Implementation of FreeRTOS based Precision Time Protocol (PTP) application as per IEEE1588v2 standards for Xilinx Zynq UltraScale Plus MPSoC devices

Priyanshu Pandey
School of VLSI and ESD
National Institute of Technology
Kurukshetra, India
priyanshupanday95@gmail.com

Bhanu Pratap
Electrical Engineering Department
National Institute of Technology
Kurukshetra, India
bhanumnnit@gmail.com

Radhey Shyam Pandey
System Software
Xilinx India Pvt. Ltd.
Hyderabad, India
radheys@xilinx.com

***Abstract*— This work proposes a method for the implementation of PTPd (Precision Time Protocol deamon) with the Gigabit Ethernet Module (GEM) Interface for test devices. The programmable real-time clock and timestamp unit was implemented in the programmable logic to achieve the synchronization accuracy of sub-microsecond. PTPd has been reconfigured and implanted into a FreeRTOS operating system to create a PTP state machine and the IEEE1588 IP core software driver has been created to give the application layer access to the precise timestamp in the link layer. This paper evaluates in detail the matter of the IEEE 1588 standard and provides a "Xilinx ZynqMP UltraScale plus SoC + MAC + PHY" technique, a hardware solution for stamping the arrival and departure period of PTP packets. Where packets are obtained between MAC and PHY layers from Media Independent Interface (MII). The result indicates that the solution can detect the messages transmit and receipt for synchronization accurately.**

Keywords*— *Xilinx ZynqMP UltraScale Plus SoC, IEEE1588v2, Time synchronization, Timestamping

I. INTRODUCTION

Nowadays, Time synchronization is alignment of time as well as advance of the over distributed network. it's critical in many applications such as real time application control system network or video applications and voice related applications [2]. there is frequency synchronization phase synchronization and timing synchronization. We can take a look at touch upon those along the way and it is important to coordinate action between distributed nodes or distributed computers and also, it's needing to trigger the measurement and then actually measure the same time over the different nodes also referenced events which can trigger all over distributed nodes.

In IEEE1588, it is about distributing the time reference time to the other nodes using the packet so it's the distribution of precise time over the packet-based system [3]. we can synchronize our frequency time or phase using IEEE1588 and it's providing high accuracy time over your network typically a sub microseconds range and it's based on hierarchical structure so master clock provides a time to your slave. the slave can synchronize the time of the master clock.

The objective of this paper is to develop a general system for IEEE1588 synchronization protocol in the GEM network, using Xilinx ZyncMP UltraScale Plus MPSoC as a platform for measuring synchronization performance parameters. For system synchronization, data synchronization and test data analysis, these parameters are essential. Based on the parameters, additional consideration is given to facts affecting synchronization and a optimized method is proposed and introduced the precise of sub-microsecond synchronization in test.

II. OUTLINE OF THE IEEE1588

The IEEE1588 system clocks are equipped in a master-slave hierarchy. Firstly, the master clock is identified and after that, it's established two-way timing exchange.

PTP message sequence with timestamp

In this figure 1, how PTP protocol message sequence actual works.

