electronics

MDPI

Article

Development Board Implementation and Chip Design of IEEE 1588 Clock Synchronization System Applied to Computer Networking

Yan-Kai Lan, Yee-Shao Chen, Ting-Chao Hou, Bo-Lin Wu and Yuan-Sun Chu *

Electrical Engineering Department, National Chung Cheng University, Chiayi 62102, Taiwan; ggyy870224@gmail.com (Y.-K.L.); s8747226@gmail.com (Y.-S.C.); ieetch@ccu.edu.tw (T.-C.H.); berlin880613@gmail.com (B.-L.W.)

* Correspondence: chu@ee.ccu.edu.tw

Abstract: With the vigorous development of industrial automation and the Internet of things, the transmission of data is more dependent on immediacy, so network devices have higher and higher requirements for time synchronization accuracy. The clock source of common network devices is provided by the transistor oscillator in the server, but the oscillator will change with factors such as aging and temperature, and it cannot be guaranteed that the oscillator in the server will work at the same frequency. Time synchronization can be achieved by technologies such as IRIG-B, NTP, or IEEE 1588 (PTP), but the hardware cost of building IRIG-B is high, and NTP has the lowest cost, but it can only provide time accuracy from milliseconds to microseconds. PTP can provide sub-microsecond or even nanosecond time precision. It is a system of time synchronization mechanisms through Ethernet transmission. In this article, we first propose a time synchronization system using the development board and PDP protocol. On the Xilinx Zynq-7000 SOC platform of Petalinux, we implement the hardware solution of Linux PTP. The hardware time stamp is 20 ns. To improve the accuracy, the congenital frequency error between the oscillators must be considered. Therefore, a PTP auxiliary time stamp with dynamic frequency compensation is proposed and designed into a chip. Experimental results show that at 45 nm (TN40G) it can operate at 370 MHz and achieve 2.7 ns resolution, which can be applied to more demands.

Keywords: IEEE 1588; precision time protocol; time synchronization; IoT

Citation: Lan, Y.-K.; Chen, Y.-S.; Hou, T.-C.; Wu, B.-L.; Chu, Y.-S. Development Board Implementation and Chip Design of IEEE 1588 Clock Synchronization System Applied to Computer Networking. *Electronics* **2023**, *12*, 2166. https://doi.org/10.3390/electronics12102166

Academic Editor: Esteban Tlelo-Cuautle

Received: 28 March 2023
Revised: 4 May 2023
Accepted: 8 May 2023
Published: 9 May 2023

1. Introduction

In order to implement Industry 4.0 to achieve applications such as smart factories, machine control, and data acquisition, it is necessary to use technologies such as the Internet of things and sensors. Among them, the Internet of things connects computers and other industrial equipment through the network, which will make the network structure more and more large, and the transmission of data depends more on real-time. Common network time synchronization methods include GNSS, NTP, and PTP, etc. The GNSS clock signal is transmitted through radio waves and special equipment must be installed on the application device, so the cost is relatively high. However, NTP has the lowest cost. In the industrial Internet of things (IIoT), the network time protocol (NTP) can be used for time synchronization [1,2]. NTP is low-cost and easy to use, but it can only provide millisecond-level time precision, which cannot meet the precision required by modern measuring instruments and industrial control. Therefore, the IEEE Society proposed the Precision Time Synchronization Protocol (PTP) in 2002, also known as IEEE 1588. The second edition of IEEE 1588–2008 [3] was revised in 2008, and the latest version is the third edition of IEEE 1588–2019 [4], released in 2019. PTP is a synchronization protocol for periodic packet exchange using Ethernet, using primary/secondary architecture; its

advantage is that it can be achieved by using the existing Ethernet line, which makes the deployment of the time synchronization network easier.

Regarding the research on IEEE 1588, B. Zhao [5] implemented a design scheme of FPGA, DM9000, and DP83640 based on the detailed analysis of IEEE 1588 protocol and the best primary clock algorithm (BMCA), through the integration of software and hardware. Verified by experiment, the system can realize sub-microsecond level time precision in synchronization with the primary clock and secondary clock. Their design improved the time synchronization precision and simplified the hardware circuit design. N. Moreira [6] discussed the implementation of the PTP function on the Xilinx Zynq-7000 SOC development board. The experimental environment is to use one development board as PTP primary and the other development board as PTP secondary and use an oscilloscope and a computer to view the PPS (pulse-per-second) signals generated by the device and the message of the PTP console at the same time. PPS is a signal that outputs a high logic level once a second and the pulse width is generally 100 ms. It can be applied to smart grids and substation automation systems and can meet reliability requirements. S. S. W. Lee [7] proposed two architectures to realize the OpenFlow switch with the PTP function. The first architecture is to embed the designed PTP module in a NetFPGA-based OpenFlow switch. The second architecture is to use the designed PTP module as an add-on module to work with a PTP unaware switch. The results show that the PTP module can implement a software-defined time synchronization network, and can meet the precision standards of microsecond-level small wireless base stations. Z. Idrees [8] compared common time synchronization technologies, such as GPS, NTP, and PTP, and concluded that PTP is the most stable clock synchronization technology, and also proposed many methods to improve the accuracy of PTP, including boundary clock optimizations, modifications in IEEE 1588 messaging, improved internal clock stability, etc. Due to the rapid development of artificial intelligence algorithms and the establishment of smart factories, especially distributed optimization (DO) on multi-agent networks, has been extensively studied [9]. Time synchronization is an important key. All devices on the network need to have a common time reference so that all devices work together and take fast actions at precise times.

In this paper, firstly, we implemented the Linux PTP hardware solution on the Xilinx Zynq-7000 SOC platform of Petalinux. The hardware timestamp is 20 ns. To improve accuracy, we propose an IEEE 1588 (PTP) chip design and implementation with dynamic frequency compensation. Experimental results show that the 45 nm (TN40G) node can work at 370 MHz with a resolution of 2.7 ns. The final calculated time offset is 10.8 ns. The core of the IC we designed can be integrated into related communication chips, it can be applied to a related communication network card, and this communication network card has the function of fast time synchronization.

The rest of this article is structured as follows. In Section 2, we will introduce PTP. In Section 3, we illustrate the implementation of system timestamps, including software timestamps and hardware timestamps. In Section 4, we illustrate the IC design and implementation of PTP hardware-assisted time stamping. Experimental results and analysis will be placed in Section 5. Section 6 provides conclusions.

2. Precision Time Protocol Introduction

The PTP protocol sends periodic packets to each node in the network through the Ethernet to confirm the system structure of the entire network and transmit time information to achieve synchronization between different clocks. PTP defines a relay request–response mechanism to measure the offset of Primary and Secondary. It calculates the sending and receiving time by sending Sync, Delay_Req, Delay_Resp, and Follow_up packets, as shown in Figure 1. Using the timestamp generated by the above-mentioned time synchronization mechanism to calculate the round-trip time of the packet, the offset time and path delay time between the two clocks can also be calculated. The Secondary will correct the

local time according to the calculated offset time to achieve time synchronization with the Primary.

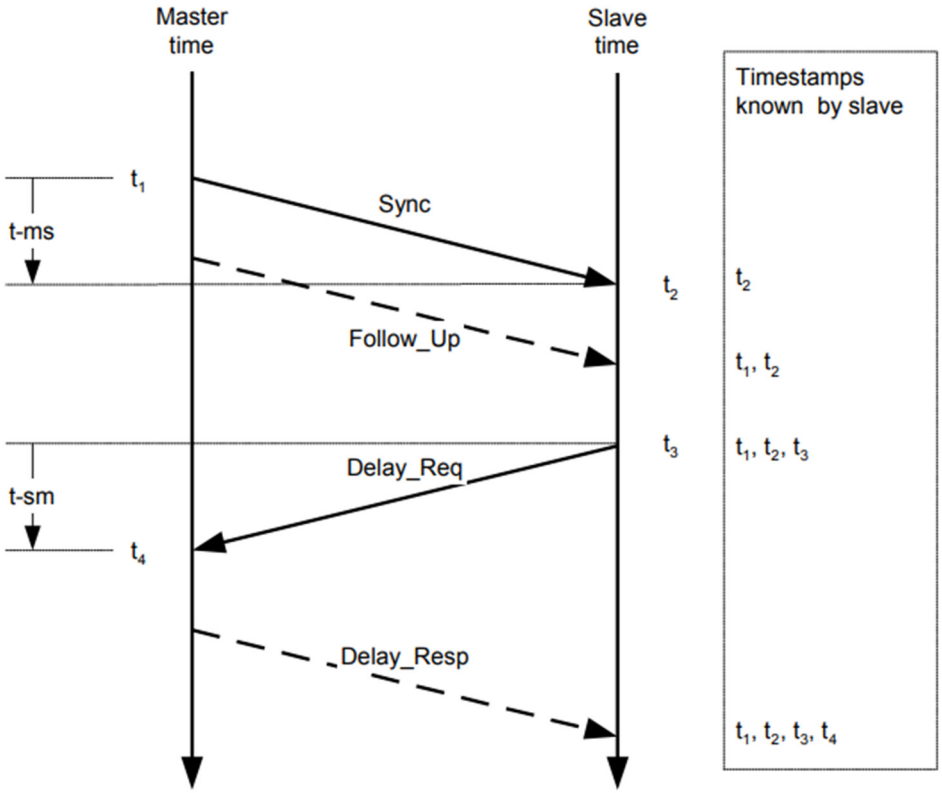

Figure 1. Delayed request–response mechanism synchronization method.

