ARGOS: A GPS Time-Synchronized Network Interface Card based on NetFPGA

Jaime J. Garnica, Victor Moreno, Ivan Gonzalez, Sergio Lopez-Buedo,
Francisco J. Gomez-Arribas, Javier Aracil

High Performance Computing and Networking Group
Universidad Autonoma de Madrid
11 Francisco Tomas y Valiente, 28049 Madrid, Spain
Tel: +34 914972249

jaime.garnica@uam.es, sergio.lopez-buedo@uam.es

ABSTRACT

In this paper we present ARGOS, a NetFPGA-based Network Interface Card featuring transparent GPS time synchronization. It allows to accurately timestamp received packets, and to send precisely spaced bursts of packets. Since NetFPGA does not include a GPS interface, we developed a sister card including RS232 and RS422 ports where standard GPS receivers can be connected. ARGOS appears as a conventional NIC to the operating system, and it uses the timestamping features offered by Linux kernels, so any conventional packet capture tool such as Wireshark can be used with it. ARGOS has been successfully deployed in the OneLab2 European Internet testbed, making it possible to launch experiments in more than 10 nodes spread all around Europe and Israel. Using ARGOS and the OneLab2 testbed, complete pictures of Internet delay and available bandwidth can be obtained.

1. INTRODUCTION

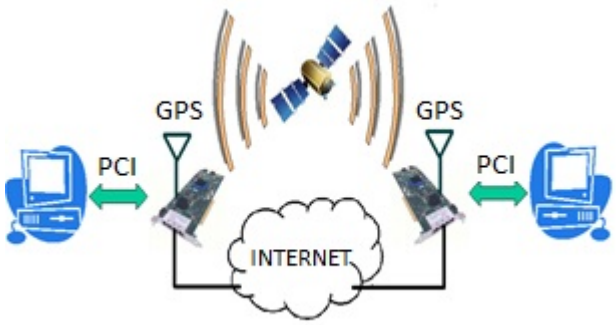
Accurate packet timestamping is an essential tool to study the behavior of communication networks. Not only it provides valuable information about the behavior of networking infrastructure (routers, switches, etc.), but also it is necessary to better understand applications and protocols. For example, an apparently simple operation such as measuring one-way delays requires both ends to be accurately synchronized. Small errors, even sub-millisecond, can ruin the measurements. As networks increase their speed, better time synchronization is required. For example, the maximum packet rate in 10 Gigabit Ethernet is 14.88 million packets per second. That is, time between packets can be as small as 67.2 nanoseconds, thus requiring nanosecond-level synchronization.

Time synchronization is usually implemented using NTP (Network Time Protocol) [1][2]. NTP typically provides millisecond accuracy, utilizing redundant servers in order to offer accuracy and reliability. An alternative to NTP is the Precision Time Protocol (PTP) described in IEEE standard 1588 [3]. PTP allows for much better accuracies, but it requires a precise clock to be available in the same local area network where the system being synchronized is connected. Whereas NTP is designed to work at the software level, with a NTP daemon running on the system, PTP provides nanosecond-level accuracy by using special NICs with hardware-assisted time synchronization. However, this level of accuracy can only be obtained if there is a high-precision clock source in the local area network (PTP Grandmaster) and network switches are also PTP-aware, thus requiring significant investments.

Actually, the most commonly used solution for high-accuracy packet timestamping is to directly attach a GPS receiver to the NIC. Firstly, GPS receivers are nowadays cheap and straightforward to operate. Secondly, accurate packet timestamping can only be performed at the hardware level, as close as possible to the input interface. Even if the host clock is perfectly synchronized, the uncertainty added by the different elements between the NIC and the host processor (FIFOs, DMA transfers, etc.) will severely impair the accuracy of the measurements. This is the option implemented in this paper: A GPS-based time source is connected to NetFPGA in order to create a NIC time-synchronized to UTC within tens of nanoseconds.

