Atheros AR8031

1588v2 Precision Time Protocol

Application Note

Draft Version 1.00

11 Jan 2011

Table of Contents

Contents

Table of Contents 2

1. Overview 5

2. 1588 theorem brief 6

2.1 Concept 6

2.2 PTP time synchronization principle 7

Delay response mechanism: 8

Peer delay mechanism: 9

2.3 Synchronous Ethernet 9

3. Atheros AR8031 Features 11

3.1 AR8031 1588v2 clock structure 12

RTC structure 13

RTC work mode 14

3.2 How to adjust the RTC 14

Directly write 14

Adjust counter step 15

3.3 PTP Parser 15

Generate interrupt 15

Generate timestamp 16

Capture message information 17

Provide correction field 18

Special work modes 19

4. Application solution 22

5. Atheros demo application with AR8031 25

5.1 System Architecture 26

5.2 System working mechanism 27

5.3 Usage 28

6. Test results 31

7. Conclusion 32

Revision History

Date	Rev	Description
March 14, 2011	1.00	Draft release

1. Overview

This application note summarizes 1588v2 basic concept and explains precision time protocol (PTP) synchronization mechanisms. Then it describes Atheros AR8031 1588v2 features and gives some typical application examples with excellent test results.

1588v2 typical application

- Time-sensitive telecommunication services that require precision time synchronization between communicating nodes
- Industrial network switches that synchronize sensors and actuators over a single wire distributed control network to control an automated assembly process
- Powerline networks that synchronize across large-scale distributed power grid switches to enable smooth transfer of power
- Test/measurement devices that must maintain accurate time synchronization with the device under test in many different operating environments
- Home networked Audio/Video appliances that distribute timing information at each media playback device to provide a precise reference point for streaming multimedia, thus enabling a convergence to Ethernet as a viable consumer electronics interconnect.

All these applications require precise clock synchronization between devices with accuracy in the sub-microsecond range. It is a remarkable feature of the IEEE 1588 that this synchronization precision is achieved via the regular Ethernet connectivity using standard Ethernet frames. This must be assumed to be non real-time capable.
This solution allows nearly any device of any performance to participate in high precision synchronized networks that are simple to operate and configure.

1588v2 Telecom network application

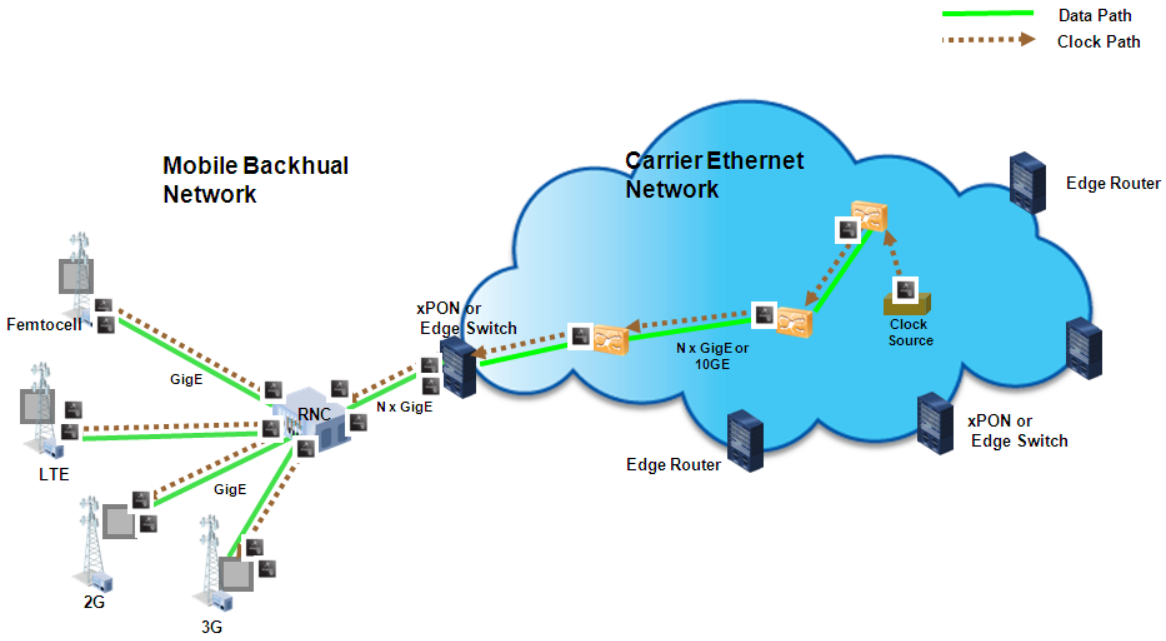


Figure 1—telecom network application

Traditional base station gets precise time information from TDM network. As the TDM network begins to convert to packet network, base station must use expensive GPS to get time information. 1588v2 provides a cheap and precise time synchronization solution. As figure above shows, all the base stations can synchronize to the master clock in the control center. Also 1588v2 can back up for the base station GPS clock.

2. 1588 theorem brief

2.1 Concept

Base on the Precision Time Protocol (PTP) work mode and position in the network, 1588v2 defines four clock modes:

Ordinary clock: A clock that has a single PTP port in a domain and maintains the timescale used in the domain. It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock. A base station which supports 1588v2 is a typical ordinary clock example. It may get precise time of day from a GPS as a master clock in a domain. Or it may serve as a slave clock synchronizes with another base station through a packet network.

Boundary clock: A clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain. It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock.
End-to-end transparent clock: A transparent clock that supports the use of the end-to-end delay measurement mechanism between slave clocks and the master clock.
Peer-to-peer transparent clock: A transparent clock that, in addition to providing Precision Time Protocol (PTP) event transit time information, also provides corrections for the propagation delay of the link connected to the port receiving the PTP event message.

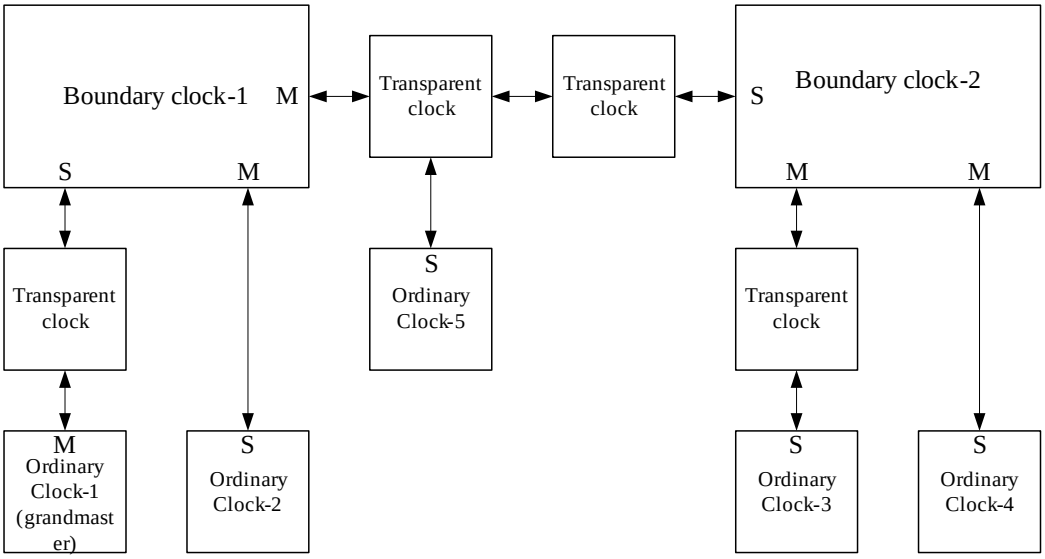


Figure 2—clock structure

2.2 PTP time synchronization principle

1588v2 protocol uses two basic mechanisms to synchronize different clocks (get **time of day offset**) and get **propagation delay** between clocks in the domain. The two basic mechanisms are delay request-response mechanism and peer delay mechanism.