Figure 1 is a schematic diagram of the implementation of the delay request–response mechanism; assume that the time difference between Primary and Secondary is offset, the transmission path delay between Primary and Secondary is delay, and the delays are symmetrical to each other.

After the above packet exchange, Secondary has four timestamps from t_1~t_4, such as Equations (1) and (2)

$$t_1 + \text{delay} = t_2 - \text{offset} \tag{1}$$

$$t_3 - \text{offset} + \text{delay} = t_4 \tag{2}$$

After finishing, if Equation (3) is the time difference t_ms from the primary terminal to the secondary terminal and Equation (4) is the time difference t_sm from the secondary terminal to the primary terminal, the offset time of Equation (5) can also be calculated (time offset) and the average path delay time of Equation (6) can also be calculated (mean path delay). This basic synchronization message exchange is only applicable when t_ms and t_sm time are approximately the same, when the difference between t_ms and t_sm exceeds the set threshold, the delayAsymmetry value is required for error correction.

$$t_{\_ms} = t_2 - t_1 = \text{delay} + \text{offset} \tag{3}$$

$$t_{\_sm} = t_4 - t_3 = \text{delay} - \text{offset} \tag{4}$$

$$\text{Offset} = \frac{(\text{tsm} - \text{tsm})}{2} \tag{5}$$

$$\text{meanPathDelay} = \frac{(\text{tms} + \text{tsm})}{2} \tag{6}$$

3. Implementation of System Timestamps (Including Software and Hardware Timestamps)

Figure 2 shows the network transmission protocol implemented by our system, which is formulated according to the simplified TCP/IP protocol [10] of the OSI (Open System Interconnection) Model. The OSI model is to formulate seven-layer standards for software and hardware in the network and uses common standards to enable different systems to communicate with each other. The TCP/IP protocol is currently a widely used network transmission protocol for the merging of functional layers.

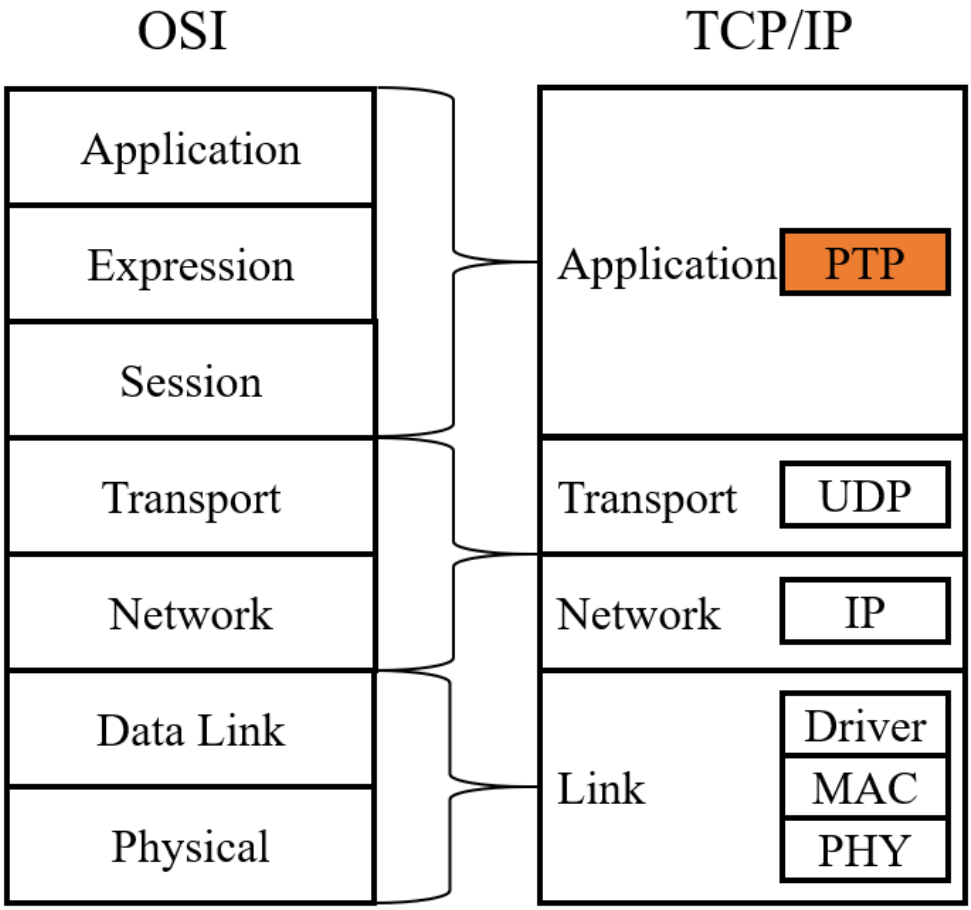

Figure 2. Network transmission protocol of PTP protocol.

Figure 3 is the implementation architecture of the PTP protocol. In the agreement, points A, B, and C are the positions where timestamps may be generated [11,12]. The synchronization accuracy is greatly affected by the position where the timestamp is generated. Point A is at the application layer mark timestamp, this is the simplest way because it can be realized by software, but the jitter of the protocol stack delay is uncertain, and the delay jitter will cause a difference between about a few milliseconds ms and tens of microseconds μs due to errors, this implementation method is not used by ordinary people. Point B is to mark the timestamp between the network layer (IP layer) and the media access control layer (MAC layer). This method avoids the uncertainty of protocol stack delay jitter, and can also be implemented by software, and the time accuracy can reach the microsecond level. Point C is to mark the timestamp at the GMII (gigabit media-independent interface) close to the physical layer (PHY). GMII defines the logical and electrical characteristics of the reconciliation sublayer (RS) and gigabit media-independent interface (GMII) between

CSMA/CD media access controllers and various PHYs. This method is based on the combination of software and hardware and requires the assistance of hardware auxiliary circuits. In the path that does not pass through the original packet to the application layer of PTP, due to the delay jitter of the physical layer is about the constant delay, which can provide sub-microsecond time precision. This section will introduce software time stamping and hardware time stamping, respectively.

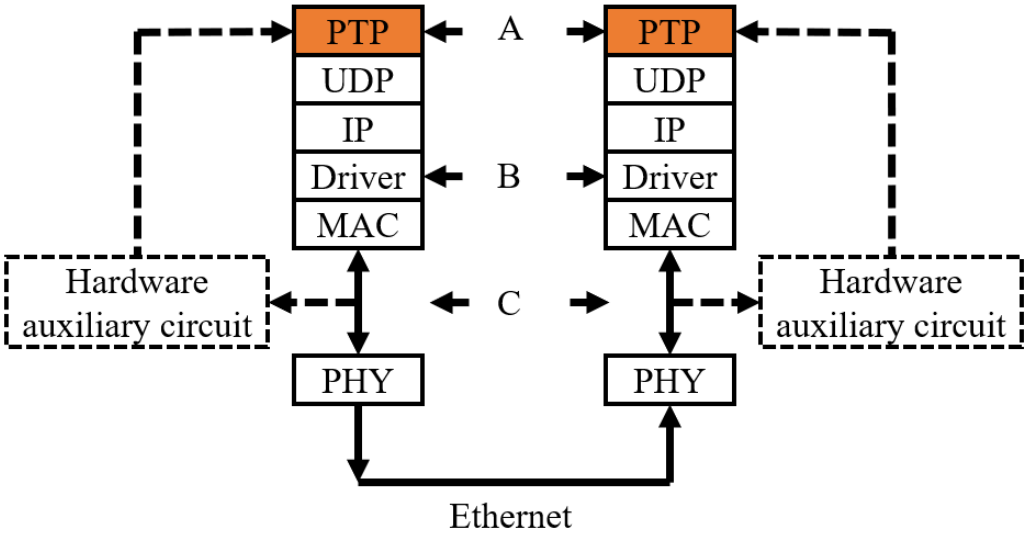

Figure 3. PTP protocol implementation architecture diagram.

In the lab environment, two PCs are set up. The specifications are shown in Tables 1 and 2. They are respectively used as the primary clock of the system and the secondary clock of the system. The network uses Ethernet to distribute fixed IP for connection and then uses the open source package software Linux PTP to realize. The software is implemented according to the IEEE 1588 protocol of Linux, provides a reliable implementation of the standard, and uses the application programming interface (API) provided by the Linux Kernel.

Table 1. Computer specifications (primary).

		Device Specifications and System Software Used		
CPU	**RAM**	**Network Card**	**OS**	**Software**
Intel core i5-9500	8 GB	Intel Corporation 82,579 LM Gigabit Network Connection	Ubuntu 16.04	C language

Table 2. Computer specifications (secondary).

