Microprocessors and Microsystems 91 (2022) 104507

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

High-precision time synchronization chip design for industrial sensor and actuator network☆

Ching-Han Chen [a], Ming-Yi Lin [b,*], Yu-Chien Shih [a], Chun-Cheng Chen [b]

[a] Department of Computer Science and Information Engineering, National Central University, Taiwan
[b] Department of Computer Science and Information Engineering, Taipei City University of Science and Technology, Taiwan

ARTICLE INFO

Keywords:
Embedded systems
Fieldbus
Industry 4.0
Industrial internet of things (IIoT)
Mission-critical sensor and actuator network (MC-SAN)
Precision time synchronization (PTS)

ABSTRACT

Mission-critical (MC) sensor and actuator networks (SANs) are being applied widely for multiaxis motion control of robotic manipulators, particularly for performing MC tasks in smart manufacturing. One of the most challenging tasks in MC-SANs is the development of a precision time synchronization (PTS) protocol. A conventional precision time system uses the IEEE 1588 v2 standard [1]; however, in MC-SANs, many factors need to be considered when designing an IoT computing system, such as robustness, cost-effectiveness, interoperability and scalability. This study proposes a high-precision time synchronization protocol based on Modbus sensors and actuator networks for an industrial fieldbus network. The PTS protocol design and hardware synthesis are conducted with a consistent and systematic design methodology. GRAFCET is used to model the individual functional modules and hierarchical behavior of the system. The behavior of each module is represented as a sequential–concurrent hybrid discrete event system. We apply high-level synthesis rules to design PTS hardware for a complex PTS system based on the VHSIC hardware description language (VHDL). The PTS hardware module has an embedded PTS algorithm with a proportional error correction model, and the time synchronization accuracy of the precision time protocol (PTP) chip is up to 42 ns. Experimental results demonstrate that our high-performance PTS controller chip can satisfy the requirements of multiaxis motion control and that it is more efficient than other protocols and products designed for similar industrial manufacturing applications.

1. Introduction

1.1. Background

In recent years, mission-critical (MC) sensor and actuator networks (SANs) have become increasingly popular for emerging industrial automation applications [2–8]. Sensors mainly collect information on manufacturing operations by transforming a physical stimulus into an electrical signal that can be analyzed by the manufacturing system and be used for decision-making. Actuators convert an electrical signal from sensors into a mechanical motion and act on industrial equipment. MC-SANs are finding extensive use, and many sensors and actuators are being used for transmitting large data volumes. In a smart factory, numerous distributed sensors and actuators collaborate to realize real-time control and data collection by using a fieldbus network.

MC-SANs are widely used in multiaxis servo motion control systems for Industry 4.0 devices such as robotic manipulators, industrial robots, industrial cyber-physical systems (CPSs), and industrial Internet of things (IIoT). In particular, in IIoT smart manufacturing systems, sensors and actuators are part of the control loops, and network performance in terms of message transfer delay and reliability is required. The MC-SAN is typically designed for time-critical applications with strict real-time constraints, such as those found in high-speed multiaxis motion in the industrial automation and smart manufacturing domains. Robotic manipulators are also multiaxis motion control systems with highly coupled nonlinear dynamics. In addition, to perform automated industrial tasks accurately and reliably, manipulator positions, actuators, and velocities are controlled digitally. It is therefore important to develop precision time PTS control methods for multiaxis motion systems.

Existing MC-SAN technologies encounter substantial challenges in providing highly efficient time synchronization. In particular, time synchronization protocol (TSP) is crucial for industrial sensor networks. TSP is intrinsically an essential part of the IIoT system and an enabling technology that helps build a synergy between industrial automation

☆ Submission for Microprocessors and Microsystems (Special Issue on Adaptive and Reconfigurable Embedded Systems)

* Corresponding author.
E-mail addresses: pierre.miat@gmail.com (C.-H. Chen), milin@tpcu.edu.tw (M.-Y. Lin), ycshih.miat@gmail.com (Y.-C. Shih), jcchen@tpcu.edu.tw (C.-C. Chen).

https://doi.org/10.1016/j.micpro.2022.104507
Received 11 April 2021; Received in revised form 27 October 2021; Accepted 2 January 2022
Available online 11 March 2022
0141-9331/© 2022 Elsevier B.V. All rights reserved.

systems and the computation and communications domain. In short, time synchronization in MC-SAN systems is a fundamental concern for the coordination of embedded devices and event control in industrial communication environments.

MC-SANs with TSP have received increased attention in process industries and are being used for many industrial operations. The clock synchronized design of the high-precision motion control is considered a necessity in many industrial case studies. In case of IIoT, each of the industrial applications requires different level of synchronization accuracy based on the criticality of applications, existing infrastructure, network bandwidth and special hardware. In most industrial automation applications, the time synchronization requirement of DI/DO is 200 μs, and high-performance synchronous motion control requires real-time response times < 250 μs [9]; reaction times may be as short as 1 μs [10]. In addition, the different kinds of industrial applications should be reinforced by industrial communication [9–12]. The authors stated the particular requirements of the industrial applications and the clock synchronization categories for the smart grid. The time synchronization classes and the respective accuracy requirements along with their applications are listed in [12]. As described in [12], the accuracy requirement of PMU and distributed measurement is 1 μs; the automated fault recording is 100 μs. Therefore, it is challenging to meet the PTS stringent latency requirements of feedback control in industrial sensor and actuator networks.

In many industrial case studies [13–16], the high-precision time synchronization control is a necessity. The information gathered from different types of sensors has an important role in the IIoT. Time alignment technique for sensor data coming from diverse equipment is very important. Algorithms of time synchronization must have the precise timing of a sequence of events. The impact is caused by the incongruity in computing from the synchronization error of the industrial sensor and actuator networks. This impact comprises the wrong alarms and judgments, needless expenditures, or even damage the equipment. Desynchronized clocks in industrial applications can degrade the system performance and damage the infrastructure [12]. Any further synchronization error will expose infrastructure in a risky industrial environment. In [14], the authors discussed the case of a smart grid that installed smart meters and intelligent electronic devices at Intelligent substation automation systems. The absence of accurate synchronization will affect the electrical safety of the equipment and lead to a cascading failure. Moreover, if robotic manipulators working in a group lose the synchronization can result in clashes of their acts and upset the production pipeline.

1.2. Motivation

Recently, academicians and commercial practitioners have focused on developing high-precision time synchronization technologies and products for smart manufacturing and IIoT applications. Ref [17]. proposed a uniform (re)configuration approach called MidSN, which is a middleware architecture for processing over embedded sensor and actuator networks. Idrees, Zeba, et al [12]. investigated the different ways for PTPv2 evaluation. The software-based PTP employment was conducted in [17] and [19] based on Linux operating systems (OS). In [17], the authors used PTPv2to measure one-way delay and quantify packet loss ratios and proposed a software-based PTP as a network monitoring tool. The authors' findings reveal different latency, latency variance, and packet loss characteristics. According to the PTP, [19] designed the message sending and receiving process at the application layer. In addition, depending on the Linux kernel, the slave clock time is adjusted. The hardware-based PTP got many benefits over the software-based PTP. The main uncertainty source of a software-based synchronization approach is the timestamp method. Grandmaster clock with devoted hardware could be able to deliver precise nanosecond timestamp resolution. The authors in [15] emphasize software-based synchronization is unsuitable for IoTs. The proof of concept confirms the feasibility of hardware-based synchronization systems for IoT.

The Field Programmable Gate Array (FPGA) is the best candidate to fulfill the flexibility requirements of the emergent Ethernet-based protocols. The White Rabbit Precision Time Protocol (WR-PTP) switch is manufactured by Seven Solutions [20]. Ethernet frames are exchanged and connected directly to a Xilinx FPGA characterized by very low latency. Ref [16]. presented a synchronization solution which enables FPGA-based devices to synchronize with sub-nanosecond accuracy over optical ber links to WR-PTP devices. Hardware timestamps are bound to one period of the reference clock in PTPv2. The time delay between signals can be measured in a straightforward way using regular FPGA logic elements. The time delay measurement is performed using FPGA logic elements. As stated in [12], the hardware-timestamping was FPGA-based, and straight integrated to transceivers. With hardware-timestamping, a mean clock synchronization error of 1.1 ns was attained in [12]. Recently, [21] proposed an enhanced clock synchronization method using FPGA and ARM software and hardware co-design, in order to improve synchronization accuracy. The experimental results presented in [21] that the slave clock is synchronized with the master clock with the time offset within the range of 40 nanoseconds. In [22] authors designed a high precision time digital converter (TDC) based on FPGA and ARM. The result presented that the precision of TDC is 43.2 ps. In [23], the authors designed and implemented a clock synchronization method that supports the AS6802 standard. This method uses FPGA to record the transparent clock, replacing the MAC IP core timestamping approach in the previous Ethernet time synchronization solution. The system has been tested and the synchronization accuracy is less than 60 nanoseconds. However, studies on well-known synchronization technologies have not considered robustness, interoperability, deployability, and scalability for IIoT scenarios. The aforementioned factors are believed to affect synchronization accuracy and the overall performance of the IIoT system. These protocols are not appropriate for the main synchronization accuracy requirements of mass data processing capability. Some of these novel proposals have lacked cost-effectiveness preventing them to be extensively used in real industrial scenarios. In addition, industry-related commercial solutions are notably expensive; therefore, the cost of integrating all performance indices into an IIoT system will be very high.

The objective of this manuscript was to develop an enhanced clock synchronization method capable of enabling time-delay control, proportional correction, precision time synchronization for improving synchronization accuracy and optimizing cost-effectiveness. The concept of the proposed method is based on the delay evaluation model of IEEE 1588 PTPv2 to be defined for the proportional gain coefficient controller in a proportional correction model. The significance and contributions of this manuscript are summarized as follows:

We proposed a clock synchronization enhancement method to optimize the link delays, in order to improve synchronization accuracy. The method has an embedded PTS algorithm with a proportional correction scheme. The prototype system has been tested in a real network, and the synchronization accuracy can be achieved to 42 ns. The results show that the slave clock of the proposed method is synchronized with the master clock, leading to better robustness and significant improvement.

The novelty of the proposed method consists in the development of the proportional correction scheme based on augmented IEEE 1588 PTPv2 and proportional gain controller for the proportional error correction model. The very existence of time offset can limit the accuracy of time synchronization, especially in applications where high-precision positioning is essential. The method defined the delay evaluation model with superior performance. Moreover, the model highlights the need to strengthen a range of essential offset, time delay, estimate proportion, and synchronization correction.

Third, we developed a development and test platform by using a high-speed FPGA controller to reduce fabrication cost and time. A consistent and systematic design methodology is adopted for the rapid

development platform. The proposed PTS hardware module is designed to realize precision time synchronization for a decentralized multiaxis motion control system. The developed platform was used to verify the accuracy of time synchronization of the proposed PTS controller hardware. Finally, using our robust PTS architecture, we fabricated a prototype PTS controller chip without an extra CPU and tested it.

1.3. Structure of this article

The remainder of this manuscript is organized as follows. The remainder of this manuscript is organized as follows Section 2. represented an overview of the well-known synchronization protocols. In Section 3, the clock synchronization correction algorithm model is introduced. In Section 4, the overall architecture of the proposed PTS-based system is described. In Section 5, detailed implementations and precision time synchronization control for multiaxis servo motion for a robotic manipulator is discussed. Finally, in Section 6, the conclusions of this study are summarized.

2. Overview of the well-known synchronization protocols

In this section, we perform a review of existing state-of-the-art time synchronization technologies, including Global Positioning System (GPS), Network Time Protocol (NTP), Simple Network Time Protocol (SNTP), PTPv2, and WR-PTP. These well-known synchronization protocols primarily relate to wired networks. These well-known synchronization protocols primarily relate to wired networks.

