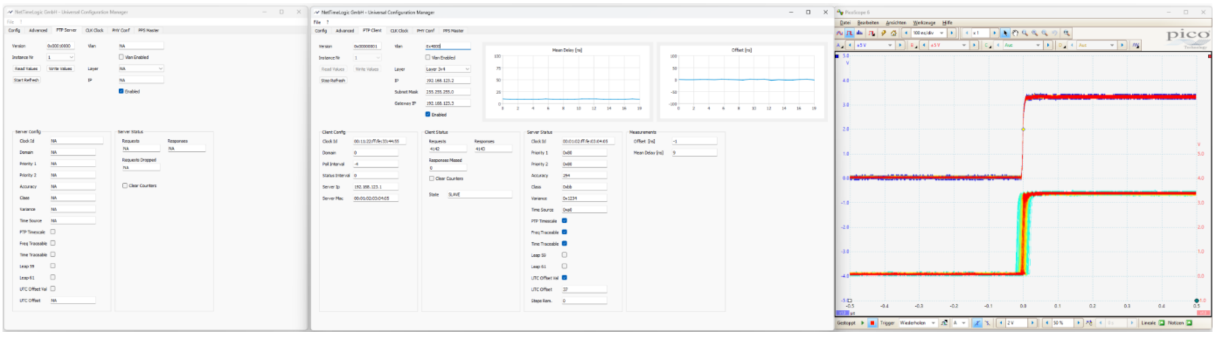


Full hardware high-performance Client-Server PTP Prototype based on FlashPTP

8. März 2024

Why all that?

In the last weeks we revisited the topic of Client-Server based PTP (CSPTP) since there was again some buzz about this.

Currently there are three different Client-Server PTP solutions (or better proposals) out there. All having the same basic concept in mind: a NTP like Client-Server scheme with PTP messages to get on one hand hardware timestamping on the Client and Server but more important to get timing support by the network and lastly to allow much faster synchronization cycles than with NTP (e.g. 128/s). This will get you the same precision as with normal PTP. The three solutions are:

- simplePTP defined by Meta
- FlashPTP defined by Meinberg
- Stateless PTP defined by Microchip.

Unfortunately they are completely incompatible with each other. The IEEE1588 WG took this up and started standardization work in this direction to have one common Client-Server PTP (which is so far the voted name, short CSPTP) but the PAR is not approved as of writing this post.

So, now you may ask, why not just using the standard Unicast scheme of PTP as defined in IEEE1588? Well there are a couple of reasons:

1. IEEE1588 Unicast PTP is a stateful scheme which makes it already complicated. CSPTP is stateless which makes it simple.
2. To get a Unicast Master to talk with a Unicast Slave many so called signaling messages are used to register the Slave on the Master and the Master needs to take track of this message registering which is complicated in the first place and adds quite some overhead. CSPTP doesn't keep track of Clients, the Server just responds to a Request sent by the Client.
3. Point number 2 is also a really dangerous attack vector: You can fake IPs, MACs and ClockIDs on behalf of another device in the network and the Master will start blasting PTP messages to it: huge amplification potential. For CSPTP you have basically no amplification, for one Request you get one Response (in best case of equal length).
4. For a Slave which just died or was disconnected the Master still sends PTP messages for some time (until it gets no registration refresh). With CSPTP on the other hand just no Request is sent anymore, so also no Response will come.
5. The minimal number of PTP messages for a complete synchronization cycle including information of the Master is 4 messages (not including the registering Signaling messages): Sync, DelayReq, DelayResp and Announce. Where CSPTP in best case only would need two messages: Request and Response (what message this shall be we discuss later).
6. Not intended for Routed Networks