		Device Specifications and System Software Used		
CPU	**RAM**	**Network Card**	**OS**	**Software**
Intel core E5-1620	8 GB	Intel Corporation Ethernet Connection (7) I219-LM	Ubuntu 16.04	C language

The actual execution flow of the system software is shown in Figures 4 and 5.

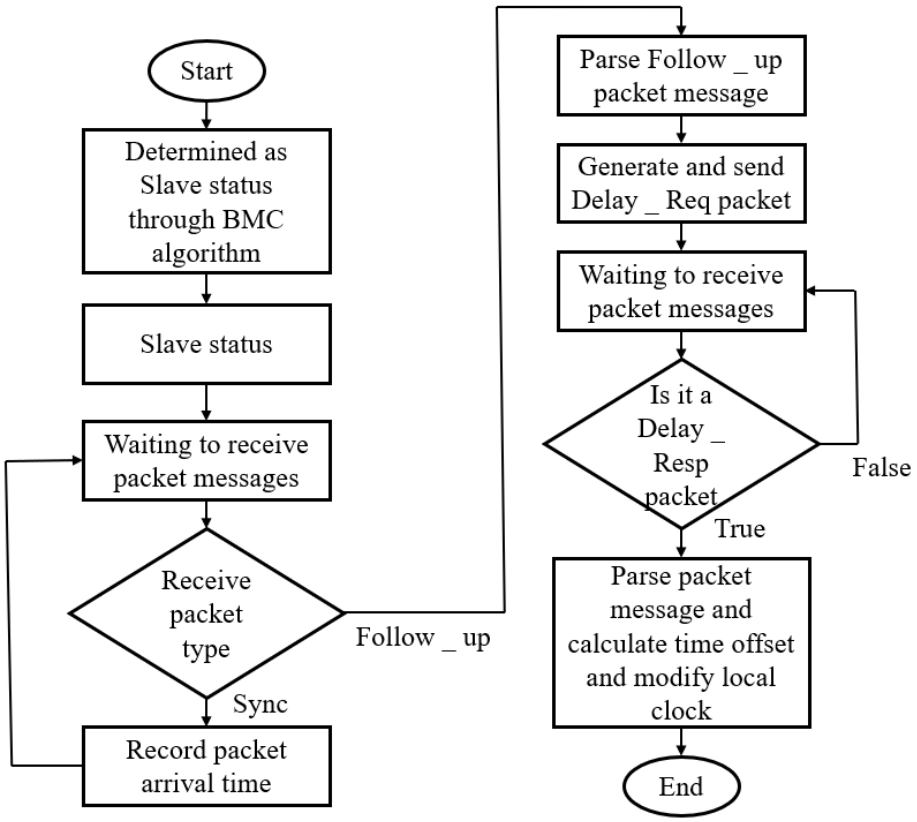

Figure 4. Secondary workflow chart.

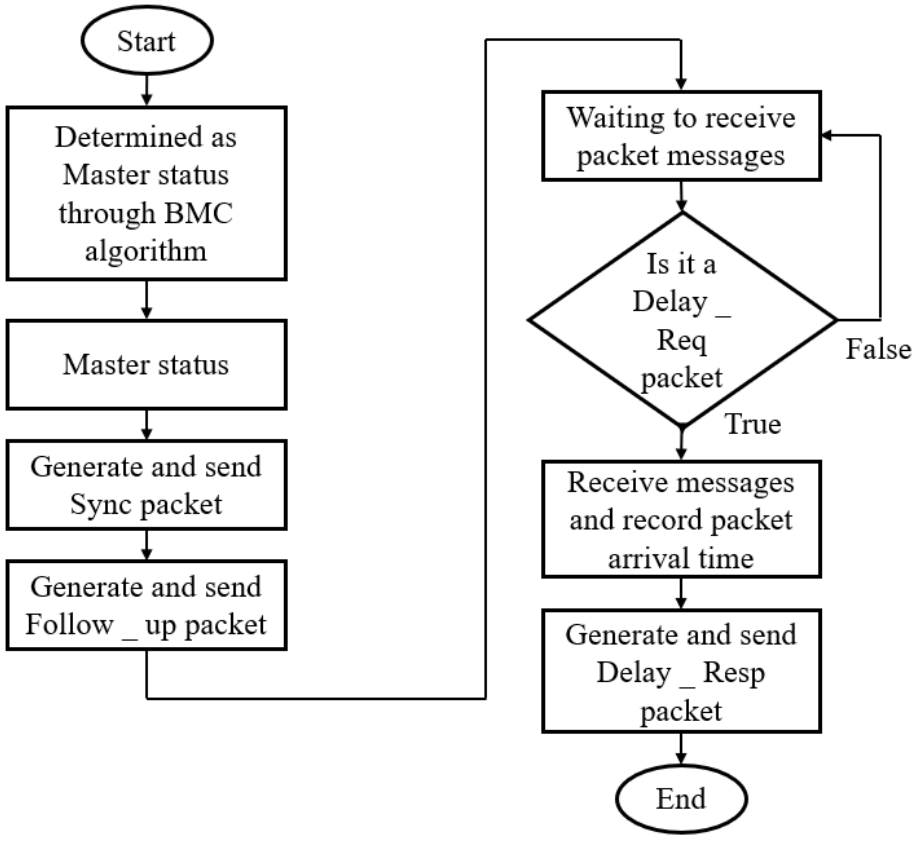

Figure 5. Primary workflow chart.

The experimental environment of our software timestamp is that the Primary transmits a synchronization packet to the Secondary every second. Figure 6 shows the measurement results. After testing 3600 pieces of data for one hour, it can be seen that most of the measured values are concentrated in ±60 between microseconds.

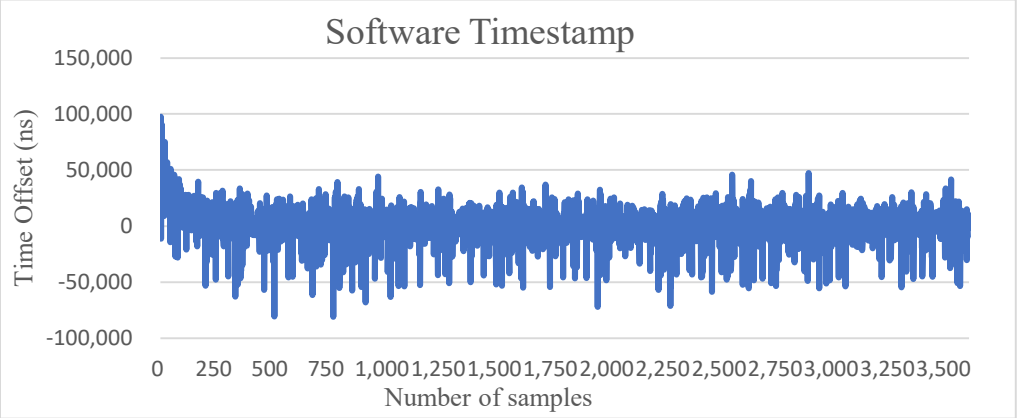

Figure 6. Software timestamp test results.

In order to improve the accuracy of clock synchronization, it is necessary to make the timestamp closer to the physical interface, that is, the MAC or PHY layer, to reduce the impact of stack protocol delay [13–15], also known as a hardware timestamp. The implementation of our hardware timestamp is shown in Figure 7. First, set up two ZedBoard development boards. Table 3 shows the equipment specifications and the system software used. They are respectively used as the Primary and the Secondary of the system. The Ethernet is assigned a fixed IP for connection, and then the software package Linux PTP is also used for time synchronization.

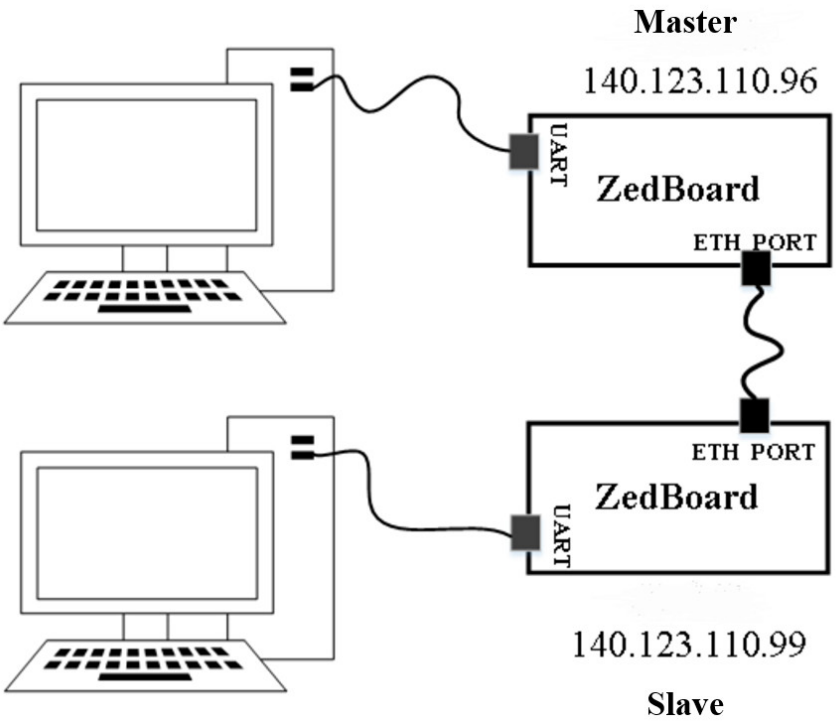

Figure 7. System hardware implementation architecture diagram.