GPS-based time synchronization is the finest way to attain nanosecond accuracy [24]. GPS devices are quite reliable in providing synchronization and localization measurements. A GPS satellite contains atomic clock information and GPS receivers obtain precise time information. This information is critical, and it is dispersed to all the associated nodes. Nevertheless, GPS deployment at every IoT node is very expensive in terms of cost and power, especially for networks with a large number of nodes [25]. Outdoor GPS antennas are required in many scenarios; these outdoor antennas are susceptible to vandalism. Furthermore, GPS performs poorly in indoor environments. GPS signals reception is difficult which makes it an unattractive approach.

NTP is the oldest internet protocol and is a high fault tolerance protocol [27]. In terms of synchronization accuracy, NTP can provide microseconds scale accuracy using a software-based solution. Although NTP may not be ideal for real-time, it is widely used in today's environment. NTP was established for packet-switched networks. NTP uses the RTT (round-trip time) as estimator. Once the RTT is calculated, it is used as a determining factor of time delay. The RTT packet-switched process as described in [15]. NTP clients send packets to the NTP server at certain times and wait for the response. The packet's RTT can be calculated from the time the response was received. From this, in turn, the latency can be estimated assuming symmetrical packet delays. NTP implements a hierarchical architecture. Each level in the hierarchy is known as a stratum. In principle, NTP synchronization accuracy is highly related to the performance of the underlying network. The lower the stratum number the closer the device is to an original time source. A server synchronized to a stratum N server runs at stratum $N + 1$. At the very top are hardware reference clocks, such as GPS or radio time receivers, known as stratum 0 hardware clocks. Stratum 1 NTP servers are synchronized to an atomic clock and therefore have the most accurate time. The distribution of the time information to multiple layers results in a loss of accuracy.

SNTP is a less complex implementation of NTP used to synchronize computer clocks in the Internet, but without requiring the storage of state over extended periods of time [28]. SNTP typically provides time within 100 milliseconds of the accurate time. SNTP also does not provide the complex filtering and statistical mechanisms of NTP. SNTP also does not realize a few of the difficult time-tracking procedures. SNTP is appropriate for embedded devices as it needs few resources as compare to the NTP at the cost of lower synchronization accuracy. The SNTP method is much simpler than the NTP, given that it bypasses various steps and only periodically adjusts the time, obtaining less accuracy. SNTP is an adaptation of the NTP, but without requiring the storage of state over extended periods of time. SNTP does not realize the difficult time-tracking procedures and also does not provide the complex filtering and statistical mechanisms of NTP. SNTP is appropriate for embedded systems and applications in which full NTP capability is not required.

PTPv2 was introduced by the IEEE 1588–2008 in order to provide time accuracies beyond those attainable using NTP, thanks to a technique called hardware-assisted timestamping. This protocol provides hardware-level time precision by a peer-to-peer link, using devoted timing hardware to every port. The exchange of time-tagged messages between master and slave clocks, used to calculate the link delay between the two clocks. In addition, the slave node can account for time offset when adjusting its clock time with respect to the one of its master clock. PTPv2 is easy to deploy and expedites convergence. It is used to synchronize real-time clocks (RTCs) in web nodes through precisely timed broadcasting. This protocol features minimal consumption of online network resources and high stability; therefore, it is commonly adopted for industrial time synchronization. However, the accuracy of the PTPv2 synchronization is implementation-dependent. PTPv2 allows attaining synchronization in the range of nanoseconds provided that all the network components are PTPv2 enabled. Ref [14]. specifies all PTPv2 network nodes are equipped with PTP-aware routers or switches, implementing the hardware-timestamping scheme in order to measure and compensate for the time spent by messages in queuing at their own ports.

One of the most important approaches based on PTP is WR-PTP. In Ref [16]., the authors present WR-PTP as pre-standard implementation of the IEEE 1588–2019 High Accuracy Default PTP Profile. The profile formalizes WR-PTP as part of the IEEE 1588 standard, which helps bring best-in-class performance to its potential applications. WR-PTP is a low latency, Ethernet-based time distribution method for general purpose data transfer and synchronization. WR-PTP protocol was initiated and developed by the European Organization for Nuclear Research (CERN) to align the clocks of their accelerator complex. WR-PTP is a multi-collaborative open hardware project aiming at the distribution of timing with sub-nanosecond accuracy to thousands of nodes connected within an Ethernet network [26]. It means that hardware designs and driver software are open and available on the internet. The project facilitates collaboration between different scientific research organizations and stimulates improvement of the equipment [29]. WR-PTP synchronization scheme uses the existing standards to achieve sub-nanosecond accuracy, these include Synchronous Ethernet (SyncE [30]), PTPv2 along with precise phase measurement technique. In addition, WR specifies many innovative technologies, such as enhancements for the use of Ethernet hardware clocks and the calibration of asymmetry in fiber-optic cabling [14]. This approach has been validated during the last years.

Ref [12, 15]. emphasized that typical approaches such as NTP and GPS are unsuitable for IIoTs, due to the special requirements of the sensor and actuator networks (limitation of energy, computing power, and memory). In [12], the authors have concluded that a non-PTP switch lowers the performance even with very accurate and expensive PTPv2supported equipment. Ref [12–16]. have also compared the well-known time synchronization technologies. The results concluded that the PTPv2 is the most appropriate solution to robust clock synchronization. Ref [16]. reviewed different issues that have prevented the industrial adoption of White Rabbit in the past. In [16], the authors enumerate the main obstacles that have prevented the industrial adoption of WR-PTP in the past. Several issues had been discussed including: immaturity of technology; deployment cost; interoperability capability; synchronization robustness; and data transmissions.

In summary, with the increasing deployment of large-scale distributed sensor nodes and associated networks in the IIoT system, highly

reliable and precise clock synchronization gains ever increasing importance [31]. While GPS, PTP, NTP, SNTP and WR-PTP are popular time synchronization technologies, there are disadvantages in terms of synchronization accuracy achieved, specialized hardware required, robustness and cost-effectiveness. Moreover, they have not considered the interoperability, interconnectivity, deployability, and scalability requirements of a complex IIoT system. This work attempted to use hardware-timestamping methods to complete the delay correction model of the proposed PTS algorithm. The goal of the proposed algorithm was set to achieve synchronization accuracy reliability and ensure timeliness of the exchanged datagrams.

3. Clock synchronization correction algorithm model

In general, a robust IEEE 1588 PTPv2 system can ease the synchronization process by correcting the offset and integrating the transmission delay by supporting interoperability for the master and slave exchange messages and diverse nondeterministic delays presented from an application level in the PTS system. This section defines the delay evaluation model and the design of a robust clock synchronization architecture with superior performance. Moreover, it presents the proportional error correction model in our proposed embedded PTS algorithm. The notation adopted in this article is summarized in Table 1.

3.1. Time offset

The time of the two terminal nodes at the two ends of the master and the slave is not synchronized; that is, there is a time difference between the local clock time of the master and the slave. If the time of the master clock is used as the benchmark, there will be a time offset (T_{offset}) between the slave clock and the master clock. If the T_{offset} between the two terminal nodes can be corrected accurately, highly accurate synchronization for the Modbus-SAN system can be achieved. PTPv2 is used to synchronize clocks throughout a network. Clocks determine the T_{offset} between themselves and their master. Let T represent the physical time. For a given master and slave devices, T_m represents the time measured by the master clock and T_s represents the time measured by the slave clock at physical time T, and T_m is defined as

$$T_m = T_S + T_{offset} \tag{1}$$

In reality, however, with the end-to-end time delay resulting from the transmission of the master local clock as processed by the system as well as the master–slave through the communication interface, it is difficult for the slave to obtain the master clock instantly and variations will exist in the calculated time offset. Therefore, with only correction to the time offset, the time synchronization might still demonstrate a significant uncertainty.

In addition, the time spent to transmit the timestamp between the master and the slave needs to be considered to further improve time accuracy.

Table 1
Notation and nomenclature used in this article.

Symbol	Description
T	The physical time
T_m	The time measured by the master clock at physical time T
T_s	The time measured by the slave clock at physical time T
T_{offset}	The master clock is used as the benchmark, the time measured by the master-to-slave offset at T
T_{delay}, T'_{delay}	The time measured by the master-to-slave delay at physical time T
T_1, T_2, T_3, T_4, T_5	Precise timestamps are captured at the master and slave clocks
M_{delay}	The mean time of T_{delay} and T'_{delay}
T_{diff}	The differential error between the master clock and slave clock as well as the error of the clock generator or environmental disturbance.
T_{error}	The time difference was caused by the clock generators of the local oscillator.
K_p	The proportional gain coefficient of the proportional error correction model
T_{diff}	Estimating the mean value from T_{diff}
T_{error}	Estimating the mean value from T_{error}
T_p	Proportional error

3.2. Transmission time delay

To calculate the T_{offset} between the master clock and slave clock, we need to transmit the timestamp to another terminal node. However, in addition to the transmission time of the physical cables, the transmission process also includes the processing of the communication modules; therefore, transmission time delay will definitely occur.

The key to improving the time synchronization performance is how to accurately estimate the transmission time delay. As illustrated in Fig. 1, the time difference between the master clock and the slave clock represents the transmission delay.

Because the transmission time delay is considered, the term on the right-hand side of (2) is added to T_{delay}, where T_{delay} represents the time measured by the master-to-slave delay at time T. Furthermore, T_m is the time of the slave clock after the time offset and delay are added.

$$T_m = T_s + T_{offset} + T_{delay} \tag{2}$$

By using the symmetry characteristics of the bidirectional communication architecture, the transmission delay is assumed to be the mean value of the bidirectional transmission delay. As illustrated in Fig. 2, T_{delay} represents the time required for the master to transmit a *Sync* packet to the slave, and T'_{delay} the time required for the slave to send the *Ack* packet back to the master. The mean value of the transmission delay can be calculated through the timestamps of the sending and receiving actions.

According to (3), the average transmission time delay M_{delay} is the mean time of T_{delay} and T'_{delay} Although the T_{delay} value cannot be obtained directly, we can calculate M_{delay} by using the difference between the timestamps.

$$M_{delay} = \frac{1}{2}\left(T_{delay} + T'_{delay}\right) \tag{3}$$

Considering the initial time offset and transmission time delay, the result obtained after time synchronization. Although M_{delay} is not the accurate transmission time delay, the accuracy of time synchronization can be guaranteed to be improved compared with that when the transmission delay is ignored. M_{delay} is estimated using the average value. Under the conditions where there is no guarantee that $T_{delay} = T'_{delay}$, M_{delay} can be defined by (4) in terms of T_{diff} as

$$M_{delay} = T_{delay} + T_{diff} \tag{4}$$

where T_{diff} represents the differential error between the master clock and slave clock. T_{diff} can be obtained using (3) and (4):

$$T_{diff} = \frac{1}{2}\left(T'_{delay} - T_{delay}\right) \tag{5}$$

Because T_{delay} cannot be calculated directly, M_{delay} is used as the estimated value of the transmission time delay. From (5), it can be proven that T_{diff} is fixed under the same system architecture. Therefore,

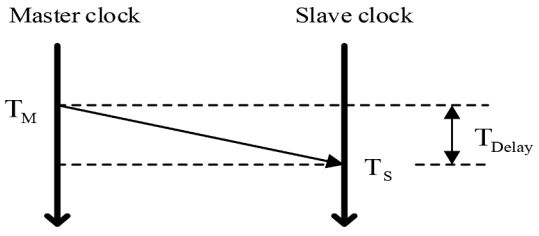


Fig. 1. Transmission time delay model.

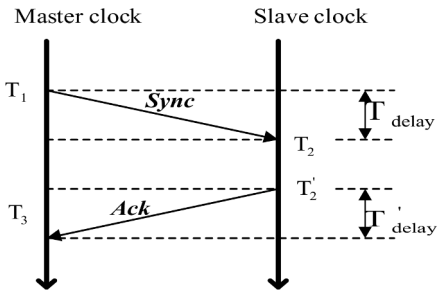


Fig. 2. Packet timestamp transmission time delay.

considering the accuracy of time synchronization, the proportional error is further corrected after time synchronization.

