PtpTransparent Clock

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Sven Meier
Reviewer(s)	-
Version	2.5
Date	25.10.2024

Copyright Notice

Disclaimer

The information available to you in this document/code may contain errors and is subject to periods of interruption. While NetTimeLogic GmbH does its best to maintain the information it offers in the document/code, it cannot be held responsible for any errors, defects, lost profits, or other consequential damages arising from the use of this document/code.

NETTIMELOGIC GMBH PROVIDES THE INFORMATION, SERVICES AND PRODUCTS AVAILABLE IN THIS DOCUMENT/CODE "AS IS," WITH NO WARRANTIES WHATSOEVER. ALL EXPRESS WARRANTIES AND ALL IMPLIED WARRANTIES, INCLUDING WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF PROPRIETARY RIGHTS ARE HEREBY DISCLAIMED TO THE FULLEST EXTENT PERMITTED BY LAW. IN NO EVENT SHALL NETTIMELOGIC GMBH BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL AND EXEMPLARY DAMAGES, OR ANY DAMAGES WHATSOEVER, ARISING FROM THE USE OR PERFORMANCE OF THIS DOCUMENT/CODE OR FROM ANY INFORMATION, SERVICES OR PRODUCTS PROVIDED THROUGH THIS DOCUMENT/CODE, EVEN IF NETTIMELOGIC GMBH HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

IF YOU ARE DISSATISFIED WITH THIS DOCUMENT/CODE, OR ANY PORTION THEREOF, YOUR EXCLUSIVE REMEDY SHALL BE TO CEASE USING THE DOCUMENT/CODE.

Overview

NetTimeLogic's PTP Transparent Clock is a full hardware (FPGA) only implementation of a three port (but extendable to N ports) Transparent Clock according to IEEE1588-2019/2008 (PTP). The whole protocol handling, algorithms and calculations are implemented in the core, no CPU is required. This allows running PTP synchronization completely independent and standalone from the user application. The Transparent Clock can act as a Spider Clock around an Ethernet Switch or a Redundancy Core like HSR or PRP or like in the reference design as a Daisy. For the Daisy Chain a special filter and mux was implemented to allow frame sending and reception on the Daisy Chains uplink port. All datasets according to IEEE1588-2019/2008 can be configured either by signals or by an AXI4Lite-Slave Register interface.

Key Features:

- PTP Transparent Clock according to IEEE1588-2019/2008
- Intercepts path between MAC, Switch or Redundancy core and PHY
- Three Port (extendable to N Ports)
- Inaccuracy introduction: +/- 25ns
- Support for Default Profile: Layer 2 (Ethernet) and Layer 3 (Ipv4/v6) support
- Support for Power Profile: C37.238-2011 including VLAN support
- Support for Utility Profile: including HSR and PRP tag handling
- Support for TSN Profile: including HSR, PRP and TSN tag handling
- Support for ITU Profiles
- One Step and Two Step support
- Peer to Peer (P2P) and End to End (E2E) delay measurement
- Optional unicast E2E handling
- Full line speed
- AXI4Lite register set or static configuration
- Datasets according to IEEE1588
- MII/GMII/RGMII Interface support (optional AXI4 stream for interconnection to 3rd party cores)
- Optional Management Message support
- Optional TwoStep Message sending support
- Timestamp resolution with 50 MHz system clock: 10ns
- Asymmetry Correction

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	28.12.2015	First draft
1.0	07.10.2016	First release
1.1	29.06.2017	E2E added
1.2	11.08.2017	Message Intervals added
1.3	14.11.2017	Delay Naming and Link Speed added
1.4	20.12.2017	Status interface added
1.5	30.08.2018	TwoStep and TSN added
1.6	29.09.2018	Added Asymmetry and some type fixes
1.7	25.04.2019	Added Default Dataset Values to Status Record
1.8	14.06.2019	Added MaxDelay
1.9	25.01.2021	Added IPv6
2.0	11.06.2021	Added VLAN remove/insert made for PTP
2.1	26.07.2022	E2E Unicast added
2.2	23.08.2022	E2E Unicast reworked
2.3	03.01.2023	Added Vivado upgrade version description
2.4	13.11.2023	Added DSCP
2.5	25.10.2024	IEEE1588-2008 => IEEE1588-2019/2008