Delay response mechanism:

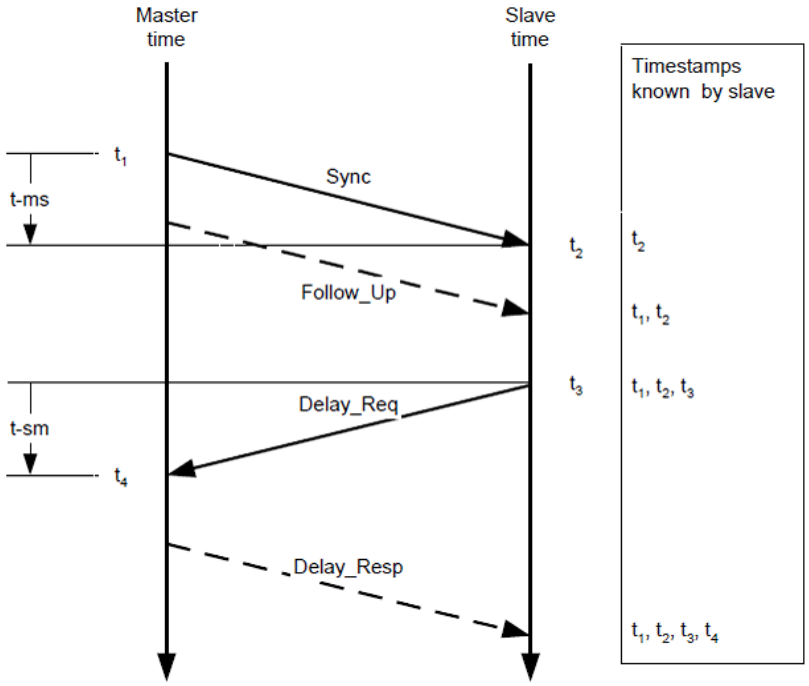


Figure 3—delay response mechanism

Aim to let the slave port get the time offset between master and slave.
-Master starts the process through sending Sync message.
-Master side capture the time (t_1) Sync message out of port.
-When Sync message arrives at slave port, the time t_2 is captured.
In two step mode, the t_1 is sent to slave port by Follow_Up message.
In one-step clock mode, the t_1 is sent by Sync message which need the master mode support this special mode.
-Then slave port sends the Delay_Req message in t_3.
-Delay_Req message arrives at master port in t_4.
-After that t_4 will be sent back to slave port by Delay_Resp message.
Now slave port got four time stamps: t_1, t_2, t_3, t_4
Assume the two direction propagation delay between master and slave port is the same.
Link delay= $[(t_2-t_1) + (t_4-t_3)]/2$
Time offset= $t_2-t_1-t_{link\_delay}$
If the propagation delay is not the same, use correction field to compensate.

Peer delay mechanism:

It measures the port to port propagation time.

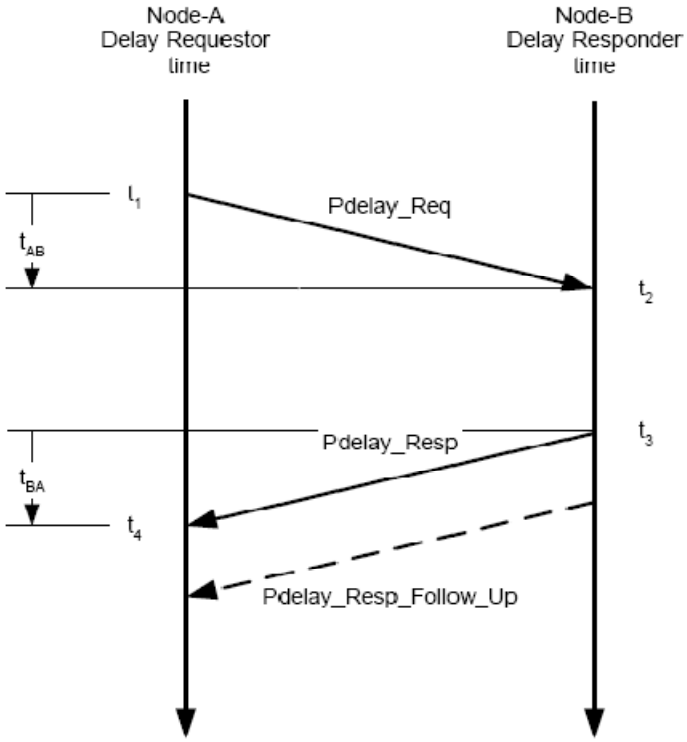


Figure 4—peer delay mechanism

t_1 and t_4 are measured by Node-A. t_2 and t_3 are measured by Node-B. (t_2-t_3) is calculated by Node-B and sent back to Node-A.

Node-A can get the Link delay= $[(t_2-t_1) + (t_4-t_3)]/2= [(t_4-t_1) + (t_2-t_3)]/2$

In two-step clock mode, need to send (t_2-t_3) back to Node-A through Pdelay_Resp_Follow_Up message.

In one-step clock mode, (t_2-t_3) is sent back to Node-A by Pdelay_Resp message by which bandwidth is saved.

2.3 Synchronous Ethernet

1588v2 periodically calculates time offset to compensate slave clock. The frequency of exchanging 1588v2 packets must be limited to save bandwidth. If the master and slave RTC reference clocks are free running, high precision time synchronization can not be achieved.

Synchronous Ethernet is a physical layer syntonization technology. Synchronization of the physical layer of a link means that the receiver can lock onto the frequency of the transmitter.
For 1000BASE-T, the slave mode PHY recovers the physical frequency of the master mode link partner. For 100BASE-TX, PHY receiver recovers the physical frequency of the link partner transmitter. Since PHY local reference clock is the line side packet transmit reference clock except 1000BASE-T slave mode PHY (which use the recovered clock to transmit). So PHY local reference clock frequency can be transferred to link partner by SYNC-E.

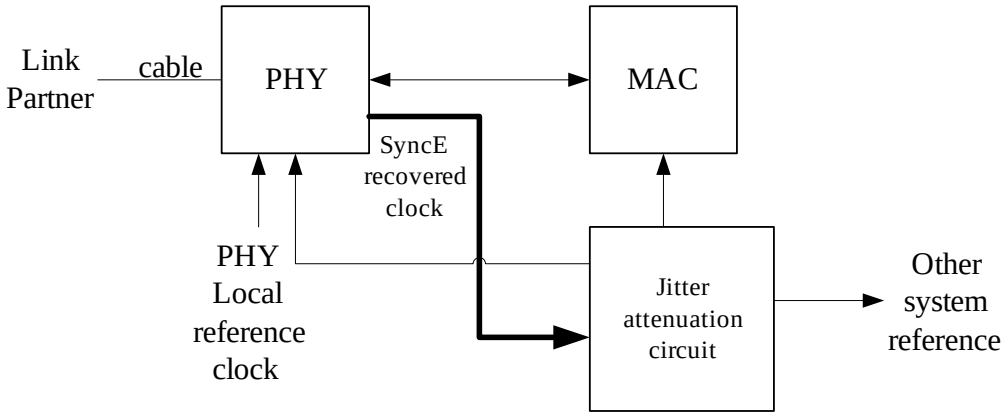


Figure 5—SYNC-E typical application

Note: SYNC-E can recover link partner frequency with high precision. But recovered clock jitter is hard to control and can be accumulated. Such as the figure above shows that SYNC-E recovered clock can be used as other system reference. If it is used as reference clock of another PHY, the clock can be passed to other equipment by SYNC-E. For typical application a jitter attenuation circuit is used to improve the jitter and also provide holdover protection.

If this recovered SYNC-E clock is used as 1588v2 RTC reference clock. The frequency difference can be eliminated. Sub-10ns synchronous precision can be achieved with typical 125MHz clock.

AR8031 supports a frequency controlled SYNC-E recovered clock output for system reference and control. See register MMD7 0x8016[4:2] description in datasheet for detail. And it supports setting MMD7 0x8012[7] =1'b1 to select SYNC-E recovered clock for 1588v2 reference clock inside the PHY chip, detail in clock structure chapter.

3. Atheros AR8031 Features

AR8031 is ATHEROS's 4th generation, single port 10/100/1000Mbps tri-speed Ethernet physical transceiver. AR8031 supports both IEEE 1588v2 and Synchronous Ethernet to offer a complete time synchronization solution to meet the next generation network requirements. It supports clock synchronization between slave and master by the exchange of Precision Time Protocol (PTP) packets. Supports IEEE 1588v2 by offering a 1588 packet parser, accurate time-stamping and insertion to support both one-step and two-step clock modes.

AR8031 is a physical device at the lowest layer of a PTP node, can provide the most precise timestamp compared with software, MAC or FPGA scheme with 1588v2 PTP protocol.
A typical Ethernet device diagram is shown below:

Figure 6—Ethernet device diagram

1588v2 is used to synchronize two PTP clocks the **time of day** and get **propagation delay**. Refers to chapter2.2, the synchronization precision affects by the symmetry of transmit and receive propagation delay and the precision of transmission and receipt timestamp generated. The delay between the timestamp is generated and the message is sent out should be corrected. And the asymmetry also should be corrected.
Any uncontrolled buffer or delay in the packets transfer and process path will affect the synchronization precision base on the 1588v2 PTP protocol.

