A Zero-Nanosecond Time Synchronization Platform for Gigabit Ethernet Links

Carles Nicolau

Dept. of Information and Communication Technologies,
Universitat Pompeu Fabra, Barcelona, Spain
carles.nicolau@upf.edu

Abstract. Ethernet is becoming the dominant data transmission technology for service providers due to its simplicity and low cost. However, the need for Quality of Service (QoS) provisioning for some time-sensitive applications drives the definition of new functionalities in Ethernet. In this work we address QoS in Ethernet by introducing a time synchronization capability at MAC level while maintaining its asynchronous and distributed architecture. We present a reconfigurable logic platform based on a Field Programmable Gate Array (FPGA) in which we embed our custom timestamping unit (TSU). Leveraging the TSU, we have implemented a synchronization mechanism with which we achieved a best-case synchronization accuracy of zero nanoseconds. The effectiveness of the method is confirmed through several experiments.

Keywords: Ethernet, Quality of Service, Time Synchronization, FPGA, HW-timestamping.

1 Introduction

Ethernet technology is becoming the dominant access technology at local, metropolitan, and wide area networks. The low cost, high data rates and low complexity and maintenance offers an opportunity to service providers for using Ethernet on a very large scale and replacing their expensive legacy networks to more simple Layer 2 (of the OSI layered model) Ethernet networks. However, low cost and simplicity are only part of Ethernet's attraction. The challenge is that Ethernet is a 'best-effort' and asynchronous oriented technology, therefore industry bodies and equipment manufacturers are making considerable efforts to ensure Ethernet services become 'carrier-class'. High-precision time synchronization is a key enabler for offering such carrier-class QoS, a functionality that present native Ethernet lacks and that provides receivers with precise information about end-to-end delays, and that would benefit e.g. effective playout of time-sensitive data [1] or high-precision network analysis [2].

A few years ago the Ethernet Passive Optical Network (EPON) specification [3] introduced a rough synchronization capability in Ethernet based on timestamping for the purpose of coordinating the slotted access of the nodes. Synchronization accuracy and precision using timestamps are affected by the time variability of inserting the time information inside the message [4]. With a well-defined and characterized timestamp mechanism, we believe there is a chance to define a synchronization mechanism for native Ethernet to better support time-sensitive applications with more control of the QoS provided. Our objective is to improve Ethernet capabilities within an acceptable range while respecting its architecture simplicity principle. Hence, we address QoS by introducing a time synchronization capability but maintaining its asynchronous and distributed architecture.

With the former perspective in mind, we present a testbed for the design extension of native Gigabit Ethernet to address the QoS through time synchronization at Layer 2. The testbed is based on a FPGA which provides the resources for designing a custom and flexible hardware (HW) Timestamping Unit with the following improved key features from our previous work [5]: **1)** high-resolution and reliable HW timestamping, **2)** precise timing of the execution/processing times of the internal platform key blocks, with which we can optimally characterize the timestamping mechanism, and **3)** high-resolution calculation of message propagation times. Leveraging the three previous properties, we reuse the synchronization mechanism specified in EPON to achieve a perfect link synchronization between two nodes.

We start this work in Section 2 which pinpoints the actual solutions akin to our work. Section 3 is devoted to clarify the requirements of our platform, as well as to explain in detail its design and the different scenarios to obtain reliable timing and synchronization performance. In Section 4 we present the experimental results for characterizing the timestamping mechanism and the achieved synchronization accuracy. We draw conclusions in Section 5.

2 Related Work

Considering the requirement of maintaining the actual native Ethernet structure, i.e. without redefining the physical layer (PHY), other standards such as the IEEE Std. 1588 Precision Time Protocol (PTP) [6] and the Network Time Protocol (NTP) [7] are being developed. Both protocols distribute timing via UDP packets that carry timestamps generated by a master (server). PTP achieves up to six orders of magnitude better synchronization accuracy than NTP by relying on HW support for generating the timestamps. The hundred-ns accuracy target of PTP is consolidating it as THE standard for synchronization timing distribution over Ethernet in many different areas.