Table 3. Development board specifications.

Device Specifications and System Software Used		
CPU	**OS**	**Software**
Cortex-A9	Petalinux 2017.4 & Petalinux 2020.2	C language

Figure 8 is a simple Zynq-embedded development board workflow chart. We want to build a development environment for an embedded platform of a Linux system with a package software Linux PTP, and the Petalinux platform is a tool developed by Xilinx for the Zynq-embedded Linux system, can run in the computer's Linux (such as ubuntu) system environment and tailor-made execution files, and can be used together with Xilinx ISE hardware design tools [16] to simplify the development process of Linux systems such as Zynq®-7000 SOC and improve design efficiency. We use Petalinux 2017.4 and Petalinux 2020.2 for implementation.

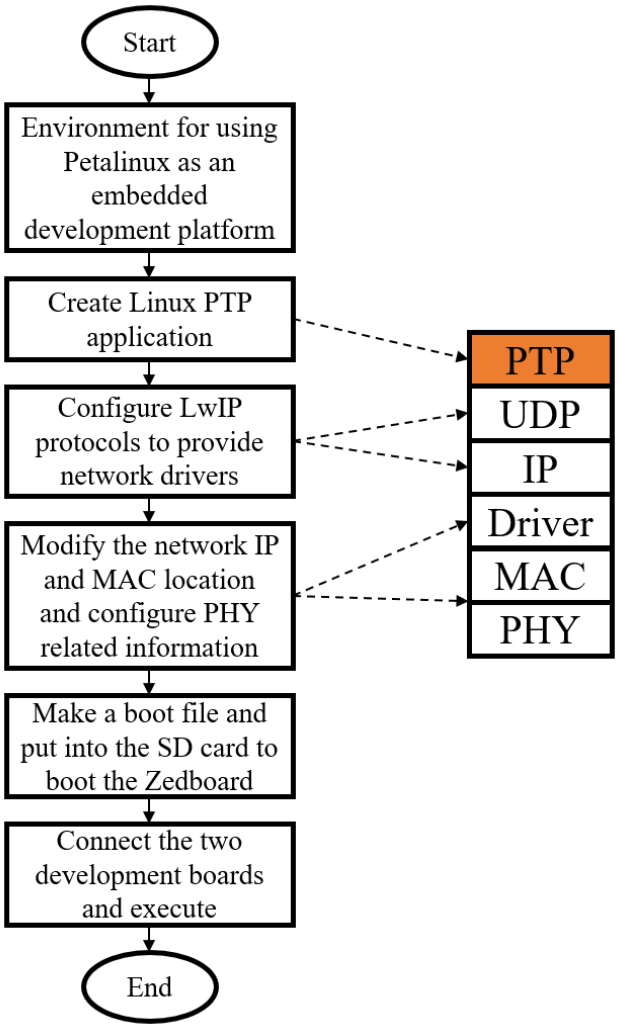

Figure 8. Workflow chart of Zynq-embedded development board.

Connect the two ZedBoard development boards with a network cable, make sure that the MAC and network IP of the two development boards are different, and run the PTP application, as shown in Figure 9.

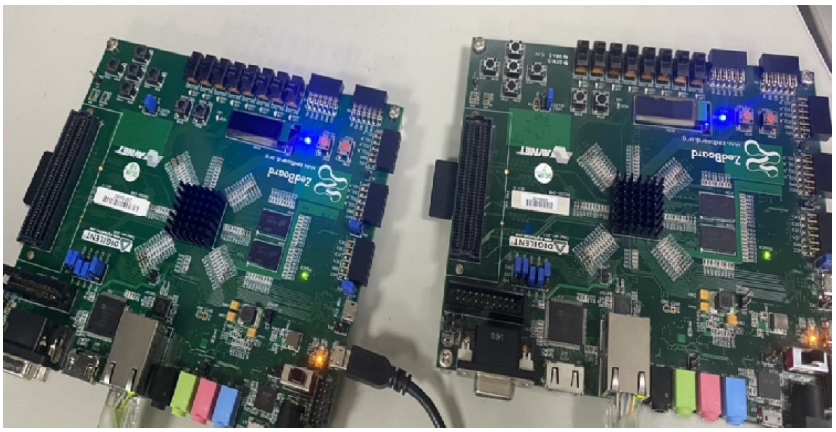

Figure 9. Diagram of connecting two Zedboard development boards.

The experimental environment for our hardware timestamp is that the Primary sends a synchronization packet to the Secondary every second. Figure 10 shows the measurement results. After testing 3600 records for one hour, it can be seen that most of the measured values are concentrated in between ±20 nanoseconds.

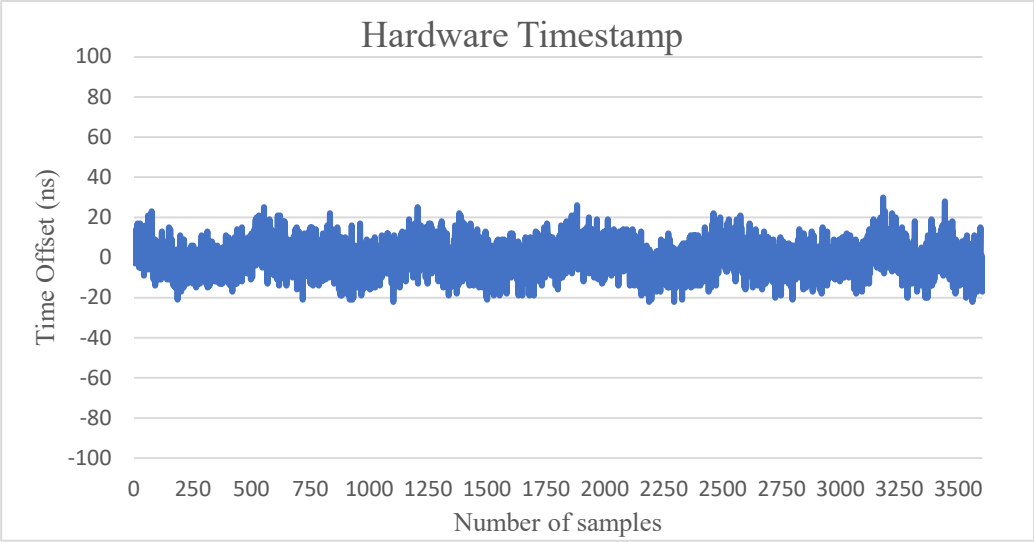

Figure 10. Hardware timestamp test results.

Based on the FPGA-based IEEE 1588 standard synchronization architecture [17], we implemented a timestamp that can be captured between the PHY layer and the MAC layer, eliminating delay jitter caused by network protocol stacking, and thereby improving synchronization accuracy. When the Primary and Secondary are connected through a network cable, the deviation of IEEE 1588 clock synchronization can be limited to ±20 ns. Compared with the related research in Table 4, the realization results of this article have high clock synchronization accuracy.

Table 4. Comparison chart.

Item	According to [6]	According to [18]	According to [19]	This Article
Development board	Xilinx Zynq-7000	Xilinx ZynqMP UltraScale	-	Xilinx Zynq-7000
Driver/bus	Axi	Macb	-	Emacps
Time offsets (ns)	±40	±100	±40	±20

From the above practice and related research, it is found that in order to improve the accuracy, the congenital frequency error between the oscillators must be considered. An IEEE 1588 auxiliary timestamp with dynamic frequency compensation is also proposed; it is designed as a chip to reduce the delay of protocol stacking and to improve the accuracy of time so that it can be applied to more needs.

4. IC Design and Implementation of PTP Hardware-Assisted Timestamp

Figure 11 is our system architecture diagram, taking Primary sending packets to Secondary as an example. The CPU software (SW) acts as the processing core, manages the synchronization process in the hardware, calculates the time offset, and adjusts the hardware's real-time clock. The hardware is located in the gigabit media-independent interface (GMII) between the MAC layer and the PHY layer. GMII can be used to detect the information of each packet on the interface and communicate with control signals.

