ISPCS 2009 International IEEE Symposium on Precision Clock
Synchronization for Measurement, Control and Communication
Brescia, Italy, October 12-16, 2009

High Accurate Timestamping by Phase and Frequency Estimation

Reinhard Exel and Patrick Loschmidt
Austrian Academy of Sciences
Institute for Integrated Sensor Systems
Viktor Kaplan Straße 2
A-2700 Wr. Neustadt
{Reinhard.Exel, Patrick.Loschmidt}@OEAW.ac.at

***Abstract*—When using network synchronization protocols like IEEE 1588 or NTP, a common approach to increase the performance is to add hardware support for timestamping the essential synchronization messages. Further, timestamping accuracy and the stability of the local oscillator of a network synchronized node are the two main influence factors for high accuracy clock synchronization. While the latter is subject to manufacturing technologies, the first is mainly given by the design of the packet timestamper of the node. While common solutions use sampling in minimized intervals to achieve high accuracy, this paper lists other approaches and proposes a method based on phase/frequency estimation. While the additional hardware effort is rather low, the presented method allows for high accuracy. The performance of the developed design is equivalent to a 5.5 GHz sampling clock, but still can be implemented even in low cost digital logic devices.**

I. Introduction

The presence of a common notion of time established by clock synchronization is an important service in any distributed network as it enables the system to generate a consistent view and perform coordinated actions. Depending on the required accuracy special measures have to be taken to achieve sufficient synchronization. These include dedicated clock synchronization protocols like IEEE 1588 or NTP and hardware assistance to accurately use and keep the timescale. These protocols have in common that the synchronization is performed by a regular exchange of messages and a local oscillator is used to maintain a continuous timescale between the messages. Apart from the stability of the oscillator, the quality of the timestamp taken at the transmission and reception of the synchronization messages is a limiting parameter for the attainable accuracy. Using accurate timestamping hardware in every network element allows to calculate the individual residence times and enables the system to derive the total end-to-end delay.

In general, timestamping methods can be categorized by their source:

- Software: This approach takes the timestamp at the instant when the sync message is visible to the operating system or application.
- Hardware: Dedicated hardware records the arrival of a sync message, typically at the interface of the MAC to the physical layer.

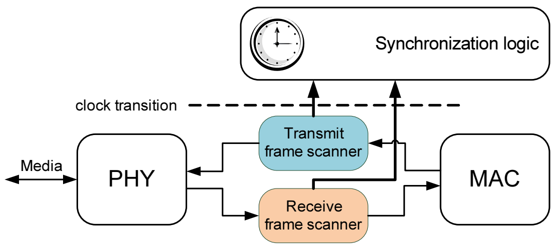

Fig. 1. Clock domains in an asynchronous communication system

While the first method is versatile and does not require any special hardware, the processing within the lower network layers before the actual timestamping point introduces jitter. For many applications the accuracy in the range of milli- down to microseconds is sufficient [1]. Hardware timestamping on the other hand requires special hardware, but is able to achieve timestamping accuracies several orders of magnitude better than pure software solutions. In this paper we want to focus on hardware timestamping with an example application for an IEEE 1588 synchronization system over 100 Base-TX Ethernet.

The paper is structured as follows: Section II gives an overview of timestamping in a general asynchronous network like Ethernet and different approaches to timestamping. Section III describes phase and frequency estimation techniques and their application for timestamping. In section IV we present preliminary results and the final section concludes this paper.

II. Timestamping in Packet-Oriented Networks

A common timestamping approach in a packet-oriented, asynchronous network (like Ethernet) uses a frame scanner on the interface between the physical and the MAC layer monitoring the content of received and transmitted frames as depicted in figure 1. When it detects a frame, which should be timestamped, the frame scanner asserts the timestamp signal $\mathcal{S}_{\mathrm{TS}}$ in order to trigger the storage of a timestamp. The assertion of this signal is observed by the synchronization logic which copies the current time together with some frame identification information into a memory block.

