

CESNET Technical Report 11/2008

Hardware Supported Precise Timestamps Generation

VLADIMÍR SMOTLACHA, JIŘÍ NOVOTNÝ, TOMÁŠ MARTÍNEK

Received 15.10.2008

Abstract

This report describes a system for precise timestamp generation dedicated for high speed network applications. High resolution and accuracy of generator, often required by network applications, is reached such that the critical parts of algorithm are placed into the dedicated hardware circuit. The proposed system was implemented in chip with Spartan3 technology and tested on COMBO-PTM card. The measured clock generator accuracy and stability shows that the system accomplishes requirements to packets timestamping at link speed up to 10Gbps.

Keywords: precise timestamps, network applications, FPGA, VHDL

1 Introduction

A timestamp denotes time in which a particular event has occurred. In most cases we require timestamps that have defined relation to some time scale, (e.g. UTC), therefore the clock which generates timestamp has to be synchronized to that timescale. The timestamps share clock parameters, especially:

- *Resolution* - the smallest step in which the clock is updated,
- *Accuracy* - time difference between reported and 'true' time - the time of referenced timescale.

Nearly all network monitoring and measurement application require timestamping of network packets. Typical examples of such applications are packet logging, network delay and jitter measurement, assessment of packets dynamics (inter-packet delays), and SLA (Service Level Agreement) verification.

In this document we describe a system issuing timestamps suitable for timestamping of data packets in the network. Such system must meet the following conditions:

1. Time resolution fine enough to assign different timestamps to two consecutive packets,
2. Accuracy suitable for the data processing application,
3. Repetition rate high enough to timestamp every packet in the monitored link.

Our timestamping system is designed for 10Gbps Ethernet links. As the smallest packetlength is 64 bytes, we need resolution better than 50ns for 10Gbps resp. 500ns for 1Gbps. Repetition rate for 10Gbps is about 20 mil. timestamps per second. Probably the most accuracy sensitive application is one-way delay measurement - delays in LAN are in the order of hundreds of microseconds, therefore the required absolute accuracy is the order of microseconds.

Packet timestamping for link speeds below 1Gbps can be arranged by the operating system of monitoring computer (assuming the system clock is NTP synchronized) but hardware timestamping unit is unavoidable for link speeds 1Gbps and higher.

2 System Architecture

An overall architecture of the clock and the timestamp generator is shown in Figure 1. The hardware circuit which represents the most important part of it is implemented in an FPGA chip. All registers are accessible through the PCI interface.