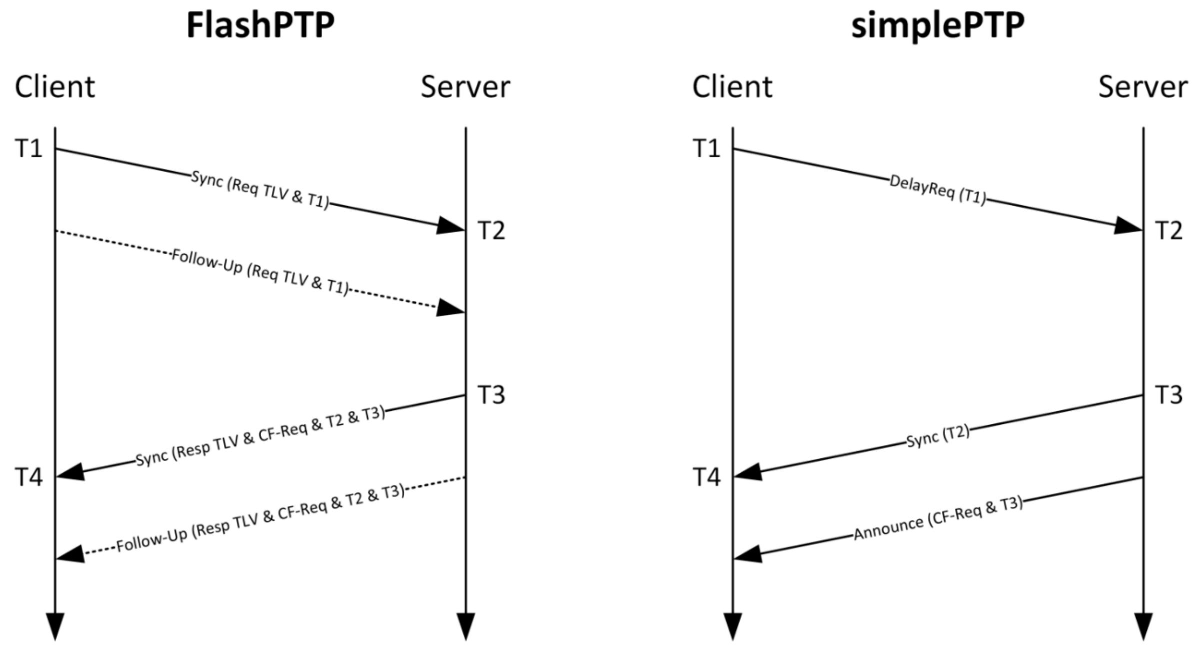
So as you see there are some good reasons to go with the new Client-Server based PTP approach.

FlashPTP vs simplePTP

Let's have a look into two of the published Client-Server PTP proposals (the ones mostly adapted as far as I can tell):

- simplePTP: https://engineering.fb.com/2024/02/07/production-engineering/simple-precision-time-protocol-sptp-meta/
- FlashPTP: https://github.com/meinberg-sync/flashptpd

Under the links above you can find details regarding the two proposals. As mentioned, both have the same principal concept. The main difference is which messages are exchanged between the Client and the Server and how the timing information is propagated between Client and Server.



Message exchange FlashPTP vs simplePTP

And this is where simplePTP took the wrong approach in our point of view. First of all simple PTP requires always 3 messages where FlashPTP only requires 2 messages at best. Secondly simplePTP requires the Network to be One-Step capable (which might be the case nowadays but not guaranteed), where FlashPTP uses the standard IEEE1588 way of Sync and FollowUp which allows One- and Two-Step operation by both the Client/Server and the Network. Thirdly, and this is probably one of the most important points, simplePTP missuses the Announce message for Timing information by filling the Timestamp and Correction fields which are not intended for this and also causes invalid Sync messages.