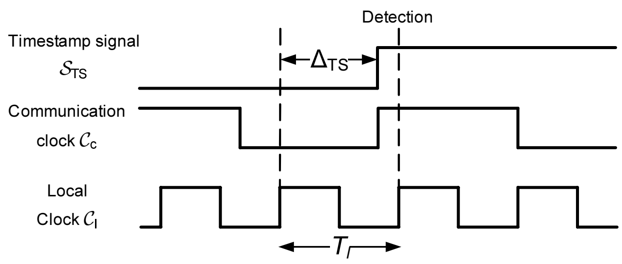
Detection
Timestamp signal
$\mathcal{S}_{TS}$
Δ_{TS}
Communication clock $\mathcal{C}_c$
Local Clock $\mathcal{C}_l$
T_l

Fig. 2. Timestamping a frame from a different clock domain

In an asynchronous system (like Ethernet) the receive clock $\mathcal{C}_\mathrm{r}$ with the period T_r, the transmit clock $\mathcal{C}_\mathrm{t}$ (period T_t) and the local clock $\mathcal{C}_\mathrm{l}$ (period T_l) are in general not equal. The receive clock is typically recovered from the received data, whereas the transmit clock and local calculation clock are sourced from local oscillators. Given that the receive and the transmit frame scanner operate at $\mathcal{C}_\mathrm{r}$ and $\mathcal{C}_\mathrm{t}$ respectively, both clocks can be treated equally (asynchronous) to $\mathcal{C}_\mathrm{l}$. Therefore, they are subsequently called communication clocks $\mathcal{C}_\mathrm{c}$ to avoid always mentioning both. When one of the frame scanners assert the timestamp signal $\mathcal{S}_\mathrm{TS}$, the signal has to be passed to the local clock domain.

A transition synchronous to one of the communication clocks sampled with the local clock causes a equally distributed jitter Δ_TS with a width of one clock period ($0 \leq \Delta_\mathrm{TS} < T_\mathrm{l}$), which results in a variance $\sigma^2_\mathrm{TS} = \frac{T_\mathrm{l}^2}{12}$. The value of Δ_TS is unknown since the timestamp signal can be asserted at any time and can just be recognized at a rising edge of $\mathcal{C}_\mathrm{l}$ as depicted in figure 2. One clock transition can be avoided by setting T_l to a multiple of T_t, since the latter has to be sourced by the node anyway. Still one jitter source remains, namely the transition between $\mathcal{C}_\mathrm{r}$ and $\mathcal{C}_\mathrm{l}$. Obviously the jitter can be decreased by increasing the frequency $1/T_\mathrm{l}$. Nevertheless, complex hardware designs limit this approach to relative low frequencies (e. g. the Syn1588® IEEE 1588 IP-core is limited to around 100 MHz even on modern FPGAs).

In order to enhance timestamping accuracy, different approaches to reduce the jitter Δ_TS can be used. These can be grouped in the following two subcategories:

- Single-Shot methods measure the time between the assertion of the timestamp signal $\mathcal{S}_\mathrm{TS}$ with respect to the rising edge of $\mathcal{C}_\mathrm{l}$.
- Phase estimating methods use the fact that the timestamp signal is asserted synchronous to $\mathcal{C}_\mathrm{c}$ and therefore it can be used to estimate the current phase of $\mathcal{S}_\mathrm{TS}$ with respect to the rising edge of the local clock, $\mathcal{C}_\mathrm{l}$.

Both methods have in common that they measure or estimate the phase Δ_TS with respect to $\mathcal{C}_\mathrm{l}$ and therefore it can be better expressed by a relative phase offset $\delta_\mathrm{TS} = \frac{\Delta_\mathrm{TS}}{T_\mathrm{l}}$. The relative phase offset δ_TS is useful in applications running an Adder Based Clock (ABC) like common IEEE 1588 cores. Assume that the ABC is increased every T_l by the increment I, the value of the timestamp is calculated by the last value of the ABC plus δ_TS times I.

An overview of available methods for measuring time intervals can be found in [2]. This paper focuses on methods which can be implemented fully into modern digital logic devices without the necessity of external components. Thus, all approaches using external ADCs or mixers are not considered, although they can deliver comparable or even better performance. In the following only the most relevant techniques for use in an FPGA are tackled.