Since the last revision of the PTP standard in 2008, a new normative for transporting PTP over IEEE 802.3 Ethernet is considered (Annex F of [6]), a feature that is akin to our work. This new specification prospects a fully IEEE 1588 Layer 2 subsystem independent from upper layer services, and thus also reinforcing our initial beliefs on the success of a pure Layer 2 solution for Ethernet. This new PTP normative will benefit from the use of shorter messages (64 bytes) and the possibility to build PTP with small memory footprint and resources, even integrated in a single Ethernet chip. To our knowledge, we have proof of one work [8] implementing this new *IEEE 1588 Layer 2* specification. The authors are targeting synchronization accuracies of tenths of ns between contiguous nodes.

3 Design

3.1 Platform Requirements

The major objective in the development of our platform is to provide a low-cost, programmable and flexible framework for experimenting with new Ethernet design extensions at low level. FPGA-based embedded platforms are the best tools for this purpose. However, one limitation of commercial Ethernet platforms is that their compliant MACs are black boxes, avoiding researchers to integrate and experiment with new MAC functionalities. Here we want to rapidly work around this problem and avoid (the hard task) of designing a compliant MAC from scratch. In this design we focus on the following high-level requirements: **1)** fast re-usability with other platforms, **2)** easy upgradeability with improved or new functionalities, **3)** compatibility with SW-based synchronization solutions [7], thus providing Ethernet with autonomy and independence from upper layers to perform fully Layer 2 functions. **4)** Keep the asynchronous and distributed Ethernet philosophy, avoiding synchronization approaches based on re-spinning the Layer 1 transceivers [9].

3.2 Embedded System

In Fig. 1 it is shown the reconfigurable platform, the Xilinx ML403 general purpose board which is based on a Virtex-4 FPGA [10]. The board mounts a tri-speed Ethernet PHY, several expansion headers (HDR) for external connectivity, 64MB of DDR SDRAM for massive storage and two XO's for clocking the PHY and the rest of the FPGA internal circuitry. The FPGA chip contains a PowerPC microprocessor (μP), an Ethernet MAC block (MAC), a digital frequency synthesizer (DFS) and reconfigurable logic with which we have implemented our Timestamp Unit (TSU).

The TSU is divided into four major blocks. The first one is the transmission block (tx block) which contains a finite state machine (tx FSM) that detects and manages the replacement of some synchronization frame fields. *tx block* operation is synchronous with the transmission clock signal (*tx_clk*) coming from the MAC ('- -' line).

The second block is the reception block (rx block) which has the same functionality as its transmission counterpart besides cyclic redundancy error checking of the incoming frames. The operation of *rx block* is synchronous with the reception clock signal coming from the on-board PHY chip ('· -' line).

The third block is the control adjustment unit (CAU) which contains a 32-bit counter (*localtime*) that is summing up clock ticks at the frequency of 269.96 MHz ('··' line), thus providing a timestamp granularity of 3.704 ns. We will use *localtime* counter to keep track of high-resolution time at MAC level and as the