Software begins the PTP protocol by sending PTP messages. If the timestamp is added in the software layer, it is hard to control the message sent out with a precise timestamp because of controller process time, interrupt response time and thread scheduling time, etc. Different process ability of master and slave controllers will also affect the asymmetry. The MAC and PHY side buffer delay and transfer delay also affect the performance.

If 1588v2 timestamp is added in MAC or FPGA part, it needs to consider the transfer delay to line side. PHY TX buffer delay can not be bypassed.

AR8031 integrates a RTC to provide local time of day and a packet parser to distinguish 1588v2 packets, generate time stamp and do special process base on message type. With AR8031 hardware assistant, system with 1588v2 can get nanosecond level synchronization precision.

3.1 AR8031 1588v2 clock structure

AR8031 integrates a RTC (real time counter) which can use (1) the local 125MHz reference clock, (2) recovered clock from Ethernet received data or (3) external input 1588 reference 50-125MHz clock to work.
Figure below shows the clock structure.

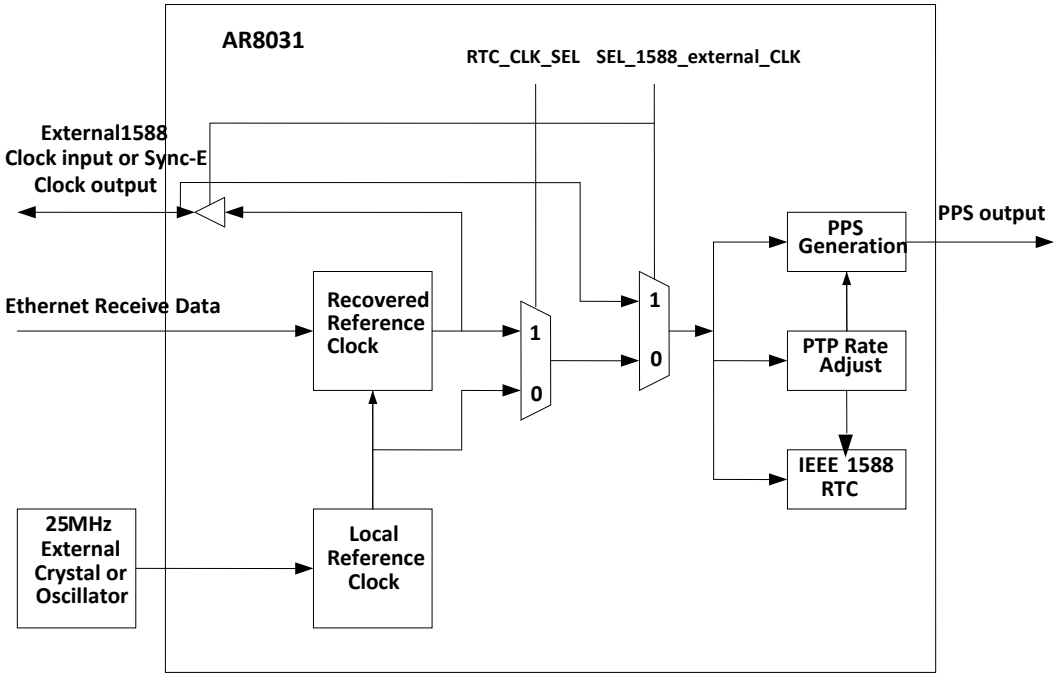


Figure 7—AR8031 clock structure

Notes:

1. External 1588 clock input share the same chip pin with SYNC-E clock output.
   Set register MMD7 0x8017[11] =1'b1 to select external 1588 clock input as reference and the pin works as an input. Set register MMD7 0x8017[11] =1'b0 to select SYNC-E clock output. (Default)

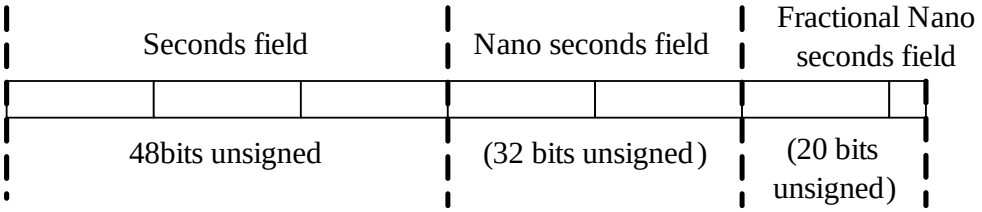
This control bit refers to the “SEL_1588_external_CLK” in the figure above.

2. Without using the external 1588 clock input, select the other two RTC reference clocks by:
   Set MMD7 0x8012[7] =1’b1 to select SYNC-E recovered clock.
   Set MMD7 0x8012[7] =1’b0 to select local 125MHz clock as reference. (Default)
   This control bit refers to the “RTC_CLK_SEL” in the figure above.

RTC structure

RTC (real time counter) controls the local **time of day.**

It has 48bits seconds field, 32 bits nanosecond field and 20bits more for fractional nanosecond field.



The real time can be read from AR8031 register through MDC/MDIO interface.

MMD3 register table

Address offset	Definition	Function	R/W
0x803D[15:0]	Real Time[79:64] current time, seconds	Real Time Clock	Read Only
0x803E[15:0]	Real_Time[63:48]	Real Time Clock	Read Only
0x803F[15:0]	Real_Time[47:32]	Real Time Clock	Read Only
0x8040[15:0]	Real_Time[31:16] Nano seconds	Real Time Clock	Read Only
0x8041[15:0]	Real_Time[15:0] Nano seconds	Real Time Clock	Read Only
0x8042[15:0]	RTC_Frac_Nano[19:4] fractional nano seconds	Real Time Clock	Read Only
0x8043[3:0]	RTC_Frac_Nano[3:0] fractional nano seconds	Real Time Clock	Read Only

RTC work mode

The second field counts from zero until full saturated, then wrap around to zero.
The nano second field counts from zero to 1S then wrap around to zero. The counter basic step can be set by register 0x8036 and 0x8037.
Example: If using local PLL 125MHz reference clock for RTC, the clock cycle is 8ns. We need to set the counter step to 8ns in register below. Write 6'b001000 to 0x8036[15:10] (nana second field), and write 0x8036[9:0] and 0x8037[9:0] to 0x0 to set fractional nanosecond field. This is called 6.20 format.
MMD3

Address offset	Definition	Function	R/W
0x8036[15:0]	increase_value[25:10] real time increase value every cycle, 6.20 format	Real Time Clock	Write/Read
0x8037[9:0]	Increase_value[9:0]	Real Time Clock	Write/Read

3.2 How to adjust the RTC

Once getting the **time of day offset** between master and slave clock with 1588v2 mechanism, the offset can be used to adjust slave clock.

AR8031 supports two mechanisms to adjust the time of day in RTC with 1588v2 protocol.

One is to write the offset directly to the PHY, the other is to adjust the increase value of RTC counter every cycle.

Directly write

Write the time offset to slave RTC directly.
MMD3

Address offset	Definition	Function	R/W
0x8038[15:0]	Nano_offset[31:16] slave offset from master, Nano Second field	Real Time Clock	Write/Read
0x8039[15:0]	Nano_offset[15:0]	Real Time Clock	Write/Read
0x803A[15:0]	Sec_offset[47:32]	Real Time Clock	Write/Read
0x803B[15:0]	Sec_offset[31:16]	Real Time Clock	Write/Read
0x803C[15:0]	Sec_offset[15:0]	Real Time Clock	Write/Read

-Divide the offset to second field offset and nanosecond field offset. The fractional nanosecond offset can be eliminated.
-Translate the offset value from signed decimal to signed two's complement and write to the register in the table above.
-This offset does not take effect immediately after writing. Write 0x1 to MMD3 0x8044 which is written only. Then the offset will be written to the RTC.

Each 1588v2 clock should be set with a local system time before synchronization process. This can be achieved by writing offset register above.

Adjust counter step

Directly write cannot give the best synchronization precision and stability. AR8031 provides a way to change the increase value each cycle. This method gives step adjustment which is a smooth and precise way to adjust the time offset. This method is also a way to compensate the frequency skew between the master and slave node.
The frequency skew of the master and slave RTC reference clock is the main reason for time offset fluctuation when the network structure is stable.