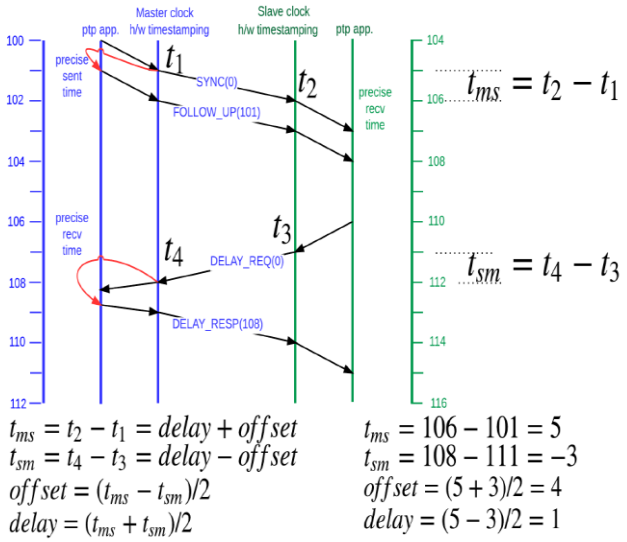

Figure 1. PTP message sequence

We put another line here that application and then this is the actual time stamping put it on a slave side as well master side. It is the hardware timestamping that reads the actual time. Master PTP application sends the packet out before we send the packet to the wire you don't know the time, so the sync that's the first message is called SYNC with no unknown time that's 0 here, but as soon as the packet is on the wire, your chipset says it's okay this packet sends t1 time. Using this t1 time put it in the next packet Follow_Up message sending. we sent previous packet at 101 second so at this point on a slave knows when it is sent the actual master says it sent and when it arrived here hardware says I receive this packet t2. The next sequence slave software

says send a packet as soon as Delay_Req packet is on the wire we know the t3 time sent it to master and then master received a packet. Then it received it knowns actual accurate timestamp t4 put it on to the next Delay_Resp packet. Which is the till they respond by the way this is still a request from the slave and then we send a delay response saying that after received your delay request packet. at this time so once sleep get all this packet collected then sleep can calculate that offset of between master and slave and then delay between two networks.

III. IEEE 1588 IMPLEMENTATION FOR XILINX ZYNQMP ULTRASCALE PLUS SOC

The zcu102 Evaluation Kit (for Xilinx ZynqMP UltraScale plus SoC) is used as the hardware platform for the hardware timestamp based IEEE1588v2 Precision Time Protocol. The solution uses the Xilinx SDK, which includes the GEM driver, the PHY driver, the ported FreeRTOS kernel, and the ported lwIP TCP/IP stack for the zcu102 board. The IEEE1588v2 Precision Time Protocol is implemented by the PTP daemon application. It's shown in fig.2.

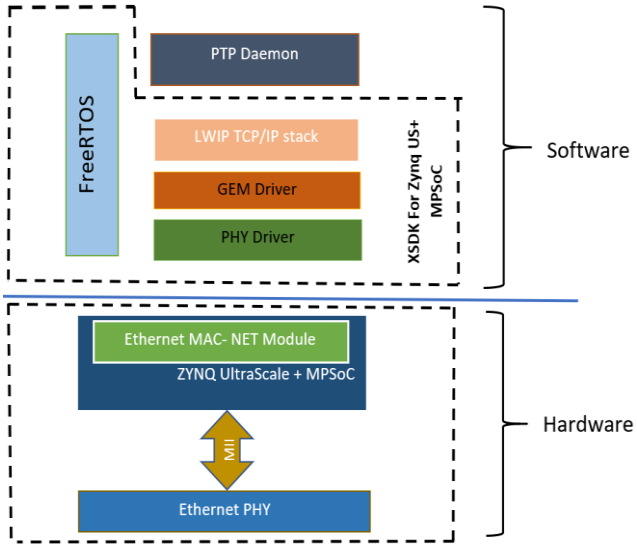

Fig. 2. PTP solution for Zynq US + MPSoC

i. Hardware components

Zynq UltraScale+ MPSoC is the Xilinx second-age Zynq platform, the combination of a processing system (PS) and user-programmable logic (PL) into the same board [8].

The PS includes the Arm® flagship Cortex®-A53 64-bit dual-core or quad-core processor and Cortex-R5 dual-core real-time processor. Apart from the earlier cost and integration benefits previously provided by the Xilinx Zynq-7000 devices, Xilinx Zynq UltraScale+ MPSoC and RFSoC boards also provide new benefits and features.

ii. Software components

The software of the IEEE 1588 usage incorporates the Xilinx SDK for the for Xilinx ZynqMP UltraScale plus SoC device hardware platform and the PTP daemon. The Xilinx SDK is a software structure for creating applications on the Xilinx MCUs, including GEM drivers, middleware, and FreeRTOS real time operating system.