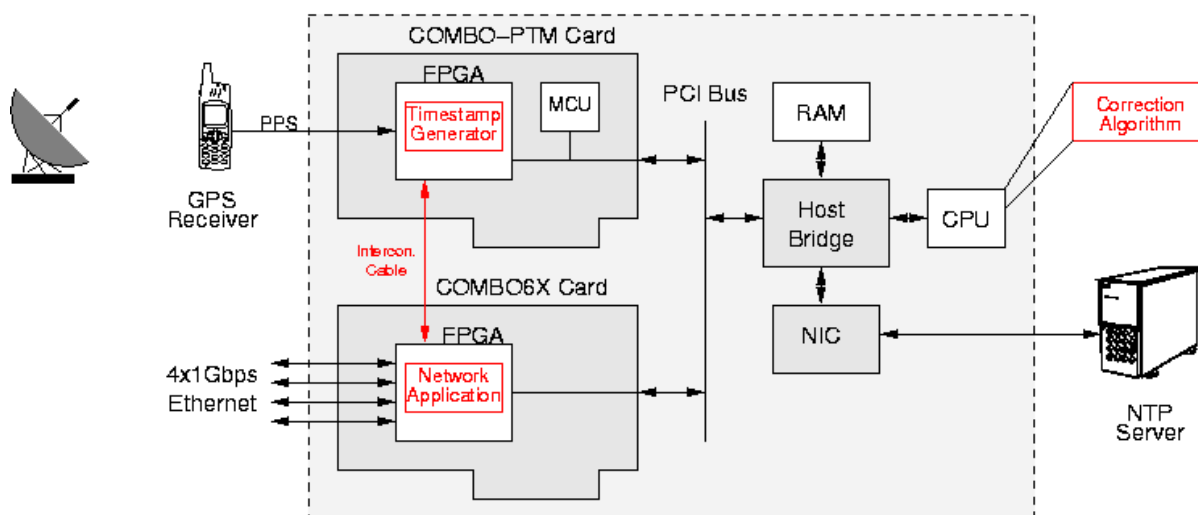


Figure 1. Architecture of the system for precise timestamps generation

The card has interface for PPS (Pulse Per Second) signal that is commonly used for time information transfer: raising edge of the pulse represents start of a second. There is no coding of the second label (i.e. which second it is) in the PPS signal. Source of PPS signal can be any clock (e.g. radio clock, atom clock) but a GPS receiver is utilized most often.

The synchronization itself is software implemented - the routine manipulates with the hardware registers. The synchronization algorithm is neither time critical nor computation intensive and therefore can be executed on the host CPU. Detailed description of the synchronization algorithm is depicted in Section 5.

Prior start of the generator, it is necessary to initialize the clock time by one of the following ways (in preference order):

1. GPS receiver if it generates labels of seconds on the serial output,
2. NTP protocol if the computer is connected to the Internet,
3. Any other external device that provides serial time information in some common format,
4. If no accurate time information is available, the card uses just the host system time.

More detailed description of the initialization process is presented in Section 4.

Note: The PTM card contains the embedded microcontroller (MCU) connected to FPGA. Based on the requirements of a target application, it is possible to move the synchronization algorithm from the host CPU to the this MCU. Similarly, the initialization process, which utilizes the GPS receiver, can be realized on MCU directly and thus the timestamp generator can run independently of the host CPU.

The COMBO-PTM can be used in two basic models:

1. *The PTM card utilized by a software application.* The PTM card works as an accurate and stable clock. The software application reads the time through the PCI bus and uses it for further processing or distribution. Typical example of is a primary NTP server utilizing COMBO-PTM card as a clock. Please note that time reading is influenced by the varying delay of the PCI bus (that might reach several microseconds) which decreases the real clock accuracy.
2. *The PTM card connected to another COMBO6 family card* Such other card is typically dedicated for acceleration of network applications. In this case, timestamps are transferred from the PTM to the processing card using dedicated bus. The processing card receives timestamps and assigns them to each packet incoming from network interfaces. Examples of such applications include: a measurement of inter-packet gaps, a measurement of the active components latency, etc.

Note: Future COMBO6X card might contain a precise crystal and GPS input. In this case, it will be possible to implement the COMBO6-PTM functionality on that card directly.

3 Timestamps generation

A hardware architecture of PTM card is shown in Figure 2. The card is equipped with a temperature compensated oscillator (TCXO) 10 MHz with frequency stability better than 1 ppm (i.e., 10^{-6}) in the range of normal ambient temperatures. This frequency 10 MHz is multiplied n -times inside the FPGA such that the PTM clock operates at frequency $SCLK=n*10$ MHz (where n is usually set to 6, 8 or 10).

The actual time is stored in a Real Time Register (RTR). This register is 96 bits wide and contains the value split into two parts. The upper 32 bits represent the number of UTC seconds since 1.1.1970 00:00:00 UTC (i.e. the standard Unix time scale). The lower 64 bits represent the fragment of the second. Note, the standard timestamp format requires 64 bits only. The extension of the fragment part from 32 bits to the 64 bits is due to the higher precision of internal time representation.

The clock operates in following steps: Each cycle of SCLK clock signal, the RTR register is incremented with a 64-bit number stored in an Incremental register (INCR). The content of INCR register should be set to the proper number, roughly $1/(n*10^7)$, in order to accumulate 1 second in RTR after $n*10$ million incremental cycles.