Observe the time offset changing reference to time, information about the frequency difference between the two clocks can be got.
Adjust the increase value each cycle to minimize the frequency skew will get better time offset performance.

3.3 PTP Parser

Generate interrupt

AR8031 integrates 1588 Packet Parser in both RX and TX direction which can distinguish 1588 packets from other packets. When receives 1588 packet from line side or MAC side, AR8031 will generate an interrupt. There are two interrupt control bits in register 0x13.

Address offset	Definition	Function	R/W
0x13[3]	INT_RX_PTP	Received RX direction 1588 packet interrupt	Self clear after reading
0x13[2]	INT_TX_PTP	Received TX direction 1588 packet interrupt	Self clear after reading

Generate timestamp

1588v2 defines event messages and general messages which are used to communicate between PTP clocks to get time information.
Event messages are timed message in which an accurate timestamp is generated at both transmission and receipt. It includes:
- Sync message
- Delay_Req message
- Pdelay_Req message
- Pdelay_Resp message

General messages do not require accurate timestamps.

Detail message definition please reference to 1588v2 standard.

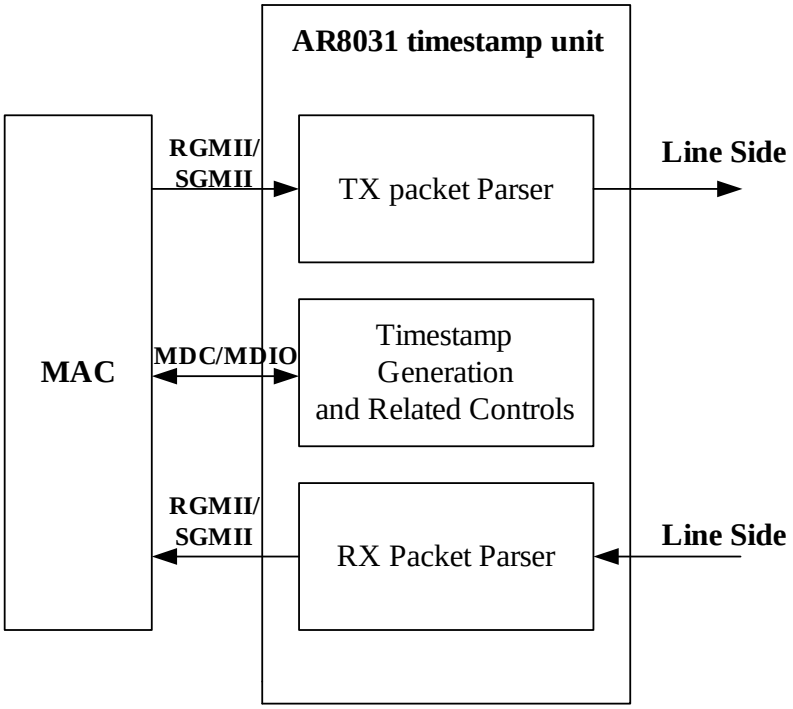


Figure 8—AR8031 timestamp unit

When event message is through PHY, timestamp will be generated which are buffered in register below.

Address offset	Definition	Function	R/W
Time-stamp Unit—RX			
0x8019[15:0]	RX_time_stamp [79:64]	Timestamp	Read Only
0x801A[15:0]	RX_time_stamp [63:48]	Timestamp	Read Only
0x801B[15:0]	RX_time_stamp [47:32]	Timestamp	Read Only
0x801C[15:0]	RX_time_stamp [31:16]	Timestamp	Read Only
0x801D[15:0]	RX_time_stamp [15:0]	Timestamp	Read Only
0x801E[11:0]	RX_frac_nano[19:8]	Timestamp	Read Only
0x801F[7:0]	RX_frac_nano[7:0]	Timestamp	Read Only
Time-stamp Unit-TX			
0x8026[15:0]	TX_time_stamp [79:64]	Timestamp unit	Read Only
0x8027[15:0]	TX_time_stamp [63:48]	Timestamp unit	Read Only
0x8028[15:0]	TX_time_stamp [47:32]	Timestamp unit	Read Only
0x8029[15:0]	TX_time_stamp [31:16]	Timestamp unit	Read Only
0x802a[15:0]	TX_time_stamp [15:0]	Timestamp unit	Read Only
0x802B[11:0]	TX_frac_nano[19:8]	Timestamp	Read Only
0x802C[7:0]	TX_frac_nano[7:0]	Timestamp	Read Only

Capture message information

Besides generating interrupt, in order to help 1588v2 protocol stack to ensure the timestamps are associated with the correct message, PHY will capture and store additional PTP message information such as sequence ID, source port identity and message type.
In MMD3 register:

Address offset	Definition	Function	R/W
Time-stamp Unit—RX			
0x8013[15:0]	RX_sequenceId	sequenceId	Read Only
0x8014[15:0]	RX_sourcePortIdentity [79:64]	sourcePort Identity	Read Only
0x8015[15:0]	RX_sourcePortIdentity [63:48]	sourcePort Identity	Read Only
0x8016[15:0]	RX_sourcePortIdentity [47:32]	sourcePort Identity	Read Only
0x8017[15:0]	RX_sourcePortIdentity [31:16]	sourcePort Identity	Read Only

0x8018[15:0]	RX_sourcePortIdentity [15:0]	sourcePort Identity	Read Only
0x801E[15:12]	RX_messageType[3:0]	Message type	Read Only
Time-stamp Unit-TX			
0x8020[15:0]	TX_sequenceId	sequenceId	Read Only
0x8021[15:0]	TX_sourcePortIdentity [79:64]	Timestamp unit	Read Only
0x8022[15:0]	TX_sourcePortIdentity [63:48]	Timestamp unit	Read Only
0x8023[15:0]	TX_sourcePortIdentity [47:32]	Timestamp unit	Read Only
0x8024[15:0]	TX_sourcePortIdentity [31:16]	Timestamp unit	Read Only
0x8025[15:0]	TX_sourcePortIdentity [15:0]	Timestamp unit	Read Only
0x802B[15:12]	TX_messageType[3:0]	Message type	Read Only
0x802D[15:0]	Origin_correction[63:48]	Correction field	Write/Read
0x802E[15:0]	Origin_correction[47:32]	Correction field	Write/Read
0x802F[15:0]	Origin_correction[31:16]	Correction field	Write/Read
0x8030[15:0]	Origin_correction[15:0]	Correction field	Write/Read
0x8031[15:0]	Ingress_trig_time[51:36]	Timestamp unit	Write/Read
0x8032[15:0]	Ingress_trig_time[35:20]	Timestamp unit	Write/Read
0x8033[15:0]	Ingress_trig_time[19:4]	Timestamp unit	Write/Read
0x8034[3:0]	Ingress_trig_time[3:0]	Timestamp unit	Write/Read
0x8035[15:0]	TX_latency[15:0] in nanoseconds	Timestamp unit	Write/Read

Provide correction field

The Origin_Correction field, ingress_trig_time field and Tx_Latency field are used for one-step clock mode.

Origin_Correction field is used for time correction of typical application.

For ordinary/boundary one-step clock, write the asymmetry information about ingress and egress latency.

For end-to-end transparent one-step clock, write the correction field in PTP Event Message of ingress port.

For peer-to-peer transparent one-step clock, write the correction field in PTP Event Message of ingress port and add link delay before enter ingress port.
AR8031 will automatically add related correction into the correctionField of PTP messages.

Ingress_trig_time field is a timestamp unit.
For ordinary/boundary one-step clock, it is the nanosecond and fractional nanosecond field of Pdelay_req RX timestamp works in peer delay mechanism.
For transparent one-step clock, it is the nanosecond and fractional nanosecond field of RX timestamp of ingress port. The timestamp is read from ingress port and written by controller.AR8031 will compare the ingress port and egress port timestamp to get the residence time.

Tx_Latency field helps correct the time difference between the moment timestamp is generated and when the packet is transmitted to the physical media.

Special work modes

Support add timestamp into packet

From above description, controller through the MDC/MDIO interface to read and write timestamp and information. It requires a high speed MDC/MDIO interface. Also it needs the controller to handle the PHY interrupt frequently. If not, the information will be overlapped by a new 1588 event message. It restricted the bandwidth of CPU.
AR8031 provides a high speed MDC interface up to 25MHz and a way to add the timestamp into the incoming packet. Then timestamp can be transferred to the 1588v2 software stack through the in-band GE interface. It controlled by register MMD3 0x8012[1].
Of course AR8031 will recalculate the CRC of the received packet and fill back to the packet to replace the original one.
For event message, the attached timestamp is RX timestamp.
For general message, the attached timestamp is the TX timestamp of the associated event message if existed.