a) lwIP TCP/IP stack

The lwIP implements a TCP / IP networking stack that is lightweight for embedded systems. lwIP is an open source available in C source code format providing under the Berkeley Software Distribution (BSD) style license. This stack is used to reduce the use of RAM through a full-scale TCP for use in embedded systems. The Xilinx SDK provides lwIP stack customized to run on different Xilinx installed frameworks [10]. It can be run on ARM-based Xilinx Zynq UltraScale+ MPSoC related all programmable SoC. The main stack is an IP implementation, on top of which you can choose to include the UDP, DHCP, IGMP, TCP and many other protocols according to your required and the memory size is available in the designed system.

b) FreeRTOS real-time operation system

FreeRTOS is an open source RTOS. It is real time operating system kernel or real time scheduler targeting embedded devices. It utilizes embedded application to fulfill its requirements in hard-real-time. The FreeRTOS features provided by the kernel are multithreading by tasks, queues, software timer, semaphore, mutexes and event groups. FreeRTOS kernel port is available for Xilinx Zynq UltraScale+ MPSoC, ZynqMP SoC and other Xilinx devices use FreeRTOS kernel 10.0 version.

c) *GEM hardware support for PTP*

The controller detects the transmission and receipt of PTP event packet Sync, Delay_Req, PDelay_Req, and PDelay_Resp. Master-slave clock synchronization is a two-stage method.

- The master will correct the offset between the master and the slave clocks by sending the slave a sync frame with a follow-up frame with the exact time the sync frame was sent. On the master hand, the GEM controller assists modules and the slave detects precisely when the master sent the sync frame and the slave got it. Then the slave adjusts his clock to suit the master clock.

- Correct the delay in transmission amongst the master and the slave. The slave sends the master a delay request frame that sends a response frame for delay reaction. On the master hand, the GEM controller assists modules and the slave detects precisely when the slave sent the delay request frame and the master got it. Then the slave has sufficient data to adjust his clock to account for the delay.

IV. TIMESTAMPING

A. Software Timestamping

Using a conventional Ethernet module, the PTP protocol can be fully implemented in the software shown in fig. 3. The delay variance implemented by the software stack operating on the master and slave systems implies that only a constrained accuracy can be obtained since the time stamp data is implemented at the application layer. As the Master has no hardware clock, both its own internal oscillator drift and the master's oscillator drift must be compensated for the

slave-running PTP daemon, degrading time synchronization precision.

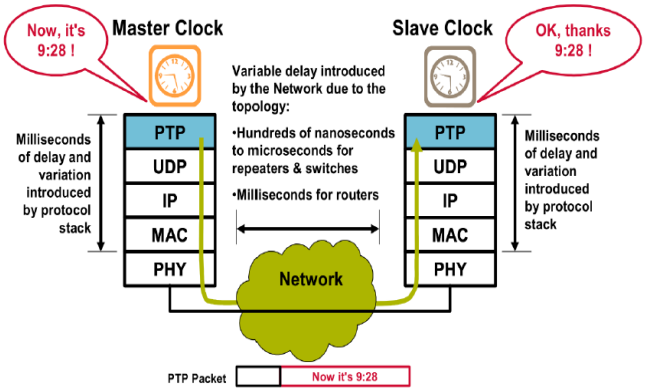

Fig. 3. Software Timestamping

B. *Hardware Timestamping*

Take timestamps nearer the physical interface, namely to the MAC or PHY layers to reduce the effect of the stack protocol delay shown in fig. 4. Master and Slaves with a Hardware time stamp then PTPd are providing time synchronizing accuracy less than 1 microsecond with independent of their operating system and network load.