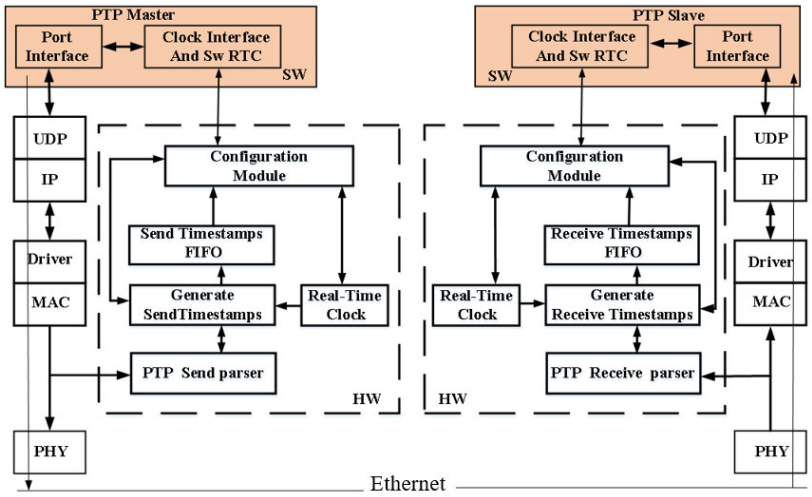

Figure 11. System architecture diagram.

Figure 12 is our Block Diagram. The hardware part completes the frequency-adjustable real-time clock (Real-Time Clock), the parser (Parser) of the PTP packets at the receiving end (Receive) and sending end (Send), and the generation of timestamp (Generate timestamps) at the receiving end (Receive) and sending end (Send), and write the timestamp with the control signal into FIFO (First In, First Out). If the control signal is recorded, write the recorded timestamp to the output, and update the corresponding control signal; otherwise, the timestamp will be discarded. The I/O pin assignment is shown in Table 5.

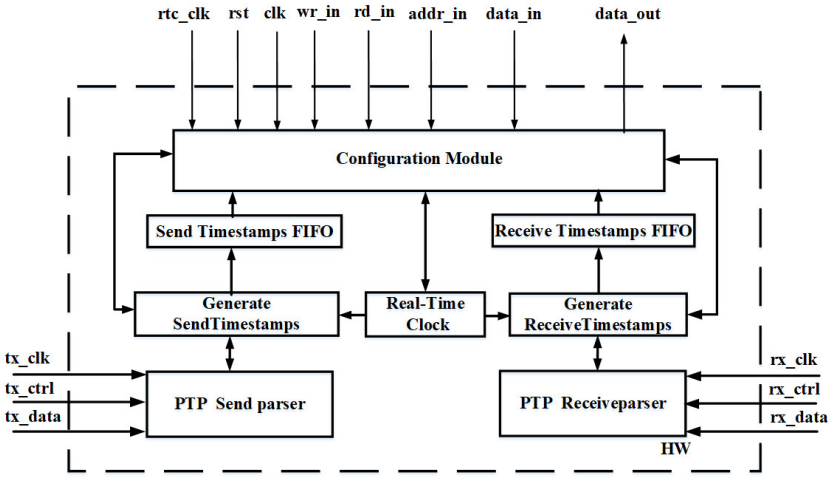

Figure 12. Block diagram.

Table 5. I/O pin assignment.

Input/Output	Pin Name	bit	Description
Input	clk	1	System clock, positive edge trigger
Input	rst	1	Reset, negative edge trigger
Input	rd_in	1	Control signal, indicating when to read data to data_out
Input	wr_in	1	Control signal indicating when to write data_in
Input	addr_in	6	Contains the address of the node that must be synchronized
Input	data_in	32	Contains the data of the nodes that must be synchronized, serialized in
Input	rtc_clk	1	From the RTC module
Input	rx_clk	1	All signals on the write and read domains are synchronized to this clock
Input	rx_ctrl	1	Control signal indicating when to read data from rx_data
Input	rx_data	8	Packet data, serial input
Input	tx_clk	1	All signals on the write and read domains are synchronized to this clock
Input	tx_ctrl	1	Control signal indicating when to read data from tx_data
Input	tx_data	8	Packet data, serial input
Output	data_out	32	Combined into timestamp, serial output

The hardware-assisted timestamp is mainly divided into two blocks: the receiver (Receive) and the sender (Send), and use the PTP parser (Parser) with the real-time clock to generate the time stamp, respectively, and store the timestamp in the FIFO register. In the process, the PTP configuration register (Configuration Module) is updated through the control signal, so that the software end can obtain the timestamp. Next, we will describe the design of the hardware, as shown in Figure 13.

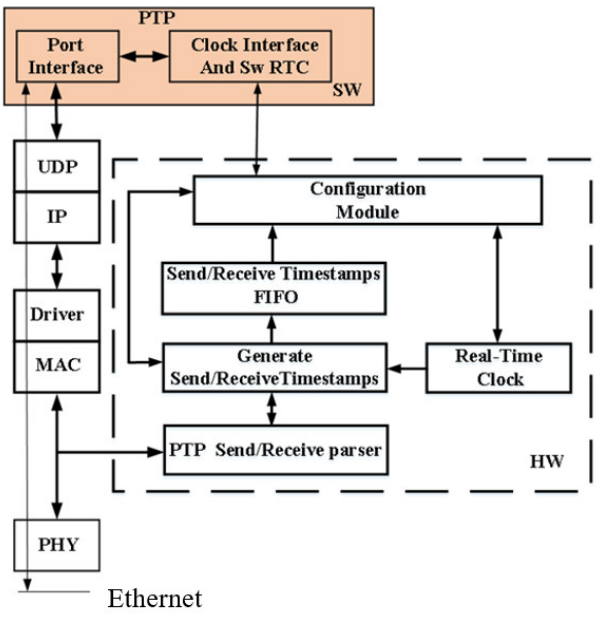

Figure 13. Block module.

The Configuration Module is a register that stores the timestamp of the PTP message and other Module-related parameters. The main structure is a multiplexer. Input the 32-bit data_in and 8-bit addr_in as the relevant parameter address of the synchronization node, and input them in series. Input 40-bit period_comp and 40-bit adj comp as dynamic frequency compensation parameters, which are mainly obtained from PTP parser Module and Generate timestamps Module, and tx_data_in and rx_data_in are two packets of information obtained between the MAC layer and the PHY layer because the total length of the packet is long and not fixed, so 8-bit serial input is used to reduce the number of IO Pads. The output is data_out that is combined with other Module-related parameters and timestamps. Since the original output data_out pins are as many as 256, it will cause a serious pad limit. Therefore, 32-bit serial output is used to reduce the number of IO Pads, as shown in Figure 14.

A PTP parser is a module used to parse whether the received packet is a PTP packet message. Its main structure is a comparator, as shown in Figure 15. The input Data_in is the packet information obtained between the MAC layer and the PHY layer and is serially input, and the multiplexer and the cnt counter are used to obtain the information used to judge the message header. In the information segment, whether it is a PTP packet is determined by identifying the message header layer by layer, and the output ptp_infor is a 32-bit PTP packet identification. ptp_found records whether it is a PTP packet, which will be sent to the FIFO Module for use. ptp_comp analyzes whether the packet is a Sync and Follow_up packet, and sends this signal to the Configuration Module for use.

The function of Real-Time Clock is to generate a real-time clock with adjustable frequency. The architecture of the real-time clock is shown in Figure 16. The input and output time accumulators adopt the PTP protocol standard: 48-bit second field, 32-bit nanosecond field, and add an 8-bit sub-nanosecond (sub-ns) field to improve clock accuracy. The input period_in is composed of 40-bit temporary registers, of which 8 bits record nanoseconds, and the last 32 bits record sub-nanometers. The main function is to accumulate each oscillation of the oscillator in the free-running mode of the system; for an example such as a reference clock for a frequency of 125 MHz, add 8 nanoseconds to each oscillation. The clock is usually implemented using a counter register, driven by a quartz crystal oscillator. The disadvantages of these oscillators are low accuracy, susceptibility to environmental influences, low long-term stability, and non-adjustable frequency. The frequency of the crystal changes over time and as the ambient temperature changes. With the help of frequency compensation technology, the crystal frequency can be dynamically compensated to keep the frequency at a stable value. Crystals with relatively poor performance can also be successfully used in high-precision clock synchronization systems. The frequency is compensated by initiating the time adjustment step by writing the period adjustment and period count to the period_adj and adj_cnt registers, respectively. The period adjustment is in nanoseconds ns, with 8 integer bits and 32 decimal bits. Since the input time accumulator is 8-bit sub-nanoseconds, the resolution is 2^{-8} nanoseconds and the register of the oscillator is 32-bit sub-nanoseconds, which means the resolution is 2^{-32} nanoseconds, so when the accumulated value of the oscillator is not an integer of 2^{-8}, use a temporary register to save its decimal place and count it in a loop. This method can improve the precision of the clock.

Figure 17 shows the dynamic frequency compensation Frequency compensation [20–22] used in this article. The PTP Primary will periodically send a synchronization packet to the PTP Secondary, record the sending time t_1, t_1' and the receiving time t_2, t_2' in the temporary register, and use a 32-bit count temporary register to calculate the compensated frequency, and calculate the compensated value FreqCompValue to compensate the Secondary clock.

