s half-duplex
Repeater	10 Mbit/s MIL-4710H half-duplex
Switch	100 Mbit/s Cisco Catalyst 3500 Series XL full-duplex
WLAN AP	11 Mbit/s Nokia A020 802.11b WLAN AP
WLAN adapter	11 Mbit/s Nokia C110 802.11b
Development board	Altera Excalibur EPXA1 Development board
Radio on PLD	Intersil PRISM MAC'less Radio

Five transmission technologies were used to analyze Linux PC implementation accuracy. The used technologies were direct Ethernet cable, repeater (hub) network, switched network, ad-hoc WLAN, and WLAN AP. The PLD implementation accuracy was analyzed on ad-hoc WLAN topology.

The achieved accuracy of the Linux PC prototype with each technology was evaluated according to measurement arrangements presented in Figure 5. The results of each test run are presented in Figure 9. According to the measurements, the average offset reached using a direct cable connection was 1.8 µs while the variance was 0.7 µs^2. A repeater network is almost as accurate as the direct cable connection with 1.9 µs average offset and 0.1 µs^2 variance. A switched network enables 0.95 µs average offset with 0.3 µs^2 variance.

In WLAN, the ad-hoc operation enables 0.66 µs average offset with 0.2 µs^2 variance. The WLAN AP enables the average offset of 4.6 µs with 2.5 µs^2 variance.

Summary of the results is presented in Figure 10. For comparison, it also contains the previous Windows software measurements reported in [3]. Although different operating systems were used, both setups were measured using the same laptop PCs. The accuracy of all setups has increased several microseconds with the driver based timestamping, except for

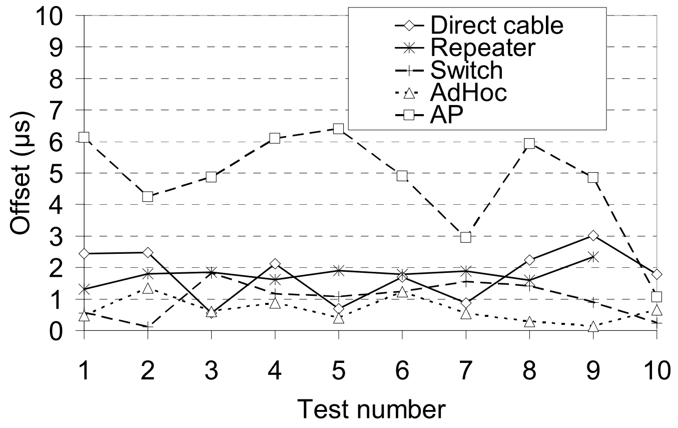

Figure 9 Achieved offsets for the Linux PC prototype with different LAN technologies.

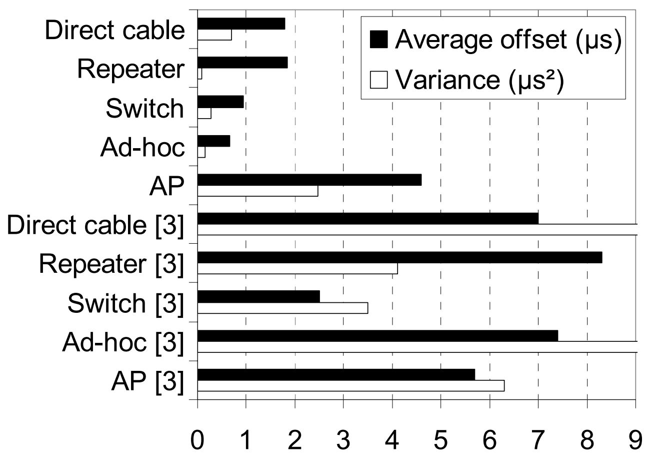

Figure 10 Summary of the Linux based prototype measurements compared to measurements in [3].

the AP setup. It shows that when AP is involved, most errors come from the delay variation in AP instead of the outbound and inbound latencies.