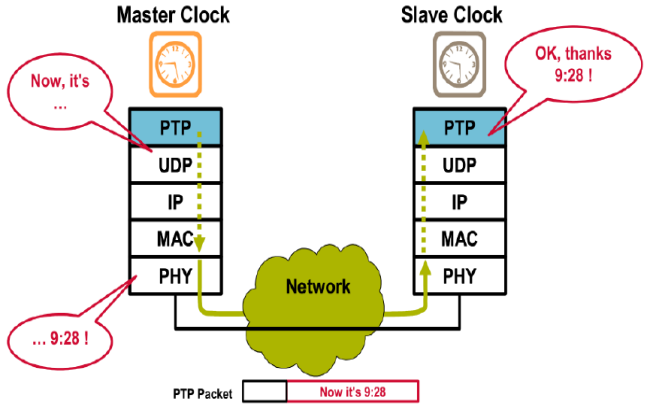

Fig. 4. Hardware Timestamping

In this proposed method, use hardware timestamping for reduce the stack protocol delay and make with FreeRTOS environment for board. It allows synchronization with significantly improved accuracy.

V. THE WHOLE PTPD DESIGN

The whole scheme is divided into three layers i.e. application, driver and hardware layer. The application layer understands the BMC algorithm, the packet processing and the PTP message transfer method based on the IEEE1588 protocol. We used PTPd is a open source software. It's achieved the software state machine. The original PTPd is established in the application layers by stacking the TCP / IP network protocol. This is a software timestamp which controls only with synchronization in milliseconds levels. We are need to solve the low precision of the software timestamp and to create an accurate hardware event trigger and improved the PTPd application.

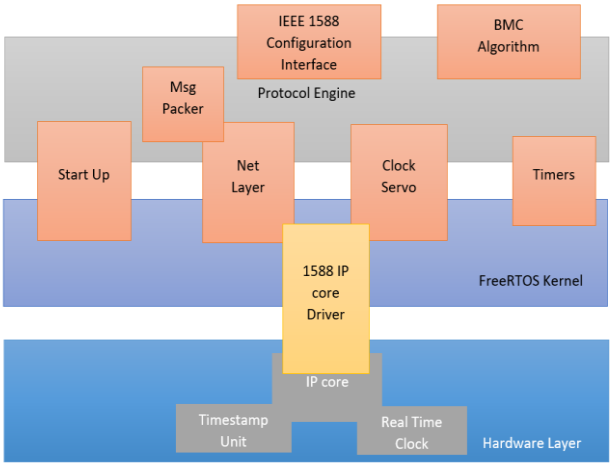

Fig. 5. The whole design of IEEE1588v2

The entire system after enhancement is shown in the Fig. 5. Here adding a scheduling process of IEEE1588 IP core for optimized the clock synchronization accuracy.

The core of the IEEE 1588 IP comprises primarily of three components: register module, timestamp module, real time clock module. The register module functions as an interface layer between the FPGA-based software and the IEEE1588 IP core.

A. *Timestamp Module*

The PTPd package format designed with ethernet based on IEEE 1588 standards, the specific standards of timestamp module are following:

a) 110bit timestamp (78bit time message, 32bit PTP packet recognition).

b) Support PTP packet format for two layers (Ethernet+PTP) and PTP packet format for four layers (Ethernet+IPv4/IPV6+UDP+PTP).

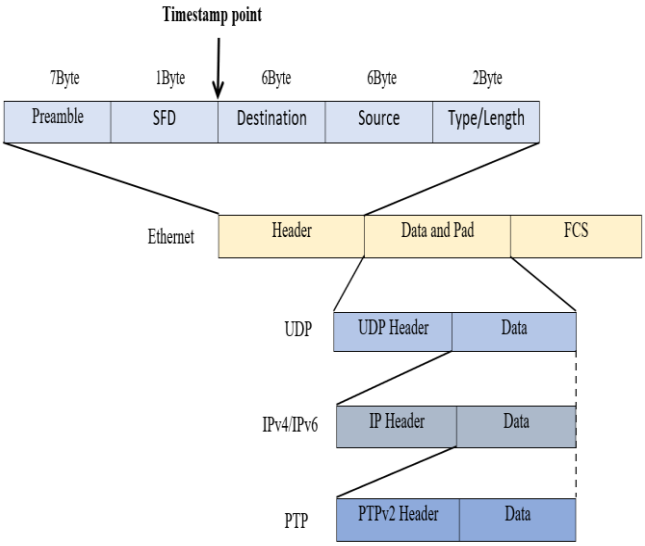

Fig. 6. IEEE1588 package format

The Timestamp module displays the data input stream consistently. PTP package format is shown in Fig. 6. Here two successive 1 at the end of the Ethernet packet SFD (start frame delimiter), this time characterize as timestamp point [6]. Currently, real-time clock time is being saved. Then we check the messages if the message is a PTP event message then the timestamp is store otherwise discard the timestamp.