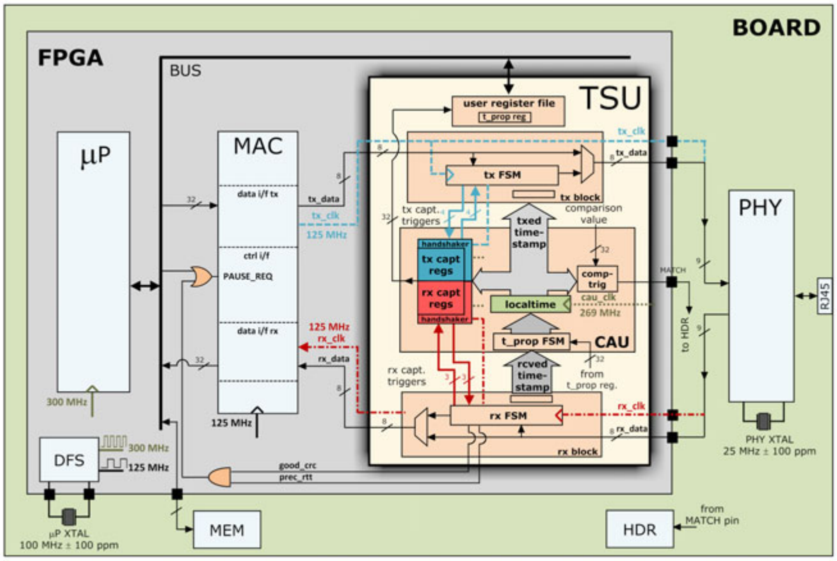


Fig. 1. Simplified block diagram of the reconfigurable platform

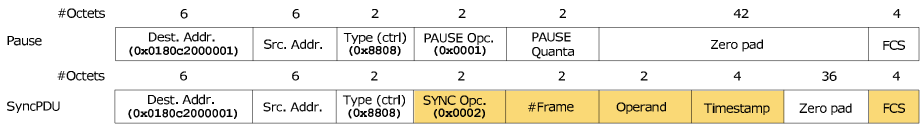
fine time measurement unit with which to compare the synchronization accuracy between nodes. As seen in Fig. 1, *localtime* register is interfaced to/from four regions:

1. to a capture register file within CAU, for storing different time freeze points (see Section 3.3) of the outbound and inbound frames (*tx capt regs* and *rx capt regs*, respectively). Each FSM triggers several timestamps when transmitting/receiving frames that are stored in *register stack*. The main problem of this approach is the interaction of the register stack with the three clock sources (*rx_clk*, *tx_clk* and *cau_clk*). This cross-clock domain scenario can hamper the reliability of the timestamps due to metastability errors [11]. In order to prevent inconsistent timestamps, we have introduced handshake operations (*handshaker*) between the capture registers and *localtime*. *handshakers* consist on a FSM and a flancter circuit [12].
2. to a comparator-trigger block (*comp-trig*) within CAU, for comparing *localtime* instantaneous values with a user-defined constant value (*comparison value*). When *localtime* equals *comparison value*, a 50 ns-wide pulse is raised on an external output (*MATCH*). This resource will provide us means to verify the synchronization accuracy between nodes (see Section 4.2).
3. to the transmission block (*tx block*) within the TSU, to provide the timestamps to be transmitted in the outbound synchronization protocol data unit (syncPDU). The timestamps are triggered at the 25^{th} byte of a current outbound frame, and inserted from 29^{th} to 33^{rd} fields (see Fig. 2).
4. from the reception block (*rx block*) within TSU, to read the incoming timestamps and syncPDUs. The time of arrival of a syncPDU is also taken at the

25th incoming byte to keep the symmetry with the outbound path. *rx block* contains a state machine (*t_prop FSM*) that performs the propagation time correction with the content stored in *t_prop reg* register.

The last block is the *data register stack* register file which contains all the information of the last sent/received syncPDU, internal timing measurements and control/status registers.

As explained before, due to the MAC inaccessibility, we reuse the existent flow control frames (i.e. Pause frames) to generate the syncPDUs. When the TSU detects that an outbound frame is a Pause, it replaces on-the-fly 5 fields (17 octets) (see Fig. 2).



#Octets	6	6	2	2	2	42	4
Pause	Dest. Addr. (0x0180c2000001)	Src. Addr.	Type (ctrl) (0x8808)	PAUSE Opc. (0x0001)	PAUSE Quanta	Zero pad	FCS

#Octets	6	6	2	2	2	2	4	36	4
SyncPDU	Dest. Addr. (0x0180c2000001)	Src. Addr.	Type (ctrl) (0x8808)	SYNC Opc. (0x0002)	#Frame	Operand	Timestamp	Zero pad	FCS

Fig. 2. Pause control frame (top). Synchronization protocol data unit (syncPDU) (bottom).

3.3 SyncPDU Timing

We aim at designing a timing model to measure with high resolution the processing and propagation times of a syncPDU from one node to the other for the purpose of characterizing the delay and jitter of the timestamping mechanism. In Fig. 3 it is shown the timing of the synchronization flow ('· -' line) at, and below the MAC level, which starts and finishes at the master side. The time freeze points of the synchronization flow are located at different strategic and symmetric physical points along the path. For a better follow up, we have assigned logical time variables to the HW registers with the purpose of storing intermediate transmission ($t_B(t_i)$-`tx_req`, $t_C(t_J)$-`st_tx`, $t_D(t_K)$-`txed_ts`, $t_E(t_L)$-`end_tx`) and reception ($t_M(t_F)$-`st_rx`, $t_N(t_G)$-`toa`, $t_O(t_H)$-`end_rx`) times.