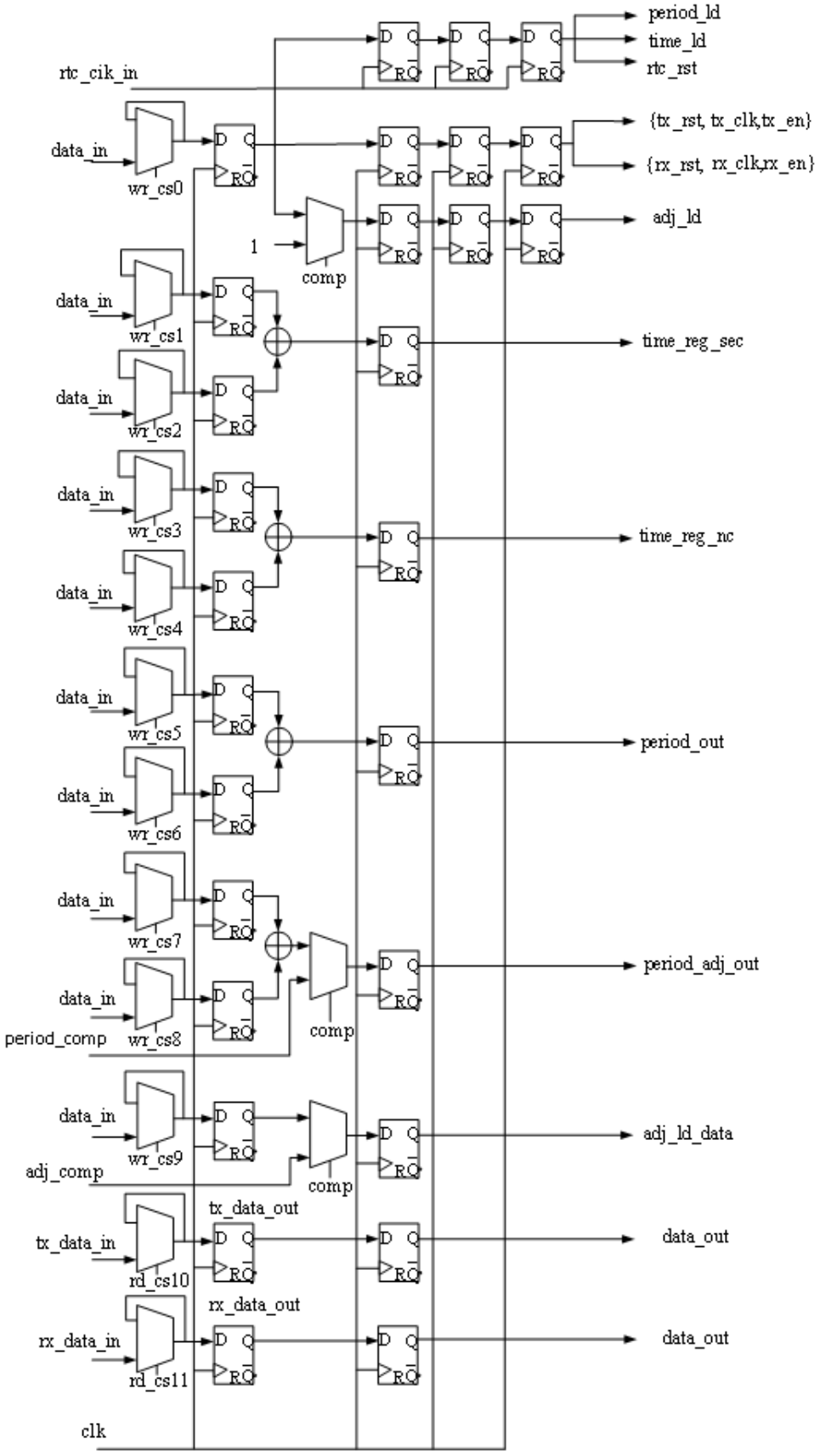

Figure 14. Configuration module.

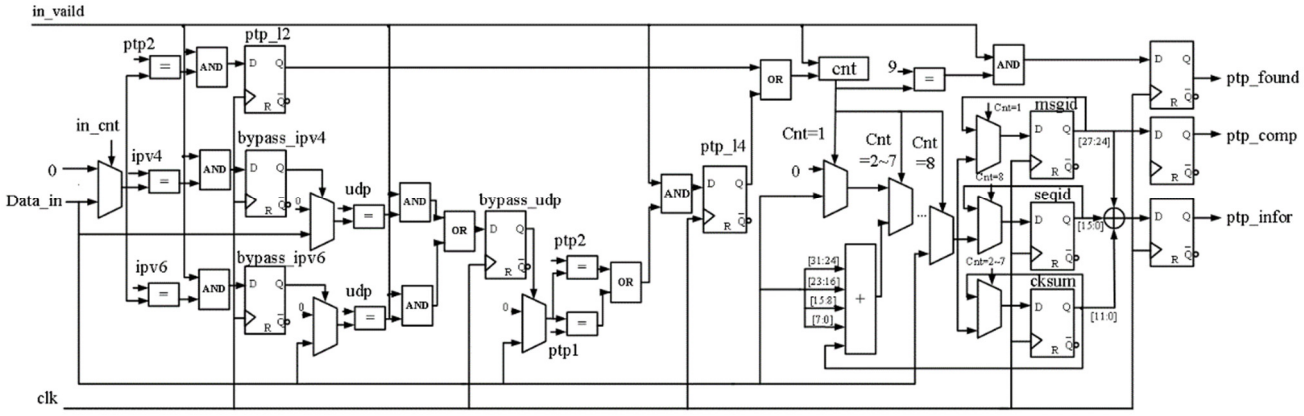

Figure 15. PTP parser module.

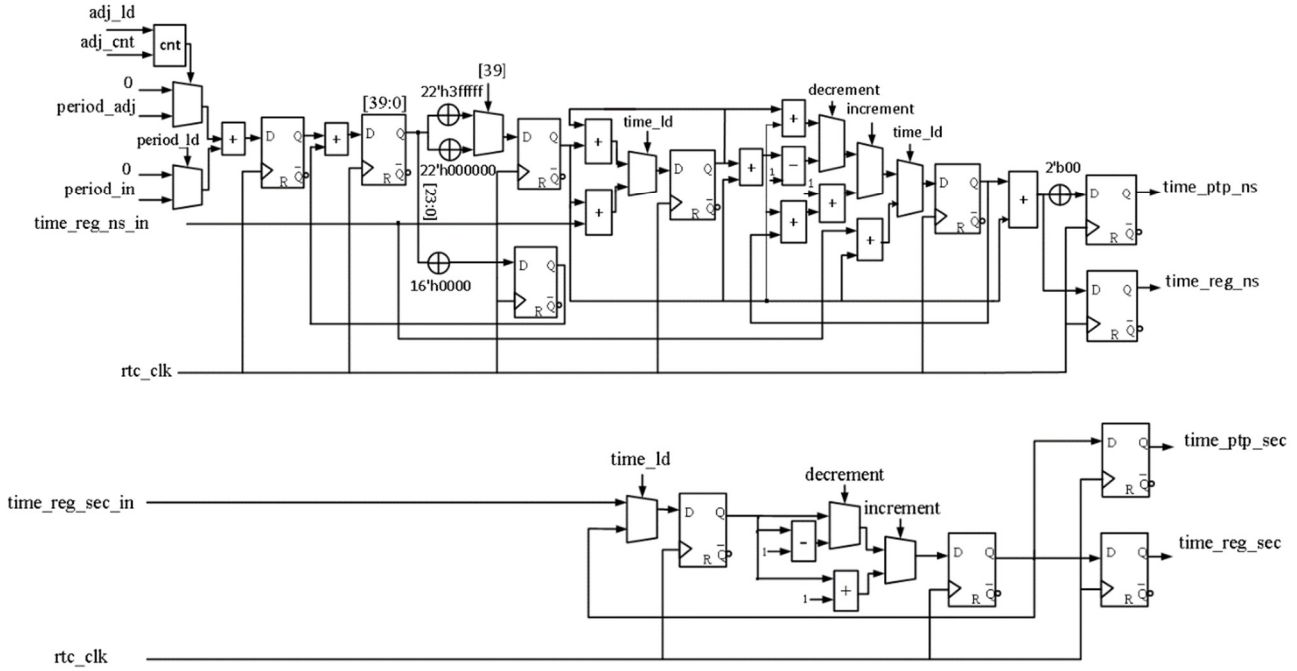

Figure 16. Real-time clock module.

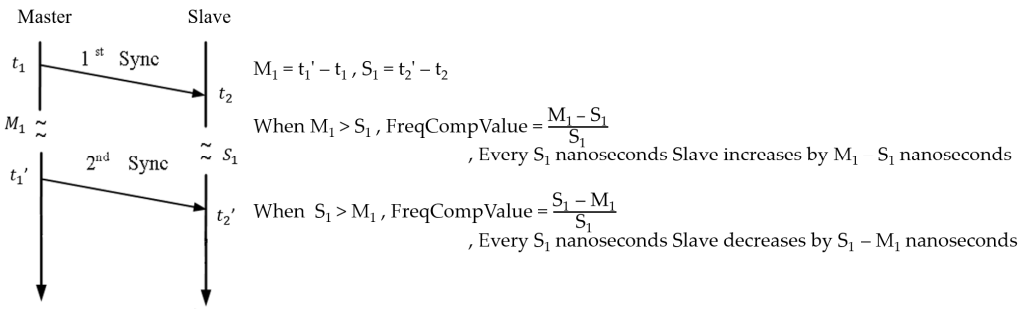

Figure 17. Frequency compensation.