B. PTP Packet Format

All PTP packets need a header, body, and suffix. This is a common header for all PTP messages [7]. IEEE1588v2 header format shown in Table 1. Here are two main points to define these messages.

Header 1588 V2									
Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
reserved								1	5
flagField								2	6
correctionField								8	8
reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

Table 1. IEEE1588v2 header format

a) messageType

MessageType's value shall show the PTP message type as described in Table 1. Message type fields (1-4) for PTP event packets are accessible in the PTP header.

Message Type	Message class	Value(hex)
SYNC_MESSAGE	Event	0
DELAY_REQ_MESSAGE	Event	1
PATH_DELAY_REQ_MESSAGE	Event	2
PATH_DELAY_RESP_MESSAGE	Event	3
Reserved		4-7
FOLLOWUP_MESSAGE	General	8
DELAY_RESP_MESSAGE	General	9
PATH_DELAY_FOLLOWUP_MESSAGE	General	A
ANNOUNCE_MESSAGE	General	B
SIGNALING_MESSAGE	General	C
MANAGEMENT_MESSAGE	General	D
Reserved		E-F

Table 1. messageType

b) sequenceId

The sequenceId field value is allocated for Sync, Pdelay Req, and Announce packets by the originator of the packet. This field needs to be recorded to match packets.Real-time clocks modules

C. Real-time module

In the IEEE1588 protocol, this module supports the format and function are as follow:

i) 48-Bit seconds, 32-Bit nanoseconds and 32-Bit sub nanoseconds domain.
ii) In the real time clock can be used external clock oscillator for better stabilization.
iii) We designed clock setting and offset setting interfaces for the PM (phase modulation) function to adjust the clock value, the modify resolution is 2^{-8}ns.
iv) We have built FM and temporary FM interfaces for FM (frequency modulation), the period modulation resolution is 2^{-32}ns. 32bit counters are given to count temporary FM's efficient time.
v) For accurate measurement can be used hardware PPS output support.

D. Software And Driver Design

The specific access to the register address through the AXI bus completes the development. The IEEE1588 IP core is consistently configured by writing the control register. At this stage we read the status of the data registers for the real-time clock and timestamp values.

The IEEE1588 IP core requirement for multiple registers is being implemented. For the FreeRTOS software platform, the more information about driver supports implementation. The default PTPd source code exists for the Linux, NetBSD, FreeBSD, Mac OS X operating systems. So PTPd porting on FreeRTOS with lwIP and Xilinx SDK drivers for the Xilinx ZynqMP UltraScale plus SoC devices. The operation network related code, system related code, Hardware timestamp related code must be added or ported in the drivers. In the PTPd, the changes or ported work on the system time routines, software timer, system services, interact with network socket and some modification of the PTP protocol. These ported work covers under the FreeRTOS. The configuration FreeRTOS tasks with Xilinx ZynqMP UltraScale plus SoC board.

We are created three task threads using FreeRTOS in the PTPd application:

- *main_thread* task: It is created in the main function. This task initializes the lwIP TCP/IP stack, static IP address setting, netmask and gateway address configuration. It's also initializes MAC address configuration; Ethernet hardware initializes and starts the PTPd application task. After initializing the lwIP and PTPd tasks it's deleted itself by calling vTaskDelete function.
- *network_thread* task: In the TCP/IP initialization, it's created by main_thread task. This task run in the main lwIP task to access the lwIP core function.
- *ptpd_thread* task: this task runs the PTPd application and created this task denotes either the master or the slave clock.

