# Fast-Convergence Microsecond-Accurate Clock Discipline Algorithm for Hardware Implementation

J. Viejo, J. Juan, M. J. Bellido, A. Millan, and P. Ruiz-de-Clavijo

***Abstract*—Discrete microprocessor-based equipment is a typical synchronization system on the market which implements the most critical features of the synchronization protocols in hardware and the synchronization algorithms in software. In this paper, a new clock discipline algorithm for hardware implementation is presented, allowing for full hardware implementation of synchronization systems. Measurements on field-programmable gate array prototypes show a fast convergence time (below 10 s) and a high accuracy (1 $\mu$s) for typical configuration parameters.**

***Index Terms*—Field-programmable gate array, hardware timestamping, Network Time Protocol (NTP), Precision Time Protocol (PTP), synchronization system.**

## I. Introduction

Time synchronization of electronic equipment is required in many applications. Typical examples are distributed measurement and control systems [1] and the synchronized phasors (synchrophasors) applied to power distribution networks [2], [3]. In these systems, measurement and control equipment is synchronized using specific protocols through the communication network itself, which is usually a switched Ethernet network [4]. Time synchronization over Ethernet networks uses two primary synchronization protocols. On the one hand, the industry norm IEC 61850 [5] defines the Simple Network Time Protocol (NTP) (SNTP) [6] as a standard way to synchronize a set of substations with a time server. SNTP is a simplified version of the more general NTP [7] that is commonly used in Internet servers and routers. On the other hand, the IEEE 1588 standard [8] specifies the Precision Time Protocol (PTP) that was designed specifically for industrial applications. These protocols are based on the exchange of packets with a time server; these packets contain a set of timestamps used for calculating the round-trip time and the time offset of the server's clock relative to the client's clock. Both NTP and PTP implement sophisticated algorithms to adjust the local clock and maintain the offset at a minimum. PTP is more focused on local area networks (LANs) and is typically hardware supported, so it can achieve much better accuracy than NTP, although a similar accuracy is achievable with NTP/SNTP if the same restrictions and hardware support are applied [6].

Recent research works have demonstrated that dedicated hardware support is necessary to achieve high synchronization precision [1], [9]. These authors define a set of synchronization functions completely implemented in hardware which can be applied for timestamping and controlling the local clock. However, higher level tasks like clock disciplining and communication protocols are implemented in software [9]. Moreover, a full or near-full hardware implementation of a synchronization process can enable more accurate, robust, power-efficient, and cost-effective synchronization systems. As a result, it is necessary to design a hardware-friendly algorithm for synchronization.

In this paper, a clock discipline algorithm for hardware implementation aiming at microsecond accuracy is presented. The algorithm is easy to implement in hardware, has very fast convergence, and is able to maintain a very accurate local time compared to a time server, by using standard digital design methodologies.

This paper is organized as follows. In Section II, the clock model and the drift control mechanism are presented. In Section III, the synchronization algorithm is described. Section IV presents some experimental results, and conclusions are summarized in Section V.

## II. Hardware Clock Model and Drift Control Mechanism

The implemented clock model is based on the computer clock model presented in [10] which is formed by a voltage-controlled oscillator (VCO), a local clock implemented in hardware, and a prescaler that reduces the oscillator frequency to a convenient frequency [Fig. 1(a)]. In this model, the frequency is controlled by a digital/analog converter (DAC), and the time is adjusted by adding the offset to the local clock. However, this model is not suitable for digital implementation, as additional external hardware components are needed. Thus, the proposed alternative is to replace the VCO and DAC with an oscillator operating at a fixed frequency and to include a digital module (*drift control*) which is in charge of controlling the oscillator frequency. The designed hardware clock model is shown in Fig. 1(b).

In the proposed scheme, time adjustment can be performed by directly adding the offset to the local clock. However, this adjustment should only be done when the local time is substantially different from the reference time since this action violates the monotonic requirement. Smaller adjustments are done by varying the local clock frequency using the drift control module. In this module, first, the prescaler is in charge of reducing the oscillator frequency $f_{\mathrm{osc}}$ to a frequency $f_{\mathrm{pres}}$ slightly higher than the nominal operating frequency $f_{\mathrm{nom}}$ of the local clock. Since the resolution of the local clock in its fractional part is $n$ bits, $f_{\mathrm{nom}} = 2^n$. The prescaler reduction factor is controlled through a configuration parameter called *prescaler factor*;