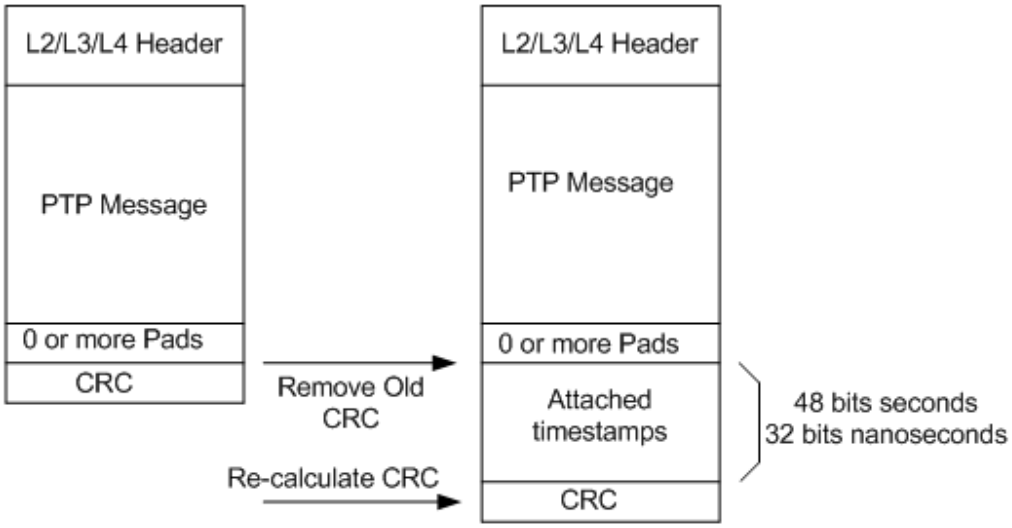


Figure 9—inband timestamp

Support one-step clock mode

The delay request-response mechanism and peer delay mechanism mentioned in chapter2.2 need to use event messages (Sync, Delay_Req, Pdelay_Req, Pdelay_Resp) and general messages (Follow_Up,Delay_Resp,Pdelay_Resp_Follow_Up) to communicate.

Two Step Clock

A clock that provides time information using the combination of an event message and subsequent general message, which means that the follow-up message carries a precise estimate of time the Sync message was placed on the PTP communication path.

One Step Clock:

A clock that provides time information using a single event message.

The complete Time Synchronization System needs to support both modes.
AR8031 supports all one-step and two-step clock modes defined in register MMD3 0x8012

Address offset	Definition	Function	R/W
0x8012[2:1]	CLOCK_MODE	00: ordinary/boundary two-step clock. (default) 01: ordinary/boundary one-step clock.	R/W

		10: transparent two-step clock. 11: transparent one-step clock.	

The behavior of one step clock deals with PTP messages is a little different from that of two step clock.

Sync message

As an ordinary/boundary one step clock:
AR8031 will add the sum of egress timestamp and Tx_Latency into the originTimestamp field of Sync message excluding any fractional nanoseconds.
The fractional nanoseconds of the sum plus the Orign_Correction will be added into correctionField of Sync message.
Recalculate the CRC of the changed Sync message before sending it out.

As a transparent one step clock:
Software needs to write the ingress_trig_time (the ingress timestamp of Sync message enter the clock port), Original_Correction (Sync message ingress correctionField value add asymmetry correction) and Tx_Latency (other corrections).
AR8031 will correct the egress timestamp with Tx_Latency. Then it uses the nanosecond field of the corrected egress timestamp and ingress_trig_time to get the residence time between the ingress and egress ports. The residence time will be added into correctionField of Sync message.
Recalculate the CRC of the changed Sync message before sending it out.

Delay_Req message

As an ordinary/boundary one step clock:
AR8031 does not modify this message.

As a transparent one step clock:
The detail process is similar as that of Sync message. AR8031 will add the residence time between the ingress port and egress port to the correctionField of this message.
Recalculate the CRC of the changed message before sending it out.

Pdelay_Req message

As an ordinary/boundary one step clock:
AR8031 does not modify this message.

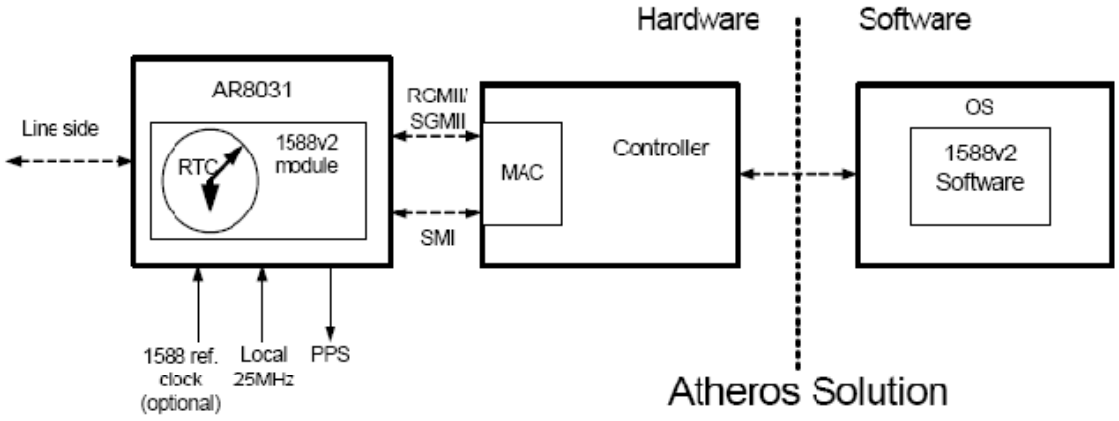
As a transparent one step clock:
The detail process is similar as that of Sync message. AR8031 will add the residence time between the ingress port and egress port to the correctionField of this message. Recalculate the CRC of the changed message before sending it out.

Pdelay_Resp message
As an ordinary/boundary one step clock:
Controller needs to write Origin_Correction, ingress_trig_time and Tx_Latency field. The ingress_trig_time is the nanosecond field of related Pdelay_Req ingress timestamp. AR8031 will calculate the difference between Pdelay_Resp egress timestamp and Pdelay_Req ingress timestamp and write it into correctionField of Pdelay_Resp message. Also it recalculates the CRC of the changed message before sending it out.

As a transparent one step clock:
The detail process is similar as that of Sync message. AR8031 will add the residence time between the ingress port and egress port to the correctionField of this message. Recalculate the CRC of the changed message before sending it out.

4. Application solution

1. AR8031 works as an ordinary/boundary PTP slave/master clock
   AR8031 can perfectly work as an ordinary and boundary PTP slave clock.

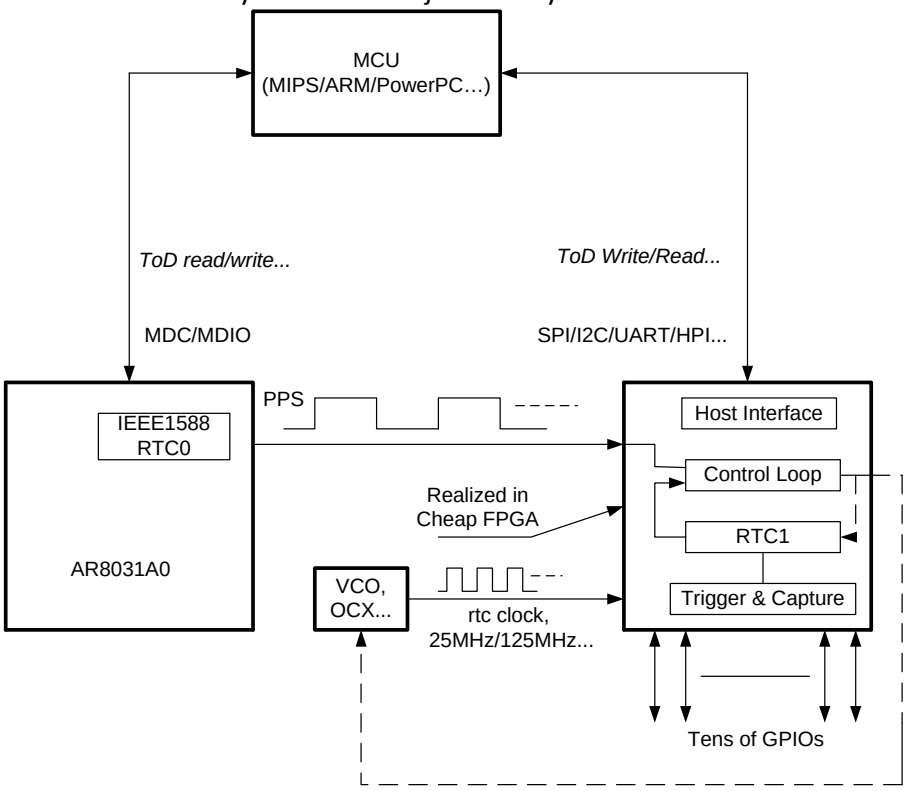


AR8031 can be designed as a normal PHY in 1588v2 supported system. It includes a RTC using local or recovered clock as RTC reference clock. It transfers the PTP packets to

controller through RGMII/SGMII MAC interface. Line side it supports copper and fiber both. When it receives or transmits 1588v2 PTP packets, it will generate interrupt to notice controller to get the time and packets info. And it supports controller to write time info and adjust the RTC clock base on 1588v2 PTP. As a PTP slave clock, it uses 1588v2 mechanism to synchronize with PTP master.