In the lwIP driver, there are initializing the IEEE1588 timer, enabling the IEEE1588 timer, Setting the time function, getting the time function, adjust the IEEE1588 timer frequency, and getting the hardware timestamp for transmit/receive frames. These are the initialization function used in the IP core to initialize all the resisters.

Based on the above process, the improved PTPd mechanisms to achieve the scheduling of PTPd hardware resources.

VI. TEST AND RESULT

A. Hardware Setup

For the test hardware setup, used two ZCU102 board (Xilinx ZynqMP UltraScale plus SoC) and connected each other. The setup has a point to point configuration where using the Ethernet cable for connected two ZCU102 board. To prevent asymmetric transmission delay interference, master and slave ports are linked with the same switch. At this stage, a significant factor influencing synchronization efficiency is the difference between slave and master oscillator frequencies. A hardware setup shown in fig. 7. Specific jumper setting is not used in the test.

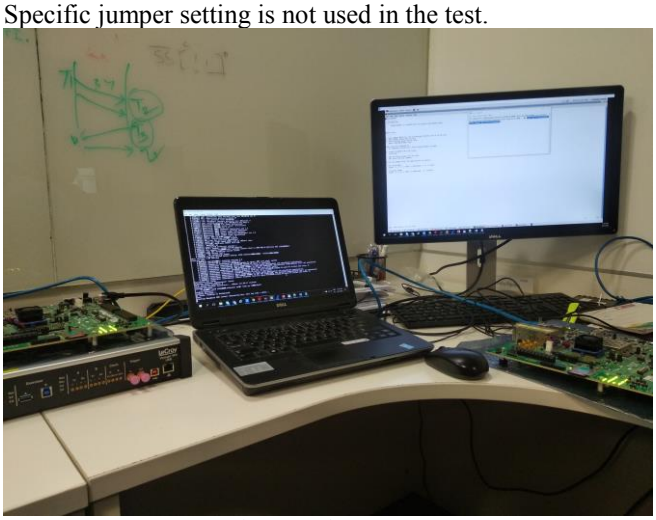

Fig. 7. Hardware Setup

B. Result

Under the FreeRTOS environment, this paper program implements the PTPd application's main function. The program runs on the psu cortexa53_0 hardware platform and used board support package (BSP) of Xilinx ZynqMP UltraScale plus SoC devices.

ptp4l Master clock output- Grandmaster:

We're going to start out listening to see if there's anybody else out there that's doing PTP. Since this is the only endpoint on the node right now, there's nobody else out there. So eventually, timeouts will start happening. You'll start realizing that, well, apparently there's nobody out there, so maybe I need to become the master. Since you're the only one out there, you'll probably get selected as the best master. And you'll become the master, and ultimately, you'll become the grandmaster it's shown in fig. 8.

```
xhdpunnaia40:~% sudo ptp4l -i eth2 -m
ptp4l[1231551.607]: selected /dev/ptp0 as PTP clock
ptp4l[1231551.608]: driver changed our HWTSTAMP options
ptp4l[1231551.608]: tx_type   1 not 1
ptp4l[1231551.608]: rx_filter 1 not 12
ptp4l[1231551.608]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[1231551.608]: port 0: INITIALIZING to LISTENING on INITIALIZE
ptp4l[1231559.144]: port 1: LISTENING to MASTER on ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
ptp4l[1231559.144]: selected best master clock 64006a.fffe.71fde6
ptp4l[1231559.144]: assuming the grand master role
```