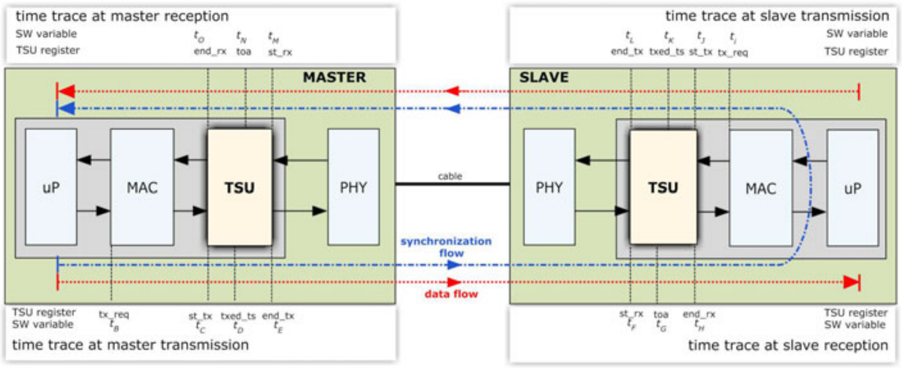


Fig. 3. Timing reference model for timestamping characterization

3.4 Synchronization Mechanism

As mentioned before we extrapolate and enhance EPON synchronization approach to synchronize two nodes in a direct link configuration. The goal using this synchronization mechanism is to measure and to adjust the clock offset between each pair of clocks in order to synchronize them perfectly. In Fig. 4 it is shown the timeline of the synchronization protocol operation at different block levels in each node. The inner columns show the instantaneous clock offset between the master and the slave and viceversa. The mid and outer columns denote which HW registers and logical variables are being written during the process.

The whole process is divided into two recursive phases, the *normal* operation phase and the *discovery* phase. In the former, the master is sending *GATE* type messages which are replied by the slave under *REPORT* type messages. At the normal phase the master is checking the progressive clock drift of the slave. If it detects a specific maximum clock drift, it enters into the discovery mode. If the clock drift keeps lower than a maximum threshold value, the master enters into *discovery* mode after a fixed and periodic interval of time ($\sim$ 14.3 sec).

The re-synchronization is performed in the *discovery* phase. As seen on the left-side in Fig. 4, the master calculates the Round Trip Time (RTT) upon the last received *REG_REQ* message according to eq. 1. This phase finishes after the master communicates the propagation delay (RTT/2) to the slave in a *REGISTER* message. The propagation delay will be used by the slave in the next normal operation phase to achieve the link-synchronization.

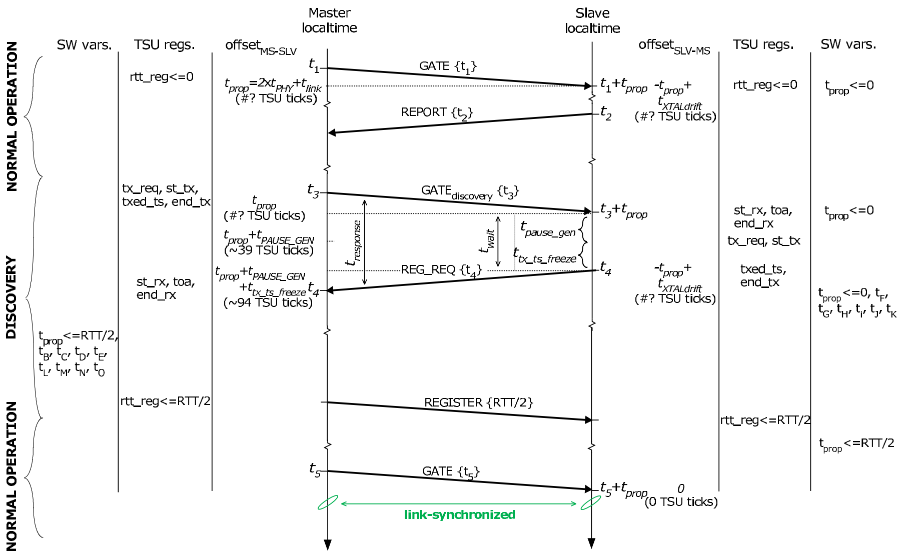


Fig. 4. Synchronization procedure adopted from EPON [3]

$$\begin{aligned} RTT &= t_{response} - t_{wait} \\ &= (t_N - t_D) - (t_{pause\_gen} + t_{tx\_ts\_freeze}) \\ &= (t_N - t_D) - (t_J - t_i) - (t_K - t_J) \\ &= (t_N - t_D) - (t_C - t_B) - (t_D - t_C) \end{aligned} \tag{1}$$