Table 1: Revision History

Content

Definitions

Definitions	
Ordinary Clock	A synchronization end node according to IEEE1588 that can take a Master and Slave role
Transparent Clock	A network node (Switch) that is IEEE1588 aware and compensates network jitter
Default Profile	PTP Profile according to IEEE1588
Power Profile	PTP Profile according to C37.238-2011
Utility Profile	PTP Profile according to IEC 61850 9-3
TSN Profile	PTP Profile according to IEEE802.1AS

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
IRQ	Interrupt, Signaling to e.g. a CPU
PRP	Parallel Redundancy Protocol (IEC 62439-3)
HSR	High-availability Seamless Redundancy (IEC 62439-3)
PTP	Precision Time Protocol (See also IEEE1588)
MAC	Media Access Controller
PHY	Physical Media Access Controller
OC	Ordinary Clock
TC	Transparent Clock
TS	Timestamp
ETH	Ethernet
TB	Testbench
LUT	Look Up Table
FF	Flip Flop

RAM	Random Access Memory
ROM	Read Only Memory
FPGA	Field Programmable Gate Array
VHDL	Hardware description Language for FPGA's

Table 3: Abbreviations

1 Introduction

1.1 Context Overview

The PTP Transparent Clock is meant as a co-processor handling PTP. It intercepts the Media Independent Interface (MII) on the Ethernet path between the MAC, Switch or Redundancy core and PHYs where it handles all PTP traffic. This means it generates and processes PTP frames directly in hardware, using the same data path as the normal traffic coming from or going to the Ethernet MAC, Switch or Redundancy core. This also means that it uses a small amount (around 2 frames per second) of the bandwidth on the MII so if 100% Network traffic shall be constantly sent by the Ethernet MAC, Switch or Redundancy core it would eventually drop some frames to still handle PTP. The PTP Transparent Clock is designed to work in cooperation with the Counter Clock core from NetTimeLogic for timer events (not a requirement, any millisecond single pulse generator is sufficient). It contains an AXI4Lite slave for configuration from a CPU, this is however not required since the PTP Transparent Clock can also be configured statically via signals/constants directly from within the FPGA.

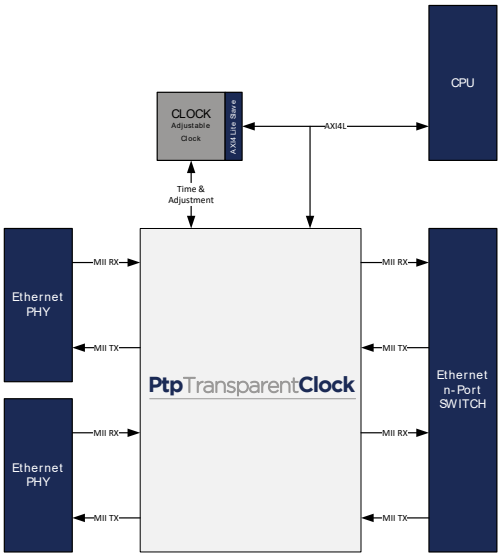

Figure 1: Context Block Diagram

1.2 Function

The PTP Transparent Clock is a PTP TC according to IEEE1588-2019/2008. It is a network node that can compensate message delays of PTP messages in a switch. It

does not synchronize itself to a PTP master and is therefore stateless. For the P2p delay measurement mode it also measures the delay to its direct neighbors using the P2P delay mechanism. It contains all Datasets defined for TCs in IEEE1588.

1.3 Architecture

The core is split up into different functional blocks for reduction of the complexity, modularity and maximum reuse of blocks. The interfaces between the functional blocks are kept as small as possible for easier understanding of the core.