Example:

Suppose the SCLK frequency is 100MHz and the RTR is initialized to zeros. The INCR initial value will be:

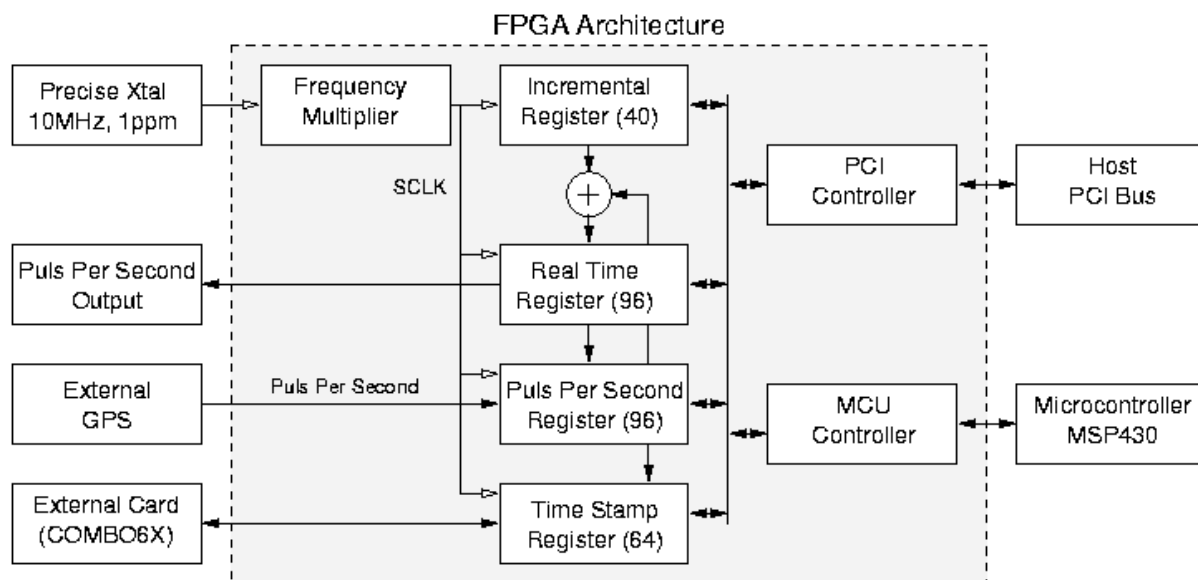


Figure 2. Hardware architecture of precise timestamps generator

$$INCR = 2^{64}/SCLK = 2^{64}/100 \cdot 10^6 = AF31DC461 \text{ (hex)}$$

In the first SCLK cycle (after the INCR is added) the RTR value is:

0000 0000 0000 002A F31D C461

In the second SCLK:

0000 0000 0000 0055 E63B 88C2

etc.

The proposed scheme of timestamps generation works precisely with the following preconditions only:

1. The frequency of the crystal is absolutely stable.
2. The result of the fraction for INCR register computation is integer value.

However, both of these preconditions are not satisfied and therefore the content of RTR register deviates from precise time gradually. The periodical correction of the RTR register is necessary. The detailed description of the correction mechanism is depicted in Section 5.

An application running on the host CPU or on the external card usually require the reading of the RTR register content on demand. For this reason, the hardware circuit provides an atomic copy of the upper 64 bits of RTR register to the auxiliary Timestamp register. The direct reading of RTR register is not possible, because the application would read the upper 64 bits of register via at least two 32-bit accesses. However the content of the RTR register is changing every SCLK clock cycle and therefore it is most likely the application would read the different parts of the different timestamps.

Please note that the resolution (not accuracy) of the 64 bits output timestamp is generally 232 ps. However, the hardware circuit for timestamps generation is

synchronized with SCLK signal, which decreases this resolution capability to the order of nanoseconds. By changing the multiply ration of input precise clock signal, the resolution can be increased arbitrary, however the resolution in orders of tens of nanoseconds is sufficient for most applications.

4 Clock Initialization

In order to achieve correct real time in the RTR register, it is necessary to initialize this register with an appropriate value before the timestamps generation starts.