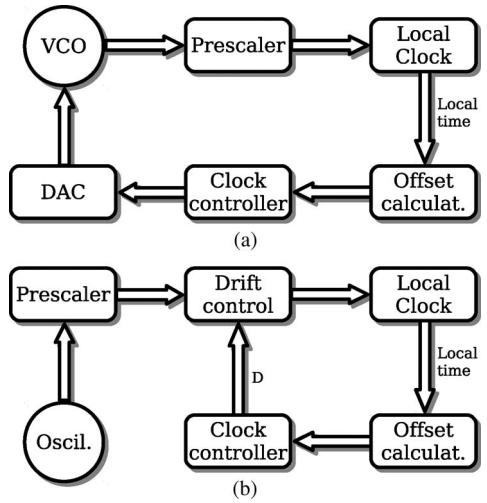


Fig. 1. Hardware clock models. (a) Computer model based on a VCO. (b) Proposed hardware clock model based on a fixed-frequency oscillator.

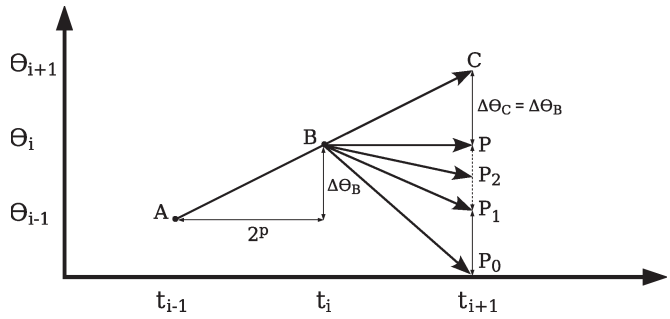


Fig. 2. Fundamentals of the calculation of corrections.

this parameter allows it to adapt this module to different local clock resolutions and system clock frequencies. Once this reduction is carried out, the drift control module performs a fine adjustment using the parameter $D$ calculated by the clock controller module. The parameter $D$ represents the number of cycles of the clock signal generated by the prescaler to be eliminated in a second. Thus, the frequency obtained by the drift control module $(f_{\text{out}})$ represents an averaged frequency which can be calculated according to

$$f_{\text{out}} \approx f_{\text{pres}} - D. \tag{1}$$

## III. Correction Calculation and Synchronization Algorithm

### A. Correction Calculation

To keep the local clock tuned to a reference clock, the adjustment parameter $D$ is corrected at every adjustment interval so that the estimated offset $(\theta)$ of the local clock with respect to the reference clock is minimized. The fundamentals applied for the calculation of this correction are shown in Fig. 2. This figure shows the estimated offset at different adjustment instants done every $2^p$ s, where $p$ is a configuration parameter that controls the adjustment interval. Assuming that, at time $t_{i-1}$, the offset is $\theta_{i-1}$ and $D$ is $D_{i-1}$ and that, at time $t_i$, the offset is $\theta_i$ (point $B$), a new value of $D$ $(D_i)$ should be calculated in order to cancel the drifting $(\Delta\theta_B)$. The correction $\Delta D_i$ applied to $D$ (2) is divided into two components according to (3)

$$D_i = D_{i-1} + \Delta D_i \tag{2}$$
$$\Delta D_i = \Delta D_i^0 + \Delta D_i^C. \tag{3}$$

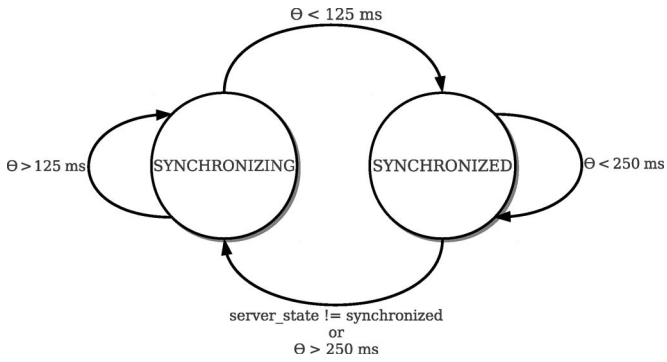


Fig. 3. States of the synchronization algorithm.