3.3. Proportional error correction model

In reality, the interference in communication between different devices should be considered; further, small errors in hardware devices, such as the clock generators of different devices, affect the accuracy of time synchronization. A clock generator is an electronic oscillator that produces a timing signal for use in synchronizing a circuit's operation. There may still be fixed differential values in clocks with the same setting, and this small error cannot be ignored as it accumulates over time. A Clock signal provides frequency and phase. If stable oscillators are used in both nodes, the frequency offset between them varies slowly and enables achieving relatively high synchronization performance. Since, cheap oscillators properties vary with power, aging, and heat. Therefore, two similar crystals are not guaranteed to oscillate at the frequency. However, a low stability reference may cause a frequency offset between the two nodes which may accumulate to a significant time difference.

The time difference T_{error} was caused by the clock generators of the local oscillator. We use T_{error} and the proportion of master–slave time difference after time synchronization to conduct proportional error correction.

After time synchronization is performed, the master sends a timestamp T_4 to the slave, as demonstrated in Fig. 3, and the slave clock receives it at timestamp T_5.

Ideally, the following equations hold after time synchronization between the master and the slave:

$$T_4 + M_{delay} = T_5 \tag{6}$$

Because the system is set up in a natural environment, a naturally occurring differential value will exist after time synchronization. The synchronized T_{error} can be obtained by transposing the term in (6), as presented in (7).

$$T_{error} = T_4 + M_{delay} - T_5 \tag{7}$$

After the time offset and transmission time delay are corrected, a fixed proportion of error still exists between the master clock and slave clock; proportional control can be used to achieve a more accurate time synchronization effect. Proportional control responds to the system by multiplying the error value and the proportional gain coefficient K_p. The K_p value may vary in different systems and needs to be determined through experimental results.

To determine the proportional gain coefficient K_p, the counter values of the master clock and slave clock need to be measured first to find the difference T_{diff} between the two based on the measurement results.

T_{diff} is derived from the differential error between the master clock and slave clock as well as the error of the clock generator or environmental disturbance. After repeating the measurement several times, multiple sets of differences T_{diff} and T_{error} are obtained. The two types of values are averaged separately, and the proportional gain coefficient K_p is obtained from the two average results, as demonstrated in (8):

$$K_p = \left(\frac{\widehat{T}_{diff}}{\widehat{T}_{error}}\right) \tag{8}$$

Precision time synchronization corrects the initial time offset and the transmission time delay of the master clock and slave clock as well as the fixed difference estimated by the error offset. Finally, the clock value is accurately synchronized with the master clock and slave clock as presented in (9).

$$T_m = T_s + T_{offset} + M_{delay} + K_p \times T_{error} \tag{9}$$

To estimate the T_{offset}, T_{delay}, M_{delay}, T_{diff} and T_{error}, the complete time synchronization algorithm requires four-stage packet switching. Each stage defines one type of packet, and the packet type name defined by the PTPv2 is adopted Fig. 4. illustrates the flow of the PTS algorithm model, recording timing of each timestamp, and timing to send a timestamp to the slave for time synchronization correction.

The specific synchronization of the proposed PTS algorithm model is described as follows: initialize the network, select 2 common clocks, and determine them as the master clock and the slave clock respectively; then synchronize the clock between the master and slave clocks. All operations are completed on the master side.

Stage 1: The master clock first sends a Sync message to the slave clock to be synchronized and records the local timestamp T_1; instructs the slave clock to receive the Sync message and record timestamp T_2. Then slave clock sends back an Ack message (Sync); and records timestamp T'_2 again.

Stage 2: The master clock sends a Follow_Up message and the T_1 information is placed in the Follow_Up message and sent to the slave

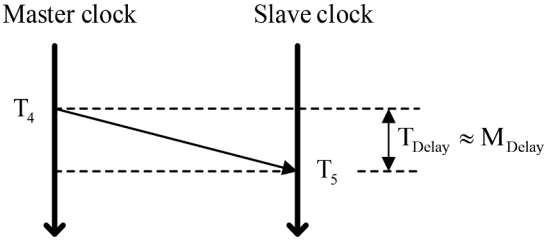


Fig. 3. Packet timestamp transmission time delay II.

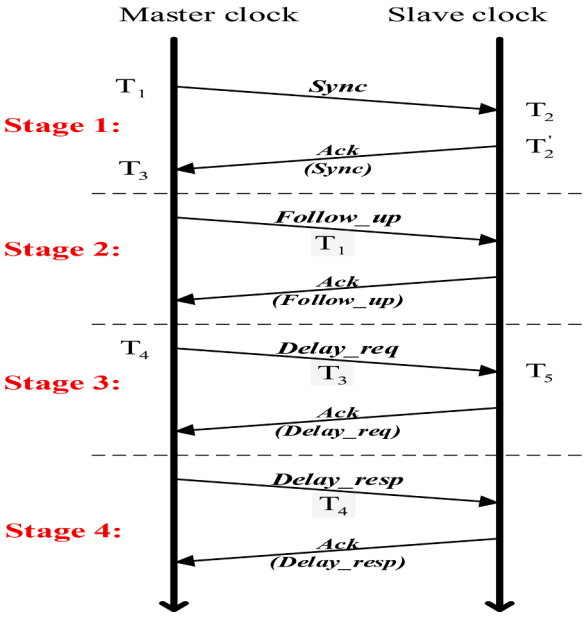


Fig. 4. Proposed PTS algorithm model.

clock. When the Follow_Up message reaches the slave clock, because packets have their corresponding identifiers (such as IP address and destination port, etc.). Hence the slave can accurately obtain T_1 from the packet at this moment; and the slave sends back an Ack (Follow_Up) message.

Stage 3: At this time, the master clock needs to send a Delay_Req message to the slave clock, and at the same time record the timestamp T_3 of the Delay_Req message sent by the master clock. The slave clock receives a Delay_Req message and sends an Ack (Delay_Req) message. When the Delay_Req message reaches the slave clock, the same as Follow_up, the slave clock can get T_3 information and sends back an Ack message (Delay_Req).

Stage 4: The master clock sends a Delay_Resp message to the slave clock. Delay_Resp message, which contains the timestamp T_4 of the master clock. When the Delay_Resp message reaches the slave clock, the slave clock can also get T_4 information and sends back an Ack message (Delay_Resp). The master clock records timestamp T_5.

So far, the timestamps of T_1, T_2, T_3, T_4, and T_5 have been accurately obtained. According to the obtained timestamps T_1, T_2, T_3, T_4, and T_5, the mean time M_{delay} of T_{delay} and T'_{delay} can be shown in the Eq. (3). The differential error T_{diff} between the master clock and slave clock can be calculated in the Eq. (5). The error value Terror between the master and slave clocks can be calculated in the Eq. (7). Next, the proportional gain coefficient K_p is performed on the proportional error correction model in the Eq. (8). When the proportional error correction is completed, the clock value is accurately synchronized with the master clock and slave clock as presented in (9). Finally, the master completes the precision time synchronization of the clock.

4. System design

In this section, we used a systematic design methodology for the hierarchical design, behavioral modeling and automatic hardware synthesis of complex computing systems which have generally multiple functional modules with complex interconnection/intercommunication. The synthesized hardware controllers can be obtained rapidly by using the synthesis rules provided in Section 4.5. This reduces the development time and cost as well as the procedure of traditional trial-and-error approaches.

4.1. The methodology of the SYSTEM () design and synthesis

Our methodology starts with the system specification, followed by the hierarchical and modular design of the system's functional architecture using the function modeling method: IDEF0 (ICOM DEFinition0). For each independent functional module, we use a discrete-event system modeling tool, namely GRAFCET to describe its behavior. Then, according to a set of proposed synthesis rules, each GRAFCET model can be synthesized as a VLSI hardware circuit using VHSIC hardware description language (VHDL). The schema of the modeling and synthesis methodology is shown in Fig. 5.

(1) Hierarchical Modular Design Using IDEF0

First, we define system specifications that must be satisfied for the functions and behaviors of a complex system. IDEF0 provides a convenient graphical modeling tool for hierarchical decomposition of functions (see Fig. 6). Through hierarchical design, a complex system can be divided into independent sub-functional modules, so as to describe more detailed functional models. The behavior in every IDEF0 functional module is taken as an independent discrete-event system (DES). We use GRAFCET to model the evolution of the discrete events.

(2) Discrete-Event Modeling Using GRAFECT

The GRAFCET language provides a formalism which takes the concurrency of a dynamic system into describes it in a intuitive way. GRAFCET was adopted to model the individual DES module and the hierarchical, pipelined behaviors of the system. A GRAFCET model can be represented by a specific hierarchical master–slave structure (Fig. 7). This structure is similar to that of the main program and its call function. The main program calls the subprogram for waking up the function and starting the independent module. When the function terminates, the finished signal will be returned to the main program and the subprogram is terminated. Because of the simple structure, the design of a complex computing system can be partitioned into a set of decoupled tasks whose results will be seamlessly integrated at last.

(3) Hardware Synthesis

Because the GRAFCET language is characterized by a rigorous state/transition structure, it is not difficult to elaborate a set of synthesis rules mapping each GRAFCET building block into hardware architecture. In Fig. 8, we propose a set of VHDL synthesis rules corresponding to the GRAFCET building blocks. Each GRAFCET structure can be synthesized in a segment of VHDL code block.

4.2. Overall architecture of PTS-based Modbus-SAN system

The system hardware architecture of the proposed PTS-based Modbus-SAN system comprises two major parts: a PTS function module backbone for PTS to execute data and instruction exchange, and an MC-SAN system module for IIoT management and database operations. This architecture is based on the functional hierarchy of automation and distributed control systems (Fig. 9).

The Modbus-SAN system module is mainly responsible for communication transmission, reception, and exchange of commands as well as the control of other function modules. The module mainly controls communication with the PTS function module and packaging processing tasks. The hierarchical architecture of the Modbus-SAN system module

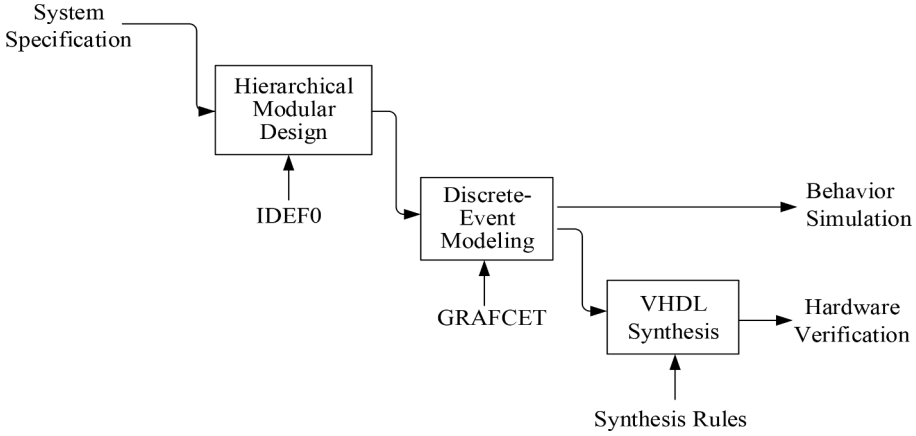


Fig. 5. The high-level modeling and synthesis methodology.