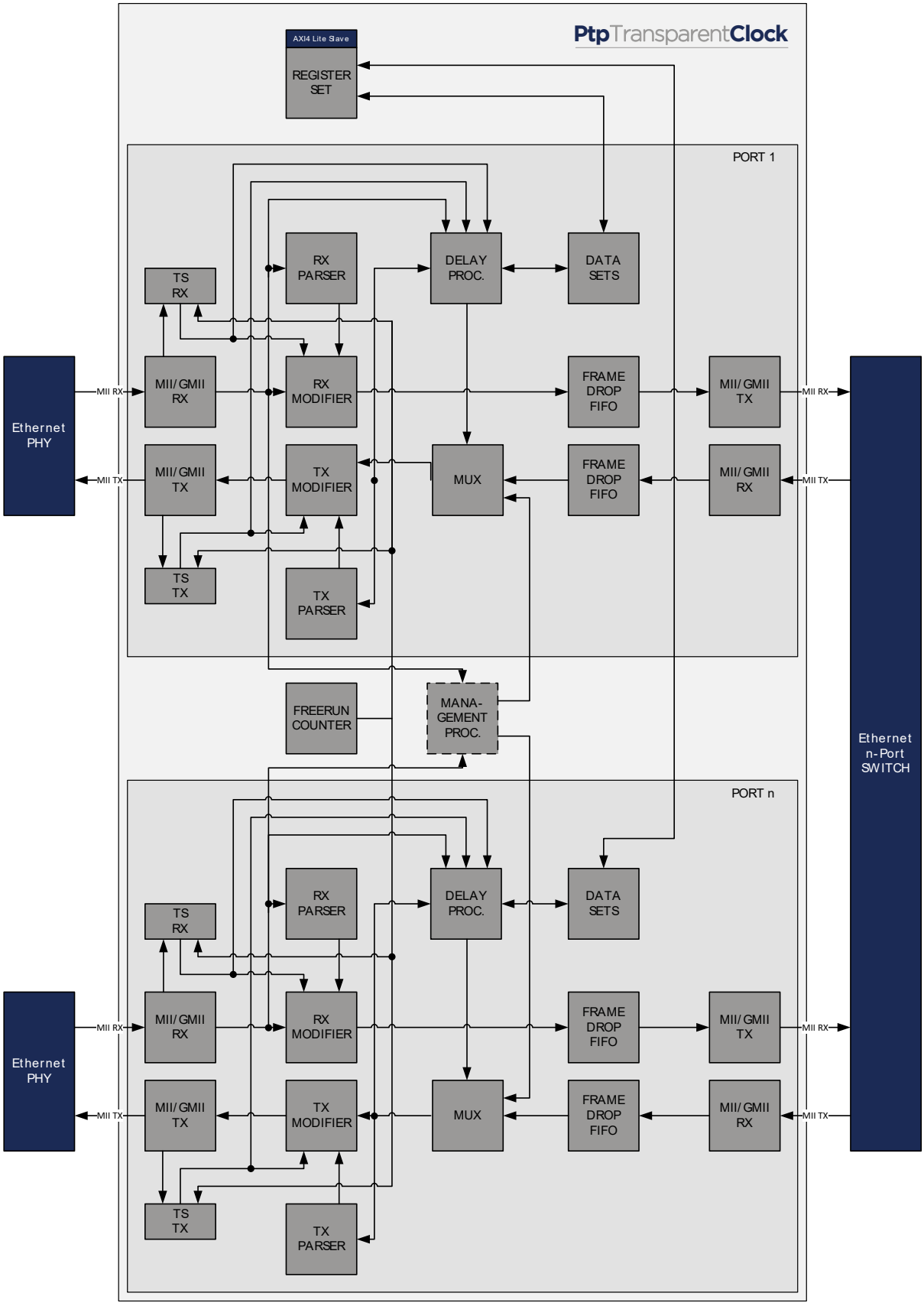

Figure 2: Architecture Block Diagram

Register Set

This block allows reading status values and writing configuration.

Data Sets

This block represents the data storage for the datasets in the data format according to IEEE1588. Multiple sources can adapt these values.

Management Processor

This block handles the PTP Management messages as a control-node, which means it is only responding to PTP management messages but no initiating PTP management messages. This block is optional for security reasons.

Delay Processor

This block handles the PTP Delay messages. In P2P mode, it will respond to PTP PDelayReq messages with PTP PDelayResp messages in one-step mode with residence time in the correction field. It also sends PTP PDelayReq and waits for PTP PDelayResp and calculates the delay for this port and averages over the last 4 measurements. In E2E mode the TC part does not issue messages.