First, the correction needed to stop the drifting $(\Delta D_i^0)$ can be calculated according to (4). With this correction, the offset is expected to stay constant after a new interval (point $P$ in Fig. 2). The second component of $\Delta D_i$ $(\Delta D_i^C)$ is intended to make the offset to converge to zero in the next adjustment intervals. The convergence component is calculated in (5), where $-(\theta_i/2^p)$ is the adjustment that would make the system converge into one interval $(P_0)$ and $q$ is a smoothing parameter that controls the convergence speed, so that $q = 1$ makes the system converge into two intervals $(P_1)$, $q = 2$ into four $(P_2)$, and so on

$$\Delta D_i^0 = -\frac{\Delta\theta_B}{2^p} = -\frac{\Delta\theta_C}{2^p} = -\frac{\theta_i - \theta_{i-1}}{2^p} \tag{4}$$
$$\Delta D_i^C = -\frac{\theta_i}{2^p} \times \frac{1}{2^q} = -\frac{\theta_i}{2^{p+q}}. \tag{5}$$

### B. Synchronization Algorithm

The objectives of the synchronization algorithm described in the following are to control the hardware clock model presented in Section II so that it reaches synchronization with a clock reference and to maintain this synchronization as accurate as possible by applying the correction calculated previously.

At the beginning of the operation, the system starts in SYNCHRONIZING state (Fig. 3). In this state, if the offset is greater than 125 ms, a hard correction of the local clock is done in order to match the server's clock time $(t_{\text{local}} = t_{\text{local}} + \theta)$. When the offset is less than 125 ms, the system transitions to SYNCHRONIZED state, and the local clock is kept synchronized to the reference clock by using the drift control mechanism and calculating the appropriate correction $(\Delta D)$ as described previously.

If communication with the reference clock is lost or the offset is greater than 250 ms, the system changes to SYNCHRONIZING state, and the convergence component $\Delta D^C$ is removed from the correction to keep the local clock drifting as small as possible until synchronization is reestablished.

## IV. Results

In order to check that the algorithm works correctly, this has been implemented on a Spartan-3E field-programmable gate array (XC3S500E) as part of an experimental SNTP client. This SNTP client has been tested against a hardware-assisted SNTP server which synchronizes its local clock using a GPS receiver. Thus, different tests have been carried out which consist of varying the values of the adjustment interval and the convergence factor. For each test, approximately 1000 successive offsets have been acquired. In Table I, the numerical values of the mean offset as seen by the client, together with the error, calculated as three times the standard deviation of the measurements are shown.

TABLE I
MEAN OFFSET ± ERROR IN MICROSECONDS

| | $q=0$ | $q=1$ | $q=2$ | $q=3$ |
|---|---|---|---|---|
| $p=0$ | $0.06 \pm 1.21$ | $0.06 \pm 1.17$ | $0.07 \pm 0.95$ | $0.05 \pm 0.85$ |
| $p=1$ | $0.06 \pm 1.49$ | $0.06 \pm 1.72$ | $0.07 \pm 1.18$ | $0.02 \pm 1.27$ |
| $p=2$ | $0.07 \pm 1.71$ | $0.05 \pm 1.51$ | $0.07 \pm 1.69$ | $0.20 \pm 1.81$ |
| $p=3$ | $0.07 \pm 2.49$ | $0.10 \pm 2.27$ | $0.14 \pm 2.98$ | $0.31 \pm 4.23$ |
| $p=4$ | $0.04 \pm 3.88$ | $0.23 \pm 4.34$ | $1.01 \pm 6.53$ | $0.55 \pm 14.86$ |
| $p=5$ | $0.18 \pm 6.74$ | $0.20 \pm 8.73$ | $0.58 \pm 16.42$ | $3.91 \pm 28.94$ |
| $p=6$ | $0.24 \pm 13.18$ | $0.59 \pm 17.97$ | $1.16 \pm 35.29$ | $9.19 \pm 64.45$ |

TABLE II
TIME TO SYNCHRONIZATION IN SECONDS

| | $q=0$ | $q=1$ | $q=2$ | $q=3$ |
|---|---|---|---|---|
| $p=0$ | 4 | 8 | 15 | 29 |
| $p=1$ | 10 | 18 | 34 | 70 |
| $p=2$ | 16 | 40 | 76 | 160 |
| $p=3$ | 32 | 80 | 176 | 360 |
| $p=4$ | 64 | 176 | 384 | 800 |
| $p=5$ | 128 | 416 | 832 | 1760 |
| $p=6$ | 256 | 896 | - | - |

It can be seen that the mean offset and error increase for larger values of $p$ because the accumulated offset in an adjustment interval is proportional to the interval duration. For intervals between 1 and 8 s ($p = 0, 1, 2, 3$), the convergence factor $q$ is not relevant because the mean offset and the error are going to be small in any case and the obtained error is always less than 5 $\mu$s. For the other intervals, the accumulated offset between adjustments can be higher, and therefore, carrying out faster corrections (using small values of $q$) can be more appropriate.