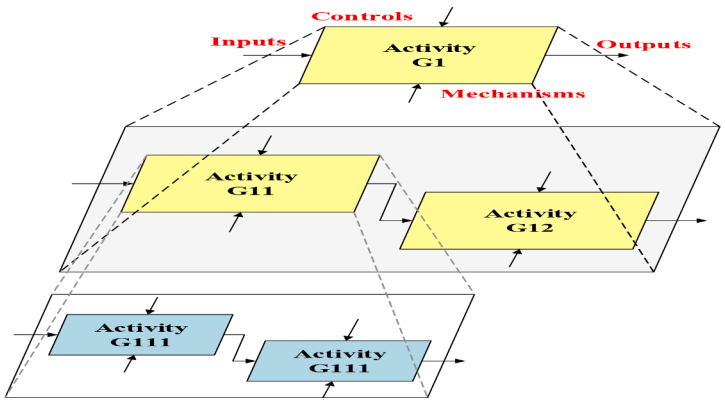


Fig. 6. Hierarchical discrete-event modeling by IDEF0. An IDEF0 model comprises functional blocks and arrows interfacing these blocks. Each functional block represents an activity, process, transportation, or operation in a system. Arrows that connect functional blocks represent relationships between them and describe the data needed for executing the system. There are four arrow types: input, output, control, and mechanism. These are collectively called ICOM; IDEF stands for ICOM DEFinition.

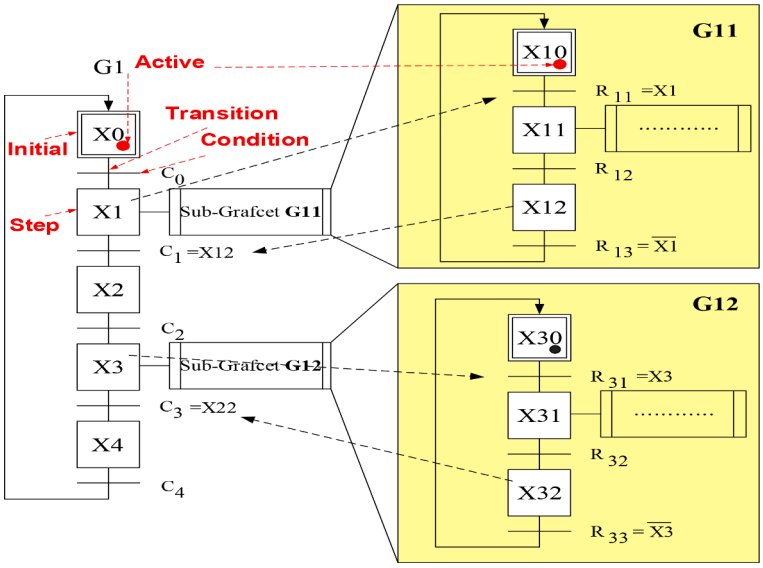


Fig. 7. Discrete-event modeling by GRAFCET. A hierarchical GRAFCET model can be represented by the specific structure of the master GRAFCET and its slave GRAFCET, termed sub-GRAFCET. sub-GRAFCET represents an independent DES that interacts with the master GRAFCET. It receives a state signal from the master as a transition signal for activating the subsystem, and it sends a finishing signal to the master when all functional steps are completed; elements of GRAFCET are highlighted in red font. G1 is a GRAFCET module; G11 and G12 are its sub-GRAFCET modules (yellow blocks); X0, X10, X20 are respectively initial states of G1, G10 and G20. As the state X1 is activated, X0 is switched to inactive. Simultaneously, the active state in sub-GRAFCET G11 is also switched from X10 to X11, and G11 module starts autonomous evolution.

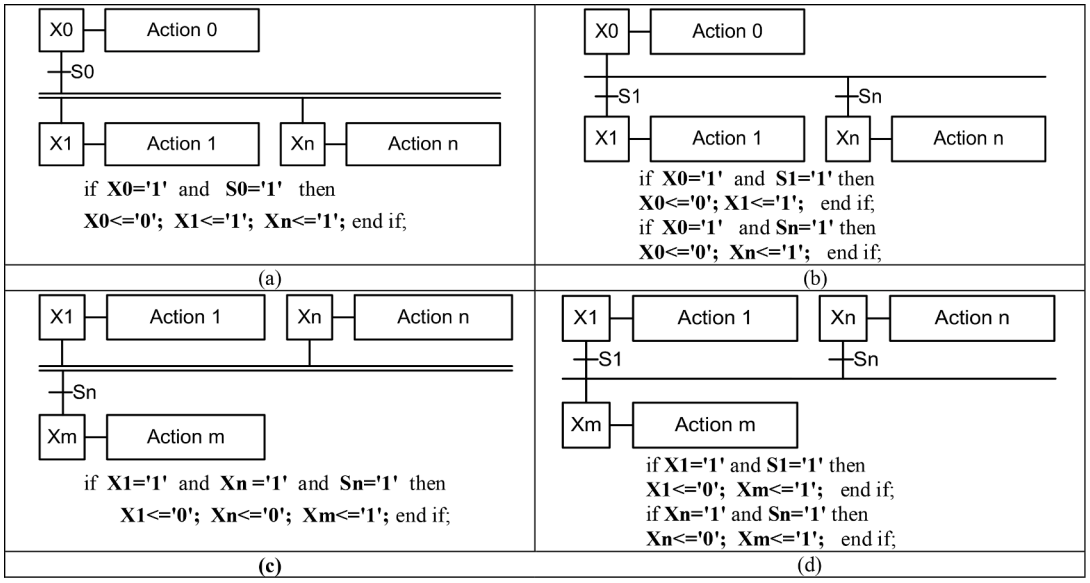


Fig. 8. Synthesis of VHDL code segment of GRAFCET model. Figure 8(a) and (c) represent the VHDL code blocks corresponding to the concurrent structures, and Figure 8(b) and (d) are the VHDL code blocks corresponding to the branching structures. Thanks to these generic synthesis building blocks, the transformation of the GRAFCET model to VHDL code is fast and straightforward.

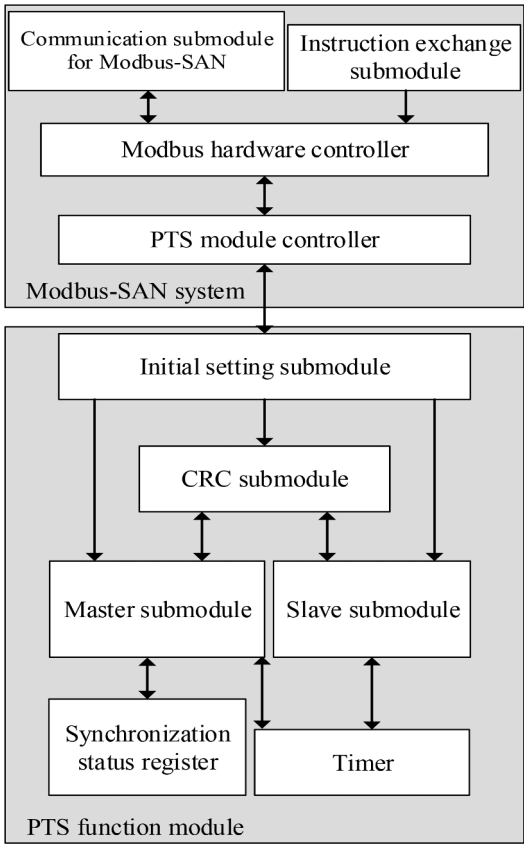


Fig. 9. Architecture of PTS-based Modbus-SAN system.

integrates the Modbus-SAN control submodule, instruction exchange submodule, Modbus hardware controller submodule, and PTS module controller.

The Modbus-SAN system module is mainly responsible for communication transmission, reception, and exchange of commands as well as the control of other function modules. The module mainly controls communication with the PTS function module and packaging processing tasks. The hierarchical architecture of the Modbus-SAN system module integrates the Modbus-SAN control submodule, instruction exchange submodule, Modbus hardware controller submodule, and PTS module controller.

The communication submodule for Modbus-SAN is responsible for the management of perception-layer sensors and actuators. The Modbus transmission mode was the high-speed, high-transmission-efficiency Remote Terminal Unit (RTU) transmission mode that is required in all devices. In addition, the Modbus function codes are adjustable. The submodule adopts the serial communications standard, which can perform one-to-many network topology transmission.

The Modbus-SAN control submodule is responsible for the management of perception-layer sensors and actuators. The submodule adopts the serial communications standard as communication submodule, which can perform one-to-many network topology transmission. By controlling discrete event signals, it is ensured that the network system only has one network port transmitting data at a time, thereby avoiding collisions and enhancing the reliability of system data exchange.

The instruction exchange submodule is the control interface for the users. It can directly control the hardware; issue instructions to the internal submodules of the Modbus-SAN system; and be responsible for initializing the devices, setting module signals, or issuing instructions for synchronization execution. The Modbus hardware controller is the bridge that links the submodules in the Modbus-SAN system and is responsible for disassembling the packets received from the communication submodule and interpreting the user's instructions, thereby initiating the corresponding function submodules or controllers based on the content of the obtained message. The PTS module controller is mainly used to receive the start and end signals and to control communication with the PTS function module.

The PTS function module forms the core of this research. The integrated Modbus-SAN system is expanded through Modbus function codes, and the Modbus hardware controller and the PTS module upper-layer controller are used to drive the PTS function submodule to achieve high-precision time synchronization. The hierarchical architecture of the PTS module integrates the initial setting submodule, cyclic redundancy check (CRC) submodule, master submodule, slave submodule, synchronization status register, and timer. The initial setting submodule is responsible for setting the ID of the devices and the initial time of the timer, determining the master or slave mode, and handling the initial parameter setting of each submodule. The CRC submodule is responsible for checking the PTS data. Only after confirming that the data is correct, the module work will continue to be performed, thereby increasing system reliability and data correctness.

After receiving the instructions or packets of the PTS module from the Modbus hardware controller, the master submodule will perform packet processing. When the synchronization is completed, the PTS packets will be output and sent out by the communication submodule. After receiving the PTS packets from the master submodule through the Modbus hardware controller, the slave submodule performs the time synchronization algorithm, corrects the time value of the timer according to the algorithm results, and sends back the PTS acknowledgement packet to the master after the completion of each operation. The synchronization status register is only used in the master mode.

Whenever the master submodule receives the acknowledgement packet from the slave submodule synchronization, the register will update the synchronization status for use by the master submodule. The timer is a 64-bit one, and signals are triggered according to the set time of the input clock to update the local clock through the continuous accumulation of values. The external module can set the time parameter and provide the time value to other submodules. After time synchronization, the slave submodule will modify the time value according to the synchronization result.

4.3. Hierarchical modular design of PTS function module using IDEF0

Fig. 10 demonstrates the IDEF0 hierarchical architecture diagram of the PTS function module. The PTS module (A0) in the top layer is responsible for controlling the communication and control parameter settings between the sixth submodules of the second layer. The master submodule (A4) and slave submodule (A5) are the main function modules of the PTS algorithm and are responsible for recording and processing the algorithm timestamp. The other four submodules are function assistance submodules. The initial setting submodule (A1) sets the initial parameter setting for other submodules according to the initial parameter it receives, which includes the address of the devices and the master–slave mode setting. After the PTS data packets are input and through the processing of the master submodule or slave submodule, the master submodule will update the synchronization status register (A2) according to the processing result, and the slave submodule corrects the timer value (A3) according to the calculated time difference. Finally, before the PTS data packets are output, a 16-bit CRC code is generated by the CRC submodule (A6). The aforementioned CRC code is attached to the PTS output data packets as the output result, and then, the external end signal is provided.

