IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 28, 2020, accepted March 20, 2020, date of publication March 26, 2020, date of current version April 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983421

An Adaptive Fuzzy-PI Clock Servo Based on IEEE 1588 for Improving Time Synchronization Over Ethernet Networks

VINH QUANG NGUYEN, TON HOANG NGUYEN, AND JAE WOOK JEON, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 440-746, South Korea

Corresponding author: Jae Wook Jeon (jwjeon@yurim.skku.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2002R1A2C3011286).

ABSTRACT A network-based control system includes many connected devices, and time synchronization plays an important role in the unified operation of those systems. A clock servo needs a guarantee time offset and a clock rate to minimize the time difference between devices. This paper proposes an adaptive clock servo to improve time synchronization. This proposed algorithm uses a fuzzy-based proportional-integral clock servo (fuzzy-PI) based on IEEE 1588 to reduce the frequency of clock compensation. This method adapts the bandwidth and enhances noise reduction, improving both the time offset and rate difference between the slave and master. The node time can synchronize with the master time after just one cycle of synchronization. Experiments validated the effectiveness of the algorithm and demonstrated that the slave can track the master with the mean and standard deviation of the time offset are 0.432ns and 4.402ns. A cycle time of one second is used to ensure a low-bandwidth network. With these results, the number of nodes over a real-time network can be increased.

INDEX TERMS Time synchronization, IEEE 1588, clock servo, fuzzy logic, real-time network.

I. INTRODUCTION

Real-time processing is a big challenge for existing networking systems [1]. The current trend is to replace central systems with distributed systems. In other words, systems with a main/core board that control and process all data are being replaced by systems that include many independent nodes that share information and data over a network [2]. The advantages of distributed systems are a reduction in wire weight and an enhancement in efficiency, flexibility, and reliability. For example, even when industrial settings change, heterogeneous devices can still cooperate through the network.

To work together, all the devices in a system need to warrant a common or global time. Normally, time in an embedded system is determined using a crystal oscillator and a counter. The peripheral and network modules are also created by combining a crystal frequency and a phase locked loop. However, crystals are heterogeneous and their frequencies are unstable because they are affected by temperature, humidity, and usage time; therefore all the devices in a system keep different times [3], [4]. It is difficult to keep time exactly in systems such as robots, integrated circuit assembly systems, and computer numerical control systems. Furthermore, because distributed systems share information across the network, noise from magnetic fields or the operating environment can cause a packet to be lost, so a common time notion is needed for the security of data exchange at each node. Because it has a high rate, large bandwidth, and short cycle time, Ethernet is expected to all the layers in many industrial architectures. When using Ethernet, data are transferred smoothly from high layers to low layers and vice versa [3] with minimal packet loss. However, the period information and schedule information need to be guaranteed. In other words, the time in the network system needs to have the same values in every networked device. Thus, time synchronization is important in real-time systems.

In time synchronization, two problems need to be considered, time offset and time skew [5]. The time offset (or offset) is the relative difference in time value between two devices in a specific moment. Time skew (or rate offset) is the clock drift between time systems with the same normal frequency

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Tang.

and can be called 'the first derivation' of time offset. Phase synchronization compensates for time offset, and frequency synchronization gets clocks running at the same rate. Thus, time synchronization needs to include both phase synchronization and frequency synchronization. Many time synchronization methods have been developed [6]. Time-division multiple access (TDMA) is based on a schedule table and time slots and is commonly used in vehicles [7]. It requires enough memory to save the time table and has an absolute time offset of several microseconds. The network time protocol (NTP) is popular in wide area networks and the Internet. It offers application-level synchronization and uses Coordinated Universal Time as the primary time standard. NTP is used in many device networks to connect many devices over a widespread area, but its time offset can reach up to numerous microseconds because of latency and delays [6]. The IEEE 1588 standard is an attractive new way to synchronize many nodes in one or two steps [8]. The Precision Time Protocol (PTP) is an IEEE 1588 profile used for precision synchronization. It can be implemented using only software or with a combination of assistant hardware and software. When using timestamped hardware, the PTP is precise to the order of nanoseconds. The PTP also has applications in many area such as wireless sensor networks and is attractive for real-time systems that need highly consistent clocks [9], [10].

The PTP is a good solution for phase synchronization when four timestamps are available. However, frequency compensation needs to accommodate small differences in the clock rates. Recent frequency synchronization research has focused on situations in which the time error reaches a bound. Beckhoff tried to modify IEEE 1588 and develop it in an EtherCAT system called a distributed clock that simply used constant maximum and minimum values of 10ns to compensate for the frequencies of the slave nodes [11], [12]. The time offset in that case cannot reach the nanosecond level. ProfiNet also used IEEE 1588 for that purpose [12]. It can compensate for the time offset, but it cannot reduce the rate of the frequency error. Previous research has used proportional integral (PI) compensation to reduce both time offset and frequency differences. PI can be implemented using both software and timestamped hardware [13], [24]. However, the latency and noise that occur during frame exchanges and quantization errors prevent the system time from reaching perfect synchronization. Therefore, filters have been attached to PI compensation, but they cause a time-constant delay before a component reaches the time error boundary [25]–[29]. That slows the initial processing phase of the system and causes lost packets. In a previous method, the compensation parameters were simply defined off-line before the system was implemented, but that is not a good solution for networked systems that contain unstable crystals and which can be affected by environmental factors. Yildirim *et al.* [30] presented an adaptive PI, but that method just adapted the integral parameter by adding or subtracting a constant value in a fixed condition, which provides inadequate compensation parameter tuning. All the parameters of a clock servo affect the stability of the clock system and need on-line determination when the system is running. In the tradeoff between the convergence time and the time offset, fuzzy logic can be used to adapt the clock servo parameters on-line. Fuzzy logic can provide a short convergence time because the training phase is ignored. In this paper, we propose an adaptive method to enhance noise reduction and achieve a small and stable time offset. Our fuzzy-PI clock servo uses fuzzy logic and a PI clock compensation algorithm to tune the PI clock servo parameter on-line and offers full phase compensation and frequency compensation. The fuzzy-PI clock servo is flexible and easy to apply to many devices that need high time synchronization. We implemented it in a real system and compared it with a PI clock servo, the optimal PI clock servo, and the Kalman-PI clock servo. Our results for the time offset, time skew, and convergence time were better than those of the other methods. Using our fuzzy-PI clock servo, the slave could track the master clocks with just 1 cycle of synchronization, and the peak-to-peak time reached 28ns with a low-cost oscillator. To use a small Ethernet bandwidth and fit into a design motion system that has many duties, we chose a cycle time synchronization of one second. Our experimental results show that the fuzzy-PI clock servo can increase the number of nodes that a network can accommodate. Furthermore, our method can easily be extended to other areas, such as wireless sensor networks and the internet of things, in which Ethernet plays a key role. An overall schematic describing the motive of this paper is presented in Fig. 1.

According to our observations, our proposed method has some advantages over previous research.

1) Use of timestamped hardware: The method uses a timestamp in the MAC layer to support high-resolution time capture and ensure low jitter and noise without needing a special physical layer chip.
2) Short time convergence: The time between a master and slave can reach a small time offset within just 1 cycle of time synchronization.
3) Small relative time error: The absolute peak-to-peak time offset can reach a value as small as 14ns when a low-cost crystal oscillator (25MHz) and the best master clock algorithm (BMC) are used.
4) Adaptive, on-line clock servo parameters: All parameters, including the proportional and integral parameters, are tuned on-line to enhance noise rejection and minimize the effects of temperature and crystal age when the system is running. The bandwidth of the clock servo is also adapted to improve the time synchronization performance.

The remainder of this paper is organized as follows. In Section II, we present recent research related to PTP synchronization and discuss algorithms that have been implemented in the past. Section III describes the problems with previous methods and our solutions. In Section IV, we propose an adaptive clock servo to improve time synchronization using a fuzzy-PI controller over Ethernet networks. Several experiments that we performed to validate the proposed

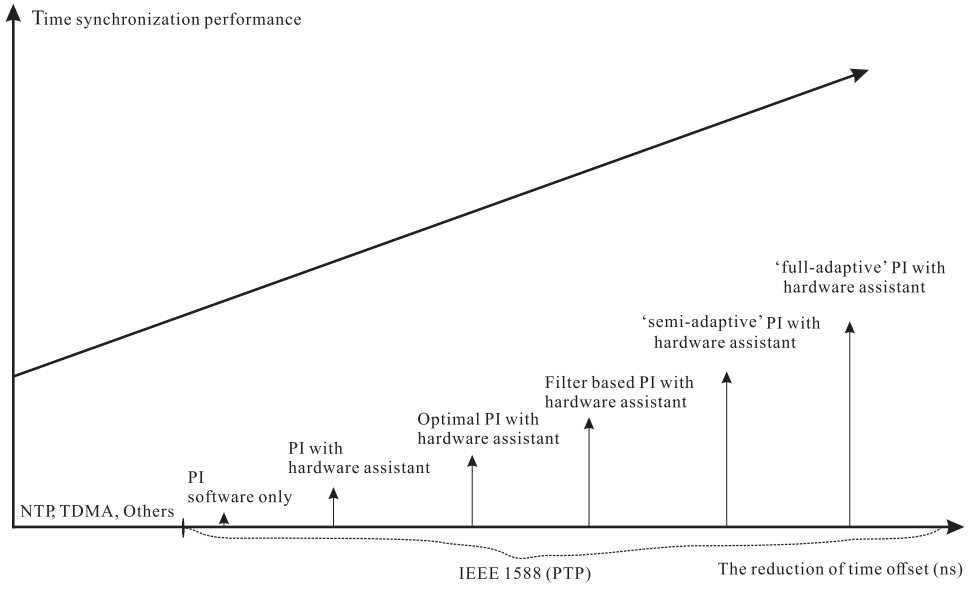

FIGURE 1. Overall time synchronization motivation.

method are described in Section V. The conclusions of this paper are presented in the last section.