The upper 32 bits of the RTR register represent the number of seconds since 1.1.1970 00:00:00 UTC. The value of actual date can be therefore calculated via number of days, hours and minutes with respect to general rules for leap years.

Example:

Suppose the date 1.1.2008, 00:00:00 UTC. The elapsed time since 1.1.1970 corresponds to 1199145600 seconds, so the RTR register have to be initialized to the hexadecimal value:

4779 8280 0000 0000 0000 0000

The actual time in seconds can be acquired in several ways:

1. If the computer is connected to the Internet, the actual time can be obtained for instance using NTP protocol (with accuracy cca 10 ms) or from others precise time sources (e.g. atomic clock)
2. Direct reading of the NMEA protocol sentence, which is the standard of GPS receiver communication protocol. This approach is more complicated, because it requires the hardware support of serial data reading and NMEA parsing inside the FPGA chip. It is necessary in cases when the computer is not connected to the Internet.

The initialization of the RTR register with accuracy inside a second is sufficient. The synchronization algorithm is capable to set the fragment part of timestamps with very high accuracy (see description in Section 5).

5 Clock synchronization

The clock synchronization is based on feedback control loop whose principle is shown in Figure 3.

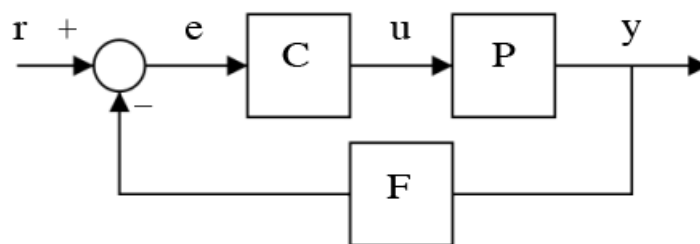


Figure 3. Control loop

Symbols mapping to our device:

- **y** - clock output, i.e. the time (or phase)
- **u** - oscillator frequency
- **e** - error, difference between captured PPS signal and local time
- **r** - reference, the PPS signal
- **P** - controlled process, i.e. clock. *Note: it is just the integrator*
- **C** - controller
- **F** - feedback, carries output phase to the comparator

We decided to utilize standard PI regulator, therefore **C** consists of proportional term and integral term and is described by just two parameters: the proportional gain C_p , and the integral gain C_i .

In our case, the clock oscillator frequency is represented by the SCLK frequency and INCR register. The SCLK is free running crystal oscillator, while INCR is periodically adjusted by the regulator. The error *err* is just the value of PPSR (PPS Register). The PI regulator calculates a *drift* which is the relative frequency difference between real and nominal SCLK value:

$$drift = drift + err \cdot C_i$$

and the INCR is adjusted by *adj*:

$$adj = -drift - err \cdot C_p$$

Although *adj* is calculated every second (assuming that PPS is present), the INCR adjustment is done in longer periods, usually one minute. The reason is to keep the noise of clock jitter small.

The regulator tuning means to change gains C_p and C_i . They depend on characteristics of the utilized crystal, mainly its frequency stability and dependence on temperature. We found out as the result of applied theory and manual tuning, that

$$C_i = 2^{-18}, C_p = 2^{-8}$$

are close to optimal values.

Described algorithm is currently implemented as the host computer program but it can be alternatively executed by the microcontroller MSP430 which is placed on the ComboPTM card.

6 Extension for External Cards

The external card communicates with PTM card using interconnection cable composed of 12 wires (see Figure 4). With respect to the limited number of wires, it is not possible to transfer whole 64 bits of timestamp value to the external card at once. For this reason a simple communication protocol containing two modes (Init and Fast) was developed.