A. Single-Shot Methods

Single-shot methods measure δ_TS directly, that is how long after the rising edge of the local clock the timestamp signal $\mathcal{S}_{TS}$ has been asserted. For that purpose a clock cycle T_l is divided into $n \in \mathbb{N}^+$ equally spaced fractions, which reduce the timestamping variance to $\sigma^2_\mathrm{TS} = \frac{T_\mathrm{l}^2}{12n^2}$.

1) High-Speed Counter: An approach presented by [3] divides T_l by a short high frequency counter with a period $T_\mathrm{h} = T_\mathrm{l}/n$. The counter is reset at every rising edge of the local clock and its value is frozen when $\mathcal{S}_\mathrm{TS}$ is high. Therefore, the counter value divided by n describes the relative phase offset δ_TS. Since the period of the local clock is exactly a multiple of T_h the clock transition between these two clocks can be designed without any additional jitter and consequently the timestamping variance reduces by n^2. Hence, the result is similar to a design completely clocked with $1/T_\mathrm{h}$ with the advantage that only a few logic elements have to run at a high frequency. Further, using double data rate (DDR) registers improves the resolution by an additional factor of 2.

2) Phase Shifted Clocks: This is another option to partition T_l. Assume that $n-1$ additional clocks are generated which are phase-shifted by $2\pi/k$ with $k = 0 \ldots n-1$ with respect to $\mathcal{C}_\mathrm{l}$. The timestamp signal is registered into n registers which generate a thermometer code if $\mathcal{S}_\mathrm{TS}$ is asserted which is converted to a binary code and used in the same manner as the high speed counter. Since $\mathcal{S}_\mathrm{TS}$ drives n registers, special care has to be taken that the clock at the registers has the designed phase-shift (i. e. the registers are timely equally spaced), since otherwise the thermometer code renders non-linear. This effect and the number of output clocks per PLL limit n to value of about 10 in state-of-the-art FPGA devices (e. g. Altera Stratix III family).

3) Tapped Delay Lines: Tapped Delay Lines (TDLs) are a common approach for digitizing times with sub-nanosecond accuracy. The basic configuration of a TDL consists of a serial chain of n latches having a delay τ_1, a non-inverting buffer with delay $\tau_2 < \tau_1$, and an output logic as described in [4]. The signal to be timestamped is then fed through all latches which freeze its current state at a rising edge. The resulting thermometer code is evaluated after the next clock edge. Nevertheless, it has to be considered, that such a design is asynchronous logic and therefore the delays τ_1 and τ_2 are not only placement, but also temperature dependent. The linearity of a TDL may be compromised by these effects and special calibration logic may be required. The possible accuracy is dependent on the intrinsic switching speed of the latches, which is typically in the range of 100 ps [5].

III. Phase estimating methods

Phase estimating methods do not measure δ_{TS} directly, but rather estimate δ_{TS} using the fact that $\mathcal{S}_{\text{TS}}$ is asserted synchronous to the communication clock. This relatively new approach for high accurate time interval measurements has been investigated by [6]. Their measurement method is based on sampling a 10 MHz atomic clock with an ADC driven by the communication clock and perform a phase estimation based on an 1024 point FFT. While the results show a very low timestamping standard deviation of about 10 ps, this approach requires an ADC per timestamper and an atomic clock to achieve this performance. Since this is rather impractical for timestamping, this paper focuses on pure digital methods implementable without additional hardware.

There are several requirements for this kind of phase estimation: First of all, the rising edge of the local clock should occur equally distributed within the clock cycles of T_{c} averaged over a given time span (i. e. the rising edge of the local clock should cover all phases of the communication clock with equal probability). This includes that T_{c} must not match T_{l} or a multiple thereof because in this case the rising edge would always coincide with a certain phase of T_{c} and the necessary averaging time span would be infinite.