The main objective of ARGOS is to be used as the building block of a distributed, time-synchronized measurement system such as the one shown in Fig. 1. Thus, NetFPGA boards can make up an accurate time testbed globally synchronized to UTC by means of a GPS receiver connected to each board.

Fig. 1. Measurement system



As opposed to other passive probes, ARGOS operates as a conventional NIC, and the driver writes the timestamp directly on the *sk_buff* structure of the Linux kernel. Therefore, timestamps can be obtained from standard, libcap-based tools such as Wireshark. On the other hand, ARGOS can act too as an active probe, being able to send trains of UDP packet bursts, as well as any other packet as if it were a regular NIC. The only difference is that the packet trains can be programmed to be sent at a given moment, and the transmit timestamp is written in the UDP payload.

The rest of paper is organized as follows. Section 2 describes the timing solution being implemented. Next, section 3 presents OneLab2, the European Internet testbed where this development has been deployed. Finally, section 4 presents some experiments using ARGOS and OneLab2.

2. TIMING SOLUTION

Implementing the accurate timestamping of packets involves many different tasks: interface to the GPS time source, design of the FPGA hardware, writing the driver, Etc. This section describes the solution developed for ARGOS.

2.1 GPS modules

Since the Global Positioning System (GPS) is based on high-precision atomic clocks, many GPS companies started selling GPS modules specialized in timing instead of positioning, providing users with nanosecond-level precision. The widespread use of GPS for positioning has dramatically reduced the price of receivers, thus becoming the solution that provides the best price/performance relationship for time synchronization.

GPS time receivers use a signal named Pulse Per Second (PPS), which is typically activated every time a UTC second starts, with less than tens of nanoseconds accuracy. Additionally, receivers have a serial port for sending position and time messages. There are many protocols for it, but the most commonly used is the National Marine Electronics Association [4] protocol, or NMEA.

NMEA defines an application layer protocol inside the GPS message system, which follows these rules:

- Each message starts with the character '$'.
- The two following characters identify the talker.
- Three next characters identify the type of the message.
- Following, there are several comma-delimited fields (containing NUL bytes if unavailable).
- An asterisk follows the last field and after it there is a message checksum.
- Messages end with <CR><LF>.

The timing message provided by the NMEA protocol is named ZDA. It presents the following structure:

$GPZDA,hhmmss.sss,dd,mm,yyyy,xx,zz*cs<CR><LF>

Time is in UTC in the format: hours, minutes, seconds, days, months and years; and "xx", "zz" respectively the hours and minutes in the local zone. Typically, a ZDA message is sent by the GPS receiver shortly after the PPS edge, in order to identify which UTC second just started.

Proprietary GPS protocols, such as the ones from uBlox [5] or Trimble, offer a bigger set of messages providing more information, which allows for a better control over the GPS information. These protocols are especially useful for accurate time synchronization applications, where the information of deviation and estimated signal error can be used to improve the timing precision.

2.2 NetFPGA and GPS communication

The sister card, figure 3, is a second board connected to NetFPGA by mean of a ribbon cable and an adaptor that plugs on the NetFPGA debug connector, see Fig. 2. The sister card obtains its power supply from the PCI or PCIe bus (there are two versions), and its main objective is to implement a serial port (RS232) that connects the NetFPGA and the GPS module. Apart from the MAX3232 level-conversion chip, the sister card includes a buffer to drive LVTTL signals through the ribbon cable. On the adaptor that plugs on NetFPGA there is an additional buffer to convert signals to the 2.5 volt levels required by the FPGA.