The Generate Timestamps Module is shown in Figure 18. It combines the information in the above-mentioned module register to form a timestamp. The timestamp [23] is composed of 110 bits, which includes 80-bit time information and 30-bit PTP Packet identification, and store this data in FIFO, using the control signal generated from the parser to indicate whether to generate a timestamp. In order to avoid the problem of metastable

state when sending relevant parameters to the corresponding registers, two DFF registers and a hand-shaking protocol are added to trigger the action, so that the output can obtain the correct control signal or Packet information.

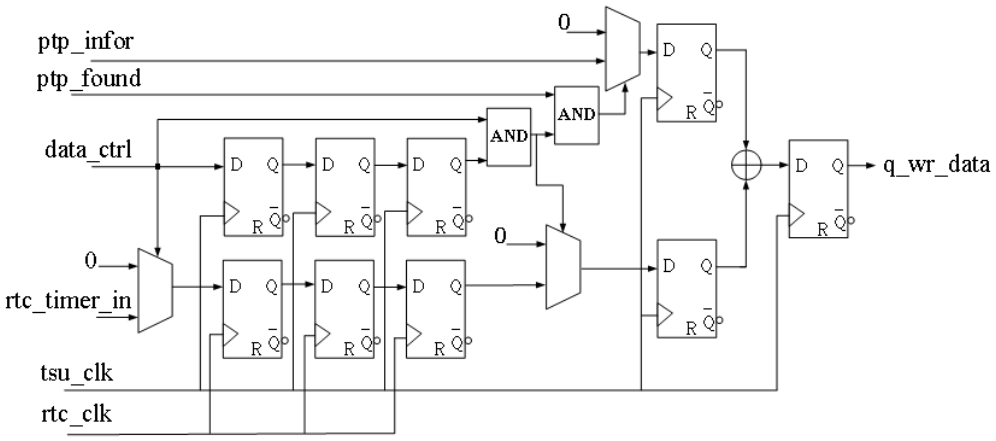

Figure 18. Generate timestamps module.

We use the IP provided by Synopsys, as shown in Figure 19; the reading and writing work under their respective clocks, and the writing clock will match the control signal to write the data into the FIFO. FIFO contains dual-port RAM (random access memory), which can read and write at the same time. When writing, wr_addr starts from 0; every time a piece of data is written wr_addr increases by one and points to the next storage unit. When the FIFO push_full is full, the data will no longer be written; otherwise, the previous data will be overwritten and the data will be lost. The read clock will be matched with the control signal to read the data from the FIFO. When reading, rd_addr starts from 0; every time a piece of data is read, rd_addr increases by one, and points to the next storage unit. When the FIFO pop_empty is empty, the data can no longer be read; otherwise, the read data will be wrong data.

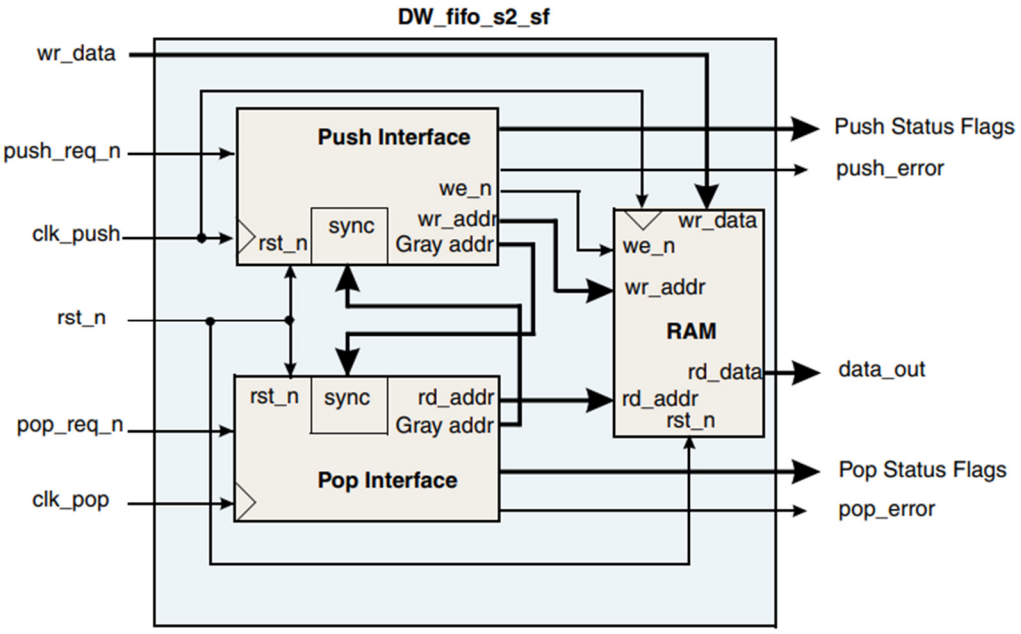

Figure 19. FIFO module [24].

5. Experimental Results and Analysis

We use Faraday's 0.18 μm and TSMC's 45 nm ARM standard component libraries to design and implement, and Gate_count is based on 2-input NAND to calculate the approximate number of logic gates used. The results of the synthesis experiments are compared in Table 6.

Table 6. Experimental results.

Technology	Faraday_U18	TSMC/ARM_T40G
Frequency	270	500
Area (μm^2)	640,396.52	45,717.89
Power (mW)	66.61	9.68
Gate Count	68,564.93	67,232.91

Our test pattern is implemented through the hardware timestamp mentioned above. Two Zedboard development boards and a PC are connected to a general router through a network cable, as shown in Figure 20, and the packet is captured through the software Wireshark. Then, you can extract PTP-related packets, and use the contents of the packets as a test pattern for hardware circuit simulation.

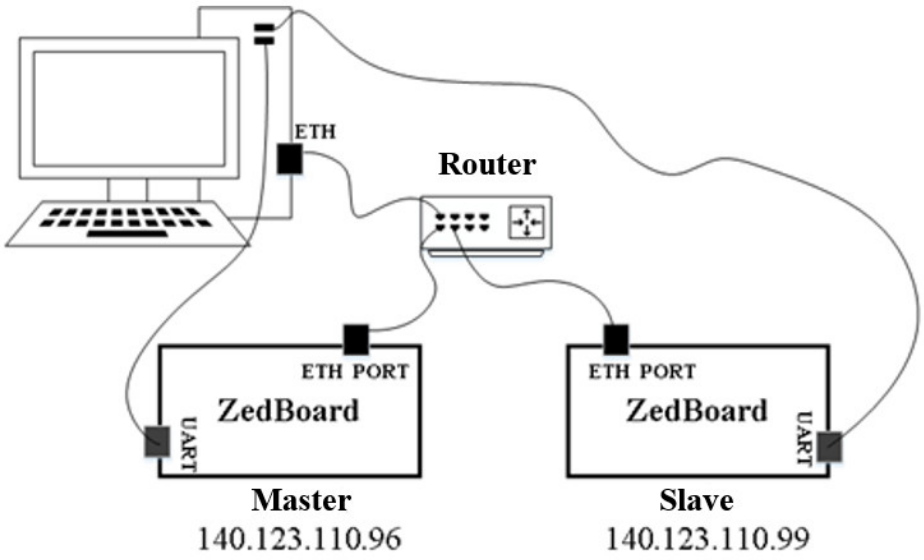

Figure 20. Capture packet structure diagram.

Our IC layout is shown in Figure 21, and the chip specifications are shown in Table 7.

Table 7. Chip specification table.

Technology	TN40G
Frequency	370 MHz
Core Area	247,894 μm^2
Gate Count	67.23 K

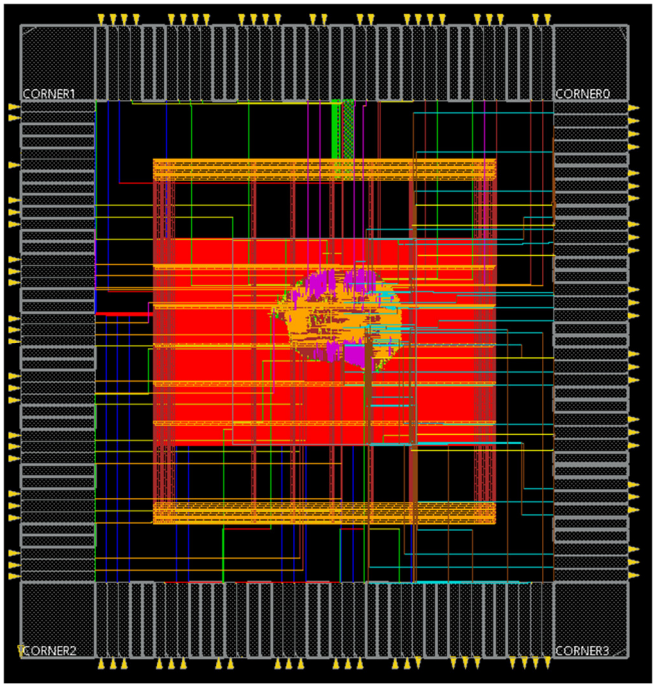

Figure 21. PTP hardware-assisted timestamp operation chip.