II. RELATED WORKS

In this section, we present algorithms used to improve the performance of IEEE 1588. The IEEE 1588 standard has been researched and implemented using both software and hardware [13]–[17]. Because the PTP uses timestamps to correct the time between slave nodes and the master, the accuracy of the synchronization method depends on the timestamped position. Using the software method, the trigger time is near the operating system, the jitter time is relatively large, and the time offset is within 100 microseconds [13]–[15]. With hardware-based timestamps, the trigger time is executed in or near a physical layer, the jitter time is less than in the software method, and the accuracy is on the order of nanoseconds [16], [17].

IEEE 1588 proposes clock offset compensation, but it does not consider frequency compensation, so it cannot provide full time synchronization. Improving its performance is thus an attractive research area. Consider the following clock model, which combines a sample holder and an integral module [9] as follows (1)-(2):

$$G_p(s) = \frac{T_S(s)}{U(s)} = \frac{1 - e^{-sT}}{e^{sT}} * \frac{K_c}{s} \tag{1}$$

$$G_p(z) = \frac{T_S(z)}{U(z)} = \frac{K_c T}{z - 1} \tag{2}$$

Here, K_c is the clock model constant, $T_S(s)$ is the slave time clock output value, $U(s)$ is the input value of clock module and T is the time synchronization interval.

With this model, a PI is used to correct both the offset and the clock rates. A method to use only software to implement this compensation (the PI clock servo) is presented in [13]. It is a simple and linear compensation offering precision inadequate for exact systems. Many factors can affect its precision, including delay time in the operating system, latency, and queue delays. The K_P and K_I parameters belong to the system in an off-line way. Using the integral square error method [18], a deadbeat PI controller (the optimal PI servo) is designed to minimize the integral square time offset [19]–[23]. The K_P and K_I parameters are fixed and belong to the time synchronized interval and clock model parameters, respectively. Using this method, the slave can track the master in 1 cycle of time synchronization, but with a tradeoff between the coverage time and the time offset. The time offset is large, and the clock cannot reduce noise from quantization errors. Chen et al. [24] presented a frequency compensation based on a fraction k of the master time and slave time (3). Their method look likes proportional compensation and offers both offset and rate offset corrections. The algorithms are good, but some noise cannot be reduced. A filter is needed to reject random noise and quantization errors [13], [25]–[28]. One simple and effective way to improve synchronization performance is a low-pass filter [13]. However, it attaches a time constant λ to the responding clock servo (4). A Kalman filter (KF) can also be used to reduce quantization errors [25]–[28]. When using a KF, process noise (Q) and measurement noise (R) need to be determined clearly because the filter performance depends on the ability to correctly determine R and Q. A KF improves the time synchronization performance, but the clock servo still needs many cycles to reach the offset boundary, making it inappropriate for real-time systems containing many devices that need to be synchronized in a short time for initial processing. Moreover, a KF needs to determine the clock servo parameters off-line, which is not good for a noisy network system. PTP implementation has also been considered in a wireless network. Tavares Bruscato *et al.* [29] presented a method using an auto-correction approach,

a prediction mechanism, and an analytical correction mechanism to enhance time synchronization support in wireless sensor networks. Yildirim *et al.* [30], presented an adaptive PI clock servo for a wireless sensor network. That proposal is good, but the adaptive parameter considers only the integral K_I parameter, and K_P is fixed to 1. Moreover, that algorithm simply adds or subtracts a fixed constant with the integral parameter at the design level of the system, making it inadequate for the fluctuations common in networked systems. Offset and frequency drift always exist in networked systems. A method with a short coverage time and full adaptive clock servo parameters is thus necessary. Therefore, using fuzzy logic to adapt the clock servo bandwidth and change the clock servo parameter is the best solution. This algorithm can track the master time in a short cycle interval because it does not use a training process. It is suitable for a real-time system affected by many noise sources.

$$k = \frac{T_n^m - T_{n-1}^m}{T_n^s - T_{n-1}^s} \tag{3}$$

$$G_f(s) = \frac{F_o(s)}{F_i(s)} = \frac{1}{(\lambda s + 1)^n} \tag{4}$$

Here, T_n^m, T_{n-1}^m are the present and previous master time counters, respectively, $F_o(s)$ and $F_i(s)$ are the output and input values of the filter and T_n^s, T_{n-1}^s are the present and previous slave time counters.

III. CURRENT PROBLEMS AND SOLUTIONS

In industrial architectures, information is transferred through many layers. Higher layers need more information but have lower real-time requirements than lower layers. In the bottom layers, which contain many devices, actuators, sensors, drives, motors, etc., the payload is small but the frequency is high [2], [3]. Clock synchronization plays an important role in ensuring accurate signals and data transfer [6]. The IEEE 1588 protocol ensures high performance, which is especially important when using mechanics that require very small errors in operation [8]. When the time is synchronized, the sampling time motion controller is also synchronized, and the message scheduling tasks are managed strictly.

Synchronization can be optimized by using temperature-compensated oscillators or oven-controlled oscillators to supply the local clock [5]. In addition, some applications use a grand master with a high-resolution timestamp in the physical layer. Such systems can improve synchronization in a network, but they also increase the cost [28]. A system that supports the BMC and uses low-cost oscillators can reduce costs when implementing a large number of nodes over a network. For this solution, the clock servo plays a vital role.

The clock servo can be implemented using a proportional-integral compensation, a proportional-integral compensation with a low pass filter, or a proportional-integral compensation with a Kalman filter [13]–[28]. It can also be implemented using frequency compensation (e.g., with a proportional controller) [24]. In any case, the core idea of the clock servo algorithm is that the PI controller provides both rate offset and offset compensation. Moreover, the noise is reduced by the low pass filter or KF, but noise is still present during the PI compensation. In addition, using a low pass filter requires that the time constant error approach the boundary. Using the KF, the time offset can reach a small value for exact processing of the noise covariance (σ_e^2) and measurement noise covariance (σ_m^2) [25]–[28]. A 'semi-adaptive' clock servo based on an additive/subtractive constant value is presented in [30], but it is inadequate for time synchronization when offset and skew still exist at all times. Our motivation in this study is to design a fully adaptive clock servo (i.e., the bandwidth of the clock servo can be changed according to the skew and time offset) to improve the time synchronization in real-time networks. Therefore, the clock servo parameters must be adaptable on-line while the system is running. The slave can ensure that its time system has only a small time difference from the master. Noise reduction is also enhanced. The proposed method, the fuzzy-PI clock servo, is presented in the next section.

IV. PROPOSED ADAPTIVE CLOCK SYNCHRONIZATION METHOD

A. PTP SYNCHRONIZATION

$$T_{owd} = \frac{t_4 - t_3 + t_2 - t_1}{2} \tag{5}$$

$$T_{offset} = \frac{t_1 - t_2 + t_4 - t_3}{2} \tag{6}$$

$$T_s^m = T_m + T_{owd} \tag{7}$$

$$e = T_s^m - T_s \tag{8}$$

The IEEE 1588 protocol uses timestamps for time synchronization, as shown in Fig. 2. In this model, the master sends a synchronized message after the T_{sync} interval. In the first timestamp, t_1 is captured when a "sync frame" is sent from the master to slaves over the network. A one-step or two-step method is configured, and a "follow-up frame" is sent right after the sync frame is transmitted. The t_1 timestamp is included in the follow-up frame. Moreover, based on the peer-to-peer transparent clock of IEEE 1588, the follow-up frame is updated according to a delay time based on the port-to-port delay measurement. Using a follow-up frame, the time of synchronization onset will be correct, which is important when frequency synchronization needs to be on the order of nanoseconds. The slave will capture t_2 when it receives the

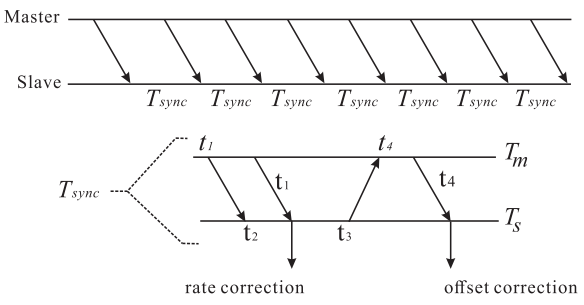

FIGURE 2. PTP synchronization.

sync frame from the master. After a random time, a request frame, which includes t_3, is sent from the slave to the master. When this frame is received, the t_4 time is captured in the master and sent back to the slave to calculate the one-way delay T_{owd} and the time offset T_{offset}. The formulas for the one-way delay and time offset calculations are presented in (5) and (6).

In the slave time, the time correction is divided into two parts: a large offset and a small offset. A time boundary is designed in this configuration. Normally, T_{offset} will be used for cycle synchronization when the offset is larger than the time boundary. When the time boundary is reached, frequency compensation is used. The master time T_m will be presented in the slave time T_s^m. Then, the time offset e between T_s^m and T_s will differ, as shown in (7) and (8). The value of this error is the input for the clock servo in the slaves.