By default, the communication operates in Fast mode. In this mode the 8 lower bits of the Timestamp Register (Figure 2) is transferred through the 8 bits data bus (TS_DATA) continuously. Since the resolution capability of the fragment part is 232ps, the lower 8 bits of the timestamp overflow roughly every 60ns. Indication

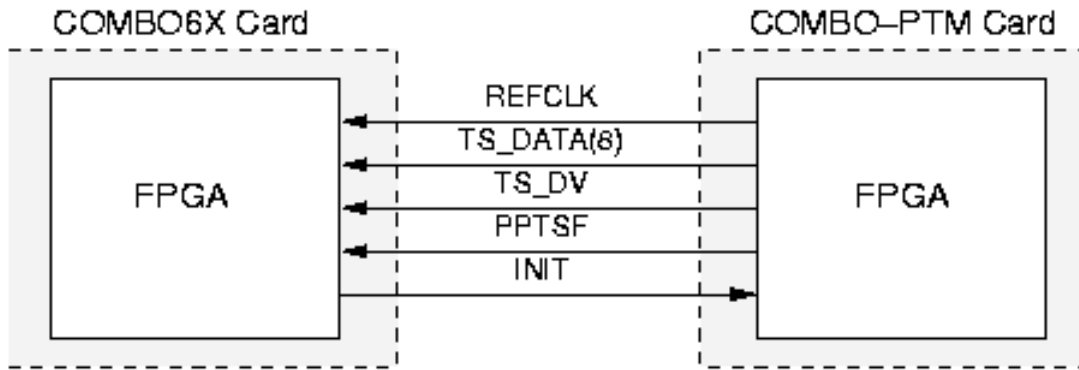


Figure 4. Interconnection scheme between the PTM and the external card

of the overflow is therefore transferred through a separate wire PPTSF (Pulse Per Timestamp Fraction). The receiving circuit placed on the external card is responsible for continuous updating of the lower timestamp part and simultaneously it has to increment the upper part of the timestamp at the moment of the overflow.

The Init mode is triggered by external card only (pulse on the INIT wire). In this mode the PTM card transmits the whole content of timestamp through 8 bits data bus (TS_DATA) continuously together with enabled valid signal (TS_DV). Since the initialization requires at least 8 steps for transfer of all 64 bits of data, the transferred timestamp value becomes invalid even during the transfer itself. For this reason, the information about the overflow of the lower 8 bits (PPTSF) is transferred together with data (in the same way as in the Fast mode). The receiving circuit on the external card is responsible for watching these pulses and incrementing of the upper part of the timestamp. As soon as the Init mode is finished, the card is switched back to the Fast mode automatically.

7 Evaluation and Results

Subject of our measurement was the card performance when it is used as a clock (mode 1 in Section 2).

7.1 Test 1 - ComboPTM clock reading

We measured duration of ComboPTM clock reading by the user level program. We wrote a simple test program that reads system clock (i.e. the system function `gettimeofday()`) then two times reads the PTM clock and system clock again.

Example:

```
# ./ptm_time_2
Device name: /dev/combosix/0
systemtime: 1227049107.794724
time: 1227049107.794724226
time: 1227049107.794728041
systemtime: 1227049107.794736
last_pps: -0.000000052
```

#

We see that PTM clock reading took about 3.8 microseconds and that last offset of PPS input signal was 52 nanoseconds. Also, that duration of PTM clock reading is about the same as system clock reading (difference 12 microseconds between two system values includes two PTM clock reading and one system clock reading). Statistical processing of several thousands of such measurements gave mean value 3.91 microseconds with standard deviation 0.04 microseconds at the Pentium4 2.6 GHz computer.

7.2 Test 2 - Time and frequency offsets

The PD regulator does digital correction of the free running oscillator. The goal is to keep time offset to PPS input signal as small as possible while the clock frequency as stable as possible. Adjustment of the control loop parameters is compromise of both requirements. Graphical representation of 10 days of measurement is shown on Figures 5 and 6.