To initiate a measurement simplePTP sends a DelayReq to the Server with the ProfileSpecific flag set to distinguish between a standard IEEE1588 PTP DelayReq (which might be an issue for PTP TCs which could potentially ignore the message then). Since there will be no DelayResp the Network can not handle Two-Step. The Server then responds with a Sync message where it puts the receive Timestamp of the Delay Request T2. This is an invalid and missused Sync message since it does not contain the transmit Timestamp T3 (and potentially has the Two-Step flag not set, since no FollowUp will follow but an Announce message). Since no Follow-Up will follow again the Network needs to be One-Step. Then the Server sends an Announce message where it puts the Correction field of the DelayReq and the transmit timestamp T3 into the corresponding fields of the Announce message. This is the second missused message since the CorrectionField of an Announce is supposed to be 0 and the Timestamp field of the Announce shall contain an estimate of the transmit Timestamp of the announce or 0. From the mechanism of 1 Request by the Client leads to 2 Responses by the Server you have an amplification factor of 2 which could again be a security issue.

FlashPTP on the other hand makes use of the standard IEEE1588 mechanism of Type-Length-Value (TLV) fields appended to PTP messages. To initiate a measurement the Client sends a Sync to the Server appends a Request TLV to the Sync (One-Step) or FollowUp (Two-Step) which allows the Server to distinguish between a Sync sent by an IEEE1588 PTP Master or a FlashPTP Client, but in any case this Sync message is a valid Sync/Follow-Up message just not from a Master but a Client. The Server then responds to the Request Sync message with a Response Sync message containing the Timestamp T3 (One-Step) where it puts the receive Timestamp T2 into a Response TLV together with the summed Correction field of the Sync/Follow-Up from the Request and additional information of the Server (basically the Information which is contained in an Announce) if the Client requested this additional Information. If the Server is Two-Step, the Response Sync is a completely normal Sync message and the TLV will be appended to the Follow-Up (same as for the Request Sync). Again the Sync/Follow-Up is completely valid just with an additional TLV and just not from a Master but the Server. The TLV lengths are also set so no amplification can happen aka Requests and Response Messages have the same length. If the Server is One-Step capable there will never be an amplification in case it is Two-Step the same amplification of 2 Responses per 1 Request as with simplePTP exists..

As you can see the FlashPTP approach is much cleaner than simplePTP when it comes to the intended use of IEEE1588 PTP messages and mechanisms and has less expectations on the Network (One-Step or Two-Step) and in best case uses one message less than simplePTP which will be the case where no amplification vector exists.

All these advantages of FlashPTP over simplePTP as well as the first documents published by the IEEE1588 WG (CSPTP will most probably and hopefully be much closer to FlashPTP than simplePTP!) made us confident enough to implement a full

hardware based (FPGA) Client and Server compatible with the FlashPTP proposal from Meinberg. Basically as preparation for the Upcoming IEEE Standard and proof of concept (and to show that we can just do it 😉)

On that note we would like to thank Thomas Behn and Meinberg for preparing the proposal and publishing the Open Source Software Client and Server implementations of FlashPTP.

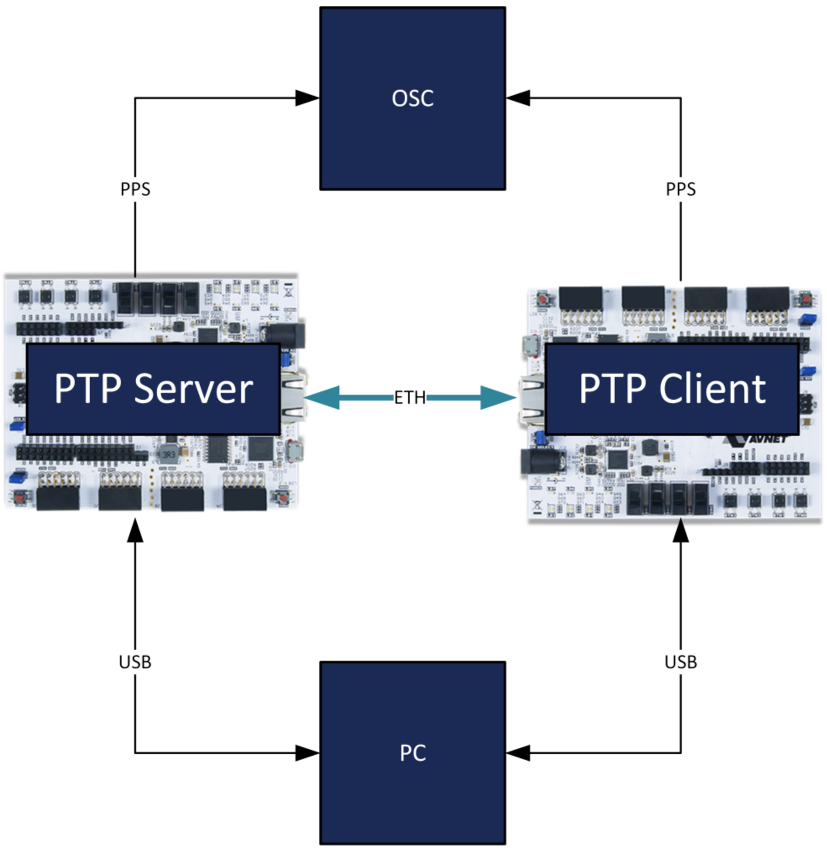
PTP Client and Server Prototype based on FlashPTP