We set the nominal period T_{l} by $n(1+\beta)T_{\text{l}} = T_{\text{c}}$ with the design parameter $0 < \beta \ll 1$ as a relative frequency offset factor and n as the nominal oversampling factor. For the sake of simplicity, we define $\epsilon = \frac{\beta}{1+\beta}$, since given that the nominal periods match the real periods, cycle slips (when the rising edges of two clock pass each other) will happen after every n/ϵ local clock periods. This implies that $\mathcal{C}_{\text{c}}$ has been sampled at $1/\epsilon$ different phase points over one cycle slip period and the maximal attainable accuracy is nominal ϵT_{l}. Selecting a very small ϵ results in a small frequency offset and great resolution, but this may result in $T_{\text{c}} \le nT_{\text{l}}$ which means that there are no cycle slips at all or even reverse cycle slips (combined with possible data loss).

Furthermore, small frequency differences are unusable since the rising edge instance is only equally distributed within T_{c} over a long averaging window and for short averaging windows the leakage effect becomes dominant.

A. Multiple Timestamps per Frame

Timestamping a frame m times is one option to estimate the phase at the timestamping event based on the assumption that δ_{TS} averages to 0.5. Given that the timestamps are centred before and after the assertion of $\mathcal{S}_{\text{TS}}$ (i. e. $\frac{m-1}{2}$ timestamps before and after the event) and that the ABC's increment is not altered during the timestamping period, the final timestamp TS is calculated by a weighted average over the timestamps TS_i following

$$\text{TS} = \sum_{i=-(m-1)/2}^{(m-1)/2} \alpha_i \text{TS}_i, \; \sum_i \alpha_i = 1. \tag{1}$$

This FIR filter can be simplified to a window integrator with $\alpha_i = 1/m$ with some limitations, namely leakage effects. The timestamping window should cover one cycle slip period or a multiple thereof to get the timestamps equally distributed over one T_{c} period. Since the cycle slip period is dependent on the current frequency offset, it varies with the oscillator drift between the communication, and local clock. One solution to this problem is to adjust m to cover always a multiple of the cycle slip periods or by capturing a big number of such periods and using a windowing function to minimize leakage effects. The leakage can also be reduced by selecting a rather large ϵ with the drawback of reduced resolution.

In the optimal case the resulting timestamping variance reduces to $\sigma^2_{\text{TS}} = \frac{T_{\text{l}}^2}{12m}$. For a typical IEEE 1588 frame with about 80 bytes frame length and a nibble-wide interface, $m = 160$ timestamps can be stored. Given that $T_{\text{l}} = 10$ ns the standard deviation resolves to 228 ps.

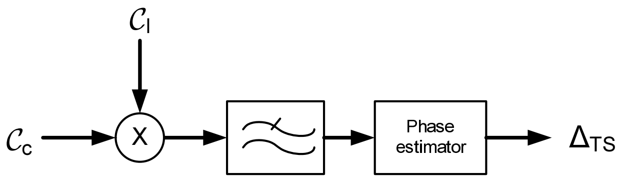

Fig. 3. Direct phase estimation by mixing

B. Digital Phase Estimation

The phase of the communication clock can be directly estimated by phase detectors. Such a detector is based on mixers which shift the spectrum of the clock to a low frequency as shown in figure 3. The output of the mixer is low-pass filtered to remove aliases at multiples of the input frequency and is conducted into a phase estimator. Given that the duty cycle of the clock input signal is constant, the low frequency part of the down mixed signal then is a measure for the phase difference at the inputs. Nevertheless, real filters with a low bandwidth have significant group delay, which has to be taken into account for the read-out. In order to allow for a digital implementation the mixer can be replaced by the output of an XOR gate filtered in the same way. A further solution would be to use an external analogue anti-aliasing filter in combination with an ADC and only perform the second filtering digitally.