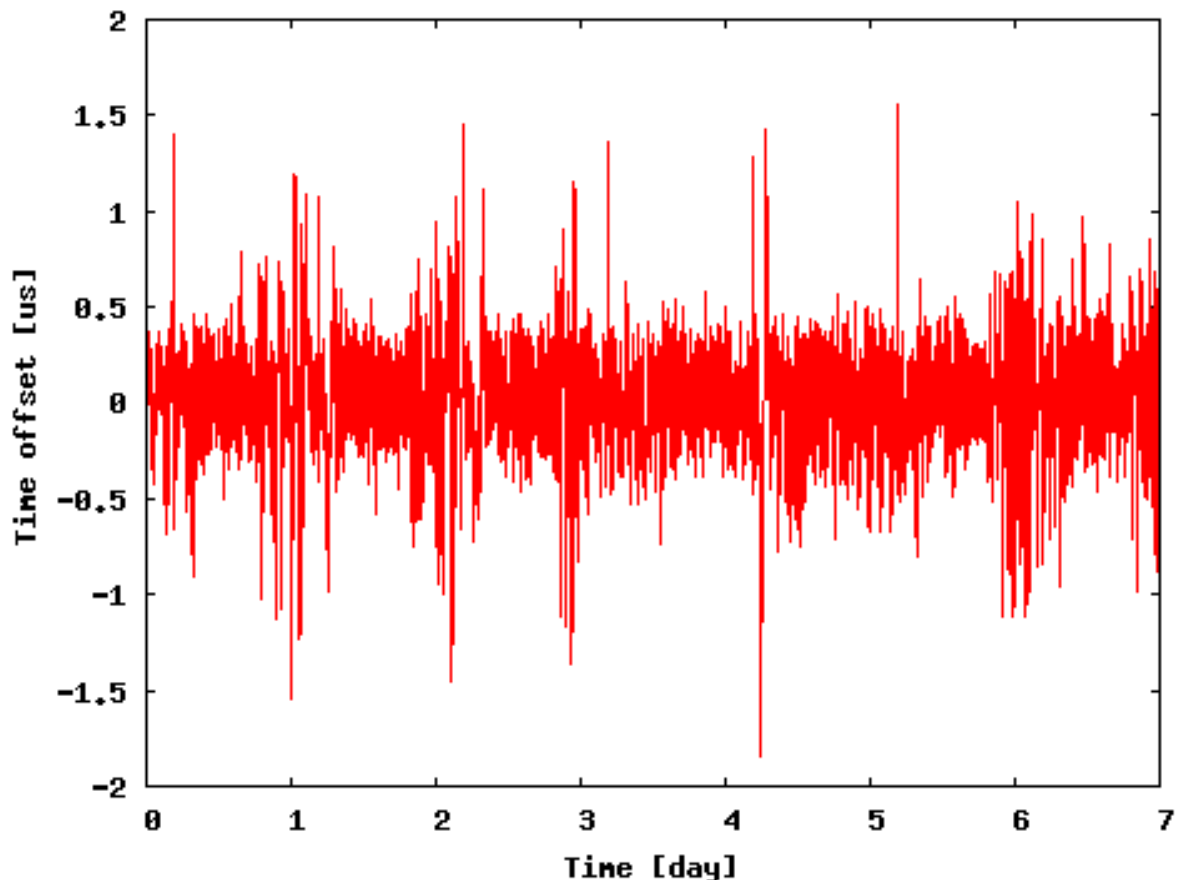


Figure 5. Time offset of ComboPTM clock disciplined by GPS receiver

We conclude that the time offset was generally between -0.5 and 0.5 microseconds with occasional peaks up to ± 1.5 microseconds. The frequency offset (caused mainly by varying ambient temperature) does not exceeded 0.02 ppm - i.e., $2 \cdot 10^{-11}$.