End of February 2024 we started the implementation of our FlashPTP Client and Server. And two weeks and a lot of Coffee later ☕ 😊 we **now have our first prototypes of the full FPGA based FlashPTP Server and Client ready**.

As with all our IP cores, the performance of our solutions is outstanding and often unmatched and our FlashPTP cores are no exception. Why am I mentioning this? Well as with standard IEEE Unicast PTP or NTP the performance of the Server defines how many Clients you can handle and at what Request rates. Our FlashPTP Server can handle Client Requests at line speed on a Gigabit Ethernet link which equals to around **one Million Requests per Second** (same as our NTP Server). This is outstanding in itself but especially if you consider that the whole Server does pull only about 3-4 Watts (go green 🌳)! This should have you covered even for a pretty large Network. And of course it does everything with One-Step messages, so we are really down to the two message exchange with no amplification vector.

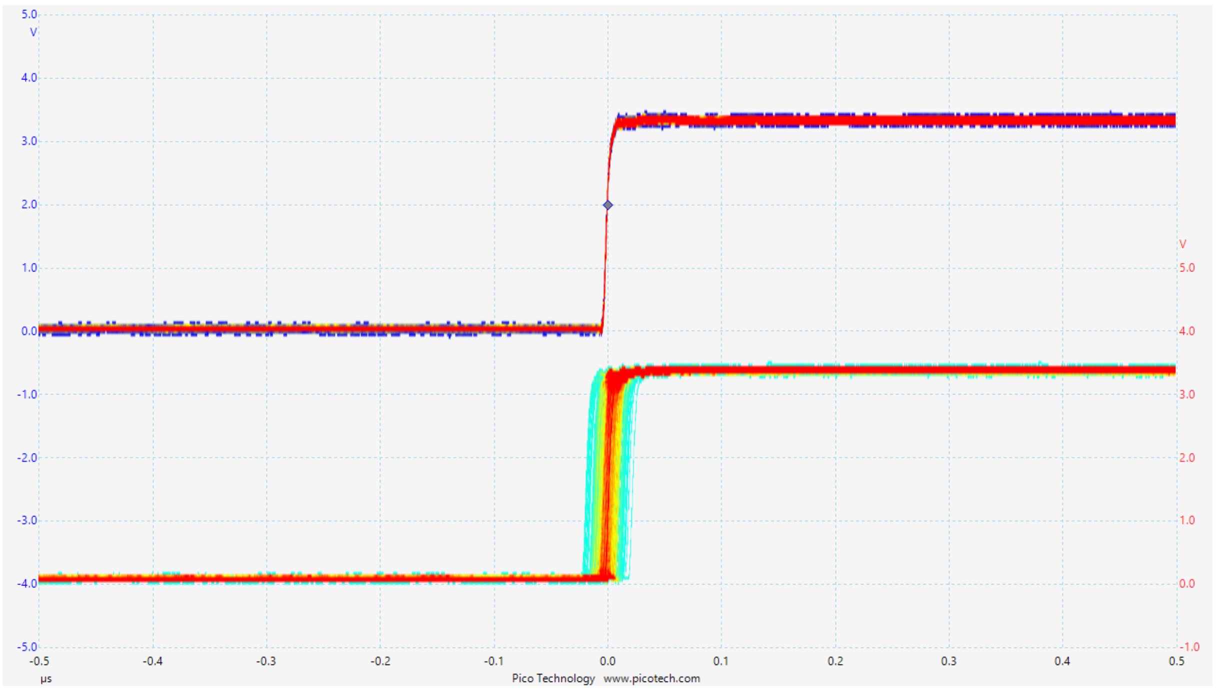
Since we always put the highest efforts into interoperability we directly tested our FlashPTP implementations against the Open Source Reference implementation provided by Meinberg and guess what: It just worked! (well after about 1h of struggling with some fields in the PTP messages and getting the Software versions correctly configured (thanks again to Thomas Behn for the support there)! **I think this is a huge step forward into standardizing a Client-Server based PTP approach**, since it showed that FlashPTP can be implemented based on the IEEE1588-2019 standard and a simple TLV scheme attached to the messages by two completely independent companies and two completely different technology approaches. The FlashPTP specification is simple and just goes hand in hand with IEEE1588 without compromising IEEE1588 in any way.