A pure digital solution without requiring external parts is possible, but under-sampling occurs. As the XORed signal is not band-limited, sampling it results in alias frequencies, that can dominate the signal (e. g. if the '1 bit' sampler is sourced from a clock correlated with $\mathcal{C}_{\text{l}}$). One feasible solution is to sample the mixed signal by a clock odd to both inputs and apply the sampled signal to a low-pass filter with very low relative bandwidth. Such filters are typically IIR type since comparable FIR filters would need a big number of filter taps. IIR filters on the other hand have a frequency dependent group delay, which means that the frequency offset between $\mathcal{C}_{\text{l}}$ and $\mathcal{C}_{\text{c}}$ must be estimated in order to compensate for the filter's group delay.

C. Combined Phase/Frequency Estimation

Rather than estimating δ_{TS} directly, it is also possible to estimate the phase by its derivative, the frequency offset, together with a reference point. The principle of phase estimation by frequency estimation can be implemented as follows: Whenever the communication clock is phase aligned to the local clock (that is when a cycle slip occurs), the phase estimation $\hat{\delta}_{\mathrm{TS}}$ is set to zero. In every subsequent clock cycle $\hat{\delta}_{\mathrm{TS}}$ is incremented by the estimated inverse cycle slip period $\hat{\epsilon}/n$ (i. e. $\hat{\delta}_{\mathrm{TS}}$ is the integral of $\hat{\epsilon}/n$ over one cycle slip period). Given that the frequency is stable in the averaging interval, $\hat{\delta}_{\mathrm{TS}}$ ideally would reach 1.0 at the next cycle slip as depicted in figure 4 for $n = 1$. For the reason of better visibility β was chosen to be 0.2.

The accurate detection of the cycle slip instant is critical for the start of the integration. To make the method independent of the communication clock duty cycle, a derived clock $\mathcal{C}_{\mathrm{d}}$ with the period $2T_{\mathrm{c}}$ is generated digitally. $\mathcal{C}_{\mathrm{d}}$ is fed into a shift register with $5 + n$ taps clocked with the local clock. Further, this clock is used for cycle slip checking and performing edge detection. While the first two shift taps are used for buffering, the middle taps $(2 + n$ down to 2) are used for cycle slip detection. If all $n + 1$ middle taps contain the same binary value, a cycle slip must have happened. The last two taps (1 down to 0) are used to detect rising and falling edges of $\mathcal{C}_{\mathrm{d}}$ at which the buffered timestamp signal $\mathcal{S}_{\mathrm{TS}}$ is checked for a high level. Since the timestamp signal is clocked into a single register the placement is not critical. Nevertheless, as for all other methods, the small (possible asymmetric) delay between receive and transmit estimator inside the IC should be measured once.

The relative frequency offset $\hat{\epsilon}$ can be calculated by monitoring the number of rising edges of the $\mathcal{C}_{\mathrm{c}}$ with respect to the $\mathcal{C}_{\mathrm{l}}$. In average every $n/\hat{\epsilon}$ local clock cycles a cycle slip will occur, which results in a missing rising edge with respect to the local clock. In order to have a continuous value of $\hat{\epsilon}$, a low-pass filter has to be applied. The bandwidth of the filter must be narrow enough to track frequency changes of the oscillator while removing the cycle slip frequency. In general, IIR filters which have poles close to 1 are ideally suited for this application. Alternatively, the frequency offset $\hat{\epsilon}$ can be calculated by the cycle slip rate $n/\hat{\epsilon}$, but this requires a division block which consumes a significant amount of logic resources. In any case, the calculation of the final timestamp involves summing up the ABC's last value plus $\hat{\delta}_{\mathrm{TS}}$ times the increment I.

IV. Results / Measurement

The effectiveness of phase estimation methods is shown by an FPGA implementation of the phase/frequency estimation method using a Altera Stratix II GX evaluation board. In order to simulate timestamping for a typical 100 Base-TX connection with the media independent interface (MII) connecting the PHY and the MAC, a communication clock of $1/T_{\mathrm{c}} = 25\,\mathrm{MHz}$ was chosen. The local clock $1/T_{\mathrm{l}}$ was selected to be 50.2272727 MHz, i. e. with an oversampling factor of $n = 2$