Receive/Transmit Timestamper

These blocks generate a snapshot of the free-running and counter clock time when the Start of Frame Delimiter (SFD) was detected by the Interface Adapters

Receive/Transmit Frame Parser

These blocks analyze the frame and extract frame information like the PTP correction field, the frame length or if the CRC ok.

Receive/Transmit Frame Modifier

These blocks modify PTP event messages (Sync and PDelay) on the fly by inserting newly calculated correction fields, timestamps and inaccuracy fields (power profile) and recreating a new CRC. It also decides whether a frame shall be dropped or not (all PTP frames are dropped in the MAC direction).

Frame Multiplexer

This block multiplexes the frames coming from all the frame processors and the forwarding path from the MAC to the PHY.

Frame Drop Fifo

This FIFO is a store and forward FIFO with drop functionality. During frame reception a drop signal can be asserted which will cause that the frame is dropped. If it will run into an overflow condition it drops the incoming frame.

Free-running Counter

This block generates a free-running counter which is used for the path delay measurement. It counts in the same format as the counter clock but with no adjustments.

(R)(G)MII Receive/Transmit Interface Adapter

These blocks convert the data stream from the (R)(G)MII to a 32bit AXI stream and back form 32bit AXI stream to (R)(G)MII.

2 PTP Basics

2.1 Protocol

PTP means Precision Time Protocol and is standardized in IEEE1588-2019/2008 (or also IEC61588-Ed.2). It describes the mechanisms how to distribute time (phase and frequency) precisely (sub-microsecond accuracy) over an Ethernet based, packet based network and determines the best clock for time distribution automatically. The principal of the protocol is based on frames that are exchanged periodically between nodes containing timestamps of when the exchanged frames where sent and received along with information of the clock quality of the nodes.

2.2 Principles

PTP defines a Master Slave system. In a PTP network there is only one active Master and multiple Slaves. As already mentioned there are messages periodically exchanged (and timestamped) between the Master and Slave to determine and correct the offset and drift of the slave against the master and to measure the network delay between the Slave and the Master to correct this also in the offset. For measuring the delay between the Master and Slave two mechanisms are defined: Peer To Peer (P2P) and End To End (E2E). As the names say P2P is measuring the delay only to the next neighbor and E2E is measuring from the Slave to the Master. We will see the advantages and disadvantages of the two mechanisms later, for now we assume a simple setup of a Slave directly connected to a Master with nothing then a cable in between:

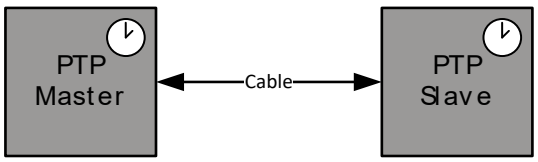

Figure 3: Simple setup

Now we look at the messages and calculations done for the two mechanisms: In both cases the Master is sending a so called Announce message and so called Sync messages to the Slave. The Master takes a timestamp T1 when it starts to send the Sync message and depending on its capabilities puts the timestamp T1 on the fly into the Sync message (one-step handling) or sends a second message called Sync FollowUp which contains T1 (two-step). On the Slave side it takes a timestamp T2

when the Sync message is coming in. With these two timestamps (T2 and T1) the slave can calculate an offset but the propagation delay between the master and slave is still missing so the Slave would have a constant offset of the delay to the Master. For calculating the delay now the two mechanisms differ:
For E2E the Slave sends a so called Delay Req message to the Master and stores the send timestamp T3. The Master takes a timestamp T4 when it receives the Delay Req message and sends this timestamp T4 via a so called Delay Resp to the Slave. Now the Slave has all four timestamps (T1-T4) to calculate the Delay according to the calculations below.
For P2P, things work a bit different. The Slave sends a so called PDelay Req message to its neighbor (in this case the Master) and stores the send timestamp T3. The neighbor takes a timestamp T4 when it receives the PDelay Req message. Then is sends a so called PDelay Resp containing T4 but and in parallel timestamps the sending moment of the PDelay Resp with T5. Again depending on the capabilities of the node it inserts the timestamp T5 on the fly into the PDelay Resp message (one-step) or sends a second message called PDelay Resp FollowUp containing T5 (two-step). The Slave takes a timestamp T6 when it receives the PDelay Resp message. Now the slave also has the four timestamps (T3-T6) to calculate the Delay according to the calculations below. In contrary to the E2E mechanism also the Master (respectively the neighbor) is also calculating the Delay the same way as the Slave.
So for E2E the Delay calculation is based on Sync messages sent by the Master where for P2P the Delay calculation is completely independent of Sync messages. This means for a high accuracy delay measurement the frequency of the two clocks have to be as close to each other as possible, where for E2E this is more important as for the P2P case. Both delay mechanisms assume a symmetrical delay which is normally the case for Ethernet.
Once the Delay is calculated the real offset can be calculated with the two timestamps (T1 and T2) from the Sync and the propagation delay calculated via one of the mechanisms. With Offset correction the phase is corrected to the one from the Master. This is done with every Sync message.
The last value that is needed to get high accuracy synchronization is the so called Drift which is the frequency difference between the Master and Slave. Since the oscillators of the Master and Slave are never 100% identical the Slave will drift away from the master during two Sync messages. To adjust the frequency the timestamps from two Sync messages are needed (T1, T1' and T2 and T2'). With these four timestamps the frequency difference can be calculated and adjusted at