Ok let's go back to our prototypes. We took two Digilent Arty-A7-100T Development Boards (with AMD Artix7 FPGA) and ported our newly developed FlashPTP Server and Client to it (which took less than an hour by the way, since our cores are really easy to integrate and port and also to Altera, Lattice Semiconductor and Microchip FPGAs). Then we extended our Universal Configuration Tool by a PTP Client and PTP Server tab to easily configure and monitor our cores. Once all that was done we put the Client and Server to the test with a simple setup as shown below.



Testsetup

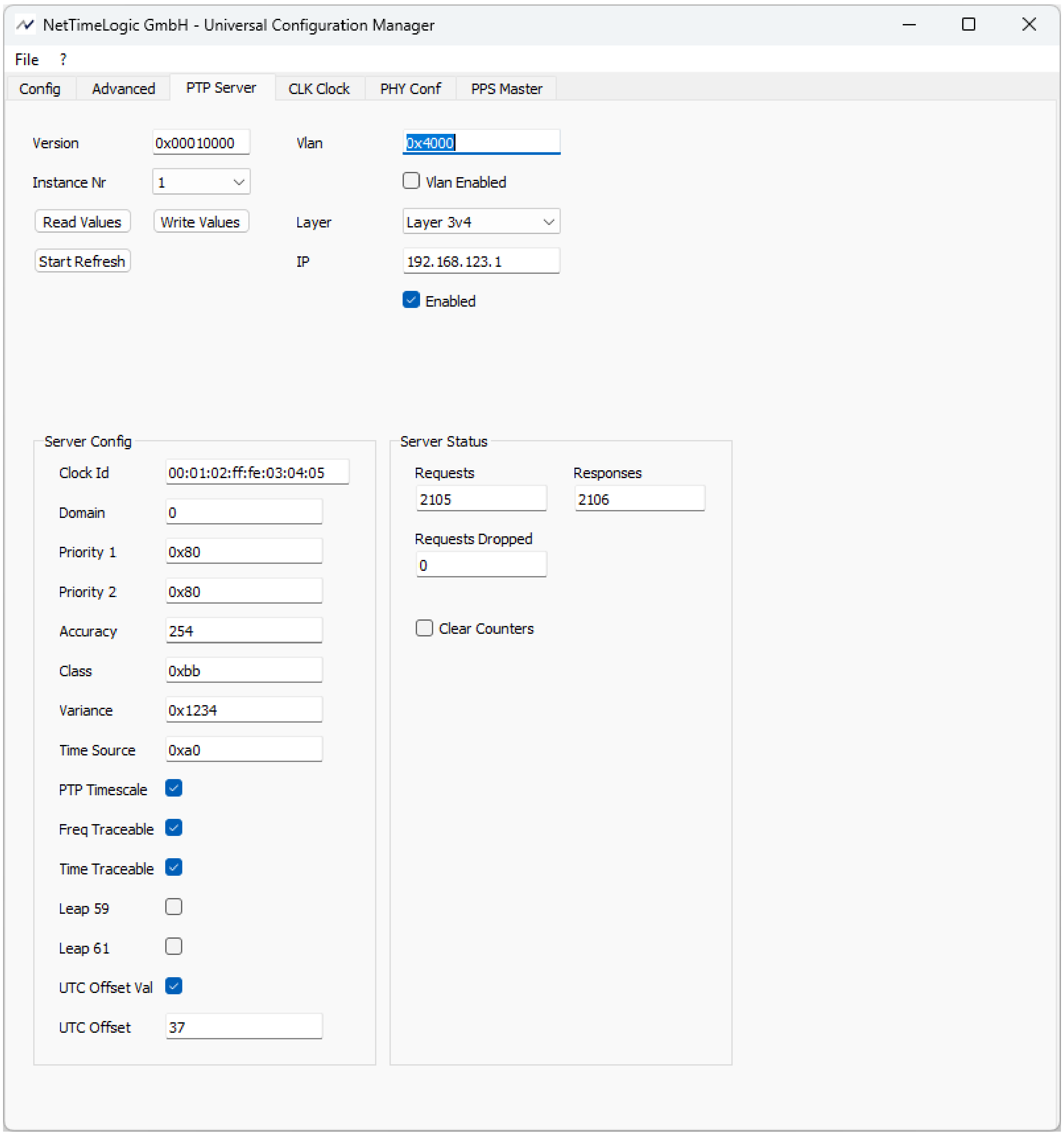
Both the PTP Server and the PTP Client put out a Pulse Per Second (PPS) based on the local clock, which we used to measure the synchronization accuracy. As shown below, the two PPS are within +/-20nswith each other over a test run of around 1h (including all outliers, 100ns per division in the graphic) with a mean accuracy of better +/-10ns.



PPS Accuracy +/-20ns

As mentioned earlier we have a tool called Universal Configuration Manager (UCM, https://www.nettimelogic.com/tools-universal-configuration-manager.php) which is used for configuration and supervision of the Server and Client. It basically allows to read and write AXI Registers over an USB/UART interface and makes a graph-ical representation of the registers.

On the Server Side you can configure the transport layer it shall use Layer2, UDP/IPv4 or UDP/IPv6 and whether frames shall be VLAN tagged or not and of course all PTP related configurations parameters like the Clock Identity and all Quality and UTC parameters.


NetTimeLogic GmbH - Universal Configuration Manager

Version	0x00010000	Vlan	0x4000
Instance Nr	1	☐ Vlan Enabled	
		Layer	Layer 3v4
		IP	192.168.123.1
		✓ Enabled	