4.4. Discrete-event modeling for PTS function module using GRAFCET

After the hierarchical modular design of the PTS function module is completed, the graphical GRAFCET language is adopted to model the individual DES module that performs control and communication

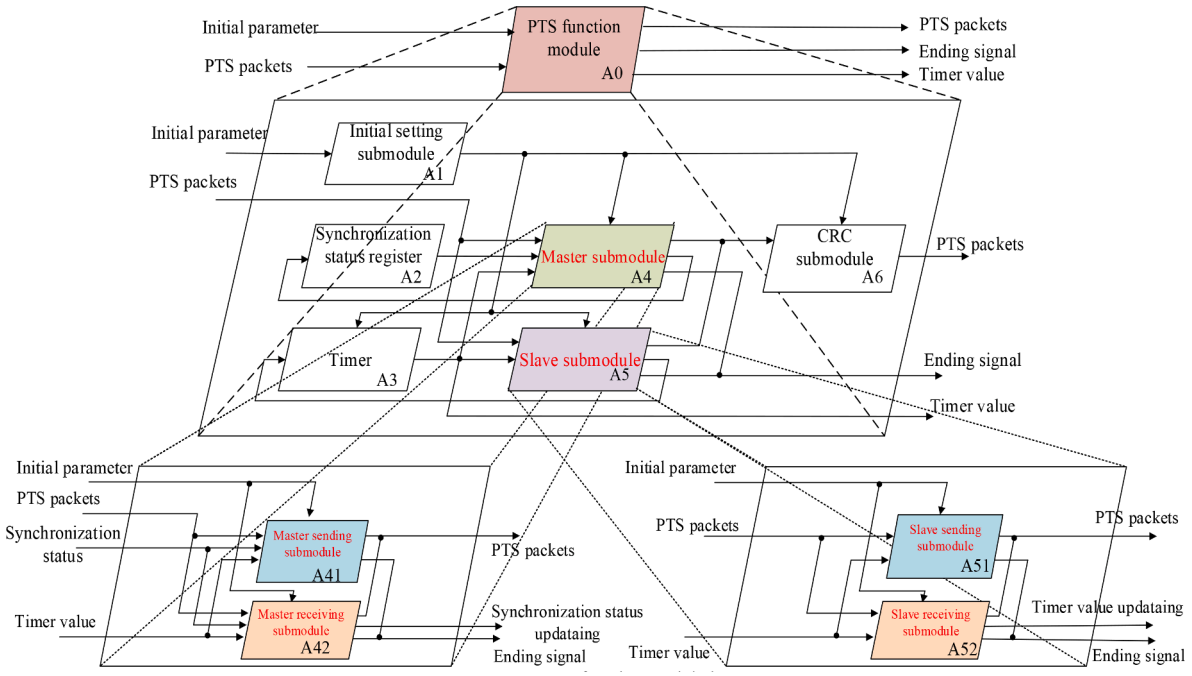


Fig. 10. PTS function module by IDEF0.

between modules through clearly defined functions and behaviors and the control of the active state and transition conditions Fig. 11.–Fig. 15 illustrate the sending (transmitting) submodule and receiving submodule of each of the five most important modules of the PTS controller chip including the PTS function module and master–slave architecture. We independently introduce the algorithm of each module and highlight their distinctive roles.

First, the definitions and features of variables and functions used by the algorithm in GRAFCET are described below:

T_n: Timestamp setting of the time data output by the submodule.

currentTime: Read the current timestamp value of timer.

Sel_Tx: Input signal responsible for choosing the sending/receiving selectors of the transmission submodule. If it is 1, select sending (transmitting) submodule, otherwise select receiving submodule.

Steps (Id, Status): Update the synchronization status register. **Id** uses a register as the array index (usually the Id number of the slave module), and the **Status** parameter is the status to be changed.

Received (T_i): Stores the timestamp data received from the outside

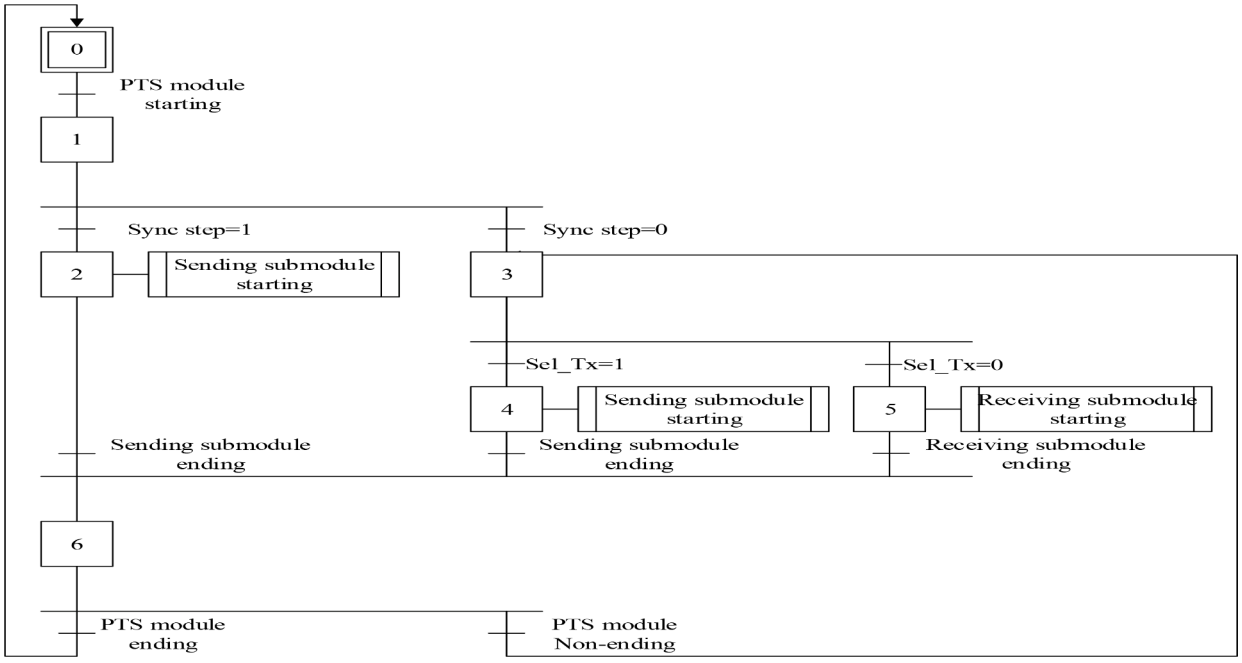


Fig. 11. GRAFCET modeling for PTS function module. When the PTS function module triggers the Modbus hardware controller, in addition to starting the initial setting submodule as a master or slave, it also needs to decide if it should execute the sending or receiving submodule. The PTS function module controller determines whether to execute the sending or receiving submodule according to the value of the input signal Sel.Tx (signal for sending/receiving selectors), and this controller can be used by both the master and the slave.

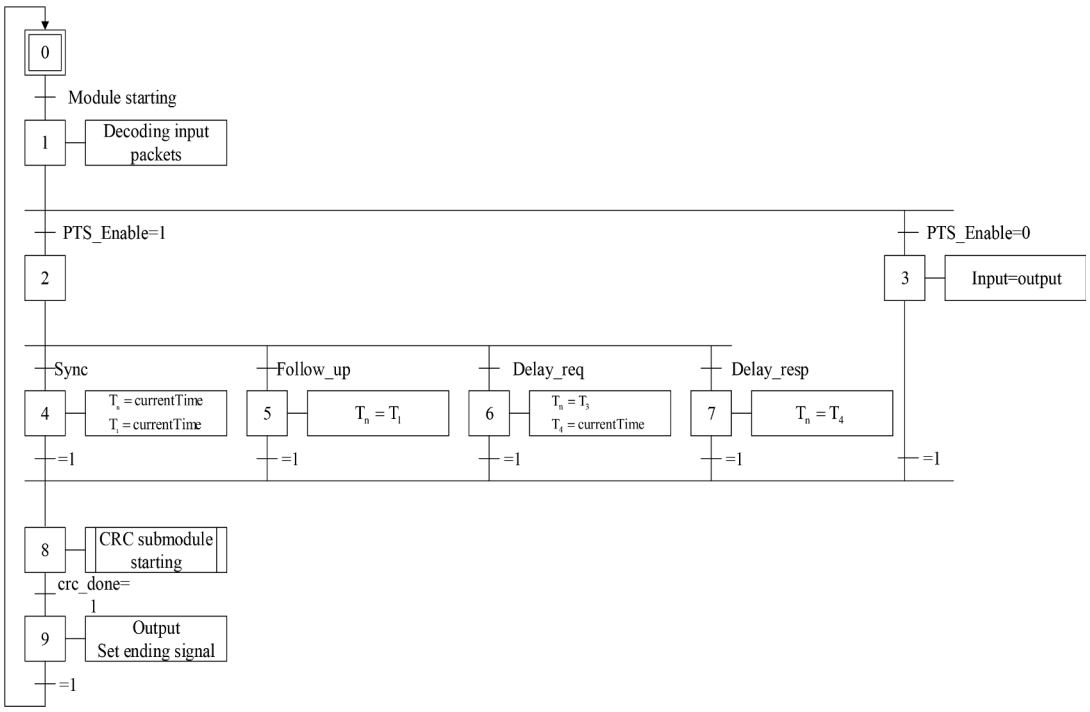


Fig 12. GRAFCET modeling for master sending submodule. After the master sending submodule is initiated, it decodes the inputted packet, indexes the state of the corresponding address in the status register according to the address information of the PTS packet, and performs different processing according to the state (Sync, Follow_Up, Delay_Req, and Delay_Resp), such as the timestamps of different meanings as well as the given output packets. Finally, the CRC code is generated by the CRC module and attached to the output result and the specific timestamp record of the PTS algorithm.

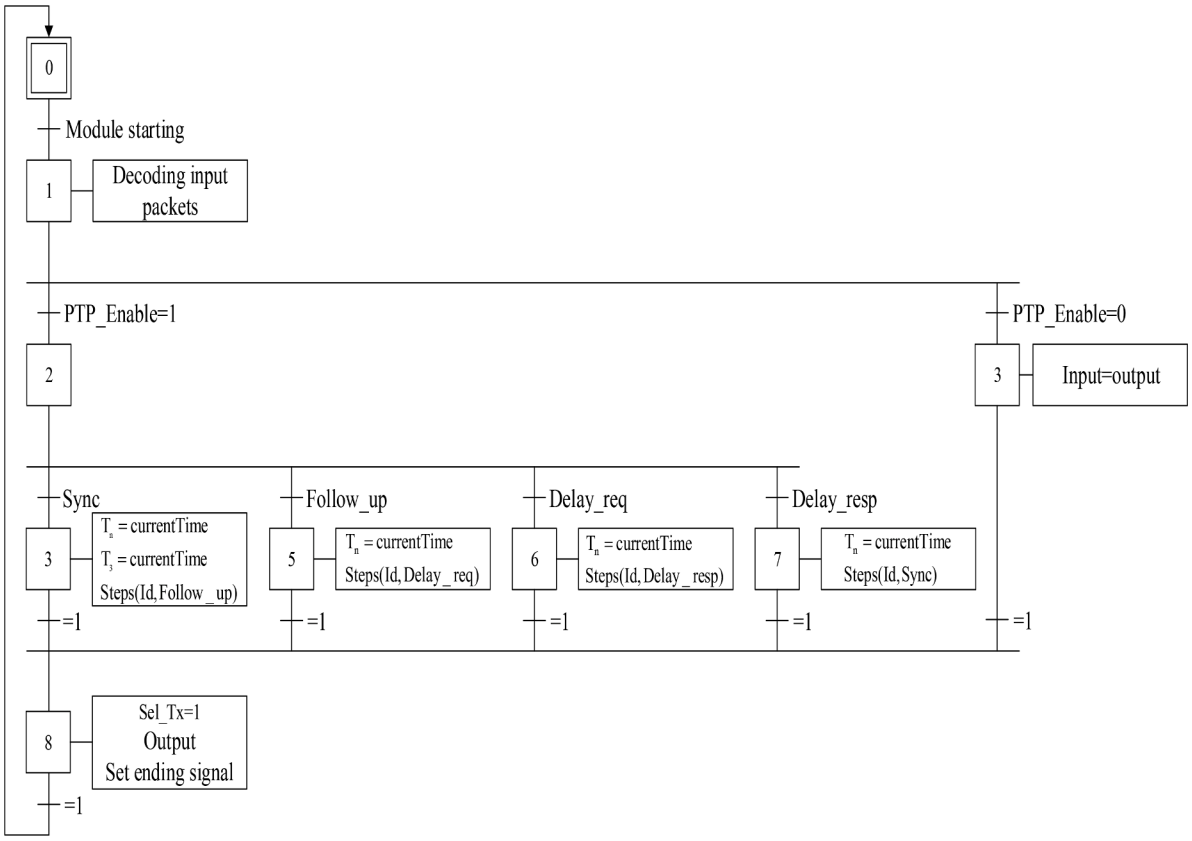


Fig. 13. GRAFCET modeling for master receiving submodule. When the master receiving submodule is initiated, it decodes the inputted packet, indexes the state (Sync, Follow_Up, Delay_Req, and Delay_Resp) of the corresponding address in the status register according to the address information of the PTS packet, The program fetches the current time as the timestamp of the output.