The clock mode of the IEEE 1588 module inside a micro-controller is illustrated in Fig. 3. This module includes a 32-bit clock second counter, a 32-bit clock nanosecond counter, a 32-bit accumulator, and a 32-bit addend register. The value of the addend register controls the normal clock f_0, ensuring that it follows the master clock. The higher the value of the addend register, the higher the frequency of the local clock and vice versa. The IEEE 1588 module has two modes of operation. If a large time offset occurs, a coarse update is run. Alternatively, a fine update is performed when a small time offset is reached. The details of this module can be found in [19].

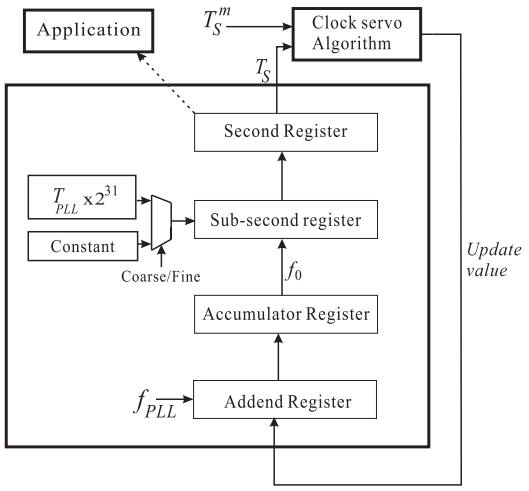

FIGURE 3. Synchronization using addend register.

B. FUZZY-PI CLOCK SERVO DESIGN

A synchronization system that needs accuracy on the order of nanoseconds, such as a motion control system, can be created based on the IEEE 1588 protocol and a clock servo algorithm. Various factors can influence time synchronization over a network, such as the crystal rate, the jitter of the operating system, the delay time in the queue, the noise of wire transmission, an incorrect clock model, quantization of the digital system, and the accumulator circuit. When using the IEEE 1588 protocol, the slave clock can compensate for a large time offset. However, the stability of time synchronization and high synchronization accuracy are based on the clock servo algorithm. In this section, we present a new synchronization method. The block state of this method is shown in Fig. 4, and the clock model is explained in detail. We also propose an adaptive PI based on a fuzzy controller to enhance noise rejection and improve the offset quality.

1) CLOCK MODEL

In this paper, we use the local clock model presented in (1). The details of this model are shown in Fig. 3. Using an approximation method, formula (1) can be presented as shown in (9). By updating the addend register value, the normalized frequency f_0 will change to match the output of the clock servo algorithm, which will be corrected according to the time counter of the master clock. To increase the resolution of the clock model, a sub-second register is implemented instead of a nanosecond register. The formula to convert from a nanosecond value to a sub-second value is presented in (10):

$$G_p(s) = \frac{T_S(s)}{U(s)} = \frac{K_c T}{s} \tag{9}$$

$$subsecond = \frac{nano * 2^{31}}{10^9} \tag{10}$$

2) FUZZY-BASED PI COMPENSATION

The IEEE 1588 protocol can compensate for large time offsets. However, it is not sufficiently stable for use in a micro-system or a synchronization motion system, such as those used by robots or exact mechanics. Our clock servo is implemented following the IEEE 1588 protocol for high offset performance on the order of nanoseconds. PI compensation is used for offset and rate offset compensation. Afterward, a filter to reduce noise with the PI controller is introduced. Although the filter can reduce noise, the K_P and K_I parameters are fixed during design, so they cannot be made adequate if the time synchronization requires a high frequency. In [30], Yildirim presented an adaptive PI clock servo that algorithm simply adds or subtracts an integral parameter with a fixed constant at the level of system design. It cannot accommodate constant fluctuations such as those found in networked systems, which always contain offset and frequency drift. A method with short coverage time and fully adaptive clock servo parameters is needed. Using the fuzzy logic, the clock servo can reach a short setting time and guaranteed high time synchronization without training time for updating weights [31]. Our main idea in this paper is to design an adaptive PI controller based on fuzzy logic [32], [33].

Normally, a PI clock servo uses a compensation method that can be described as follows:

$$G_c(s) = K_p + \frac{K_i}{s} \tag{11}$$

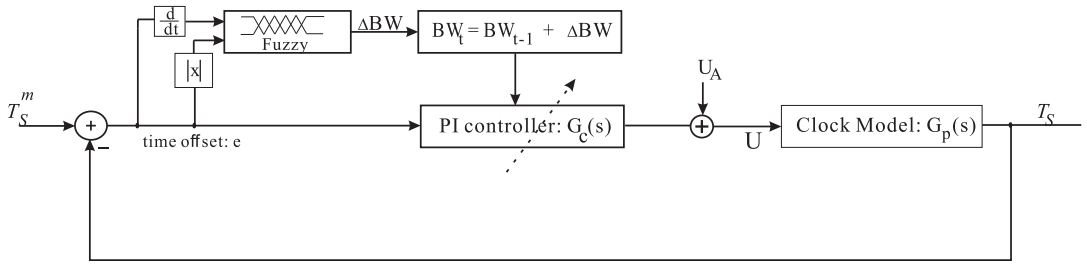

FIGURE 4. Fuzzy-PI clock servo.

By combining (9) and (11), the transfer function of the closed loop clock servo can be described as:

$$H(s) = \frac{T_S(s)}{T_S^m(s)} = \frac{K_p K_c T s + K_i K_c T}{s^2 + K_p K_c T s + K_i K_c T} \tag{12}$$

If we set $k_p = K_P K_c T$, $k_i = K_I K_c T$, (12) can be rewritten as (13). Here, T is the T_{sync} cycle of the IEEE 1588 protocol. The clock servo in (13) can be presented as a two-order system with a natural frequency ω_n and a damping coefficient ζ, as shown in (14).

$$H(s) = \frac{k_p s + k_i}{s^2 + k_p s + k_i} \tag{13}$$

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{14}$$

where $\omega_n^2 = k_i$ and $2\zeta\omega_n = k_p$.

In this transfer function, response time, noise reduction, and overshoot time belong to ω_n and ζ. Two poles of this transfer function are $p_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{\zeta^2 - 1}$, which influence the stability of the clock servo. The response time is $\frac{1}{\zeta\omega_n}$. Accordingly, when the damping coefficient is constant, the natural frequency will determine clock servo quality. If ω_n is small, the response time is large; the reverse is also true. However, when ω_n is small, noise reduction is enhanced.

If $s = j\omega$, the bandwidth BW of the clock servo can be presented by (15) [32].

$$BW^2 = \omega_n^2[1 + 2\zeta^2 + \sqrt{(1 + 2\zeta^2) + 1}] \tag{15}$$

If we consider (15) when the damping coefficient is fixed, the bandwidth of the clock servo is proportional to the natural frequency. Changing the natural frequency value will thus change the bandwidth of the clock servo, and a larger bandwidth will equal a larger cut-off frequency (and vice versa). Additionally, a larger bandwidth equals a shorter acquisition time. In the initial phase, a system that needs a large bandwidth to reach the bound error in a short time, and the offset includes more random noise. When the system is stable, a suitable bandwidth is required for noise rejection. Therefore, we propose a fuzzy-PI clock servo to adapt the clock servo bandwidth. In this paper, the absolute time offset $|e|$ and offset derivation de are two inputs of the fuzzy block controller. The membership of these inputs is shown in Fig. 5 and Fig. 6. To smooth the bandwidth BW of the clock servo, a delta bandwidth ΔBW is added to the last

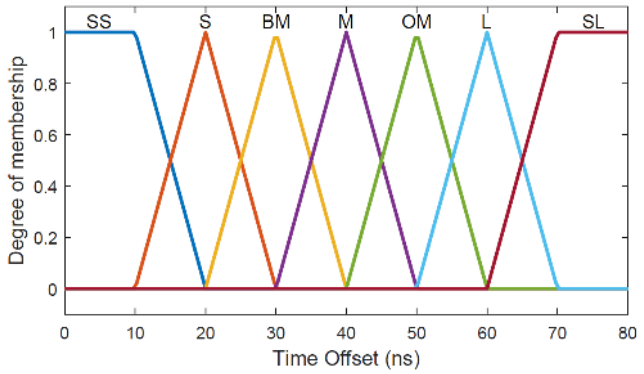

FIGURE 5. Time offset fuzzy.

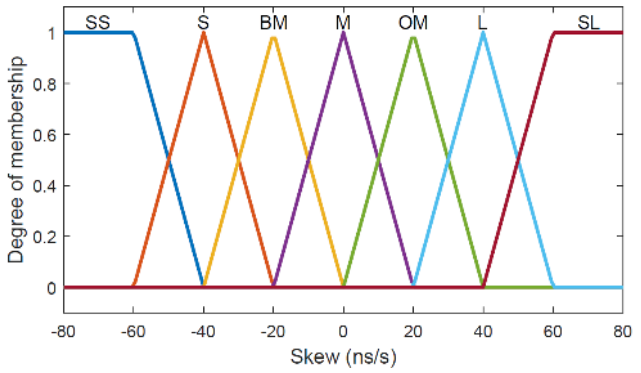

FIGURE 6. Skew fuzzy.

bandwidth BW_{t-1} to replace the system bandwidth (16). The membership of the output is shown in Fig. 7. A fuzzy surface that presents the relationships among $|e|$, de, and ΔBW is shown in Fig. 8.

$$BW_t = BW_{t-1} + \Delta BW \tag{16}$$

The fuzzy logic block changes the bandwidth of the clock servo based on a human understanding of time clock synchronization. In this paper, the minimum and maximum values of the two fuzzy inputs are set as 0ns and 80ns, respectively, in the absolute time offset and -80ns and 80ns, respectively, in the skew. The output for the fuzzy block is between -1 and 0.2 rad. Based on [32] and [33], seven fuzzy sets of absolute offset and skew were designed as super small (SS), small (S), below medium (BM), medium (M), over medium (OM), large (L), and super large (SL). The delta bandwidth is included in six fuzzy sets: SS, S, BM, M, OM, and L.