the Slave. After this both frequency and phase are adjusted and the Slave is synchronized to the Master.

PTP Nodes have to be able to handle both types of messages: one-step and two-step, but they don't need to generate two-step frames if they are one-step capable and vice versa.

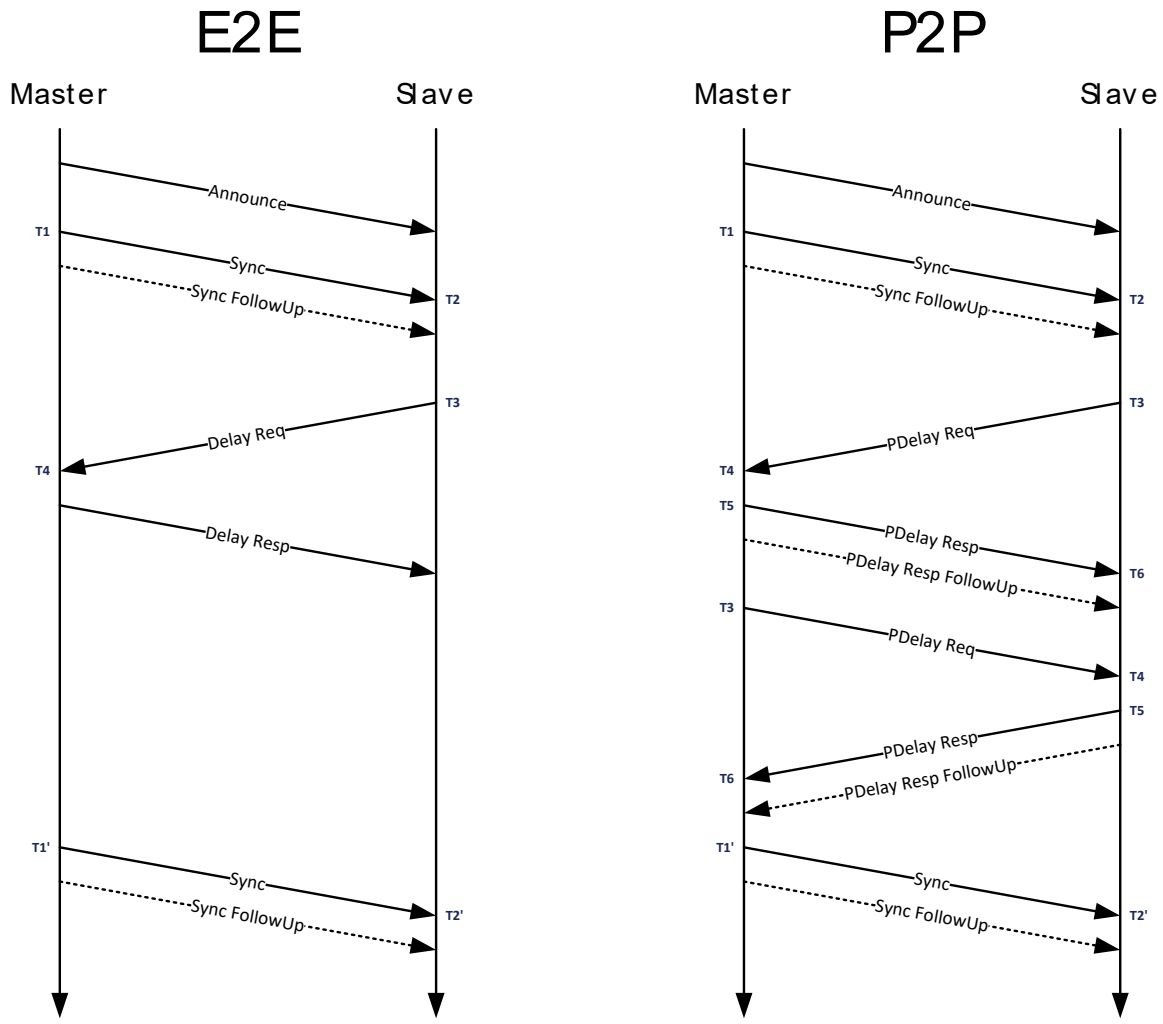

Figure 4: Message exchange simple setup

$$Delay = \frac{(T4 - T1) - (T3 - T2)}{2} \qquad Delay = \frac{(T6 - T3) - (T5 - T4)}{2}$$

$$Offset = (T2 - T1) - Delay$$

$$Drift = \frac{(T2' - T2) - (T1' - T1)}{(T1' - T1)}$$

In this example one Master was connected to exactly one Slave. In a normal setup there are many Slaves and one Master. PTP is self-organizing, which means it chooses the best available Master in a Network and all Slaves are then synchronizing to this Master. In a PTP network there are normally multiple Master capable nodes, therefore the Announce messages exist. With the Announce messages the Master capable device announces its clock quality in the network as long as no Announce message from a better node is received or a timeout occurred. This way in a steady state only one node is sending Announce messages and therefor is the Master in the network. Also the node which is sending Announce messages has to send Sync messages since it is the Master in the network. The comparison of the clock quality parameters and the state machine is defined in the Best Master Clock (BMC) algorithm.

2.2.1 PTP Nodes

IEEE1588 defines seven types of PTP nodes which all have different functions in a PTP network

2.2.1.1 Ordinary Clock (OC)

An Ordinary Clock (OC) is defined as a PTP clock with a single PTP port. It can operate either as a Master or Slave in the PTP network. The mode is selected via the BMC algorithm. Ordinary Clocks are the most common node type in a PTP network as they are generally used as end-nodes within a network requiring synchronization between each other. One of the OCs will act as a Master and all other ones will stay in Slave mode. If the current Master goes away one of the OCs will take over the Master role and synchronize the other nodes.