4 Evaluation

4.1 Timestamp Variability

To characterize the timestamp variability of the timing model of Fig. 3, we observe the relative time differences between the time freeze points. In table 1 it is shown the statistics of the timing of the synchronization flow over 50k packet and under different load scenarios. The theoretical values of each difference can be expressed in number of TSU clock ticks taking into account the CAU and the PHY interface frequencies, i.e. $\frac{f_{cau\_clk}[MHz]}{f_{tx/rx\_clk}[MHz]} = \frac{269.96}{125} = 2.15968$. For instance, a timestamp that is triggered at the 25th byte of an outbound syncPDU is equivalent to $t_{ts\_tx\_freeze} \times 2.15968 \approx 54$ TSU clock ticks. The small differences between the experimental and the theoretical values observed

Table 1. Histogram of synchronization flow timing (Fig. 3) over 50k packet run test (expressed in number of TSU clock ticks (percentage over 50k samples))

		transmission			reception	
		t_{pause_gen} (t_C-t_B)	$t_{ts\_tx\_freeze}$ (t_D-$t_C \approx 54$)	t_{tx_rem} (t_E-$t_D \approx 104$)	$t_{ts\_rx\_freeze}$ (t_N-$t_M \approx 54$)	t_{rx_rem} (t_O-$t_N \approx 104$)
Master	no load	36 (16.1%)	57 (51.3%)	113 (100%)	56 (85.0%)	110 (10.2%)
		37 (10.6%)	58 (49.7%)		57 (15.0%)	112 (45.3%)
		38 (49.7%)				113 (44.5%)
		39 (25.6%)				
	w/64B data	37 (40.2%)	57 (55.3%)	109 (18.6%)	54 (30.8%)	112 (38.2%)
		38 (39.1%)	58 (45.7%)	113 (81.4%)	56 (69.2%)	113 (61.8%)
		39 (20.7%)				
	w/1500B data	36 (17.7%)	57 (55.5%)	109 (10.1%)	55 (23.5%)	112 (40.5%)
		37 (25.9%)	58 (44.5%)	112 (20.7%)	56 (76.5%)	113 (59.5%)
		38 (56.4%)		113 (69.2%)		
		t_{pause_gen} (t_J-t_i)	$t_{ts\_tx\_freeze}$ (t_K-$t_J \approx 54$)	t_{tx_rem} (t_L-$t_K \approx 104$)	$t_{ts\_rx\_freeze}$ (t_G-$t_F \approx 54$)	t_{rx_rem} (t_H-$t_G \approx 104$)
Slave	no load	36 (19.9%)	57 (54.5%)	113 (100%)	55 (78.4%)	112 (19.6%)
		37 (32.3%)	58 (45.5%)		56 (19.6%)	113 (75.4%)
		38 (47.8%)				
	w/64B data	36 (17.1%)	57 (55.3%)	109 (18.6%)	55 (80.4%)	112 (21.1%)
		37 (40.2%)	58 (45.7%)	113 (81.4%)	56 (19.6%)	113 (78.9%)
		38 (23.8%)				
		39 (18.9%)				
	w/1500B data	36 (23.4%)	57 (52.1%)	112 (9.8%)	55 (28.2%)	112 (30.3%)
		37 (35.8%)	58 (47.9%)	113 (90.2%)	56 (71.8%)	113 (69.7%)
		38 (40.8%)				

in the table stem from the additional clock ticks needed by the handshakers to perform reliable timestamp readings. We are interested on the timestamp variability as we will use them in eq. 1. The experimental data is well discretized, precise and confined, thus proving the reliability of the timestamping mechanism.