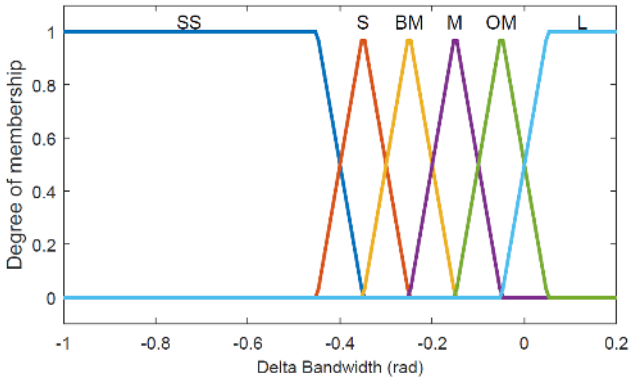

FIGURE 7. Delta bandwidth fuzzy.

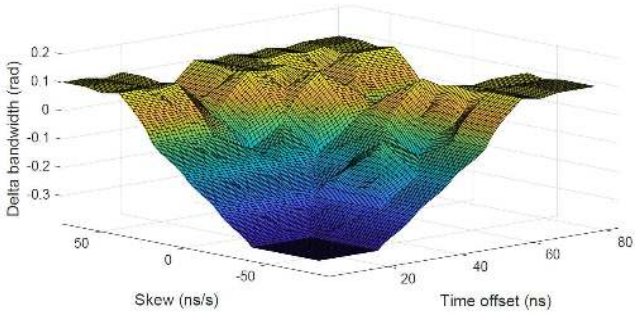

FIGURE 8. Fuzzy surface.

The input and output values are chosen based on algebraic values of those sets. The principle of the fuzzy rules is implemented as follows:

- In the initial phase, $|e|$ and de have large values, and the bandwidth of the clock servo will be large to reduce the response time and allow the error to reach the boundary value quickly. The dominant errors in this scenario are random noise and quantization error.
- If $|e|$ and de have a moderate value, the bandwidth of the clock servo will have a moderate value, and the noise can be reduced.
- If $|e|$ and de have a small value, a suitable bandwidth of the clock servo is maintained, and noise reduction is enhanced.

The fuzzy rules table for the clock servo is presented in Table 1. In practice, if the damping coefficient is held to a constant value, $\zeta = 0.5$, the bandwidth value is $BW^2 = w_n^2 * 3.08$. Based on (15), the values of k_p, k_i can be expressed by (17) and (18), respectively:

$$k_p = 2\zeta\omega_n = BW * 0.57 \tag{17}$$

$$k_i = \omega_n^2 = BW/3.08 \tag{18}$$

A normal oscillator has a high frequency variation, and the value after fuzzy-PI compensation is multiplied by $\frac{U_A}{1000000000}$ to adjust the part per billion units of the oscillator frequency [13].

TABLE 1. Fuzzy rules.

|e|/de	SS	S	BM	M	OM	L	SL
SS	S	S	BM	M	OM	L	L
S	S	S	BM	M	OM	L	L
BM	BM	BM	BM	M	M	OM	OM
M	M	OM	M	M	OM	L	L
OM	OM	M	OM	OM	L	L	L
L	L	L	L	OM	OM	OM	L
SL	L	L	OM	OM	OM	L	L

V. EXPERIMENTS AND RESULTS

This section explains how we designed and implemented our method, including the hardware and software. We conducted some experiments to confirm the reliability of our method, evaluated it under various conditions and compared the results of the experiments with those of previous methods.

A. HARDWARE AND SOFTWARE DESIGN

1) HARDWARE

An STM3220G evaluation board based on the ARM-Cortex 3 iwas used for all implementations described in this paper. An inexpensive 25 MHz crystal was mounted on the board. Inside the microcontroller, an IEEE 1588 block module was implemented. The hardware used for timestamping was triggered by the MAC layer. An addend register and sub-second register were programmed to change the normal frequency resolution. The setup for these experiments is shown in Fig. 9. Each node has an IP address that was allocated during the configuration phase. In this study, a BMC was implemented according to the IEEE 1588 standard. The IP address with the lowest value was the master over the network.

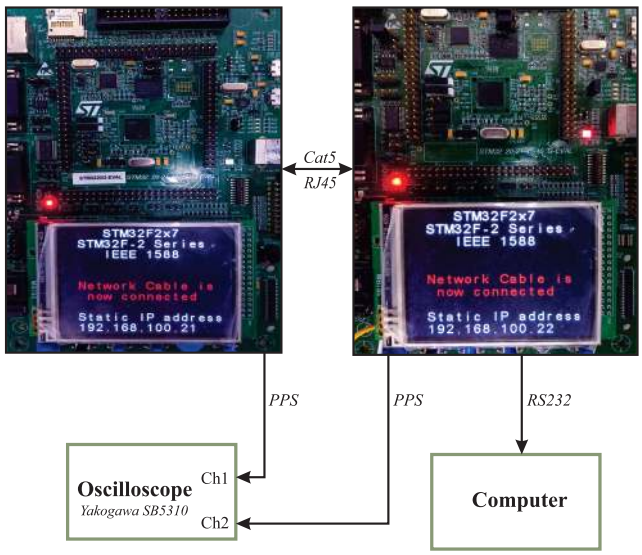

FIGURE 9. Experiment setup.

2) SOFTWARE

To implement the IEEE 1588 standard, we used the following third-party software:

1) FreeRTOS: A networked motion control system uses many commands other than time synchronization,

such as those related to motion profile and feedback status. A real-time operating system, such as FreeRTOS, is required to manage resources, tasks, and threads smoothly and exactly. For this paper, we made a thread for PTP communication that included all PTP states, such as the initial clock, BMC algorithms, and PTP algorithm. A system interrupt from the operating system of 1000??s was made as a time base for the synchronization interval cycle in both master and slave nodes. We reserved another thread for networked motion control duty. Moreover, a web server for communication between the system and the outside internet was created as a premise for an industrial internet of things application.

2) Light weight IP (LwIP): According to the IEEE 1588 standard, the protocol can be implemented in an Ethernet or TCP/IP layer. In this study, we used a lightweight IP stack and implemented PTP protocol version 2 in the TCP/IP layer. When the system is connected by switches, the system examination will be easier over a LAN network. In this paper, we set three boards with Internet protocol addresses (IPs) of "192.168.100.21", "192.168.100.22" and "192.168.100.23". Because the frequency and drift from the crystal is the same (25MHz and +/100ppm, respectively) in every node, the board with the lowest IP will become the master.
3) PTP daemo (PTPd): This is a popular open source code for the IEEE 1588 standard. It includes two versions. Version 1 was developed for the IEEE 1588 (2002) standard, and version 2 was developed for the IEEE 1588 (2008) standard, which considers the boundary clock and transparent clock. Normally, open source software is implemented in the Linux operating system. In this paper, we use this stack with version 2 for the ARM-Cortex 3. The two-step with follow-up frame is used to correct the timestamping and reduce the effects of FreeRTOS. Given the settings of the board we used, we chose "End to End" communication between the master and slave instead of "Peer to Peer" communication to calculate the one-way delay time.

B. RESULTS

In the IEEE 1588 protocol, an increasing frequency synchronization frame (reducing T_{Sync}) or increasing normal frequency f_0 can affect the time synchronization performance [9], [28]. However, when T_{Sync} is reduced, the bandwidth of the network increases. This is not good for a motion control system that is transferring many motion frames over a network. Therefore, the synchronization interval was kept at one second for all our experiments. The normal frequency was considered for the time synchronization method.

Consider the K_c parameter in the clock model. The details for the calculation of this value are presented in (19) to (21). In formula (21), K_c belongs only to the phase lock loop frequency f_{PLL}. In [28], f_{PLL} is changed to increase the supply frequency and measure the results. However, changing f_{PLL} will change the K_c parameter and the clock model. The values of K_P and K_I for PI compensation will change similarly. To preserve the clock model parameters, a pair of numbers, i.e., f_0 and the addend register values U_A, will be changed at the same time. Increasing the normal frequency f_0 can reduce the quantization error. Because the IEEE 1588 module is a digital system, the normal cycle $T_0 = 1/f_0$ is set as $T_0 = 14$ns or $T_0 = 7$ns to ensure that the constant value added to the sub-second register is an integer.

$$K_c = \frac{f_0}{U_A} \tag{19}$$

$$U_A = 2^{32} * \frac{f_0}{f_{PLL}} \tag{20}$$

$$K_c = \frac{f_{PLL}}{2^{32}} \tag{21}$$

In this study, when the absolute time offset and rate offset between the local clock and master clock are within 80ns, we consider the node to be synchronized with the master clock. The values of the time offset and rate offset are used to adapt the bandwidth of the clock servo based on the fuzzy-PI controller. Those values are transferred to a PC through an RS232 interface and drawn with Matlab.