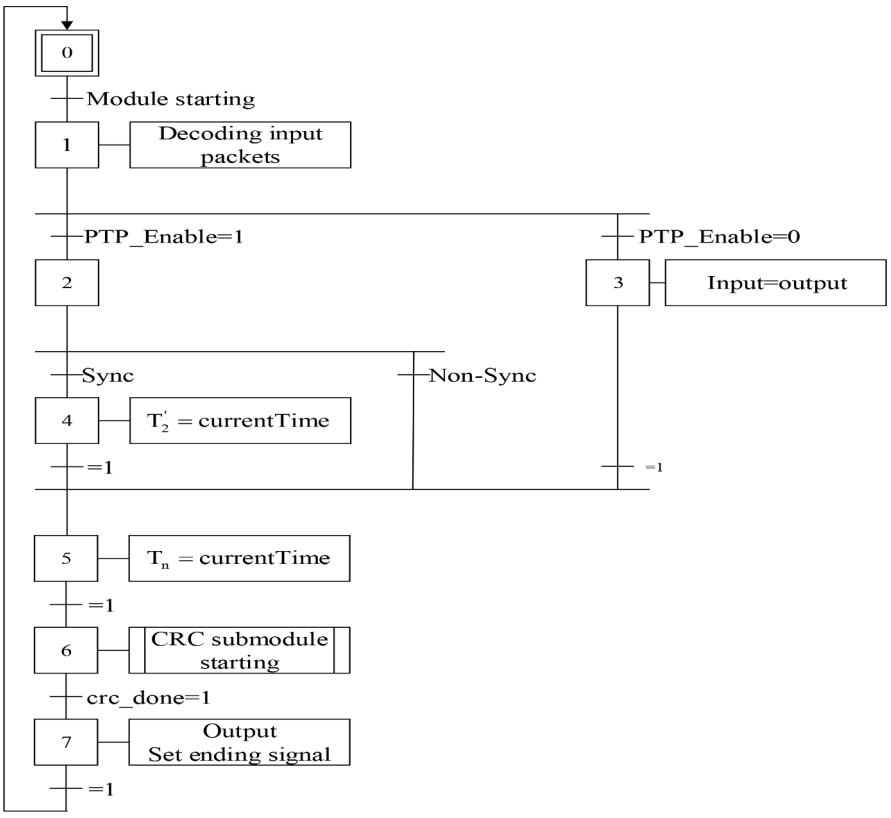


Fig. 14. GRAFCET modeling for slave sending submodule. After the slave sending submodule is initiated, it decodes the inputted packet and, according to the state information of the PTS packet, it can determine the synchronization steps that are being performed. If it is the Sync step, the current timestamp is recorded as T'2. The program fetches the current time as the timestamp of the output. Finally, the CRC code is attached to the output result and the specific timestamp.

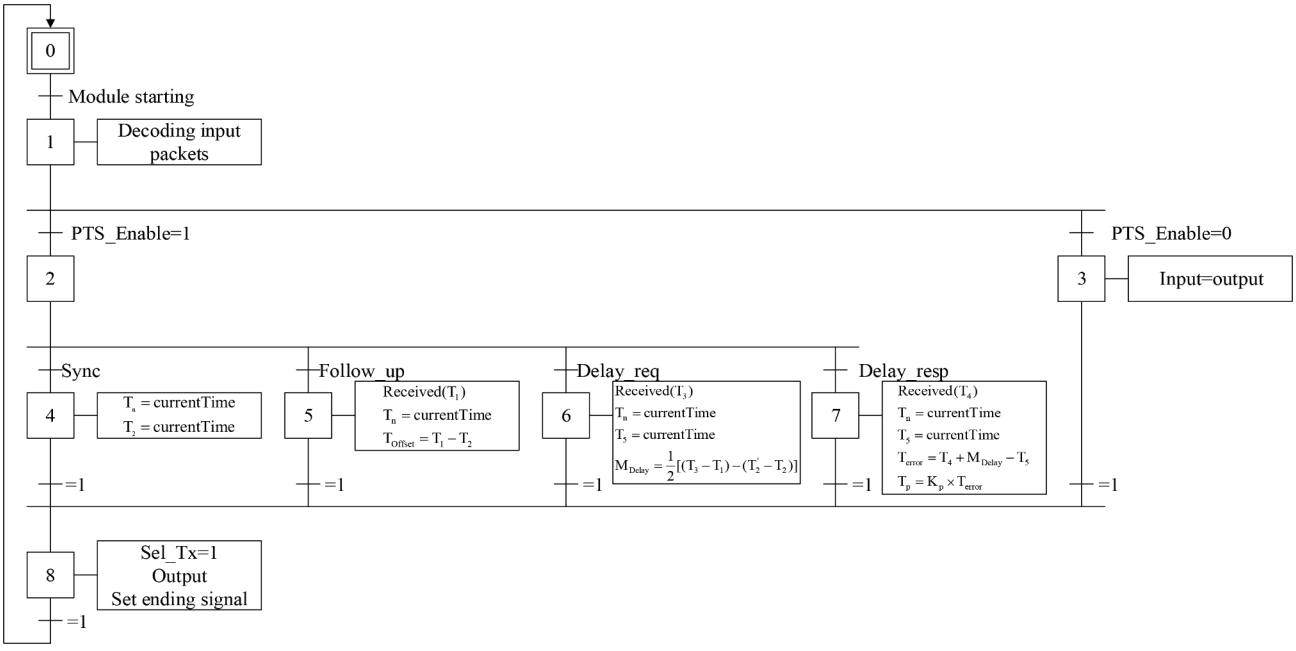


Fig. 15. GRAFCET modeling for slave receiving submodule. After the slave receiving submodule is initiated, it decodes the inputted packet and, according to the state information of the PTS packet, it can determine the synchronization steps that are being performed; then, it performs the corresponding processing in different state steps. In addition, the core time correction model of this PTS algorithm is embedded in this module mainly to correct the time offset, transmission time delay, and error offset as well as to adjust the timer value three times.

in a specific time variable for algorithm calculation.

The main tasks of the PTS algorithm of the PTS function module include sending/receiving selectors, updating statuses (Sync, Follow_Up, Delay_Req, and Delay_Resp), and fetching and recording timestamps (Fig. 11). Because our proposed PTS algorithm needs the specific timestamp of the master and slave to calculate the value of each

discrete-time model of the algorithm, the sending submodule and receiving submodule are core modules of the PTS algorithm, and their functions and natures are more complex and are responsible for different timestamps and distributed computing, this section describes the sending and the receiving submodules of the master–slave module.

The transmission module responsible for PTS includes the sending (transmitting) and the receiving submodules of the master module (Fig. 12 and Fig. 13) as well as the sending (transmitting) and the receiving submodules of the slave module (Fig. 14 and Fig. 15).

In addition, Fig. 15 reveals the timestamp computation models of the PTS algorithm and the correction timing for the timer value. In state 5, receive the timestamp T_1 from the master to obtain T_{offset} and conduct offset correction for the timer. In state 6, calculate M_{delay} and conduct transmission delay correction for the timer. In state 7, calculate T_{error} and multiply it with K_p to obtain the proportional error T_p and provide it to the timer for error adjustment.

4.5. High-level synthesis for PTS function module

The controllers of the PTS function module are easily obtained from the proposed template model by GRAFCET-to-VHDL synthesis. A reconfigurable FPGA was used to implement the PTS function module. However, the main purpose of using the PTS function module was to achieve high-precision time synchronization.

Fig. 16 presents the entire VLSI hardware circuit of the PTS function module according to our design methodology. Moreover, the major function modules responsible for the time synchronization algorithm have been embedded in the VLSI hardware circuit to integrate the master/slave mode selectors, sending/receiving selectors, sending/receiving submodules, timer, and PTS core controller. The PTS function module hardware circuit operates as follows.

The master/slave mode selectors select the master or slave mode based on the external input signals. After the mode is selected, the sending/receiving selectors decide whether they should execute send or receive tasks based on the judgment of the PTS core controller. As a local clock, the timer submodule continuously accumulates time for synchronizing the master and slave. The master and slave adjust the timer value according to the time synchronization results. In addition, by making the timer value an external output, it can be used by the entire system.

5. Implementation

The system architecture of the PTS-based Modbus-SAN system comprised two major modules, as introduced in Section 4.2. The PTS-based Modbus-SAN system architecture was developed and implemented for PTS of an IIoT-based smart manufacturing application. The hardware architecture of the PTS-based Modbus-SAN system is illustrated in Fig. 17.

As illustrated in Fig. 17, the architecture of the PTS-based Modbus-SAN system constructed in this study included a reconfigurable FPGA controller with a high-speed parallel architecture and an MC-SAN system including sensors, actuators, and servo motors for manipulators. Both the Modbus master and the Modbus slave were implemented and tested on a real FPGA platform. The PTS function hardware module is the core of the PTS-based Modbus-SAN system for PTS, and it controls the interconnection and interaction with major function modules of the PTS algorithm as well as intercommunication and interoperation with the perception-layer Modbus-SAN system. Furthermore, it provides a time synchronization data report and manipulates interfaces. In this study, the architecture of the FPGA function hardware module was implemented using the resources mentioned in Table 2. The Modbus-SAN system was connected to multiple sensors, actuators, and servo motors for manipulators in the perception layer.

We first measured the time error generated by the hardware of the FPGA clock generator. The time error of the hardware itself should also be verified. Next, we calculated the proportional gain coefficient K_p of the proportional control algorithm and provided the process and value. After the addition of the proportional error correction of the proportional gain coefficient K_p, the PTS result was obtained. Finally, we implemented a precision time control system for multiaxis servo motion for robotic manipulator applications.