Fig. 8. Master clock output

Ptp4l slave clock output:

we become slave, we're going to start locking to that master. The strings s0, s1, s2 show the different state of the clock servo: s0 is unlocked, s1 is the phase of the clock and s2 is locked. The clock will not be jumped (only slowly adjusted) once the servo is in the locked condition. INITIALIZING, LISTENING, UNCALIBRATED and SLAVE are some possible port systems that modify in the events INITIALIZE, RS SLAVE, MASTER CLOCK SELECTED. The master offset value is the master's measured nanoseconds offset. This reduced from 1262284679078 to 970 showing successful synchronization with master change from UNCALIBRATED to SLAVE to port state. It's shown in fig. 9.

```
root@xilinx-zcu102-2018_3:~# ./ptp4l -i eth0 -m -s &
[1] 2603
ptp4l[3144.781]: selected /dev/ptp0 as PTP clock
ptp4l[3144.782]: port 1: INITIALIZING to LISTENING on INITIALIZE
ptp4l[3144.782]: port 0: INITIALIZING to LISTENING on INITIALIZE
root@xilinx-zcu102-2018_3:~# ptp4l[3152.039]: selected best master clock 000a35.fffe.00220
ptp4l[3157.137]: port 1: new foreign master 64006a.fffe.71fde6-1
ptp4l[3159.243]: selected best master clock 000a35.fffe.002201
ptp4l[3161.137]: selected best master clock 64006a.fffe.71fde6
ptp4l[3161.137]: port 1: LISTENING to UNCALIBRATED on RS_SLAVE
ptp4l[3162.136]: master offset 1262284969078 s0 freq      +0 path delay         0
ptp4l[3163.136]: master offset 1262284979028 s1 freq   +9949 path delay         0
ptp4l[3164.137]: master offset        970 s2 freq  +10919 path delay         0
ptp4l[3164.137]: port 1: UNCALIBRATED to SLAVE on MASTER_CLOCK_SELECTED
ptp4l[3165.137]: master offset       3288 s2 freq  +13528 path delay         0
ptp4l[3166.137]: master offset      -2120 s2 freq   +9107 path delay      3913
ptp4l[3167.137]: master offset        674 s2 freq  +11265 path delay      3466
ptp4l[3168.137]: master offset       3451 s2 freq  +14244 path delay      3019
ptp4l[3169.137]: master offset       1947 s2 freq  +13775 path delay      3019
ptp4l[3170.137]: master offset         88 s2 freq  +12500 path delay      3394
ptp4l[3171.137]: master offset      -1755 s2 freq  +10684 path delay      3770
ptp4l[3172.137]: master offset        963 s2 freq  +12875 path delay      3394
ptp4l[3173.137]: master offset       -909 s2 freq  +11292 path delay      3792
ptp4l[3174.137]: master offset       1422 s2 freq  +13350 path delay      3792
ptp4l[3175.137]: master offset        -87 s2 freq  +12268 path delay      3815
ptp4l[3176.137]: master offset      -1611 s2 freq  +10718 path delay      3833
ptp4l[3177.137]: master offset        777 s2 freq  +12623 path delay      3792
ptp4l[3178.137]: master offset       -734 s2 freq  +11345 path delay      3833
ptp4l[3179.138]: master offset       1601 s2 freq  +13459 path delay      3821
ptp4l[3180.138]: master offset         73 s2 freq  +12412 path delay      3872
ptp4l[3181.138]: master offset      -1552 s2 freq  +10809 path delay      3993
ptp4l[3182.138]: master offset        791 s2 freq  +12686 path delay      3993
ptp4l[3183.138]: master offset       -692 s2 freq  +11440 path delay      3993
ptp4l[3184.138]: master offset       1689 s2 freq  +13614 path delay      3942
ptp4l[3185.138]: master offset        411 s2 freq  +12842 path delay      3746
ptp4l[3186.138]: master offset      -1268 s2 freq  +11287 path delay      3942
ptp4l[3187.138]: master offset       1065 s2 freq  +13239 path delay      3942
ptp4l[3188.138]: master offset       -548 s2 freq  +11946 path delay      4060
ptp4l[3189.138]: master offset      -1927 s2 freq  +10402 path delay      3972
```

Fig. 9. Slave clock output

Over that hour, after we locked, we stayed within +/- 100 nanoseconds of the master offset. It's shown in figure. PTP application is precise clock synchronization in nanoseconds is successfully used hardware timestamping. It's shown in fig. 10.