Fig. 10 plots the bandwidth, setting time, time offset, and rate offset when using the adaptive fuzzy-PI method with $T_0 = 14$ns. The measurement time is 20 minutes to ensure the accuracy of the time synchronization performance. Fig. 10(b) shows that, when using the proposed method, the node needs one synchronization interval to track with the reference clock. As calculated using the statistical measures method, the offset mean value is 0.314ns, and the standard deviation is 11.129ns. In the rate offset case, those values are -0.0109ns/s and 15.940ns/s for the mean and standard deviation, respectively. Moreover, the peak-to-peak time offset value (which is the maximum time offset value minus the minimum time offset value) is 57ns, and the peak-to-peak time rate value is 86ns/s, as shown in Fig. 10(c) and Fig. 10(d). Additionally, the time offset and rate offset are almost inside the values [−14ns, 14ns] and [−29ns, 29ns], which demonstrates the stability of the proposed method under the influence of noise.

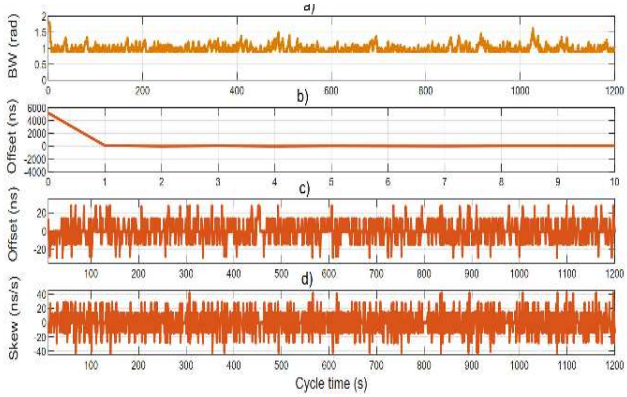

FIGURE 10. Fuzzy-PI clock servo performance in $T_0 = 14$ns.

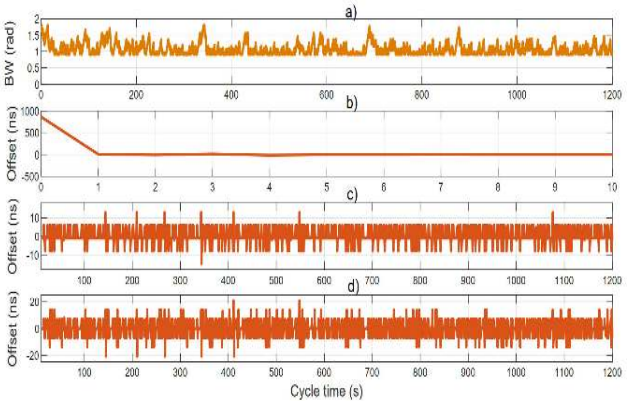

FIGURE 11. Fuzzy-PI clock servo performance in $T_0 = 7$ns.

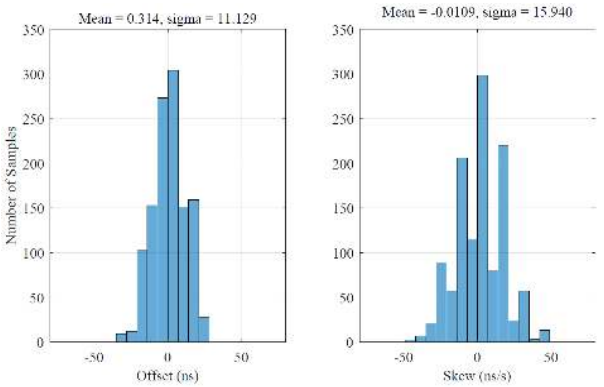

FIGURE 12. Offset and skew histogram in $T_0 = 14$ns.

A histogram of the time offset and rate offset in this case is shown in Fig. 12.

Likewise, Fig. 11 plots the bandwidth, setting time, time offset, and rate offset when using the adaptive fuzzy-PI method with $T_0 = 7$ns. The measurement time was also 20 minutes to ensure the accuracy of the time synchronization performance. In this case, the node also needs one synchronization interval to track with the reference clock when using the proposed method, as shown in Fig. 11(b). With the statistical measures method, the mean offset value is 0.432ns, and the standard deviation is 4.402ns. In the rate offset case, those values are -0.012ns/s and 6.596ns/s for the mean and standard deviation, respectively. The peak-to-peak time offset value is 28ns, and the peak-to-peak rate offset value is 42ns/s, as shown in Fig. 11(c) and Fig. 11(d). Moreover, the time offset and rate offset are almost inside the values $[-7\text{ns}, 7\text{ns}]$ and $[-14\text{ns}, 14\text{ns}]$. Thus, the fuzzy-PI clock servo is also stable with $T_0 = 7$ns. The time offset in this case is smaller than when $T_0 = 14$ns. A histogram of the time offset and rate offset in this case is shown in Fig. 13.

The bandwidth adaption is presented in Fig. 10(a) and Fig. 11(a). With a fast and high resolution such as in a time synchronization system, the bandwidth is limit from 0.9 rad to 1.8 rad. Considering the offset and rate time, a suitable

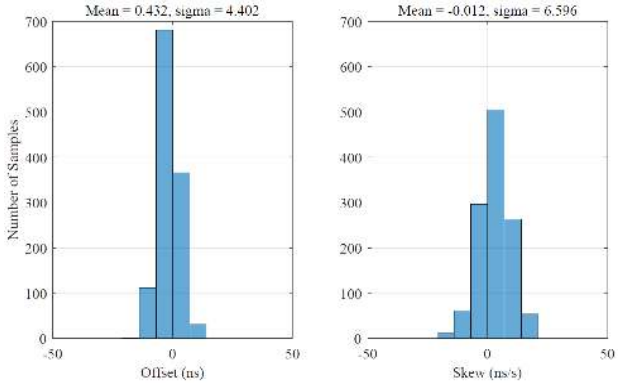

FIGURE 13. Offset and skew histogram in $T_0 = 7$ns.

bandwidth value is calculated based on the fuzzy logic. In the initial phase, the bandwidth value reaches a maximum value for a fast response time and decreases later when a boundary error encountered. When a big time offset or big rate value is considered, a large bandwidth is used. Based on the bandwidth adaption value, a suitable k_p and k_i are calculated following Eq. 17 and Eq. 18. In both case, $T_0 = 14$ns and $T_0 = 7$ns, the bandwidth is sensitive to fluctuations in the system, thus guaranteeing a high performance for time synchronized systems.

In a two-case study, the offset and rate values are quantized because the trigger time is made from the addend register and a constant value (Fig. 3). Thus, when the addend register runs over, a tick occurs, and a constant value will be added to the sub-second register. For example, when the normal frequency is $\frac{1}{14ns}$, the constant value added to the sub-register value is 14ns. This mechanism means that the tick from the addend register can deviate by some value when two clocks cannot be synchronized perfectly. Those results have the same form of time offset and rate offset, as shown in [28].

One pulse per second (PPS) measurements are used to externally determine the time offset values in the two cases, and timer 2 is used to measure the PPS. This timer is triggered from the IEEE 1588 module after synchronization. A high-performance oscilloscope with a resolution of 1 ns was used to measure these values. The timer in master was selected as the trigger source. In normally, the timer 2 has also a counter. Additionally, small amounts of noise and delay time can be added to the measuring system out-side of the Ethernet module. Jitter exists in each timer, and a small bias can occur between the PPS from the master and the PPS from the slave. Fig. 14 and Fig. 15 show that the peak-to-peak offset value is about 60ns when $T_0 = 14$ns and 30ns when $T_0 = 7$ns.

C. EVALUATION OF THE PROPOSED METHOD UNDER VARIOUS CONDITIONS

To evaluate the proposed method based on the PTP protocol (IEEE 1588 standard), some experiments are conducted under various conditions. The time synchronization interval is changed to evaluate the fuzzy-PI when the controlled environmental parameters are changed. Then, a star topology is setup

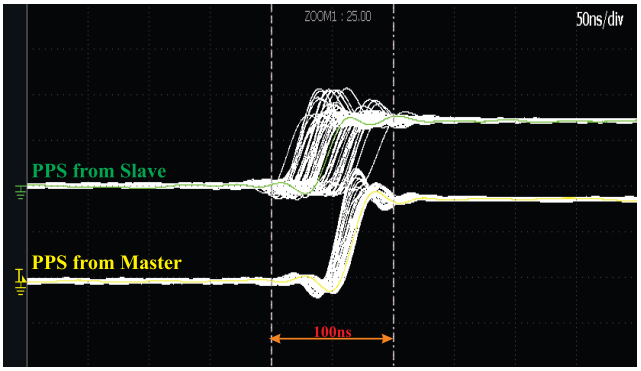

FIGURE 14. Captured oscilloscope images of the synchronized clock signals in $T_0 = 14$ns.

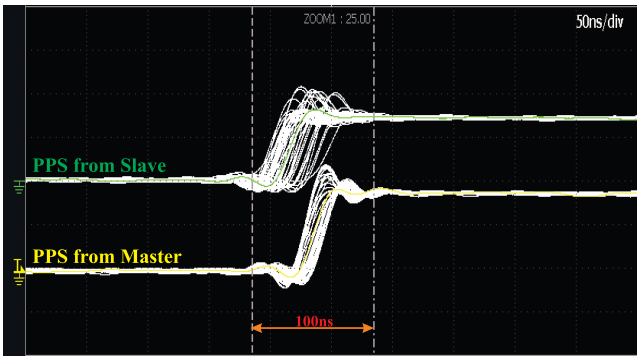

FIGURE 15. Captured oscilloscope images of the synchronized clock signals in $T_0 = 7$ns.

between the three evaluated boards to confirm the protocol extension with the proposed clock servo algorithm. In each case, the experiments were conducted for both $T_0 = 14$ns and $T_0 = 7$ns, separately.