5.1. Time error of clock generator

This section emphasizes the time difference between the clock generators of the local oscillator, as introduced in Section 3.3. The master and the slave each use two FPGA hardware modules to measure the time difference per second generated by the clock generator of the two hardware modules at 50 MHz. As presented in Fig. 18, through the synchronizing trigger signals, it is ensured that the timers in the two

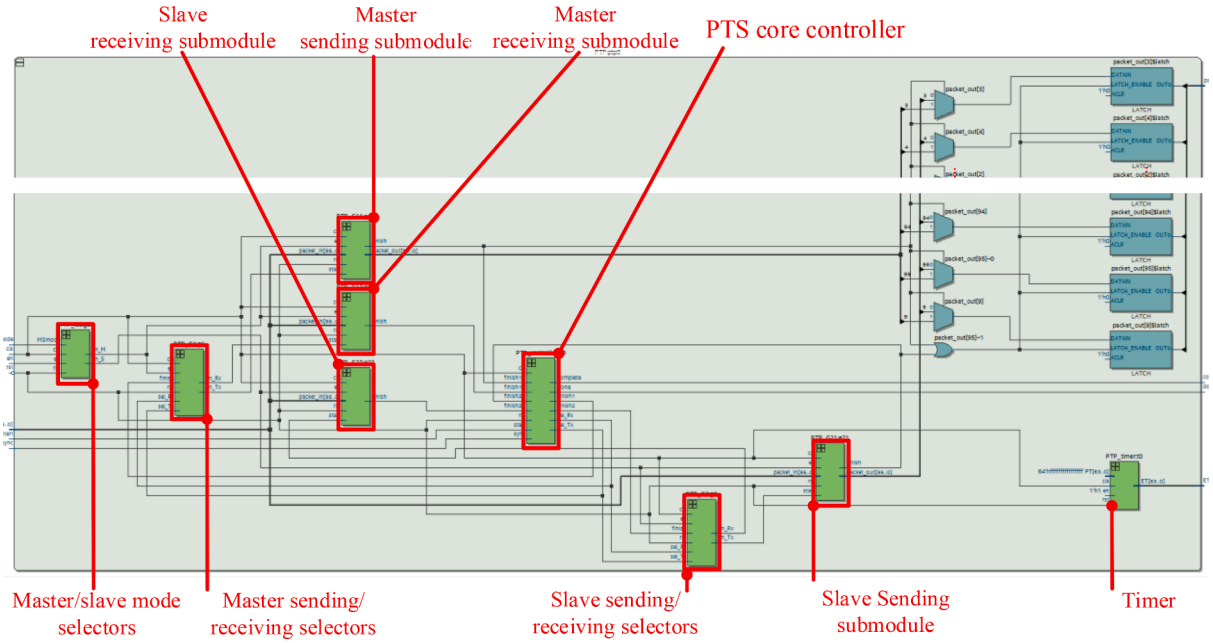


Fig. 16. VLSI hardware circuit for PTS function module.

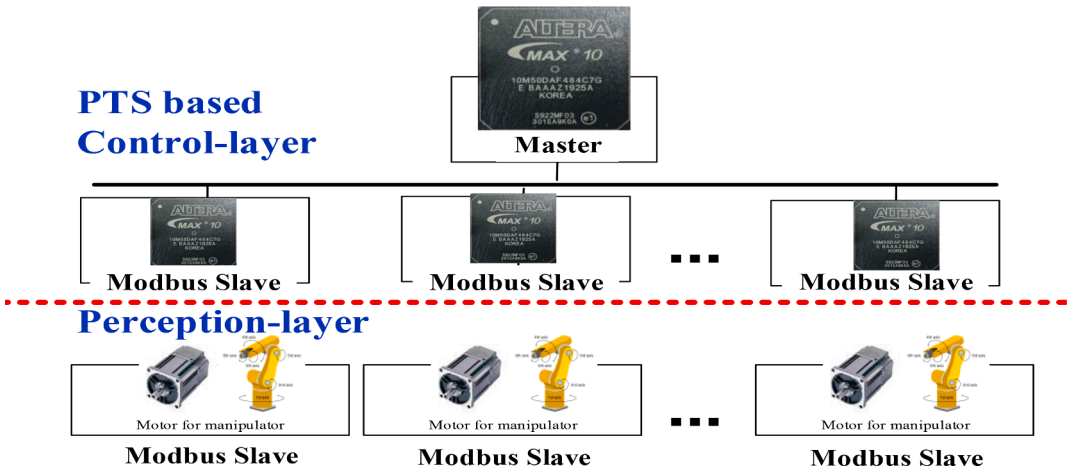


Fig. 17. Prototype of PTS-based Modbus-SAN system.

Table 2
Resource Utilization of Implemented PTS Function Hardware Module (Numbers in Brackets Indicate Device Utilization).

	PTS function hardware module
Device	10M50DAF484C7G
System Clock(Max)	50MHz
Embedded Memory	1638 K Bits
Total logic element	14,134/50,000 (28%)
Total register	7701
Total pins	264/360 (73%)
Total PLLs	1/4 (25%)

master–slave hardware modules are initiated simultaneously. After the timer outputs the digital output (DO) signals, the logic analyzer is used to measure the time difference of the timer of the two hardware modules' clock generator after 1 s, and the time difference is 0.18 μs. The measurement result reveals a significant difference in the rates generated by clocks in different hardware modules with the same clock setting.

5.2. Proportional gain coefficient

In the experiment, by using the complete signal of the master, we can determine whether synchronization is completed. When the complete signal rises from the falling edge to the rising edge, it can be determined that the time synchronization has ended. Afterward, the master device outputs one signal to the slave. When time synchronization is completed, the master–slave immediately record their clock values to calculate T_{diff}, and then, T_{error} is used to calculate the proportional gain coefficient K_p The aforementioned time synchronization process is repeated 1000 times and the measurement results are recorded Table 2. enumerates 10 sets of T_m, T_s, T_{error}, and T_{diff}. The master–slave timer counts at a frequency of 50 MHz, and each unit is converted to a real time of 20 ns ($1/50{,}000{,}000 = 0.00000002\ s = 20$ ns).

The T_{error} and T_{diff} in Table 3 are substituted into (8) to obtain K_p, which is calculated to be 0.3797. In addition, by observing the T_{diff} of the timers of the master–slave in Table 3 after time synchronization, we can determine the time error before proportional control, which found to be approximately 20,858 ns.

5.3. Proportional error correction

After calculating the coefficient K_p, of the proportional error model is used for the proportional error correction of time synchronization, and then, the error results are remeasured.

Table 4 presents the records of measurement results after 10 professional error corrections. The T_{diff} is 2.1 units (±42 ns), the maximum

Table 3
Value of Time Difference and Error.

No.	T_m	T_s	T_{error}	T_{diff}
1	1,460,831565	1,460,832607	2746	1042
2	4,669,720848	4,669,721888	2752	1040
3	7,189,908202	7,189,909242	2752	1040
4	9,281,168807	9,281,169854	2743	1047
5	11,539,975448	11,539,976491	2748	1043
6	13,589,047768	13,589,048810	2747	1042
7	16,397,155073	16,397,156117	2744	1044
8	19,586,783777	19,586,784818	2747	1041
9	22,343,585323	22,343,586369	2744	1046
10	25,086,334182	25,086,335226	2747	1044



Fig. 18. Time error of clock generator.

Table 4
Value of Time Difference and Error Offset after PTS.

No.	T_m	T_s	T_{diff}
1	6,541,730418	6,541,730421	3
2	12,917,408384	12,917,408386	2
3	16,249,025634	16,249,025635	1
4	18,699,062624	18,699,062626	2
5	20,854,033479	20,854,033484	5
6	29,337,668296	29,337,668295	−1
7	32,289,441396	32,289,441394	−2
8	36,423,441268	36,423,441266	2
9	40,848,052288	40,848,052287	−1
10	43,394,022824	43,394,022822	−2

is 5 units (100 ns), and the minimum is 1 unit (20 ns).

5.4. PTS control of multiaxis servo motion for robotic manipulator

Robotic manipulators are now widely used in IIoT smart manufacturing systems. To perform industrially automated tasks accurately and reliably, manipulator positions, actuators, and velocities are controlled digitally. There is a very close relationship between the PTS performance and manipulator kinematics. In addition, robotic manipulators are multiaxis motion control systems with highly coupled nonlinear dynamics. Therefore, it is important to develop a PTS control solution for them.

This study presents a prototype of a PTS-based Modbus-SAN system for the time synchronization of manipulators. The performance of the PTS hardware module designed for verification satisfies the requirements of industrial applications. The constructed prototype included a master, a slave, sensors, actuators, and servo motors for manipulators. The prototype also integrates the Modbus-SAN system and transmits time synchronization packets via the universal asynchronous receiver/transmitter (UART). The microcontroller (MCU) is responsible for sensing the trigger signals and servo motors for the kinematic control of robotic manipulators.

In the architecture of the PTS-based Modbus-SAN system with the PTS function, the master and slave each connect to one servo motor and output one cycle of a pulse width modulation (PWM) signal per second to control each motion or degree and velocities of the servo motors. However, for the two servo motors controlled by different devices, the manipulator kinematics requires manual time synchronization, and the synchronization error must be within 1 μs for industrial automation. The master and slave devices sense the trigger signal of MCU and initialize the timer value, and according to the local timer value, set to output one cycle of a PWM signal per second within 10 s. The status of the Modbus-SAN system is measured with and without PTS and the results for different time values are output Table 5. lists the time difference value from 1 to 10 s without time synchronization. The measurement result indicates a time difference exceeding 1 μs in 4 s, and this error increases with time.

Table 6 indicates that with PTS, the maximum time difference per second of the Modbus-SAN system is only 140 ns, and the mean difference is 68 ns.

Table 5
Time Difference Value Without PTS.

Second	Time difference (ns)
1	319
2	660
3	979
4	1300
5	1600
6	1920
7	2260
8	2580
9	2900
10	3220

Table 6
Time Difference Value with PTS.

Second	Time difference (ns)
1	40
2	60
3	60
4	40
5	40
6	100
7	120
8	140
9	40
10	40

After the integration of the changes in time difference in the PWM output from Tables 5 and 6, the results are presented in Fig. 19. When the system does not have time synchronization, after the initial counter value starts, the time difference increases almost linearly. When PTS is performed periodically in the system, the measured time difference remains small; the mean time difference is within 100 ns, and the convergence is 140 ns.

5.5. Performance evaluation

We validated the PTS hardware performance improvement through comparisons with state-of-the-art synchronization technologies and commercial PTPv2 solutions; the set of parameters included the synchronization protocol and time accuracy for operating one set. In [12], their findings reveal synchronization precision, scalability, cost, and hardware requirements. The results are listed in Table 7. In Table 7, the most popular time synchronization technologies: GPS, WR-PTP, PTP, and NTP are compared in terms of their architecture and other relevant properties. Commonly main features have been discussed below.

Despite of the high accuracy, GPS has numerous disadvantages. For example, GPS is very expensive. GPS entails that all the participating nodes are furnished with a GPS receiver. Further, inside buildings, GPS signals reception is difficult which makes GPS an unattractive approach for further consideration.

WR-PTP extends the PTP to provide synchronization with sub-nanosecond accuracy. The original WR-PTP implementation guarantees a synchronization accuracy below 1 ns for a small number of cascaded nodes [13]. identified the performance degradation when a considerable number of devices are deployed in WR-PTP. On the other hand, [12] was noted that in IIoT applications, WR-PTP may not be a cost-effective solution. The main disadvantage is that specialized White Rabbit hardware would be prohibitively expensive. These include both the intermediate nodes (switches, routers) and the end nodes (workstations, PCs).

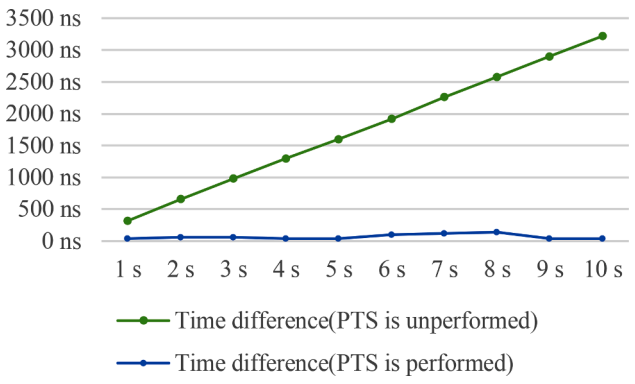


Fig. 19. Comparison of time difference value of PTS-based Modbus-SAN system for multiaxis motion control.

Table 7
Comparisons with State-of-the-Art technologies and commercial PTP solutions.