AR8031 also can work as a master clock which supports µs level precision. Because the master time of day is written by controller through MDC/MDIO interface which limits the time precision. If more precise master time is needed, please reference to scheme No.3 below as a grandmaster.

2. AR8031 works in system clock adjust and system control.



1. RTC1 synchronized with RTC0 through PPS, accuracy < 10ns

2. RTC1 tick rate is adjustable, via virtual frequency (increment value) or VCO output

Figure 10—control system application

AR8031 provides a PPS clock output as a system reference. PPS is a 1 second period clock signal which is synchronous with internal RTC.
Some industrial control system needs to use 1588v2 PTP to synchronize local RTC then to control local system. Figure above shows a typical application solution.
The PPS is sent to an external I/O extend device such as a FPGA which has a RTC inside. Controller system interface such as MDC/MDIO, UART, SPI helps synchronize basic time of day. The PPS can help FPGA RTC synchronize with AR8031 RTC with nanosecond precision.

3. AR8031 works as grandmaster

AR8031 can work perfectly as an ordinary master/slave clock. If system needs a high precision time of day grandmaster clock, a solution is provided below.

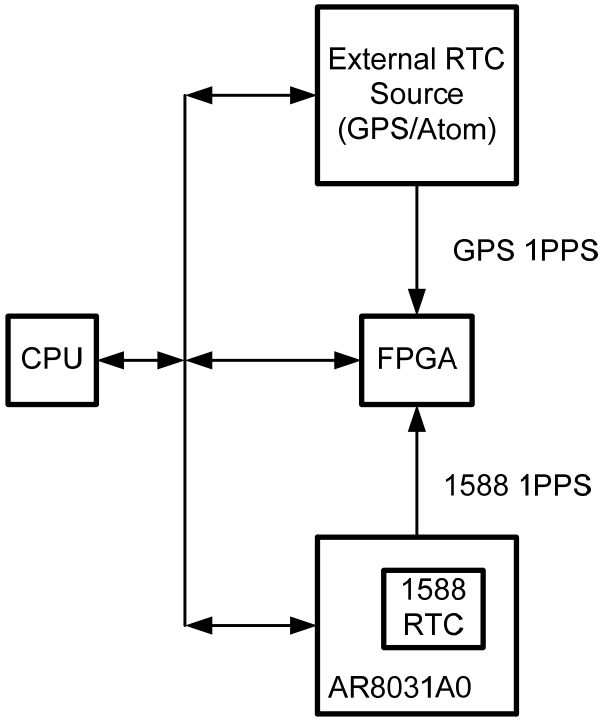


Figure 11—AR8031 grandmaster application

As a grandmaster clock the time of day is normally from GPS. GPS will send out a high precision PPS signal. As 8031 provides a PPS clock output. System can use these two PPS to get the time difference back to adjust AR8031 internal RTC. Detail adjust method can refer to “how to adjust the RTC clock” chapter.

4. AR8031 works in boundary/transparent clock

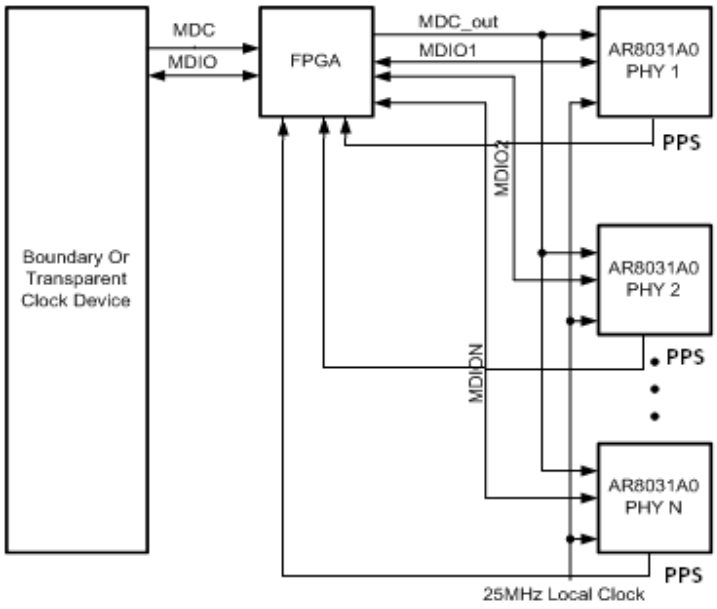


Figure 12—AR8031 transparent clock application

The boundary or transparent clock device all ports should use the same reference clock and with the same time of day. The PPS signal also should be collected to get a more precision time synchronization of each port.

5. Atheros demo application with AR8031

Atheros develops one 1588v2 demo application who works together with the demo board consists of two 8031 PHYs, FPGA and other necessary components. The demo application is used to show the PTP implementation on Atheros 8031 PHY and verify the inter- operability between Atheros 8031 PHY and the 3rd-party 1588 devices.
It is developed base on one open source 1588 daemon software. It runs on the desktop PC, and communicates with the demo board via RS232 port. Refer to Figure 1 for the topology.

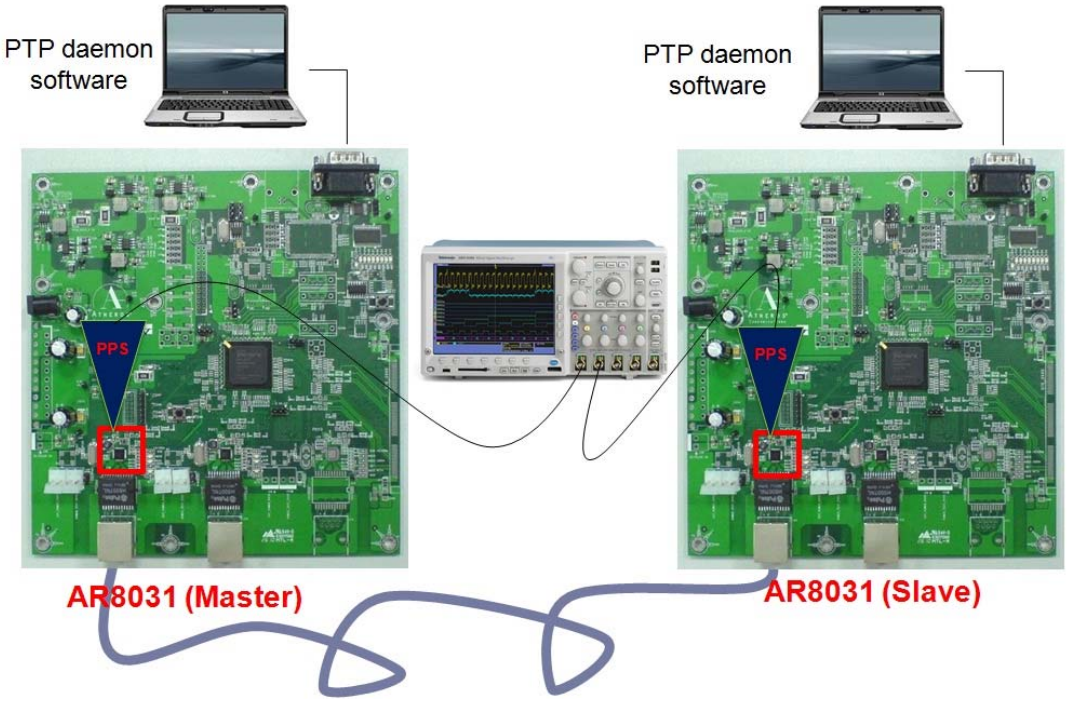


Figure 13---Connection diagram of demo board

Detail information of demo board please reference to demo board user guide.