Fig. 2 Sister Card on DL360 chassis

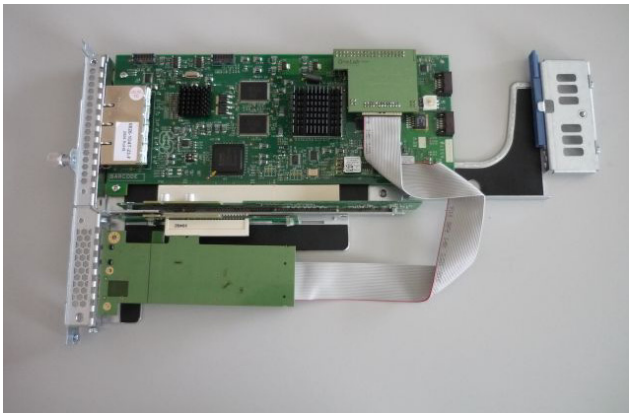
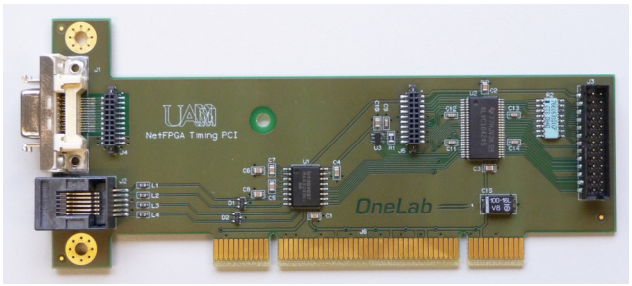
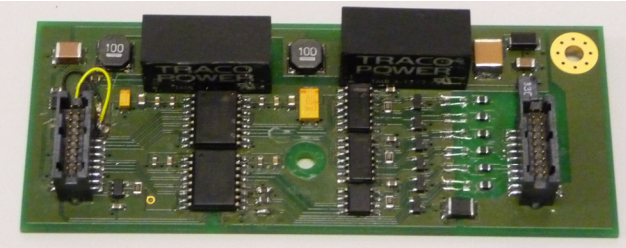


Fig. 3 Sister Card



Additionally, the sister card provides a slot for a module that provides RS-422 communications and remote +24V power for roof-installed GPS receivers such as Trimble Acutime Gold GPS Smart Antenna. Using this module, see fig. 4, the GPS can be up to 200m away from NetFPGA.

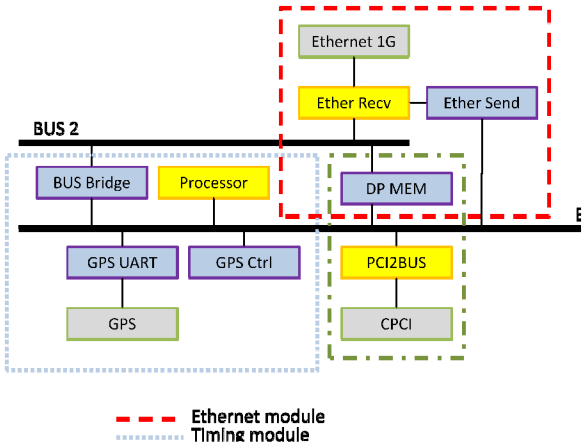
Fig. 4 RS-422 Module



2.3 Packet management and time stamping

The system architecture is shown in Fig. 5. Packets are received and sent by an Ethernet core that stores input frames into a double port memory appearing at the PCI side as if it were a FIFO. When a frame is going to be stored in the double port memory, first the timestamp is obtained from the timing module (GPS Ctrl) and it is also saved in the DP memory. The same behavior occurs when a packet is being sent, first the timestamp is read and it is written in the UDP payload just when the frame goes out of the NIC. In the figure, the thick lines are PLB buses. We decided to implement a design based on Xilinx EDK in order to continue a previous work for another platform. However, future versions will be integrated in the standard NetFPGA FPGA architecture. The drift correction algorithm runs on a MicroBlaze processor embedded in the FPGA [6], which interacts to the timing module (GPS_Ctrl) and the UART where NMEA commands are received (GPS_UART). A host-based solution could be used in spite of MicroBlaze, but this one makes easier the interface between the host and the FPGA board, avoiding the use of more registers and simplifying the driver design.