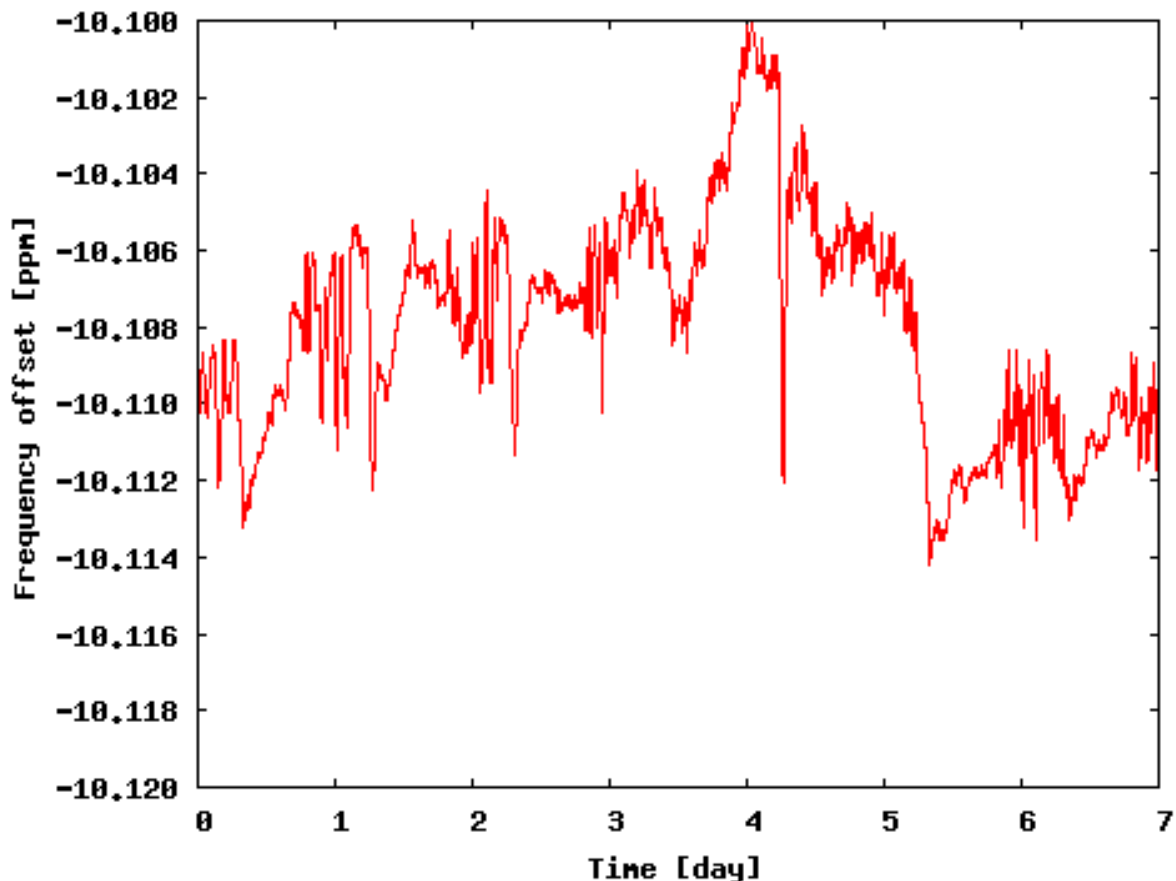


Figure 6. Frequency offset of ComboPTM clock disciplined by GPS receiver

8 Conclusions

We verified that the new synchronization algorithm for ComboPTM clock is functional and efficient. As we programmed it as a digital PI regulator, the code is easy to manage or tune in the future. The measured clock accuracy and stability meets our expectations and is convenient to packets timestamping at link speed up to 10Gbps.

Further improvement of accuracy is possible but it would require to take into account detailed model of the TCXO module. We also plan to move the synchronization routine to on-board processor.

References

- [1] Mills, David L.: Computer Network Time Synchronization: the Network Time Protocol, CRC Press 2006, 304 pp., ISBN 10: 0-8493-5805-1
- [2] Pantry S., Griffiths P.: The Complete Guide to Preparing and Implementing Service Level Agreements, 208pp, 2001, ISBN 1-85604-4106
- [3] Novotný T., Smotlacha V., Bardas R.: Schematic of COMBO-PTM, Technical Report 15/2003¹, Praha: CESNET, 2003.

¹ <http://www.cesnet.cz/doc/techzpravy/2003/comboptmschematic/comboptmschematic.pdf>