The time to full synchronization, considered as the time spent by the algorithm from the start of the operation to reach an offset below 25 $\mu$s, is shown in Table II. These times depend on the adjustment interval, obtaining better results for smaller values of $p$. As it has been commented earlier, this is because, using a small adjustment interval, the corrections are performed more often, accelerating the convergence process. Moreover, the time until synchronization also depends on the convergence factor since larger values of $q$ cause slighter corrections, so that the algorithm takes longer to reach the desired synchronization. In the worst cases ($p = 6$ and $q > 1$), a synchronization better than 25 $\mu$s is never reached.

In a typical LAN configuration, $p = 0$ and $q = 2$ are adequate values of the algorithm's parameters, providing a local clock accuracy of 1 $\mu$s, which is among the better results reported for hardware-assisted SNTP implementation (1–25 $\mu$s) [4], and largely improving the better results of the software NTP implementation (tens to a few hundreds of microseconds) [11], [12].

Under these same conditions, the time to synchronization is only 15 s for an adjustment interval of 1 s. This time compares very well to the drift adjustment time of about 10 s reported in [9] for a PTP implementation with a 125-$\mu$s synchronization interval.

## V. CONCLUSION

In this paper, a new clock discipline algorithm that is suitable for hardware implementation has been presented. The algorithm includes a drift control mechanism and a hardware clock model that have been implemented in programmable hardware as part of an experimental SNTP client. Measurements show that the algorithm is able to converge to an accuracy below 25 $\mu$s in less than 10 s and to maintain an accuracy of 1 $\mu$s for typical configuration parameters.

## REFERENCES


[1] J. Han and D. Jeong, "A practical implementation of IEEE 1588–2008 transparent clock for distributed measurement and control systems," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 2, pp. 433–439, Feb. 2010.

[2] A. Carta, N. Locci, C. Muscas, and S. Sulis, "A flexible GPS-based system for synchronized phasor measurement in electric distribution networks," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 11, pp. 2450–2456, Nov. 2008.

[3] A. Carta, N. Locci, C. Muscas, F. Pinna, and S. Sulis, "GPS and IEEE 1588 synchronization for the measurement of synchrophasors in electric power systems," *Comput. Stand. Interfaces*, vol. 33, no. 2, pp. 176–181, Feb. 2011.

[4] T. Skeie, S. Johannessen, and Ø. Holmeide, "Highly accurate time synchronization over switched Ethernet," in *Proc. 8th IEEE Int. Conf. ETFA*, Antibes-Juan les Pins, France, Oct. 2001, pp. 195–204.

[5] *IEC 61850 Communication Networks and Systems in Substations*, I.E.C. Technical Committee 57, IEC 61850 Edition 2 and other extensions, Jun. 2008.

[6] D. L. Mills, *Simple Network Time Protocol (SNTP) version 4 for IPv4, IPv6 and OSI*, RFC 4330 (Informational), Jan. 2006.

[7] D. L. Mills, J. Martin, J. Burbank, and W. Kasch, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, RFC 5905 (Standards Track), Jun. 2010.

[8] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE, PTP Version 2 (1588-2008), 2008.

[9] S. Meier, H. Weibel, and K. Weber, "IEEE 1588 syntonization and synchronization functions completely realized in hardware," in *Proc. IEEE ISPCS*, Ann Arbor, MI, Sep. 2008, pp. 1–4.

[10] D. L. Mills, "Modelling and Analysis of Computer Network Clocks," Elect. Eng. Dept., Univ. Delaware, Newark, DE, Rep. 92-3-1, 1992.

[11] D. L. Mills, "Adaptive hybrid clock discipline algorithm for the Network Time Protocol," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 505–514, Oct. 1998.

[12] D. L. Mills and P. H. Kamp, "The nanokernel," in *Proc. PTTI Appl. Plan. Meeting*, Reston, VA, Nov. 2000, pp. 423–430.