5.1 System Architecture

The demo application is developed in Microsoft Windows, and the porting to Linux is ongoing.

This application is divided into four layers:

1. UI layer. It is responsible for processing the interaction between the software and the user.
2. Protocol layer. It is responsible for handling 1588 protocol.
3. UDP/IPV4/MAC simulation layer. It is responsible for simulating the underlying transporting layer for 1588 PDUs.
4. Communication layer and physical device layer. It constructs/sends, receives/analyses the command packets to/from the FPGA lying on the demo board via RS232 port, then dispatch to the corresponding handler.

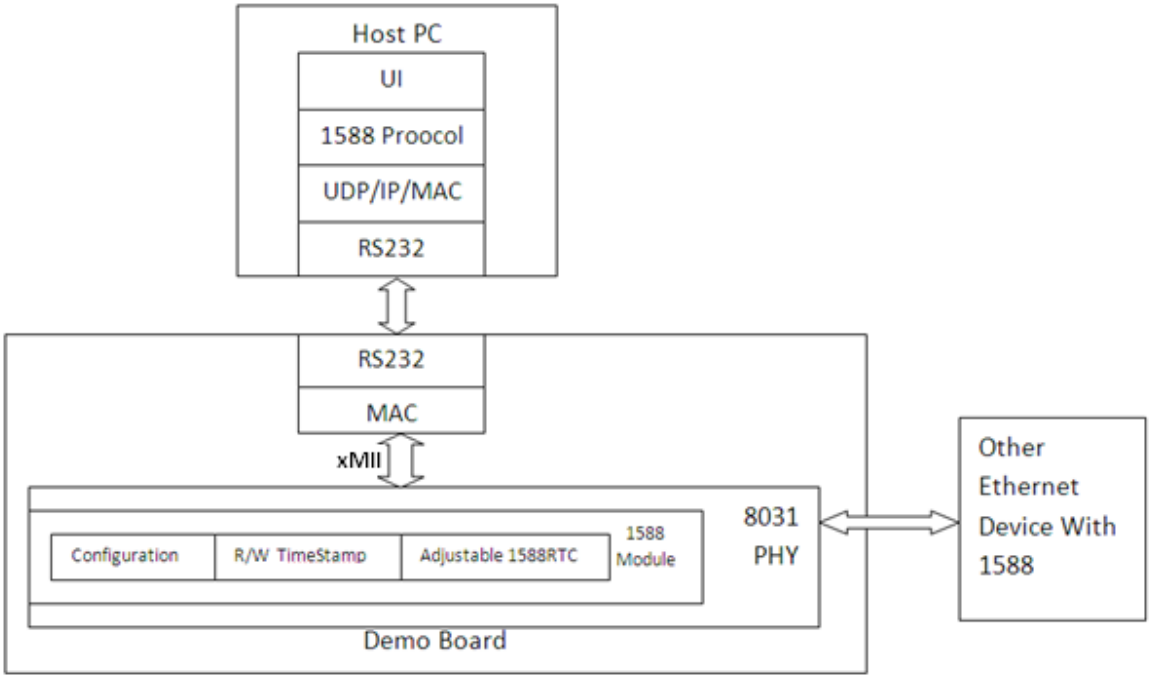


Figure 14---Architecture of Demo Application

5.2 System working mechanism

In the following section, we want to introduce how the demo application works together with the 8031PHY lies on the demo board to act as an ordinary/boundary clock. Assume they use delay response mechanism as described in figure 3.

First, we need configure 1588 configuration registers in 8031 PHY via MDIO operations according to the role of the Demo System.

Case 1: The demo system acts as a slave clock, and the other 1588 device acts as a master clock.

Step 1: The demo board receives a Sync message, 8031 PHY generates precise timestamp *t2* for it. When the demo application get the Sync message and need *t2*, it can retrieve *t2* via MDIO operations from the underlying 8031 PHY. In one-step mode, the demo application extract *t1* from received Sync message, in two-step mode, the demo application waits for the subsequent Follow_Up message. Once it gets the exact Follow_Up message, it extracts *t1* from this message.

Step 2: The demo application construct a Delay_Req message, and 8031 PHY will generate a precise timestamp *t3* for it. The demo application should retrieve *t3* via MDIO from PHY for its calculation.

Step 3: The master receives the Delay_Req message, it sends back the Delay_Rsp message with *t4*, where *t4* is the exact time that it receives Delay_Req.
Final Step: The demo application extracts *t4* from the Delay_Rsp message sent by the master. Now the demo application can calculate OffsetFromMaster and MeanPathDelay and write to the adjustable 1588 RTC registers.
In this case, the slave who use 8031 PHY can retrieve the most precise timestamp for *t2 and t3* from 8031 PHY for its calculation and improve the accuracy of the OffsetFromMaster and MeanPathDelay greatly.
Case 2: The demo board acts as a master clock, and the other 1588 device acts as a slave clock.
Step 1: The demo application starts to send Sync message, 8031 PHY will generate a precise timestamp t1 for this message. In one-step mode, 8031 PHY will re-construct this Sync message with *t1* and re-calculate CRC for this message. In two-step mode, 8031 PHY will keep this message, and the demo application need retrieve *t1* via MDIO from 8031 PHY and put it into the subsequent Follow_Up message.
Step 2: The slave clock gets Sync/Follow_Up message, it sends a Delay_Req message. The demo board receives this message, and 8031 PHY generates a timestamp *t4* for it. The demo application retrieves *t4* via MDIO operations from the underlying 8031 PHY, put it into Delay_Rsp message then sends it back to the slave clock.
In this case ,the master clock who uses 8031 PHY can retrieve the most precise timestamp for *t1* and *t4* from 8031 PHY for the slave clock's calculation so that the slave clock can improve the accuracy of the OffsetFromMaster and MeanPathDelay greatly.

5.3 Usage

The demo application can be configured to be any role with the configurations combined with the following columns in table 1. The user can configure it according to the customers' requirements and the external environment.
The demo application provides two kinds of command line styles:

1. Ptp_cmd –a demo_ip/demo_mac [-u r_ip/r_mac] [-g] [-n] [-e][-o]

	Role	1588 Clock source	Transporting layer	Synchronization Clock mode	Communication mode

	Slave only/Role decided by BMC	SyncE/Free running Clock	MAC or UDP/IPV4	One-Step/Two-Step Clock	Unicast/Multicast
Command line Parameter	-g	-n	-e	-o	-u r_ip/r_mac

Table 1

Description of Command parameters:

-g: Set the demo application as a slave only clock. The demo application's role will be free if the command line does't contain this option.

-n: Set the 1588 clock source as the clock recoved by Synchronous Ethernet. The demo application will use free running clock if the command line does't contain this option.

-e: Set the Transporting layer as the MAC layer. The demo application will use UDP/IPV4 if the command line does't contain this option.

-o: Set Synchronization Clock mode as One-Step clock. The demo application will use Two-Step clock if the command line does't contain this option.

-u: r_ip/r_mac: Set the Communication mode as Unicast Mode. The subsequent argument following with –u is the Remote IP address/Remote Mac address.

-x: Shutdown the 1588 application.

-a: Set the demo application's IP address and Mac address.

For example: We want to configure the demo board to act as a slave only clock, and the clock source is Sync-E, Transporting Layer is MAC layer, Synchronization Clock mode is one-step clock ,communication mode is Multicast, we need type one command in the UI input interface: ptp_cmd -g -n -e -o -a 10.0.0.66/11.22.33.44.55.66.

Take another example: most of the configurations are like the above example, but the Transporting layer is UDP/IPV4, the command should be: ptp_cmd -g -n -o -a 10.0.0.66/11.22.33.44.55.66.

Here, we gave the user two snapshots of 1588 demo application. Please refer to Figure 2 and Figure 3.

2. Ptp demo_mac [type]

   Where demo_mac is the last 3 significant bytes of mac address of the demo board.

   The meaning of type:

   0: one-step clock, Use Sync-E recovered clock as the 1588 clock source.

1: two-step clock, Use Sync-E recovered clock as the 1588 clock source.
2: two-step clock, Use free running clock as the 1588 clock source.
3: one-step clock, Use free running clock as the 1588 clock source.
For example, “ptp 30 2” means the demo board use free running clock as the clock source, and the communication mode is two-step clock mode. The last 3 significant byte of MAC address of the demo board will be 00.00.30.
Note: Using this kind of command, the demo board cann’t be configured as Slave-Only clock, Unicast Mode ,MAC Mode.