4.2 Link Synchronization Accuracy: Phase and Time

We have used two methods for the evaluation of the synchronization accuracy and precision between the two nodes at link level. The first method consists on comparing the phase error between TSUs just after a *localtime* adjustment. To observe this, we configure the comparator-trigger blocks inside the TSUs to raise a 50ns-wide pulse at periodic intervals of 225μs through *MATCH* output pin (see right-side of the Fig. 5). On the left side of the Fig. 5 it is shown a zoom in the time adjustment for the best and worst case synchronization accuracy. To obtain this screenshot, we have forced the slave node to re-synchronize every 5 milliseconds to better observe the variability of a re-synchronization event. Over the ~200 overlapped signal traces, we have achieved best-case link synchronization accuracies of 0 ns between TSU's *localtime* counters, and a worst-case synchronization accuracy of 130 ns.

The second method consists on comparing the instantaneous *localtime* values at each node during the *normal operation* phase of the synchronization procedure (see Fig. 4) under three different load scenarios: no background data (no bg data), with minimum-length data packets (w/64B pckts) and maximum-length data packets (w/1500B pckts).

The plots in Figs. 6a,b,c refer to the synchronization accuracy at t_N and t_K time freeze points. Fig. 6a zooms in a 150 sec window to show the 'sawtooth' shape of the synchronization trace, which reveals the clock drift of the slave relative to the master (~3.26μs/s). The sawtooth is not uniform, which means that the re-synchronization events are not executed exactly every 5 sec. This behavior owes to the fact that the μP handles multiple interruption requests from different HW blocks inside the platform and serve them undeterministically. Fig. 6b shows that the achieved synchronization accuracy is mostly confined within ±10 TSU clock ticks. The histograms in Fig. 6c better show the clock offset distribution in such region. For the three load scenarios, mean (μ) and standard deviation (σ) keep below 1 and 10 ns, respectively. The different σ values might be caused by the extra clock ticks needed by the HW mechanisms inside the MAC and the PHY to generate and process a packet.

The plots in Figs. 6d,e,f are equivalent to those of the first row but choosing different time freeze points (t_D and t_G). We observe several differences in the plot 6e compared with its top counterpart. Here the clock offset is bounded in the [20, 30] TSU ticks region, denoting an asymmetry when compared to t_N and t_K. The 20 cycles score is due to the round off error committed by the program code of the master when calculating the RTT. The bulk of the data in Fig. 6e and the zone with higher density of traces on the left plot of the Fig. 5 are equivalent

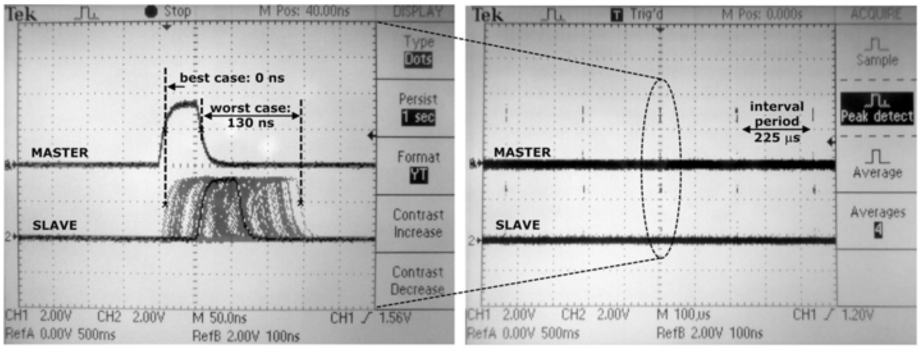


Fig. 5. Phase error of the synchronized timing signal (right). Re-synchronization variability (left).