We have implemented three kinds of timestamps, and their comparison is shown in Table 8. It can be seen that the offset of the software timestamp is the largest, which also means that its error is the largest.

Table 8. Time comparison table.

Type	Software Timestamp	Hardware Timestamp	IC Design
Time offsets (ns)	±60,000	±20	±10.8

6. Conclusions

In this paper, based on the detailed analysis of the PTP protocol, we first use the Ethernet network and fixed IP address for connection, and then use the open source software package Linux PTP for measurement. Its software timestamp is 60 microseconds. Then, we build a clock synchronization system using FPGA to match the PTP protocol. On the Xilinx Zynq-7000 SOC platform of Petalinux, a hardware solution based on the IEEE1588 protocol in the Linux PTP application is implemented. The experimental results show that the implementation results have high clock synchronization accuracy, and the hardware timestamp is 20 nanoseconds. Compared with related implementations, this design improves the time synchronization accuracy, simplifies the hardware circuit design, greatly reduces the development cost and difficulty, and has excellent portability and scalability, which provides a certain reference value for future research.

After matching the implementation of the FPGA and the PTP protocol in this paper, it is found that in order to improve the accuracy, the congenital frequency error between the oscillators must be considered. An IEEE 1588 auxiliary time stamp with dynamic frequency compensation is also proposed; it is designed as a chip and can operate at 370 MHz under 45 nm (TN40G), uses four-cycle serial output, and the Latency is 10.8 (2.7 × 4) ns resolution, which reduces the delay of the protocol stack and improves the accuracy of the time, so that it can be applied on more demand.

This thesis designs PTP auxiliary time stamps and completes the realization of detecting PTP packet messages and recording time stamps. However, in order to verify the

complete PTP system clock synchronization function, it is necessary to use a software processor as the processing core and use the ARM platform of the Linux system. The synchronization process in the FPGA is managed, offsets are calculated, hardware clocks are adjusted, etc. If the system can be integrated to complete the SOC single chip with the complete functions of the PTP protocol, the implementation cost and volume can be reduced, making it beneficial to be applied to more industrial Internet of things devices.

Author Contributions: Methodology, T.-C.H. and Y.-S.C. (Yuan-Sun Chu); Software, Y.-K.L.; Data curation, Y.-K.L.; Writing—review & editing, Y.-S.C. (Yee-Shao Chen) and B.-L.W.; Supervision, T.-C.H. and Y.-S.C. (Yuan-Sun Chu). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, L.-H.; Chen, Y.; Chu, Y.-S.; Hou, T.-C. Optimization and realization of Network Time Protocol on IIoT. In Proceedings of the IEEE International Conference on Consumer Electronics-Taiwan, Penghu, Taiwan, 15–17 September 2021.
2. Hou, T.-C.; Liu, L.-H.; Lan, Y.-K.; Chen, Y.-T.; Chu, Y.-S. An Improved Network Time Protocol for Industrial Internet of Things. *Sensors* **2022**, *22*, 5021. [CrossRef] [PubMed]
3. *IEEE Standard 1588–2008*; IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE: Piscataway, NJ, USA, 2008.
4. *IEEE Std 1588–2019*; IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. IEEE: Piscataway, NJ, USA, 2019.
5. Zhao, B.; Wang, N. The implementation of IEEE 1588 clock synchronization system based on FPGA. In Proceedings of the Fifth International Conference on Intelligent Control and Information Processing, Dalian, China, 18–20 August 2014; pp. 216–220.
6. Moreira, N.; Lázaro, J.; Bidarte, U.; Jimenez, J.; Astarloa, A. On the Utilization of System-on-Chip Platforms to Achieve Nanosecond Synchronization Accuracies in Substation Automation Systems. *IEEE Trans. Smart Grid* **2017**, *8*, 1932–1942. [CrossRef]
7. Lee, S.S.W.; Lee, T.; Li, K. NetFPGA based IEEE 1588 module for timesynchronized software-defined networking. In Proceedings of the 6th International Conference on Information Communication and Management (ICICM), Hertfordshire, UK, 29–31 October 2016; pp. 253–259.
8. Idrees, Z.; Granados, J.; Sun, Y.; Latif, S.; Gong, L.; Zou, Z.; Zheng, L. IEEE 1588 for Clock Synchronization in Industrial IoT and Related Applications: A Review on Contributing Technologies, Protocols and Enhancement Methodologies. *IEEE Access* **2020**, *8*, 155660–155678. [CrossRef]
9. An, L.; Yang, G.-H. Distributed Optimal Coordination for Heterogeneous Linear Multiagent Systems. *IEEE Trans. Autom. Control* **2022**, *67*, 12. [CrossRef]
10. Mills, D.L. Internet time synchronization: The network time protocol. *IEEE Trans. Commun.* **1991**, *39*, 1482–1493. [CrossRef]
11. Scheiterer, R.L.; Na, C.; Obradovic, D.; Steindl, G.; Goetz, F.-J. Synchronization performance of the precision time protocol in industrial automation networks. *IEEE Trans. Instrum. Meas.* **2009**, *58*, 1849–1857. [CrossRef]
12. Giorgi, G.; Narduzzi, C. Performance Analysis of Kalman-Filter-Based Clock Synchronization in IEEE 1588 Networks. *IEEE Trans. Instrum. Meas.* **2011**, *60*, 2902–2909. [CrossRef]
13. Ouellette, M.; Ji, K.; Liu, S.; Li, H. Using IEEE 1588 and boundary clocks for clock synchronization in telecom networks. *IEEE Commun. Mag.* **2011**, *49*, 164–171. [CrossRef]
14. Garg, A.; Yadav, A.; Sikora, A.; Sairam, A.S. Wireless Precision Time Protocol. *IEEE Commun. Lett.* **2018**, *22*, 812–815. [CrossRef]
15. Ferrant, J.-I.; Gilson, M.; Jobert, S.; Mayer, M.; Montini, L.; Ouellette, M.; Rodrigues, S.; Ruffini, S. Development of the First IEEE 1588 Telecom Profile to Address Mobile Backhaul Needs. *IEEE Commun. Mag.* **2010**, *48*, 118–126. [CrossRef]
16. Holler, R.; Sauter, T.; Kero, N. Embedded SynUTC and IEEE 1588 clock synchronization for industrial Ethernet. In Proceedings of the 2003 IEEE Conference on Emerging Technologies and Factory Automation, Lisbon, Portugal, 16–19 September 2003; Volume 1, pp. 422–426.
17. Moreira, N.; Astarloa, A.; Lazaro, J.; Garcia, A.; Ormaetxea, E. IEEE 1588 transparent clock architecture for FPGA-based network devices. In Proceedings of the 2013 IEEE International Symposium on Industrial Electronics (ISIE), Taipei, Taiwan, 28–31 May 2013; pp. 1–6.
18. Pandey, P.; Pratap, B.; Pandey, R.S. Implementation of FreeRTOS based Precision Time Protocol (PTP) application as per IEEE1588v2 standards for Xilinx Zynq UltraScale Plus MPSoC devices. In Proceedings of the 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 17–19 July 2019; pp. 1968–1973.

19. Yin, H.; Fu, P.; Qiao, J.; Li, Y. The implementation of IEEE 1588 clock synchronization protocol based on FPGA. In Proceedings of the 2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Houston, TX, USA, 14–17 May 2018; pp. 1–6.
20. Ingram, D.M.; Schaub, P.; Campbell, D.A.; Taylor, R.R. Performance Analysis of PTP Components for IEC 61850 Process Bus Application. *IEEE Trans. Instrum. Meas.* **2013**, *62*, 710–719. [CrossRef]
21. Decusatis, C.; Lynch, R.M.; Kluge, W.; Houston, J.; Wojciak, P.A.; Guendert, S. Impact of Cyberattacks on Precision Time Protocol. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 2172–2181. [CrossRef]
22. Itkin, E.; Wool, A. A Security Analysis and Revised Security Extension for Precision Time Protocol. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 22–34. [CrossRef]
23. Dong, M.; Qiu, Z.; Pan, W.; Chen, C.; Zhang, J.; Zhang, D. The Design and Implementation of IEEE 1588v2 Clock Synchronization System by Generating Hardware Timestamps in MAC Layer. In Proceedings of the 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), Colmar, France, 11–13 July 2018; pp. 1–5.
24. Synopsys. DW_fifo_s2_sf. Available online: https://www.synopsys.com/dw/ipdir.php?c=DW_fifo_s2_sf (accessed on 5 December 2021).