Server Config

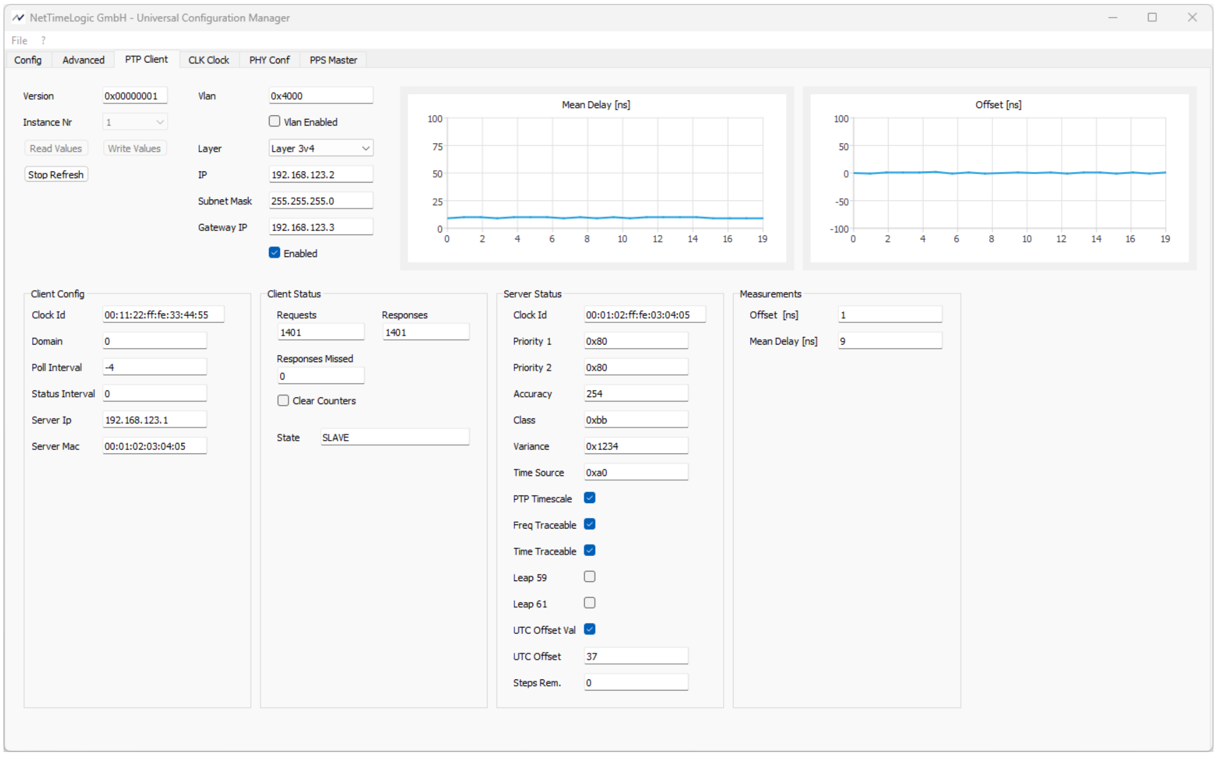
Clock Id	00:01:02:ff:fe:03:04:05
Domain	0
Priority 1	0x80
Priority 2	0x80
Accuracy	254
Class	0xbb
Variance	0x1234
Time Source	0xa0
PTP Timescale	✓
Freq Traceable	✓
Time Traceable	✓
Leap 59	☐
Leap 61	☐
UTC Offset Val	✓
UTC Offset	37

Server Status

Requests	2105
Responses	2106
Requests Dropped	0
Clear Counters	☐

PTP Server Configuration

On the Client side you can configure again the transport Layer and VLAN and if IP is used also the Subnetmask and a potential Gateway. In addition you need to configure the Poll Interval for measurements and as a specialty of FlashPTP you can also define how often a the Server Status is requested in addition (as part of the same TLV, no additional message). This allow to minimize bandwidth usage since you can synchronize e.g. 16 times per Second but only get the Server Status once per Second. Even though FlashPTP is Stateless we have added a State indication to show if the Client can synchronize to the Server or if it is unreachable or doesn't Respond to Requests or if it is not within the desired accuracy, this is however something we came up to give the User an easy indication if everything works fine. Also there are frame counts which show how many requests were done and how many Responses were received or missed. The most interesting part is however the measured Path Delay and Calculated Offset (not what we adjust but the input to the regulation)



Version	0x00000001
Instance Nr	1
Vlan	0x4000
Vlan Enabled	☐
Layer	Layer 3v4
IP	192.168.123.2
Subnet Mask	255.255.255.0
Gateway IP	192.168.123.3
Enabled	✓