1) DIFFERENT TIME SYNCHRONIZATION INTERVALS T_{sync}

The time synchronization interval between the master and slave is changed to different values to evaluate the proposed method. According the IEEE 1588 standard, T_{sync} is the squared of 2. In this case, the values of 1s, 2s and 4s are chosen for low bandwidth networks, which conform with industrial applications of 1s as the time base setting. In Fig. 16, the time offset of different time synchronization intervals in a normal cycle of $T_0 = 14$ns is shown. Following this, the mean and standard deviate are 0.0225ns and 11.9201ns in $T_{sync} = 1$s, 3.4935ns and 19.4833ns in $T_{sync} = 2$s, 2.6104ns and 52.9142ns in $T_{sync} = 4$s. The minimum and maximum time offset value are -29ns and 28ns, -56ns and 56ns, -112ns and 140ns for $T_{sync} = 1$s, $T_{sync} = 2$s and $T_{sync} = 4$s, separately.

Likewise, in Fig. 17, the experiments are also is proceeded in $T_0 = 7$ns. In this case, the mean and standard deviate are 0.2746ns and 4.2838ns in $T_{sync} = 1$s, 0.2625ns and 4.8932ns in $T_{sync} = 2$s, 0.9161ns and 8.8159ns in $T_{sync} = 4$s, respectively. The minimum and maximum time offset value are -15ns and 13ns, -15ns and 14ns, -22ns and 27ns for

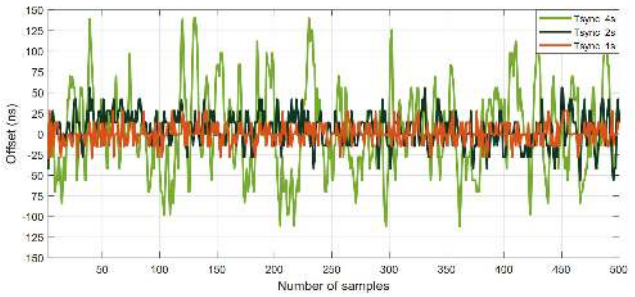

FIGURE 16. Evaluation under different time synchronization intervals in $T_0 = 14$ns.

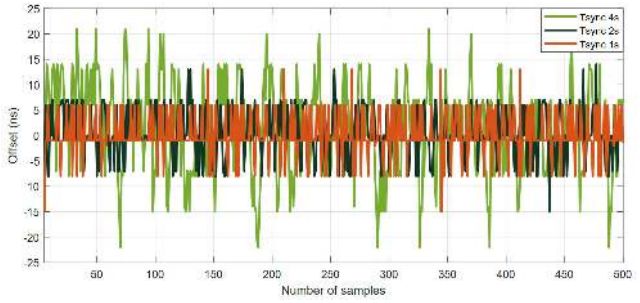

FIGURE 17. Evaluation under different time synchronization intervals in $T_0 = 7$ns.

TABLE 2. Summarized performance with different time synchronization intervals in $T_0 = 14$ns.

	$T_{sync}(1s)$	$T_{sync}(2s)$	$T_{sync}(4s)$
Min. (ns)	-29	-56	-112
Max. (ns)	28	56	140
Mean (ns)	0.0225	3.4935	2.6104
σ (ns)	11.9201	19.4833	52.9142

TABLE 3. Summarized performance with different time synchronization intervals in $T_0 = 7$ns.

	$T_{sync}(1s)$	$T_{sync}(2s)$	$T_{sync}(4s)$
Min. (ns)	-15	-15	-22
Max. (ns)	13	14	27
Mean (ns)	0.2746	0.2625	0.9161
σ (ns)	4.2838	4.8932	8.8159

$T_{sync} = 1$s, $T_{sync} = 2$s and $T_{sync} = 4$s, separately. Table 2 and Table 3 show the result details. Here, the time offset in $T_{sync} = 4$s is larger than the other cases in both normal cycles. The time offset for $T_{sync} = 2$s is bigger than the time offset for $T_{sync} = 1$s. These results agree with the IEEE 1588 standard. The longer the synchronization interval, the bigger the time offset value. The results in $T_0 = 7$ns are better than $T_0 = 14$ns because the accumulated error of the timer quantization is reduced when a high-resolution timer is applied. The absolute time offset is below 140ns, which is enough for systems that need high accuracy motion and can also be applied to industrial Internet of things (IIoT) applications.

2) EXTENDED NETWORK WITH A CONVENTIONAL SWITCH

To evaluate the method performance and test the extension ability of protocol, a star topology is formed between the three boards and a conventional switch HP J9794A. The lowest IP address "192.168.100.21" are master and two remain IP "192.168.100.22" (Slave 1) and "192.168.100.23" (Slave 2) are slaves following the best master clock algorithm. The normal cycle $T_0 = 14$ns and $T_0 = 7$ns is setup to demonstrate the system performance under time synchronization interval $T_{sync} = 1$s.

Figure 18 presents the offset when three boards are connected through the switch in $T_0 = 14$ns. The maximum and minimum offset value are 558ns and -546ns for Slave 1 and 544ns and -546ns for Slave 2. The mean and standard deviation are 0.1387ns and 234.4872ns for Slave 1 and 0.1352ns and 221.7675ns for Slave 2, respectively. Although the switch has a time delay and jitter when stored and forwarded to the Ethernet frame, the results are still better when compared with previous research [24], [30].

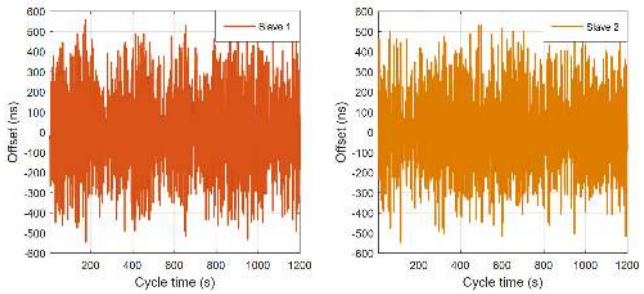

FIGURE 18. Two slaves connect with a master through a conventional switch in $T_0 = 14$ns.

Likewise, the performance of when three boards are connected through the switch in $T_0 = 7$ns is presented in Fig. 19.The maximum and minimum offset value are 90ns and -85ns for Slave 1 and 84ns and -84ns for Slave 2, respectively. The mean and standard deviation are 0.0205ns and 38.4818ns for Slave 1 and 0.0912ns and 35.2566ns for Slave 2, respectively. In this case, the absolute offset value below 100ns confirms the effectiveness of the proposed method based on IEEE 1588. The offset value also has the best performance even though it is connected using a conventional switch. Moreover, the performance in $T_0 = 7$ns is

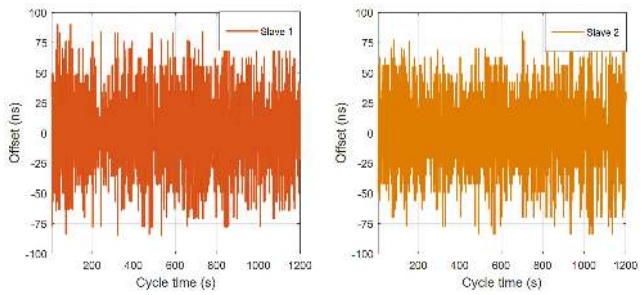

FIGURE 19. Two slaves connect with a master through a conventional switch in $T_0 = 7$ns.

better than $T_0 = 14$ns because when reducing the normal cycle time combined with the highest resolution, the quantization error and accumulated error are reduced significantly. In both cases, the proposed method reaches a best performance and the amount of networks that support time synchronization mechanisms can extended. With these results, the fuzzy-PI method can support motion control systems. Details about the testbed performances are shown in Table 4 and Table 5.

TABLE 4. Summary performance when extension with a conventional switch in $T_0 = 14$ns.

	Slave 1	Slave 2
Min. (ns)	-546	-546
Max. (ns)	558	544
Mean (ns)	0.1387	0.1352
σ (ns)	234.4872	221.7675

TABLE 5. Summary performance when extension with a conventional switch in $T_0 = 7$ns.

	Slave 1	Slave 2
Min. (ns)	-85	-84
Max. (ns)	90	84
Mean (ns)	0.0205	0.0912
σ (ns)	38.4818	35.2566

Following these results, the fuzzy-PI clock servo based on IEEE 1588 guarantees a high time synchronization and networking extension ability. It is flexible and reaches high performance for distributed networking systems.