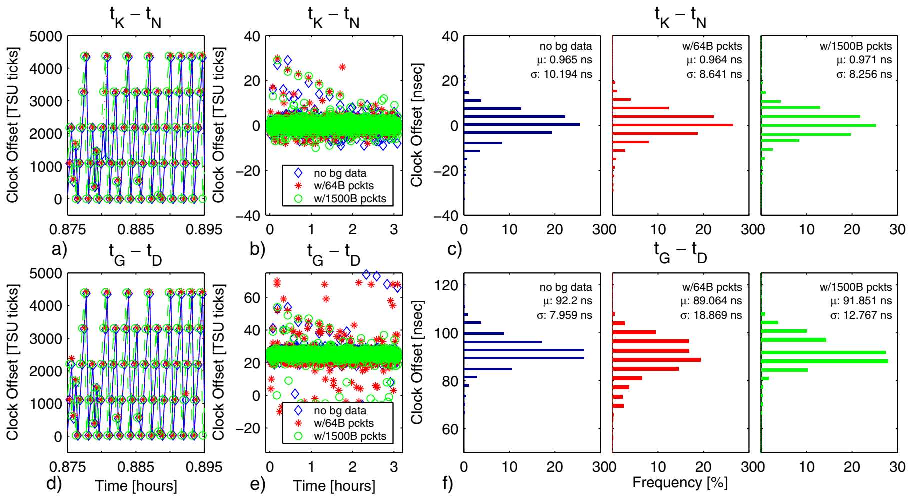


Fig. 6. a,d) Accumulated clock offset. b,e) Synchronization accuracy. c,f) Synchronization accuracy distribution.

([74, 111] ns). μ and σ statistics of histograms in Fig. 6f are well confined over 90 and 12 ns, respectively.

5 Conclusion

In this work we have presented an FPGA-based platform for developing and testing new Ethernet functionalities. We envision Ethernet as a technology capable to offer carrier-class QoS while maintaining its simplicity and low-cost if time synchronization is addressed. For this reason, we have implemented within the programmable resources of the FPGA an improved version of our custom hardware Timestamping Unit with several key functionalities: high-resolution and error-free on-the-fly timestamping, high-resolution timing of inter-MAC events and frame propagation times. Leveraging these features, we have characterized the timestamping mechanism in terms of time variability and re-used the accurate timings in the EPON synchronization exchange mechanism to achieve best-case synchronization accuracies of zero nanoseconds at MAC level.

Acknowledgements

The author gratefully acknowledges key ideas on timing flow from Prof. Dolors Sala, implementation-related suggestions from Prof. Enrique Cantó and profitable comments from Prof. Johan Zuidweg.

References

1. Steinmetz, R.: Human Perception of Jitter and Media Synchronization. IEEE Journal on Selected Areas in Communications 14(1), 61–72 (1996)
2. Arlos, P.: On the Quality of Computer Network Measurements, Ph.D. Thesis, Blekinge Institute of Technology (2005)
3. IEEE Standard for Local and Metropolitan Area Carrier Sense Multiple Access with Collision Detection Access Method and Physical Layer Specifications, IEEE Std 802.3-2005 (2005)
4. Kopetz, H., Ochsenreiter, W.: Clock Synchronization in Distributed Real-Time Systems. IEEE Transactions of Computers C-36(8), 933–939 (1987)
5. Nicolau, C., Sala, D., Cantó, E.: Clock Duplicity for High-Precision Timestamping in Gigabit Ethernet. In: 19th Int. Conf. on Field Programmable Logic and Applications (FPL), Czech Republic, August 31–September 2 (2009)
6. IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEC 61588:2009(E), pp. C1–274 (2009)
7. Mills, D.L.: Network Time Protocol (Version 3) Specification, Implementation and Analysis, RFC-1305 (March 1992)
8. Kutschera, C., et al.: Background IEEE 1588 Clock Synchronization over IEEE 802.3/Ethernet. In: IEEE Symp. on Precision Clock Synchronization (ISPCS 2008), September 22-26 , pp. 137–141 (2008)
9. Finn, N., Cisco Systems Inc.: Sync PDUs and the MAC Stack - Help needed from 802.3 to properly process IEEE 1588/P802.1AS sync PDUs. In: IEEE 802.3/IEEE 802.1 joint session, Denver, CO, USA (July 2008)
10. Xilinx, Inc.: ML40x Evaluation Platform User Guide, ug080 v2.5 (2006)
11. Stephenson, J., Altera Corp.: Don't Let Metastability Cause Problems in Your FPGA-Based Design (2009), http://www.pldesignline.com/220300400
12. Weinstein, R.: The "Flancter" (App. Note), Memec Design(2000), http://www.floobydust.com/flancter/Flancter_App_Note.Pdf