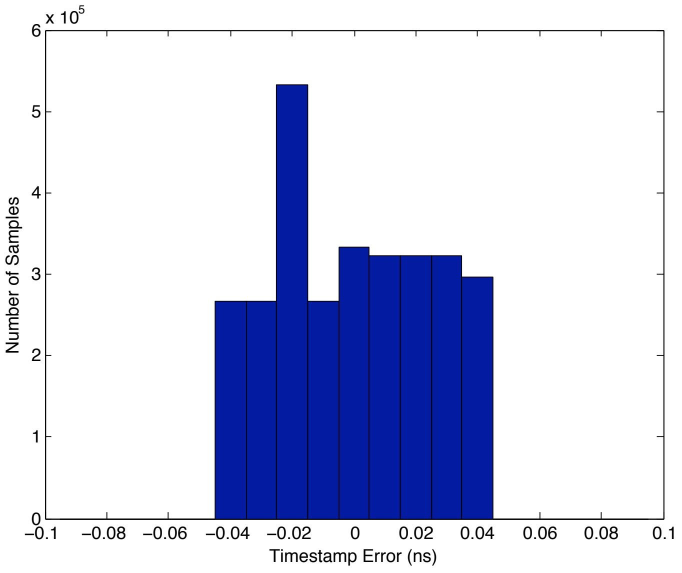

Fig. 5. Measured histogram of the timestamping error for the combined phase/frequency estimation method

and $\beta = 0.00454545$ and $\epsilon = 0.00452489$. Both clocks are generated via two PLLs from a single 100 MHz oscillator. Obviously, in real applications the communication clock and the local clock do not have a fixed frequency relationship (at least for the receive side). Nevertheless, this has to be done in the test setup to study the performance without disturbing effects caused by oscillator drift.

Every 250th clock cycle (equals 10 µs, arbitrarily chosen) the timestamp signal $\mathcal{S}_{\mathrm{TS}}$ is asserted for a single clock cycle which triggers the timestamper. The time source for the timestamper is an ABC with fixed increment and an 8 bit subnanosecond resolution. The 64 bit timestamps calculated by the estimation logic are transferred to a PC where the ideal time is subtracted from the measured timestamps. Furthermore, for long-term measurements the very slight drift of the ABC caused by the limited granularity of the ABC's increment is compensated in the PC as well.

In case of a simple clock transition (without taking δ_{TS} into account), we expect the timestamps to be equally distributed from $-T_{\mathrm{l}}/2$ to $+T_{\mathrm{l}}/2$. In fact the implementation matches exactly the expected equal distribution. If δ_{TS} is taken into account, the standard deviation of the timestamps improves significantly from 5.76 ns to 26 ps. The timestamp error is almost equally distributed with slight imperfections around the centre as shown in figure 5.

The theoretical resolution for the given parameters is $\epsilon\, T_{\mathrm{l}}$ = 90.09 ps and therefore $\sigma_{\mathrm{TS}} = 26\,\mathrm{ps}$. Given the (cheap) layout of PLLs in FPGA devices, it is surprising that these PLLs can produce such low-jitter clocks which enable such accurate measurements. In fact this result was only possible with one of four FPGAs, whereas the others generated timestamps with $\sigma_{\mathrm{TS}} = 52\,\mathrm{ps}$ instead of 26 ps. The higher standard deviation was caused by some timestamps about 130 ps away from the main equal distribution. The reason for this can be found in

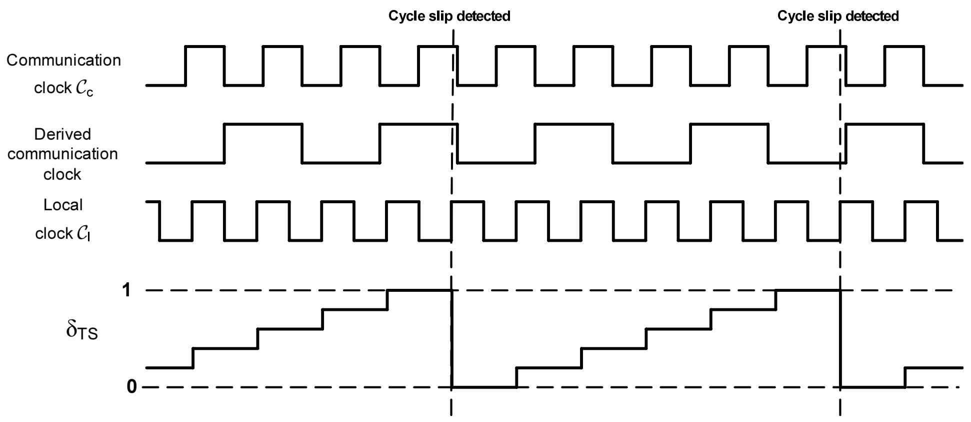
Cycle slip detected
Cycle slip detected
Communication clock $\mathcal{C}_c$
Derived communication clock
Local clock $\mathcal{C}_l$
1
δ_{TS}
0