Client Config

Clock Id	00:11:22:ff:fe:33:44:55
Domain	0
Poll Interval	-4
Status Interval	0
Server Ip	192.168.123.1
Server Mac	00:01:02:03:04:05

Client Status

Requests	1401
Responses	1401
Responses Missed	0
Clear Counters	☐
State	SLAVE

Server Status

Clock Id	00:01:02:ff:fe:03:04:05
Priority 1	0x80
Priority 2	0x80
Accuracy	254
Class	0xbb
Variance	0x1234
Time Source	0xa0
PTP Timescale	✓
Freq Traceable	✓
Time Traceable	✓
Leap 59	☐
Leap 61	☐
UTC Offset Val	✓
UTC Offset	37
Steps Rem.	0

Measurements

Offset [ns]	1
Mean Delay [ns]	9

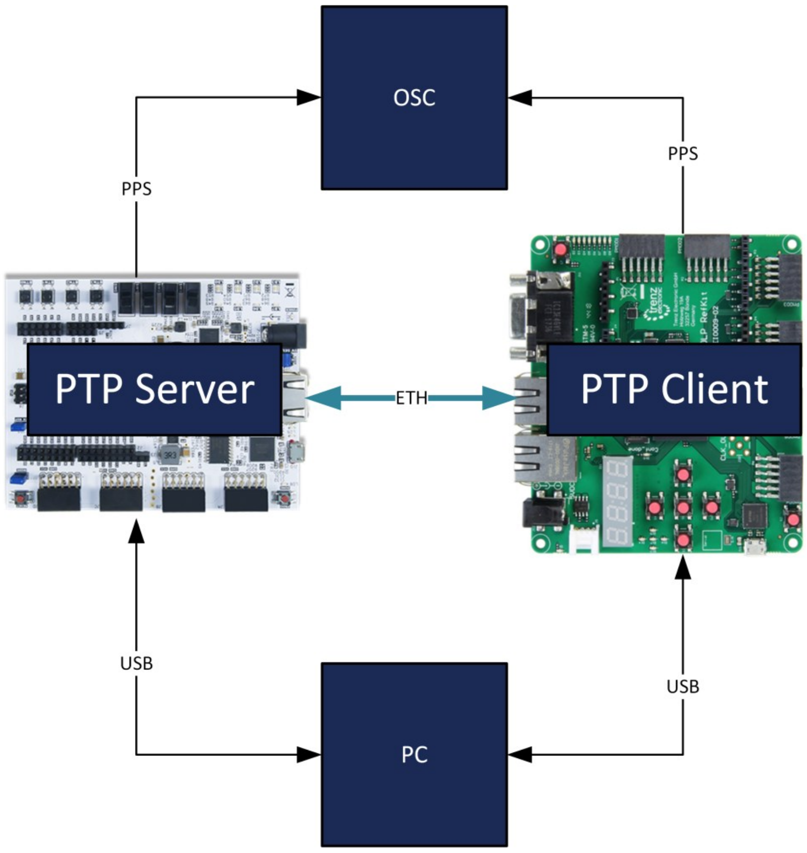
PTP Client Configuration/Supervision

Summary

NetTimeLogic has up and running prototypes for both a PTP Client and a PTP Server (with unmatched performance) based on FlashPTP fully implemented in an FPGA (no CPU, no OS, no Drivers, no Software Stack) which is fully compatible with the Open Source reference implementation from Meinberg. We choose Flash-PTP over simplePTP for many reasons but mostly because we believe it will be the base for the CSPTP which will be standardized by the IEEE1588 WG. We will adapt our prototypes as soon as the standardization progresses. If you are interested in integrating FlashPTP into your device contact us.

Update

We ported shortly (~15min) the PTP Client also to a Trenz Electronic GmbH C10LP RefKit Development Board (with an Altera Cyclone 10 LP FPGA) just to show that our IP Cores are all FPGA vendor independent. Performance wise it is identical to the previous setup.



Testsetup with mixed FPGA vendors (AMD<=>Intel/Altera)