Comparison of the state-of-the-art synchronization technologies					
	GPS[24]	WR-PTP[26]	PTP[1]	NTP[27]	
Protocol	Global Positioning System	IEEE 1588v2	IEEE 1588v2	Network Time Protocol	
Level of accuracy	nanosecond level	Sub-nanosecond level	Sub-microsecond level	millisecond level	
Reference value	100ns	10ps	1μs	1ms	
Scalability[669]	Poor	Good	Good	Good	
Cost[669]	High	Medium	Medium	Low	
Hierarchy Model[669]	Client/Server	Master/Slave	Master/Slave	Distributed Client/ Server	
Hardware Requirements [669]	GPS receivers	Optional-PTP enabled devices	specialized WR hardware	No	
Comparison of commercial PTP solutions					
	Proposed Solution	Altera 1588 System Solution[32]	SEVEN WR-LEN White Rabbit Switch [20]	Hirschmann TB1001 [33]	SEGGER PTP Solution [34]
Protocol	IEEE 1588v2+Modbus	IEEE 1588v2	IEEE 1588v2	IEEE 1588v2	IEEE 1588v2
Level of accuracy	nanosecond level	nanosecond level	Sub-nanosecond level	nanosecond level	millisecond level
Reference value	42ns	6.99 ns only timestamp accuracy	Deterministic time distribution with accuracy 1 ns.	typically 30–50ns	100ns
Operating system	no	Linux (kernel v3.16)	Linux (Kernel v2.6.39)	No	embOS/IP
Microprocessor or CPU	no	Altera Arria V SoC	ARM Atmel AT91 SAM9G45	No	Measured on a Cortex-M4

PTPv2 allows attaining synchronization in the range of nanoseconds provided that all the network components (like switches and routers) are PTPv2 enabled. However, such performance is not achieved in typical PTPv2 implementations. As PTPv2 assumes the network delay to be constant and symmetrical, delay variations can reduce its accuracy. The PTPv2 is typically used in conjunction with hardware support. Such variations might occur, if non-PTP routers and switches are used in the network. The accuracy of the clock synchronization is affected by multiple factors, including (1) Limited precision and resolution of PTP timestamps and (2) Unknown physical link asymmetry. WR-PTP achieves sub-nanosecond accuracy by basing its time distribution on the PTP and addressing the aforementioned issues limiting PTP's performance.

NTP has an accuracy of approximately 1 ms on the Internet. On the network level, there is a hierarchical structure of NTP. The advantage is that large networks can be synchronized more efficiently with several devices of up-layer and the load is reduced [15]. However, each layer adds additional delay which continues to aggregate till the last layer. The accuracy decreases (in this case) due to the propagation of errors along the hierarchy. Consequently, NTP is not well suited for current delay-sensitive applications due to its limited accuracy, which decreases even further when delay variations occur.

In summary, Ref [12, 15]. emphasized that typical approaches such as NTP and GPS are unsuitable for IIoTs. Ref [12, 15, 16]. have also compared the well-known time synchronization techniques. The results concluded that the PTP is the most appropriate solution to robust clock synchronization.

Nowadays there are several types of commercial PTP solutions in the market used to carry out time synchronization problems. However, while these equipment present good characteristics and performances they are usually very expensive. We validated the improvement in enhanced PTS performance by comparing the performance with those of other industry-related products [20], and [32–34]. In addition, we discussed several performance indices, such as synchronization accuracy, software requirements, operating system kernel, and hardware resources of the enhanced PTS protocol, so as to verify whether the proposed solution can meet the technology innovation indices described in Table 6.

As shown in Table 6, SEVEN WR-LEN White Rabbit Switch [20] demonstrated the best PTP performance with accuracy of 1 ns. However, it usually requires Ethernet, SyncE, an operating system (OS), a microprocessor, and a GPS clock receiver to implement, thus requiring more additional software and hardware resources compared to other PTP solutions. Apart from [20], the performance indices of the proposed solution are superior to those of other competing products in the market. This is because our study adopted a high-speed FPGA controller and designed the PTS function hardware, so the synchronization accuracy can be achieved to 42 ns. In addition, this study adopted PTS hardware chip, which is without the extra operating system to prevent the piling up and switching of software stacks, as well as significantly reducing the power consumption by adopting distributed computing. Furthermore, in terms of performance index for high accuracy enhancements, a complete communication process involves polling by using the master to slave and the calculation of timestamps returned from the DI/DO Slave.

6. Conclusion and future work

For realizing high-precision time synchronization control, this study proposes an enhanced PTS protocol for improving the reliability of real-time multiaxis motion synchronized control of an IIoT system and designs a PTS hardware chip according to the specifications of the proposed architecture. The proposed solution is based on enhanced PTPv2 that involves the use of a PTS algorithm combined with the transmission time delay model and the proportional correction model. The PTS function hardware module was designed using a design methodology. By using GRAFCET modeling and VHDL synthesis, efficient controller hardware for the PTS protocol can be designed rapidly. The hardware module was generated automatically to integrate all the core time correction models of the time synchronization algorithm into a complex embedded system. In addition, the proportional controller module was embedded into the PTS algorithm to adjust the system time error. This hardware module is capable of stand-alone operation and does not require an additional CPU. The measurement results showed that the synchronization accuracy of the proposed solution can be achieved to 42 ns.

Because of the limitations of the FPGA hardware implementation, our PTS sacrifices portability to achieve high performance. In the future, a processor could be added to this PTS architecture to realize PTS functionalities for providing greater design flexibility for application developers. Additionally, further work on high accuracy enhancements can address an in-depth case study for asymmetries calibration. We will try to enhance the synchronization accuracy by using machine learning.

Declaration of Competing Interest

The authors declare that they have no known competing financial

interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

[1] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems IEEE Std. 1588-2008, 2008.
[2] T. Qiu, Y. Zhang, et al., A robust time synchronization scheme for industrial internet of things, IEEE Trans. Ind. Informat. PP (99) (2017), 1–1.
[3] S. Liu, X. Hao, Z. Zhang, 'Pedestrian retrieval via part-based gradation regularization in sensor networks, IEEE Access 6 (2018) 38171–38178.
[4] Al Anbagi, M. Erol-Kantarci, H.T. Mouftah, 'Delay critical smart grid applications and adaptive QoS provisioning, IEEE Access 3 (1) (Aug. 2015) 1367–1378.
[5] Ioan Ungurean, Industrial internet of things with data distribution service for real time systems, Int. Res. J. Eng. Technol. (IRJET) 02 (05) (Aug 2015).
[6] Ju Ren, Hui Guo, Chugui Xu, Yaoxue Zhang, Serving at the edge: a scalable IoT architecture based on transparent computing, IEEE Netw. (2017), https://doi.org/10.1109/MNET.2017.1700030.
[7] Yaoxue Zhang, Ju Ren, Jiagang Liu, Chugui Xu, Hui Guo, Yaping Liu, A survey on emerging computing paradigms for big data, Chin. J. Electronics 26 (1) (2017) 1–12.
[8] K. Xu, Y. Qu, K. Yang, 'A tutorial on the Internet of Things: froma heterogeneous network integration perspective, IEEE Netw. 30 (2) (Mar. 2016) 102–108.
[9] Advantech Co., Ltd.: PCIE-1203 L http://advdownload.advantech.com/productfile/PIS/PCIE-1203 L/Product%20-%20Datasheet/PCIE-1203L20180910103050.pdf.
[10] IEEE Std 802.1AS™-2011, March. 30, 2011, DOI: 10.1109/IEEESTD.2011.5741898.
[11] J.R. Moyne, D.M. Tilbury, The emergence of industrial control networks for manufacturing control, diagnostics, and safety data, Proc. IEEE 95 (1) (Jan. 2007) 29–47.
[12] Zeba Idrees, et al., IEEE 1588 for clock synchronization in industrial IoT and related applications: a review on contributing technologies, protocols and enhancement methodologies, IEEE Access 8 (2020) 155660–155678.
[13] José Luis Gutiérrez-Rivas, et al., Enhancing white rabbit synchronization stability and scalability using P2P transparent and hybrid clocks, IEEE Trans. Ind. Inf. 17 (11) (2021) 7316–7324.
[14] Terry Jones, et al., Recent advances in precision clock synchronization protocols for power grid control systems, Energies 14 (17) (2021) 5303.
[15] Henning Puttnies, et al., Estimators for time synchronization—survey, analysis, and outlook, IoT 1 (2) (2020) 398–435.
[16] Francisco Girela-López, et al., IEEE 1588 high accuracy default profile: applications and challenges, IEEE Access 8 (2020) 45211–45220.
[17] J. Cecilio, P. Furtado, Architecture for uniform (re) configuration and processing over embedded sensor and actuator networks, IEEE Trans. Ind. Inf. 10 (1) (Feb. 2014) 53–60.
[19] Xiaohan Wei, Kaixing Cheng, Time Synchronization of distributed systems based on IEEE1588, World Scientific Res. J. 7 (3) (2021) 30–36.
[20] "Seven Solutions White Rabbit Switch," accessed:2021 -10-26. https://sevensols.com/white-rabbit-technology/.
[21] Xiaohan Wei, et al., An enhanced IEEE1588 clock synchronization for link delays based on a system-on-chip platform, Int. J. Electron. Telecommun. 67 (2) (2021) 289–294.
[22] MengDi Zhang, HuaChuang Wang, Bo Liu, A high precision TDC design based on FPGA+ ARM, J. Phys. Conf. Ser. 1486 (7) (2020). IOP Publishing.
[23] Jinlong Dong, et al., Design and Implementation of AS6802 Clock Synchronization System in TTE thernet, in: 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), IEEE, 2021.
[24] Parkinson; Spilker, The Global Positioning System, American Institute of Aeronautics and Astronautics, 1996. ISBN 978-1-56347-106-3.
[25] Saeid Sedighi, et al., Localization performance of 1-bit passive radars in NB-IoT applications using multivariate polynomial optimization, IEEE Trans. Signal Process. 69 (2021) 525–2540.
[26] Asja Derviškadić, et al., The white rabbit time synchronization protocol for synchrophasor networks, IEEE Trans. Smart Grid 11 (1) (2019) 726–738.
[27] David L. Mills, Computer Network Time synchronization: the Network Time Protocol On Earth and in Space, CRC press, 2017.
[28] Mills, David. Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI. RFC 2030, October 1996..
[29] Istvan Godor, et al., A look inside 5 G standards to support time synchronization for smart manufacturing, IEEE Commun. Standards Magazine 4 (3) (2020) 14–21.
[30] Timing characteristics of a synchronous Ethernet equipment slave clock (EEC), ITU-T Std (2007). G.8262.
[31] Rahul N. Gore, et al., Towards robust synchronization in IoT networks, in: 2019 11th International Conference on Communication Systems & Networks (COMSNETS), IEEE, 2019.
[32] Altera 1588 System Solution: https://www.intel.com/content/www/us/en/programmable/documentation/kly1423707073680.html#kly1425868778283.
[33] Hirschmann TB1001HE https://hirschmann.com/en/Hirschmann_Produkte/Industrial_Ethernet/Technologies/Precision_Time_Protocol/index.phtml.
[34] SEGGER PTP Solution: https://www.segger.com/news/precise-clock-synchronization-with-new-ptp-module-ieee1588/.



Ching-Han Chen received the Ph.D. degree in 1995 from the Franche-Comte University, France. He was an Associate Professor in the department of electrical engineering, I-Shou University, Taiwan, R.O.C. from 1995. He is now a Professor in the department of computer science and information engineering. His-research interests include embedded system design, multimedia signal processing, robotics.



Ming-Yi Lin received Master degree from Yuan-Ze University and received Ph.D. degree of computer science from National Central University in 2011 and 2016, respectively. He is now an Assistant Professor in the department of computer science and information engineering at Taipei City University of Science and Technology, Taiwan. His-research interests include embedded systems, smart sensor networks, and Internet of Things.



Yu-Chien Shih received Master degree of computer science from National Central University in 2017. His-research interests include embedded system, Fieldbus protocol, and industrial sensor network.



Chun-Cheng Chen received Ph.D. degree of National Yunlin University of Science and Technology in 2018. He is now an Assistant Professor in the department of computer science and information engineering at Taipei City University of Science and Technology, Taiwan. His-research interests include high-level thinking, creative thinking, critical thinking, problem-solving ability, invention patent