Fig. 4. Phase estimation between cycle slips

physical effects mainly the phase noise of the clock source. Random clock jitter causes cycle slips to be detected too early or too late which in consequence shifts the timestamps as well. In such cases clocks with better stability are just one possible solution, but also combining phase/frequency estimation with multiple timestamps per frame can be considered.

Using the same clock source for $\mathcal{C}_l$ and $\mathcal{C}_c$ verifies the operability and accuracy of the timestamping scheme, but for practical applications the two clocks are sourced from different oscillators. As the cycle slip detection relies on a single clock edge, a clock source with low phase noise is required. Since the clock output of Ethernet PHYs is just designed to drive digital logic, the quality of the clock outputs is limited (as measured for the SMSC LAN8700 PHY) and it might be required to feed the communication clocks of the PHY through a PLL to stabilize them. Another option is to estimate the cycle slip instant over several periods if the clock stability over multiple cycle slip periods is guaranteed.

An application for the accurate timestamping method is stability measurement. Given that one of the clocks is sourced by a stable reference oscillator, the stability of the other oscillator can be measured and characteristic parameters like the Allan variance can be calculated. This was tested with a 10 MHz Rubidium clock versus the local 100 MHz oscillator with 50 ppm advertised stability. The resulting graph is shown in figure 6.

In IEEE 1588 networks this measurement can be used for oscillator classification in the nodes. Assume that multiple nodes are directly connected to a single switch. Typically a switch drives all its PHYs by the same transmit clock and therefore the receiving PHYs in the nodes recover the switch's clock from the received data. In this topology all nodes can be supplied with the clock of the switch. If this clock is used for the reference clock, all nodes can measure their frequency departure with respect to the switch's clock. In consequence it is possible to calculate the frequency departure between every node. This information can be useful for the master election process, for a dynamic adjustment of the control servo parameters or just for the detection of faulty oscillators.

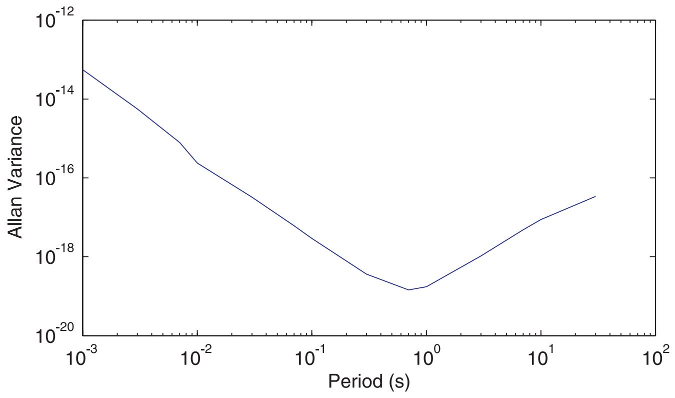

Fig. 6. Measured Allan variance of a 100 MHz oscillator

Given the possible timestamp granularity in the ps-range, the question may arise about the possible gain over traditional timestamping approaches with granularities in the range of about 10 ns. As shown in [7] it mainly depends on the oscillator and the synchronization interval. In summary the higher the stability of the oscillator and the longer the synchronization interval the more beneficial a very accurate timestamping method can render. For cheap oscillators which cause the clock in the slave node to drift away between the synchronization intervals in the order of several ns, a highly accurate timestamping method will result only in a marginal gain. Significant enhancements can be seen for highly stable oscillators or even for synchronous systems for delay measurements. Combing Synchronous Ethernet with IEEE 1588 allows all nodes to use the same frequency which simplifies the synchronisation. Despite that, such a system has to measure the line delay between master and slave. With the proposed timestamping method this delay can be measured very precisely enabling synchronous systems to synchronize virtually offset-free given that the physical connection is symmetric.