Fig. 5. Solution design



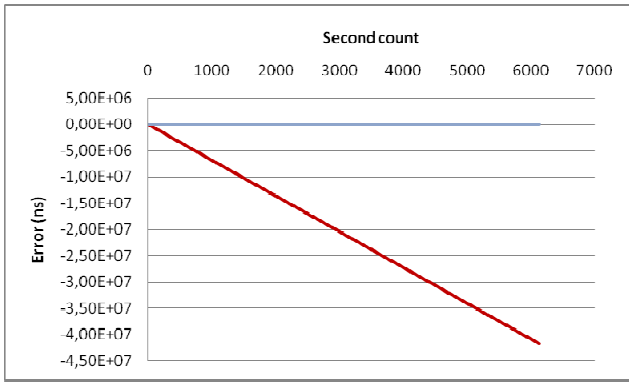
Finally, figure 6 shows the transmission mode of UDP trains of packet bursts, which are very useful for bandwidth calculations [7]. Inter-burst, inter-packet times are specified by means of a driver ioctl() call.

Fig. 6. Burst train transmission

2.4 Clock error correction

The NetFPGA local quartz oscillator has a certain drift, which should be compensated using the PPS signals that tells when a new UTC second starts. Actually (see Fig. 7) the drift of the NetFPGA local oscillator is linear, about microseconds per second (tens of milliseconds per hour).

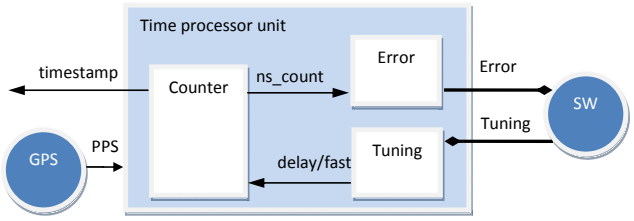
Fig. 7. Error between NetFPGA clock and PPS (ns)



Additionally, the received PPS signal also presents errors arousing from propagation delays in the antenna, cables, and buffers inside the sister card. Fortunately, it is reasonably safe to consider these delays as constant. Thus, they can be easily compensated by adding an offset to the PPS signal, something that can be easily programmed on most GPS receivers.

The local oscillator drift is compensated by means of a hardware module, which is in turn controlled by the error correction algorithm running on the MicroBlaze processor. Figure 8 shows the architecture of this module. Firstly, there is a counter running from the local oscillator. This counter provides the reference timestamp, and may be advanced or delayed according to the information coming from the tuning module. This tuning module is the one in charge of correcting the drift, and is updated by the processor based on the information gathered by the error module, which keeps track of the difference between the main counter and the reference PPS signal.

Fig. 8. Clock correction hardware architecture



The tuning value for second t is defined as the number of nanoseconds that will elapse before the count must be advanced/delayed by 1 ns during second t. For example, if the tuning value is 1, it means that every nanosecond the count will be advanced by other nanosecond, so the count will accumulate one second in only half a second. On the other hand, if the tuning value is greater than half a second the count will be advanced by just 1 ns during second t. The tuning is estimated by the software algorithm every second t from the error expected for the next second.

$$tuning(t) = {}^{1\ s}/_{expected\ error(t)\ ns}$$

The first step in the algorithm is an estimation of the NetFPGA local oscillator drift, which is at this step considered to be constant. Then, the best way to calculate this initial mean drift is to leave the counter run without any correction for some seconds, and then measure the accumulated error:

$$drift\ error(0) = \sum_{i=0}^{MAX\_SAMPLES} error(i)$$

Afterwards, there is still present an error per second because of the FPGA clock stability [8] and with it the need to perform a continuous fine tuning of the drift error:

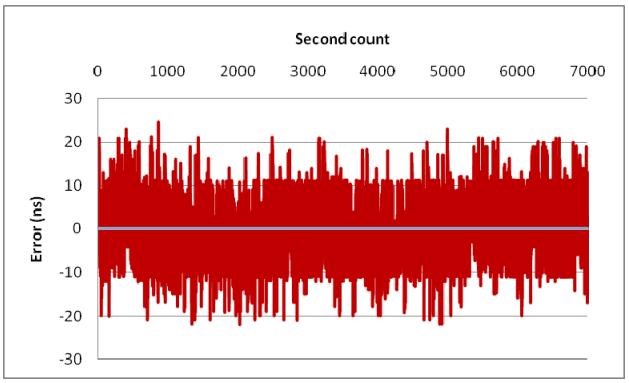
$$expected\ error(t+1) = error(t) + drift\ error(t+1)$$

Finally, some stability peaks appear as oscillations in the order of hundreds of nanoseconds. The last step is to reduce this oscillation as much as possible. In order to do that, the base error calculated with the initial examples must be adapted every second, depending on the last observed NetFPGA clock deviation. That is, a constant is added or subtracted to the drift error depending on the sign of the observed error:

$$dirft\ error(t+1) = drift\ error\ (t) \pm k$$

The error obtained shown in figure 9, after the correction reaches the stability around zero, oscillates between -20 and 20 nanoseconds. Therefore the precision obtained is on the order of tens of nanoseconds.

Fig. 9. Error obtained after correction



3. OneLab2 Testbed

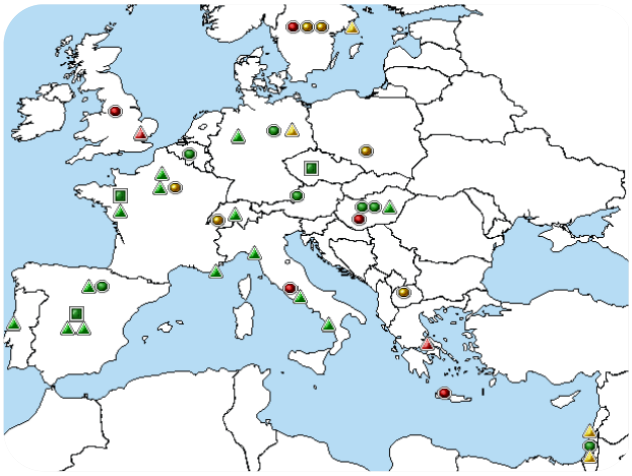
OneLab2 [9] is a European open federated laboratory supporting network research for the future Internet. The proposal was funded as an IST project under the EU FP7 funding program. It runs and operate PlanetLab Europe [10], federated with PLC.

3.1 Etomic

The European Traffic Observatory Measurement Infrastructure (ETOMIC) was created in 2004-05. Its goal is to provide an open access, public testbed for researchers investigating the Internet with active measurement methods [11]. The deployed measurement nodes developed are fully reconfigurable, extremely accurate (nanoseconds) and GPS-synchronised. ETOMIC initially used DAG cards from Endace, but with its integration in OneLab2 testbed it changed to ARGOS cards, based on NetFPGA.

Nowadays are more than 30 GPS-synchronized measurement nodes among Europe. The location of measurement nodes is illustrated in figure 10. Triangles show new ARGOS nodes, based on NetFPGA.

Fig. 10. Etomic nodes geographical location



3.2 OneLab nodes

The hardware used in OneLab2 nodes is a HP Proliant ML370 server, in which is contained an Argos Card working as a NIC. This equipment is the named Advanced Network Monitoring Equipement (ANME). The server kernel, typically runs on RedHat or Debian OS, and kernel recompilation adds the functionality to change the millisecond-accuracy time stamping brought by the time-of-the-day of UNIX systems to the new nanosecond time stamping.