D. COMPARISON WITH OTHER METHODS

Many methods have been proposed to reduce the time offset. PI clock servo uses a random couple k_p and k_i belonging to the system when implementing the different phases.Optimal PI clock servo uses the minimal integral squared error method to define the k_p and k_i parameters. With this method, $k_p = 1$ and $k_i = 1$ are fixed when $T_{sync} = 1$s. A Kalman filter based on the PI clock servo uses the same parameters as the Optimal PI clock servo while also integrating a Kalman filter into it. To demonstrate the high performance of the proposed method, we compared our fuzzy-PI clock servo with the conventional PI clock servo, optimal PI clock servo, and KF-based PI (KF-PI) clock servo using the same test bench. In the conventional PI clock servo, $k_p = 1$ and $k_i = 0.5$. For the final two methods, $k_p = 1$ and $k_i = 1$ in accordance with the optimization integral square error performance rules [18]. In KF-PI, the processing noise covariance as $\sigma_e^2 = 20$ns, the measurement noise covariance as $\sigma_m^2 = 80$ns when $T_0 = 14$ns and as $\sigma_e^2 = 10$ns and $\sigma_m^2 = 40$ns when $T_0 = 7$ns are determined. In these comparisons, we used the mean and standard deviation σ, the setting time (or converge time, when the node is synchronized with the reference clock)

and the peak-to-peak time offset as the performance indices to evaluate the time synchronization accuracy.

There is a tradeoff between performance and setting time. A longer setting time produces a better performance (i.e., a smaller time offset). As shown in Fig. 20(a) and Fig. 21(a), the fuzzy-PI and optimal PI clock servo had the shortest setting time of one cycle. The longest setting time of seven cycles was observed with the conventional PI clock servo and KF-PI clock servo. A comparison of the coverage time is shown in Fig. 20(a) and Fig. 21(a). When using the time offset index, the fuzzy-PI clock servo became much better than the other methods. With $T_0 = 14$ns, the mean and standard deviation of the time offset were 0.314ns and 11.129ns, whereas the mean and standard deviation values were −0.196ns and 13.242, −0.052ns and 17.357, −0.012ns and 16.521 for the other methods, as shown in Fig. 22. Additionally, the proposed method has a peak-to-peak offset of 57ns. For the conventional PI clock servo, optimal PI clock servo, and KF-PI, the peak-to-peak offsets were 83ns, 99ns, and 86ns, respectively. Moreover, the time offset of the fuzzy-PI is almost inside [−14ns, 14ns], which demonstrates the stability of the proposed method.

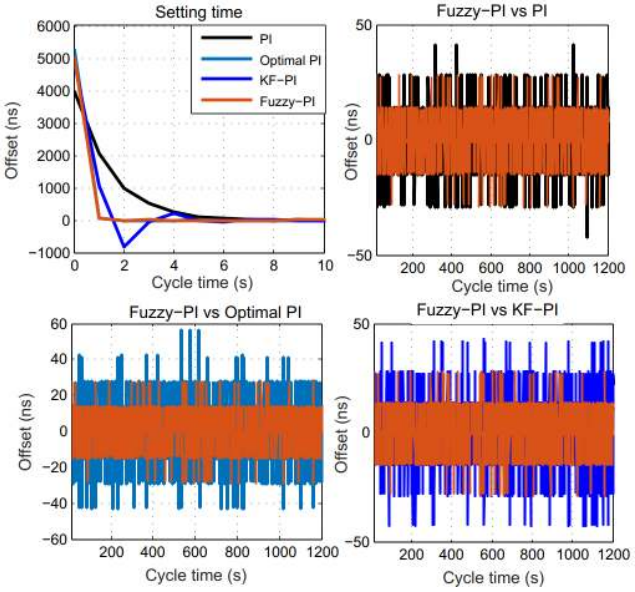

FIGURE 20. Comparison of the fuzzy-PI clock servo with other methods in $T_0 = 14$ns.

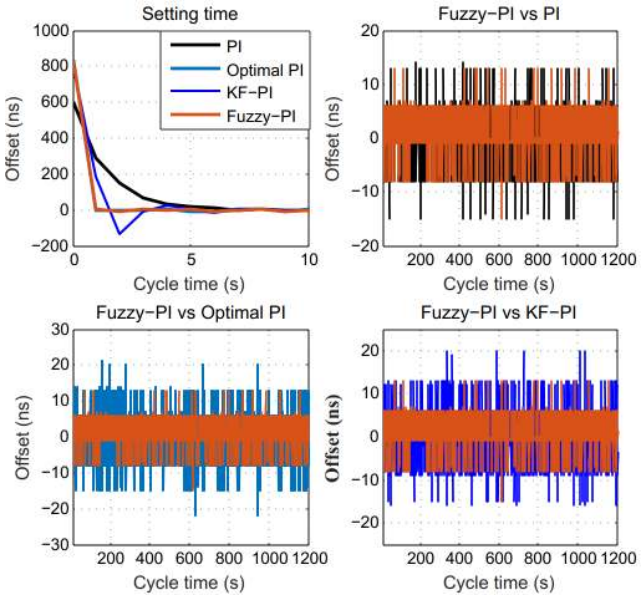

FIGURE 21. Comparison of the fuzzy-PI clock servo with other methods in $T_0 = 7$ns.

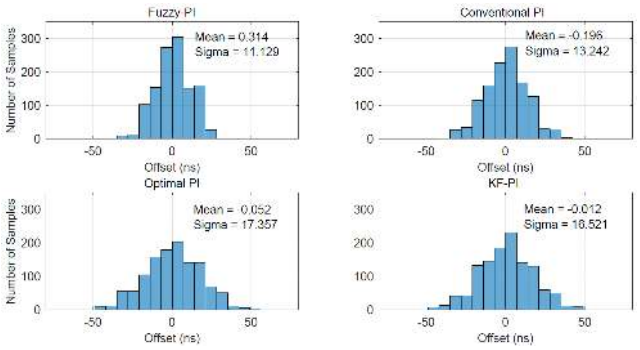

FIGURE 22. Offset histogram comparison in $T_0 = 14$ns.

Likewise, the fuzzy-PI clock servo also has the best performance when $T_0 = 7$ns. Fig. 21 shows the comparison results. The mean and standard deviation of the time offset are 0.432ns and 4.402ns, respectively, whereas the mean and standard deviation values were −0.005ns and 5.536ns, −0.017ns and 7.122ns, 0.084ns and 6.742ns with the other methods, respectively, as shown in Fig. 23. Additionally, the peak-to-peak time error value for this case was 28ns, whereas the peak-to-peak time offset values were 29ns, 43ns, and 36ns for the conventional PI clock servo, optimal PI clock servo, and KF-PI, respectively. Moreover, the time offset of the fuzzy-PI clock servo is almost between −7ns and 7ns. Thus, the performance of the proposed method is better than those of the previous methods. In Fig. 20 and Fig. 21,

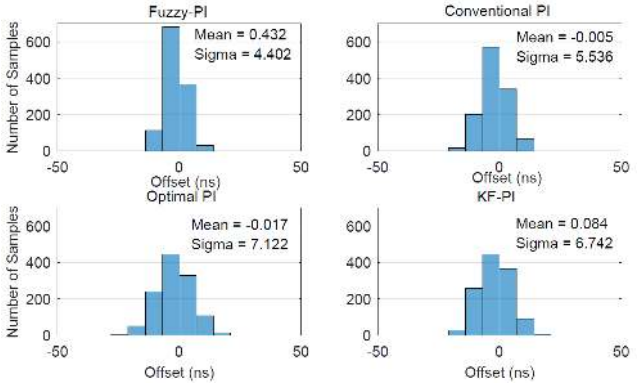

FIGURE 23. Offset histogram comparison in $T_0 = 7$ns.

TABLE 6. Comparison with other methods in $T_0 = 14$ns.

	Setting time/cycle	Min. ns	Max. ns	Mean ns	σ ns
Fuzzy-PI	**1**	**-29**	**28**	**0.314**	**11.129**
PI	7	-42	41	-0.196	13.242
Optimal PI	1	-43	56	-0.052	17.357
KF-PI	7	-43	43	-0.012	16.521

TABLE 7. Comparison with other methods in $T_0 = 7$ns.

	Setting time/cycle	Min. ns	Max. ns	Mean ns	σ ns
Fuzzy-PI	**1**	**-15**	**13**	**0.432**	**4.402**
PI	7	-15	14	-0.005	5.536
Optimal PI	1	-22	21	-0.017	7.122
KF-PI	7	-16	20	0.084	6.742

the fuzzy-PI clock servo shows outstanding performance, with the shortest setting time and a stable time offset. Details of the comparisons are shown in Table 6 and Table 7. The histograms of the time offset and rate are shown in Fig. 22 and Fig. 23. In this paper, the confided time offset value when using the mean and standard deviation is given by "time offset value = mean ± standard deviation". Although the mean time offset values presented in Tables 6 and 7 is larger than the others, the confided time offset is better than when calculating according to this way. Moreover, we also used the peak-to-peak time offset value, which is the maximum time offset value minus the minimum time offset value, as one performance index to confirm the proposed algorithm. By using two performance indices, the proposed method has a performance better than the other methods.