The PLD implementation accuracy was evaluated according to the test arrangement shown in Figure 8. It uses an ad-hoc WLAN between the master and slave clock. The result is shown in Figure 11. According to the measurements the reached average offset was 1.1 ns with 3.1 ns^2 variance. The synchronization accuracy is almost three decades better than with Linux PC implementation. The extremely high accuracy of this prototype results mainly from reducing the outbound and inbound latencies using hardware for the timestamping. The use of ad-hoc topology also reduces the variation in the transmission latency, since there are no additional devices involved.

Usage of WLAN AP between the clocks would increase the variation in the transmission latency and decrease the accuracy. However, in [3] we presented a method called *external echo* that was used to completely remove the inaccuracy caused by AP. This method is based on the normal operation of an IEEE 802.11 WLAN AP. When the master clock sends a message to the slave, it is first transmitted to AP and AP transmits it to the slave. However, each transmission from the AP to the slave can also be received by the master clock because both use the same IP multicast address. The master clock takes a precise timestamp for the sync message from the reception of the frame send by AP. The precise timestamp is transmitted with the follow-up message. Combining the hardware implemented timestamping and the external echo method can be used to implement nanosecond

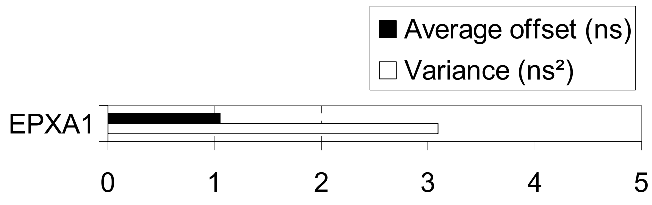

Figure 11 The PLD based prototype results.

accuracy also in the WLAN AP topology.

VI. Conclusions

Two IEEE 1588 prototypes for WLAN were presented in this paper. The first one was implemented using a Linux PC platform and timestamps the synchronization messages in the device driver. The second prototype was implemented using an embedded development board and hardware based timestamping. The accuracy of the prototypes was measured in different setups and compared to previous work with software based timestamping prototype.

According to the results, the microsecond accuracy achieved by the network device driver based timestamping can be further increased to nanosecond level by implementing the timestamping with hardware. Timestamping below the MAC protocol effectively reduces the measurement errors caused by the master and slave clock implementation even in the WLAN environment and significantly improves the synchronization accuracy. Only significant error source left is the network devices between the clocks.

Combining the hardware based timestamping and a method for removing the accuracies caused by AP is left for the further development. Nanosecond scale accuracy enables the implementation of time critical applications also using WLAN as a transmission medium. Since the time critical applications are increasingly implemented using wireless technologies, it is essential that also WLAN device manufacturers would implement the PTP functionality in their products.

References

[1] IEEE std. 1588-2002 “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems”, 2002

[2] IEEE Std 802.11-1997 “Wireless Lan Medium Access Control (MAC) And Physical Layer (PHY) Specifications”, 1997

[3] Juha Kannisto, Timo Vanhatupa, Marko Hännikäinen, Timo D. Hämäläinen, ”Precision Time Protocol Prototype on Wireless LAN”, International Conference on Telecommunications (ICT 2004), August 1-6, 2004, Fortaleza, Ceara, Brazil, pp. 1236 - 1245.

[4] Eidson, J.C.; Kang Lee, “Sharing a common sense of time”, Instrumentation & Measurement Magazine, IEEE , Volume 6, Issue 1, March 2003, pp. 26 – 32

[5] Altera Corporation, [Online], Available: http://www.altera.com

[6] Intersil, 2.4GHz 11Mbps MACless DSSS Radio HW1151-EVAL, [Online], Available: http://www.intersil.com

[7] Petri Kukkala, Väinö Helminen, Marko Hännikäinen, Timo Hämäläinen, ”UML 2.0 Implementation of an Embedded WLAN Protocol”, IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC’2004), 2004

[8] A. Pakdaman, J. Eidson, T. Cooklev, "IEEE 1588 over IEEE 802.11b", IEEE 802.11 interim meeting, Berlin, Germany, Sept. 2004, [Online], Available: http://www.ieee802.org/11/DocFiles/04/11-04-1080-00-0wng-ieee-1588-over-ieee-802-11.ppt