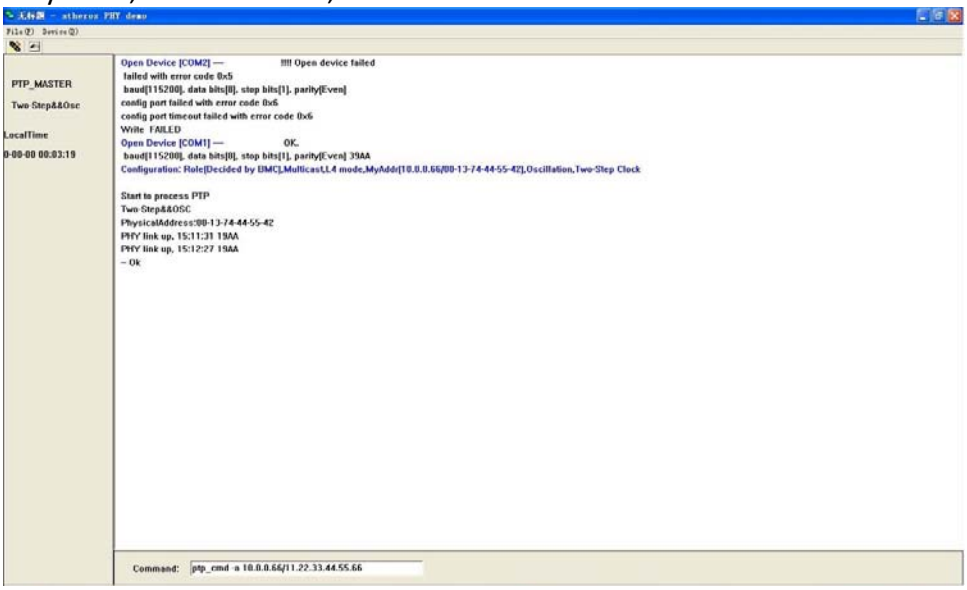


Figure 15--runtime-snapshot when demo application acts as 1588 master

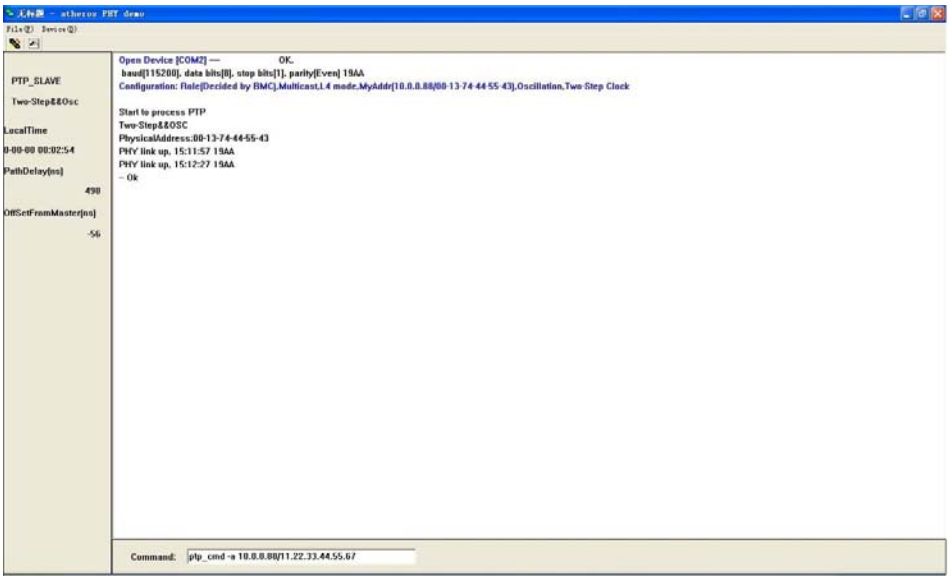
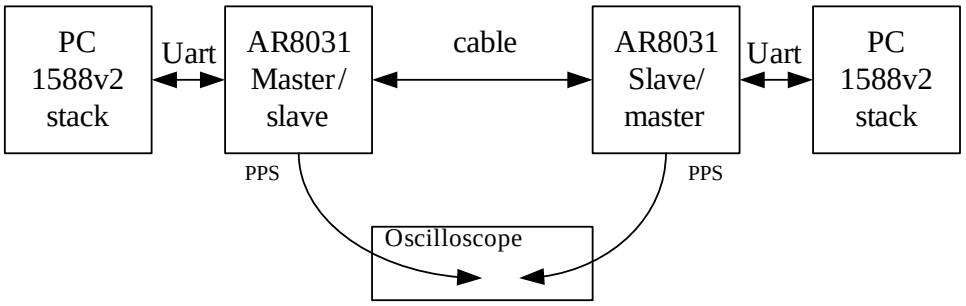


Figure 16-- runtime-snapshot when demo application acts as 1588 slave

6. Test results

Test environment:



Two AR8031 demo boards with cable connect together.
PC communicates with demo board through UART.
Use oscilloscope to test the PPS output, get the PTP time offset.
AR8031 uses free running +/-50ppm 25MHz crystal each at default.

Test results:
Restricted to the UART interface bandwidth, only send four PTP messages each second.

Work mode	Time precision

1588v2 without SYN-E	sub+/-200ns
1588v2 with SYN-E	sub+/-10ns

Test waveform with SYN-E enabled
Tektronix TPO7254 with P6139

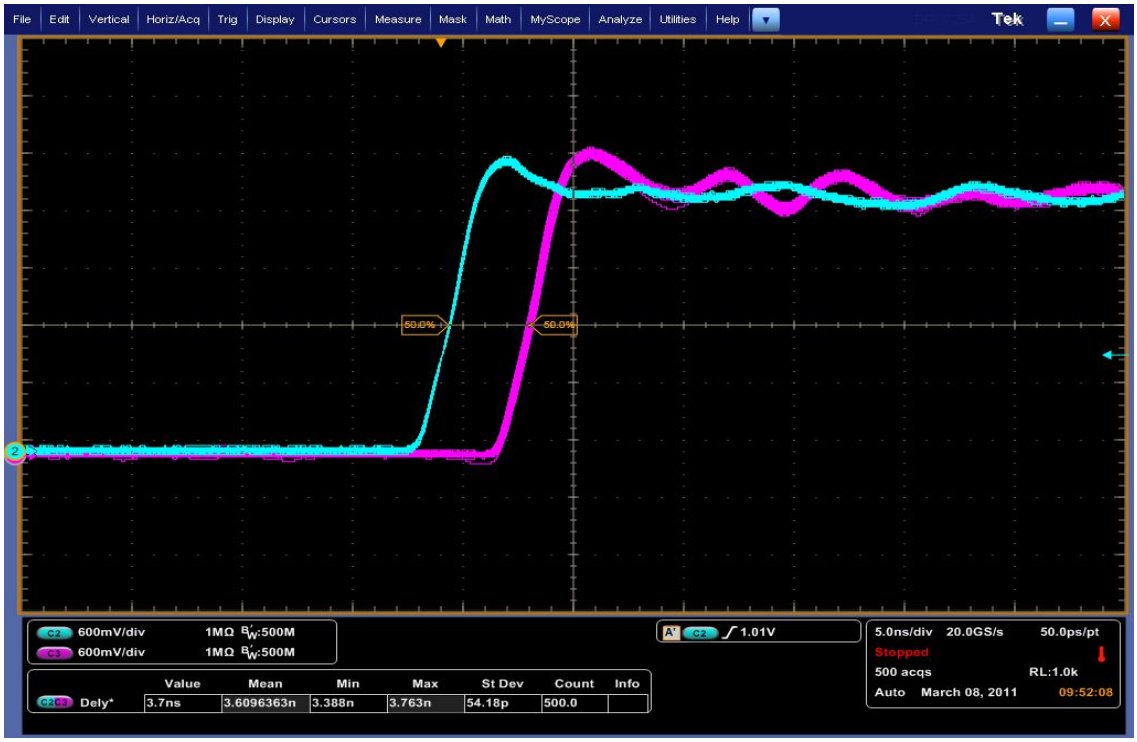


Figure 17— PPS signals of master and slave clocks

7. Conclusion

Atheros has provided a verified high performance and low cost 1588v2 solution for customer with high time precision demand base on ethernet.