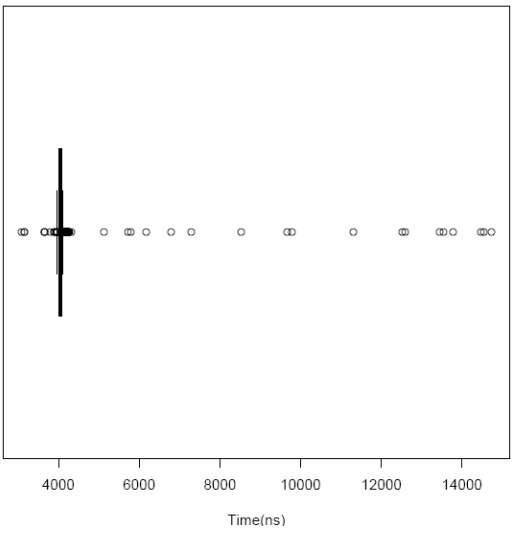
4. EXPERIMENTS AND RESULTS

The experiments for this paper have been done in the OneLab2/ETOMIC environment. The experiment consists in sending homogeneous-size UDP packet burst train, where a packet-per-burst number and a burst-per-train number are chosen. Since every node is UTC-synchronized, two of them are run the sending script in the first node and another script that open a socket to pick up the received UDP packets for the specified flow. The reception host computer calculates the difference between sending and reception timestamps in order to calculate the delay of the line.

In this case 4777 packets have been sent into 94 bursts inside two typical scenarios. The first scenario is a local network inside the Universidad Autónoma of Madrid (UAM), whereas the second scenario is a transmission between a Madrid, Spain node and another one in Eötvös Loránd University in Hungary through the Internet.

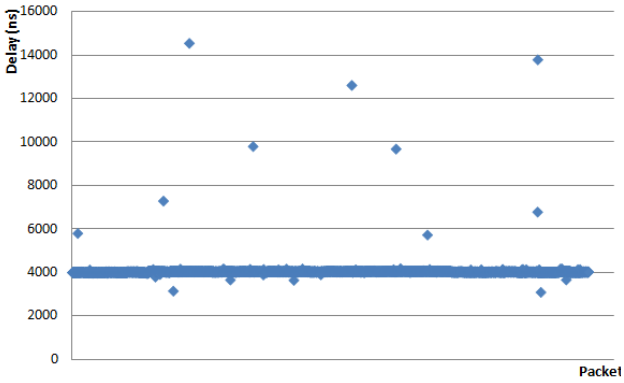
In a local network scenario, the delay distribution between different packets is very close. As can be seen in fig. 11 the boxplot shows the quartiles, that represents the 75% of the results (sorter lines), limits very close to the median (blue rectangle), 50% of the data results, which is practically equal to the average (the rectangle cannot be seen), atypical results are a few (points at both sides of the quartiles lines).

Fig. 11. One way local delay boxplot



In the figure 12 is presented the delay for each packet transmission describing an horizontal line that almost corresponds to a constant delay line, with a few exceptions corresponding to the atypical results.

Fig. 12. Solution design



For the second scenario, measuring delays in the Internet, the boxplot, figure 13, shows a less close distribution parameters, where them can be seen better than in the local one. The quiartile lines (smaller lines) are further from the median (blue section), which can be distinguised from the average. The atypical result count has been increased. In the figure 14 the results for each packet are clerarer, where the line is wider and appears a noise because of the distribution. This distribution appears as a result of the enrouting in the Network between nodes that are not in a local scenario, where the delay is minimum and the path resources do not vary as in the Internet.