E. DISCUSSION

Time synchronization can be influenced by several factors, such as an unstable oscillator, temperature, wire noise, and an incorrect clock model. Normally, a PI controller and filter-based PI are designed to achieve nanosecond-scale synchronization in the IEEE 1588 standard. Although many algorithms have been presented, the core component of the clock servo is the PI controller. In this study, we designed an adaptive bandwidth clock servo, the fuzzy-PI clock servo, to improve the time synchronization performance by using fuzzy logic and an inexpensive 25 MHz crystal and ARM-Cortex 3. The experimental results shown in from Table 2 to Table 7 demonstrate that the proposed method can synchronize the slave clock with the master within one cycle. Moreover, based on the calculated results from Table 6 and Table 7, the peak-to-peak time offset was 57ns, the mean was 0.314ns and the standard deviation was 11.129ns when $T_0 = 14$ns and these values were 28ns, 0.432ns and 4.402ns when $T_0 = 7$ns, respectively. In both cases, most offset values were within $[-T_0, +T_0]$. Thus, the fuzzy-PI clock servo has enhanced noise reduction compared with previous methods. Additionally, with a high normal frequency f_0, a high performance can be achieved. However, due to the limits of the micro-controller power, the frequency cannot increase to an infinite value, but the presented fuzzy-PI clock servo can improve the time synchronization performance. Moreover, when the rate offset is small and cycle synchronization for a low bandwidth is ensured, the maximum number of nodes connected in a networked motion control system can be increased. Our method is flexible and can be applied to many areas, such as vehicles and wireless communication [6]. In vehicles with a completed Ethernet backbone, our method is necessary for time synchronization of the gateway system [7]. In wireless sensors, our method ensures that the nodes keep uniform time. This method can also contribute to the Industrial Internet of Things (IIoT) [1], [6]. Following the results, the time offset value considering the IEEE 1588 module resolution T_0 and system parameter K_c can not be exactly defined because the clock always changes with the disturbances. Following these, an observer needs to consider reducing the time offset value and adjusting the fuzzy-PI clock servo.

VI. CONCLUSION

In this paper, we have presented a new method for increasing time synchronization performance. This algorithm is based on the IEEE 1588 protocol and uses a fuzzy-PI controller to adapt the bandwidth of the clock servo. A test bench was built using an inexpensive crystal and ARM-Cortex 3 to demonstrate the proposed algorithm. Our results show that the mean and standard deviation of the time offset are 0.432ns and 4.402ns with a low-cost oscillator. Moreover, the algorithm has outstanding performance in terms of setting time and stability of the time offset, compared with previous methods. The high resolution normal cycle and one-second time synchronization interval ensure that a maximum number of node connections can be maintained in a network using the fuzzy-PI clock servo. In the future, we will apply the method to a system that does not support the addend register, such as an EtherCAT embedded master, or to a system with different normal frequencies in different devices. We will also implement the fuzzy-PI clock servo in a system with many hops to analyze the time performance.

REFERENCES

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.

[2] R. A. Gupta and M.-Y. Chow, "Networked control system: Overview and research trends," *IEEE Trans. Ind. Electron.*, vol. 57, no. 7, pp. 2527–2535, Jul. 2010.

[3] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 860–880, 2nd Quart., 2013.

[4] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1352–1364, Jun. 2011.

[5] J. Laird, "Clock synchronization terminology," Interoperability Lab., University of New Hampshire, Durham, NH, USA, Tech. Rep., Jun. 2012.

[6] A. Mahmood and R. Exel, "Clock synchronization over IEEE 802.11—A survey of methodologies and protocols," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 907–922, Apr. 2017.

[7] Y. S. Lee, J. H. Kim, and J. W. Jeon, "FlexRay and Ethernet AVB synchronization for high QoS automotive gateway," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 5737–5751, Jul. 2017.

[8] J. Eidson, *Measurement, Control, Communication Using IEEE 1588*. New York, NY, USA: Springer-Verlag, 2006.

[9] Z. Li and B. Qin, "Research on Synchronization Technology Based on IEEE 1588 Protocol," *Appl. Mech. Mater. J.*, vol. 310, pp. 634–639, 2013.

[10] S. Dietrich and G. May, "Latency in cascaded wired/wireless communication networks for factory automation," in *Proc. Int. Conf. Ind. Netw. Intell. Syst. (NISCOM)*. Leicester, U.K.: Springer, Nov. 2016, pp. 50–61.

[11] S.-M. Park, H. Kim, H.-W. Kim, C. N. Cho, and J.-Y. Choi, "Synchronization improvement of distributed clocks in EtherCAT networks," *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1277–1280, Jun. 2017.

[12] X. Wu, L. Xie, and F. Lim, "Network delay analysis of EtherCAT and PROFINET IRT protocols," in *Proc. 40th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2014, pp. 2597–2603.

[13] K. Correll, N. Barendt, and M. Branicky, "Design considerations for software only implementations of the IEEE 1588 precision time protocol," in *IEEE Conf. IEEE 1588 Stand. Preci. Clock Synch. Prot. Net. Mea. Cont. Syst.*, 2006, pp. 1–6.

[14] P. Tosato, D. Macii, and D. Laverty, 'A software-based low-jitter servo clock for inexpensive phasor measurement units," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Sep./Oct. 2018, pp. 1–6.

[15] A. Mahmood, R. Exel, and T. Sauter, "Delay and jitter characterization for software-based clock synchronization over WLAN using PTP," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1198–1206, May 2014.

[16] E. Kyriakakis, J. Sparsø, and M. Schoeberl, "Hardware assisted clock synchronization with the IEEE 1588-2008 precision time protocol," in *Proc. 26th Int. Conf. Real-Time Netw. Syst. (RTNS)*, 2018, pp. 51–60.

[17] H. Yin, P. Fu, J. Qiao, and Y. Li, "The implementation of IEEE 1588 clock synchronization protocol based on FPGA," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (IMTC)*, May 2018, pp. 1–6.

[18] J. Leea and T. F. Edgarb, "Technical Communique: ISE tuning rule revisited," *Automatica*, vol. 40, no. 8, pp. 1455–1458, 2004.

[19] J. Liu, X. Li, M. Liu, X. Cui, and D. Xu, "A new design of clock synchronization algorithm," *Adv. Mech. Eng.*, vol. 2014, Feb. 2014, Art. no. 958686.

[20] R. Exel and F. Ring, "Improved clock synchronization accuracy through optimized servo parametrization," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control Commun. (ISPCS)*, Sep. 2013, pp. 65–70.

[21] D. Fontanelli and D. Macii, "Accurate time synchronization in PTP-based industrial networks with long linear paths," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control Commun.*, Sep. 2010, pp. 97–102.

[22] X. Xu, Z. Xiong, J. Wu, and X. Zhu, "High-precision time synchronization in real-time Ethernet-based CNC systems," *Int. J. Adv. Manuf. Technol.*, vol. 65, nos. 5–8, pp. 1157–1170, Mar. 2013.

[23] J. Hun Cho, H. Kim, S. Wang, J. Lee, H. Lee, S. Hwang, S. Cho, Y. Oh, and T.-J. Lee, "A novel method for providing precise time synchronization in a distributed control system using boundary clock," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 8, pp. 2824–2829, Aug. 2009.

[24] X. Chen, D. Li, S. Wang, H. Tang, and C. Liu, "Frequency-tracking clock servo for time synchronization in networked motion control systems," *IEEE Access*, vol. 5, pp. 11606–11614, 2017.

[25] Z. Chaloupka, N. Alsindi, and J. Aweya, "Clock skew estimation using Kalman filter and IEEE 1588v2 PTP for telecom networks," *IEEE Commun. Lett.*, vol. 19, no. 7, pp. 1181–1184, Jul. 2015.

[26] G. Giorgi and C. Narduzzi, "Performance analysis of Kalman-Filter-Based clock synchronization in IEEE 1588 networks," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 8, pp. 2902–2909, Aug. 2011.

[27] G. Giorgi, "An event-based Kalman filter for clock synchronization," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 2, pp. 449–457, Feb. 2015.

[28] X. Xu, Z. Xiong, X. Sheng, J. Wu, and X. Zhu, "A new time synchronization method for reducing quantization error accumulation over real-time networks: Theory and experiments," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1659–1669, Aug. 2013.

[29] L. Tavares Bruscato, T. Heimfarth, and E. Pignaton de Freitas, "Enhancing time synchronization support in wireless sensor networks," *Sensors*, vol. 17, no. 12, p. 2956, 2017.

[30] K. S. Yildirim, R. Carli, and L. Schenato, "Adaptive Proportional–Integral clock synchronization in wireless sensor networks," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 2, pp. 610–623, Mar. 2018.

[31] T. T. Xie and H. Yu, "Comparison of fuzzy and neural systems for implementation of nonlinear control surfaces," in *Human-Computer Systems Interaction: Backgrounds and Applications*, vol. 2. Berlin, Germany: Springer-Verlag, 2012, pp. 313–324.

[32] C. Shan, Z. Chen, Y. Li, and H. Yuan, "All DPLLs based on fuzzy PI control algorithm," in *Proc. 2nd Int. Conf. Mechanic Autom. Control Eng.*, Jul. 2011, pp. 7150–7153.

[33] L. Y. Zhang and H. Y. Liu, "Adaptive bandwidth PLL design based on fuzzy logic control," *Appl. Mech. Mater.*, vols. 543–547, pp. 1393–1396, Mar. 2014.

VINH QUANG NGUYEN received the B.S. degree in physics and engineering physics from the University of Science, Ho Chi Minh City, Vietnam, in 2012, and the M.S. degree in electrical and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2015, where he is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering.

His research interests include real-time networks, motion control, and embedded systems.

TON HOANG NGUYEN received the B.S. degree in mechatronics engineering from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 2016. He is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Sungkyunkwan University, Suwon, South Korea.

His research interests include signal processing, motion control, robotics, and embedded systems.

JAE WOOK JEON (Senior Member, IEEE) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1990.

From 1990 to 1994, he was a Senior Researcher with Samsung Electronics, Suwon, South Korea. Since 1994, he has been with Sungkyunkwan University, Suwon, where he was first an Assistant Professor with the School of Electrical and Computer Engineering. He is currently a Professor with the School of Information and Communication Engineering, Sungkyunkwan University. His research interests include robotics, embedded systems, and factory automation.