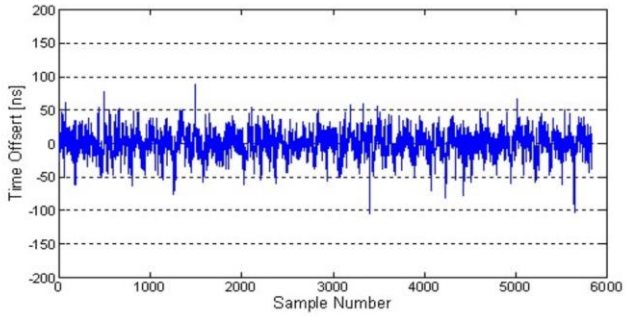

Fig. 10. Clock Synchronization test

VII. CONCLUSION

In this paper, we achieve under the FreeRTOS environment the hardware solution of GEM Ethernet in PTPd application based on IEEE1588v2 on the Xilinx ZynqMP UltraScale plus SoC platform, the program has high clock synchronized with high accuracy, excellent portability and scalability.

REFERENCES

[1] F.-Steinhauser, C.-Riesch, M.-Rudigier, "IEEE 1588 for time synchronization of devices in the electric power industry", in: Proceedings of International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication, Klaus, Austria, pp.1-6, Oct. 2010.

[2] Y. Peng, Q.-H. Luo, and Z.-Q. Liu, “An automatic evaluation system for IEEE1588 synchronization clock unit,” in: Proceedings of the Ninth International Conference on Electronic Measurement and Instruments, Beijing, China, pp.408-413, Aug. 2009.

[3] IEEE1588, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Standard, Jul. 2008.

[4] X.-Q. Li, Y. Chen, and S. Liang, “Improvement of precise time synchronization algorithm based on IEEE1588,” in: Proceedings of International Conference on Computer, Mechatronics, Control and Electronic Engineering, Changchun, China, pp.70-73, Aug. 2010.

[5] W. Jiang, and P. Robert, “Synchronizing device clocks using IEEE 1588 and Blackfin embedded processors,” Analog Dialogue, vol. 43, no 11, pp.1-5, Nov. 2009.

[6] Zhaoqing Liu, Dongxing Zhao, Min Huang, Yigang Zhang, “A Universal Method for Implementing IEEE 1588 with the 1000M Ethernet Interface”, 2016 IEEE AUTOTESTCON Anaheim, CA, USA, 12-15 Sept. 2016.

[7] Zhao, Hongkun, X. Wang, and Z. Yang. "Design and implementation of precision time synchronization system based on IEEE1588." In International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, 2011, Vol.47, pp, 610-613.

[8] Zynq UltraScale+ Device Technical Reference Manual UG1085 (v1.8) August 3, 2018.

[9] Gigabit Ethernet MAC (GEM) User Guide, ©2002 Cadence Design Systems, Inc. All rights reserved, Document Number: I-IPA01-0100-USR Rev 22 March 2010.

[10] Anirudha Sarangi, Stephen MacMahon and Upender Cherukupaly, “LightWeight IP (IwIP) Application Examples”, Xilinx, Vol. 5, No. 1, pp. 1-31, 2014.

[11] Cochran, R.; Marinescu, C.;, "Design and implementation of a PTP clock infrastructure for the Linux kernel," Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2010 International IEEE Symposium on, vol., no., pp.116-121, Sept. 27 2010-Oct. 1 2010.

[12] Meng Dong, Zhiliang Qiu, Weito Pan, Can Chen, Junxiang Zhang, Dong Zhang, “The Design and Implementation of IEEE 1588v2 Clock Synchronization System by Generating Hardware Timestamps in MAC Layer,” 2018 International Conference on Computer, Information and Telecommunication Systems (CITS), pp. 1-4, 2018.

[13] Priyanshu, Bhanu Pratap, and Radhey Shyam. “Analysis and Design of Precision Time Protocol System Based on IEEE1588 Standards” unpublised.