Fig. 13. Solution design

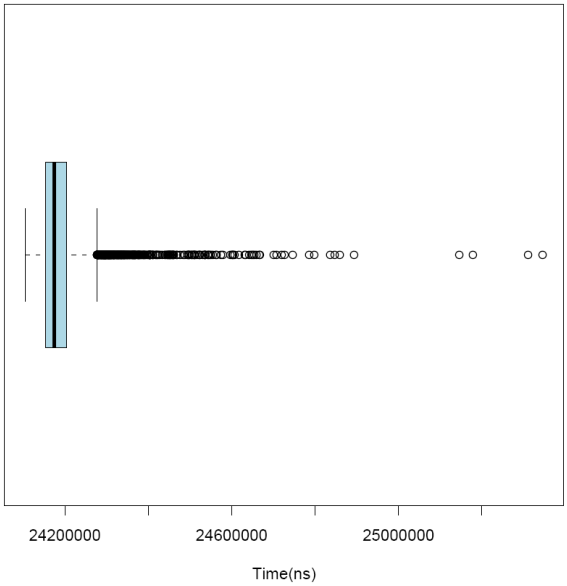
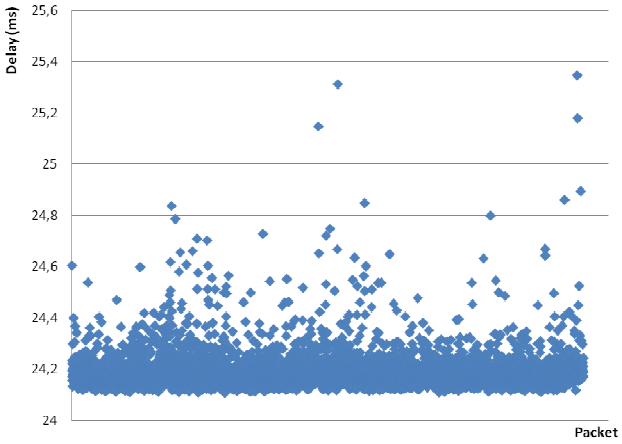


Fig. 14. Solution design



5. CONCLUSIONS

ARGOS provides a highly accurate timestamping solution that enables big Internet measurement testbeds to work. Project such as OneLab2, based on ETOMIC, provides users a testbed with the precision needed to run their experiments between nodes which are geographically located around the whole Europe. This project uses in their nodes NetFPGA-based ARGOS NICs with GPS timing, a low cost solution to achieve a timestamping as accurate as tens of nanoseconds. A big advantage of ARGOS is that timestamping is managed using the standard structures inside Linux Kernel, so any conventional application such as Wireshark can be used to analyze the captured packets. Although NetFPGA does not provide a GPS port, we showed how a sister card can be designed to add these capabilities by means of the debug port of NetFPGA.

Apart from research applications, measurements such us one-way delays or available bandwidths are very useful for companies to know the QoS they are offering. Finally, future work includes the development of a newer version compatible with the existing NetFPGA developments, to be added to the collection of open-source projects of the NetFPGA community.

6. REFERENCES

[1] "Network Time Protocol project", http://www.ntp.org/

[2] Mills, D.L.; , "Internet time synchronization: the network time protocol," Communications, IEEE Transactions on , vol.39, no.10, pp.1482-1493, Oct 1991

[3] Jiho Han; Deog-Kyoon Jeong; , "Practical considerations in the design and implementation of time synchronization systems using IEEE 1588," Communications Magazine, IEEE , vol.47, no.11, pp.164-170, November 2009

[4] "National Marine Electronics Association ", http://www.nmea.org/

[5] "uBlox GPS modules", http://www.u-blox.com/en/gps-modules.html

[6] "Xilinx Inc. MicroBlaze Processor Reference Guide", http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf.

[7] Vern Paxson, "Measurements and Analysis of End-to-End Internet Dynamics", PhD thesis, University of California, Berkeley, April 1997.

[8] M. Zieliński, M. Kowalski, R. Frankowski, D. Chaberski, S. Grzelak and L. Wydźgowski, "Acumulated jitter measurement of standard clock oscillators", Metrology and Measurement Systems, in press, 2009.

[9] "OneLab, Future Internet Test Beds", http://www.onelab.eu/

[10] "Planet Lab Europe", http://www.planet-lab.eu/

[11] "European Traffic Observatory Measurement Infrastructure", http://www.etomic.org/