V. Conclusion

Timestamping a frame in an asynchronous communication system leads to the problem of passing the timestamp signal from the communication clock domain to the local one. Such a transition introduces one of the main jitter sources of such a network link. Despite different single-shot methods which have certain advantages but several physical constraints difficult to fulfil, this paper presents three phase estimating methods which offer a comparable or even better accuracy than single-shot methods. The presented solutions use the fact that the timestamping signal is always synchronous to the communication clock. Given that there is a certain frequency offset between the communication clock and the local clock, the simplest approach is to timestamp a frame multiple times.

The two other solutions continuously track the phase relation so that for every possible timestamping event the current phase information is available. While a direct phase estimation is possible, it seems to be simpler to estimate the phase by a combined phase/frequency estimation. The latter detects the instants of cycle slips and tries to forecast the phase based on the filtered frequency offset. Both methods rely on the permanent availability of all clocks. This prerequisite does not hold for the outdated 10 Base-T standard of Ethernet, which is nowadays only rarely used.

A sample implementation emulating a typical 100 Base-TX Ethernet connection on an Altera FPGA board was used for the evaluation of the design. The results show that the timestamping jitter can be significantly reduced by almost three orders of magnitude to a standard deviation of 52 ps, which is comparable to a local clock frequency of 5.5 GHz. The remaining jitter is primarily caused by the phase noise of the PLLs and the oscillator.

Acknowledgement

The work is partially co-funded via the Austrian FIT-IT Project ϵ-WiFi (Nr. 913310). Further, the authors would like to thank Oregano Systems - Design & Consulting Ltd. for their project support.

References

[1] T. Skeie, S. Johannessen, and O. Holmeide, “Highly Accurate Time Synchronization over Switched Ethernet,” in *Proc. 8th IEEE International Conference on Emerging Technologies and Factory Automation*, Oct. 2001, pp. 195–204.

[2] J. Kalisz, “Review of methods for time interval measurements with picosecond resolution,” *Metrologia*, vol. 41, pp. 17–32, Feb. 2004, provided by the SAO/NASA Astrophysics Data System. [Online]. Available: http://stacks.iop.org/Met/41/17

[3] R. Exel and G. Gaderer, “Boundaries of Ethernet Layer 2 Hardware Timestamping,” in *Proceedings of the 2008 IEEE International Workshop on Factory Communication Systems*, G. Cena and F. Siminot-Lion, Eds., 2008, pp. 255–258.

[4] J. Kalisz, R. Szplet, J. Pasierbinski, and A. Poniecki, “Field-programmable-gate-array-based time-to-digital converter with 200-ps resolution,” *IEEE Trans. Instrum. Meas.*, vol. 46, no. 1, pp. 51–55, Feb. 1997.

[5] R. Szplet, J. Kalisz, and R. Szymanowski, “Interpolating time counter with 100 ps resolution on a single FPGA device,” *IEEE Trans. Instrum. Meas.*, vol. 49, no. 4, pp. 879–883, Aug. 2000.

[6] X. Zhu, G. Sun, S. Yong, and Z. Zhuang, “A High-Precision Time Interval Measurement Method Using Phase-Estimation Algorithm,” *IEEE Trans. Instrum. Meas.*, vol. 57, no. 11, pp. 2670–2676, Nov. 2008.

[7] P. Loschmidt, R. Exel, A. Nagy, and G. Gaderer, “Limits of synchronization accuracy using hardware support in ieee 1588,” in *ISPCS 2008, International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Ann Arbor / U.S.A., Sep. 2008, pp. 12–16.