2.2.1.2 Grandmaster Clock (GM)

A Grandmaster Clock (GM) is defined as a PTP Ordinary Clock with either an external time source (GPS, IRIG) or a very high accuracy time (ATOM). It can only act as a Master in the PTP network and will win the Master role according the BMC. In the case that more than one Grandmaster is connected to the same PTP network the one which is worse according to the BMC will go in a Passive state where it remains as long as the Master is active. This is used in the case of backup Grandmasters.

2.2.1.3 Slave Only Clock (SC)

A Slave Only Clock (SC) is defined as a PTP Ordinary Clock which can only act as a Slave in the PTP network and will never win the Master role according the BMC; it

will therefor also never send Announce Messages. Slave Only Clocks are always end nodes, so if no Master is available in the network they will be unsynchronized. Since they don't really participate in the BMC selection a very lightweight implementation of slave only clocks is possible.

2.2.1.4 Boundary Clock (BC)

A Boundary Clock (BC) is a network element with PTP functionality. It has in contrary to the OC more than one port. A Boundary Clock is normally an Ethernet Switch or Router. The Problem with normal Switches and Routers is that the forwarding delay between a frame coming in and going out of the device is not deterministic. Therefore the concept of Boundary Clocks was introduced, where all PTP messages end and are sourced by this node rather than forwarding the PTP messages. A Boundary Clock synchronizes itself on one of the ports to the Master, so it is Slave on that port and acts on all other ports as Master synchronizing the other nodes. The state decision on the Ports is again based on the BMC. If no better Master is available it can also take the role of the Grandmaster in the network, in that case it is Master on all ports. A BC can also act as a bridge between different PTP network configurations.

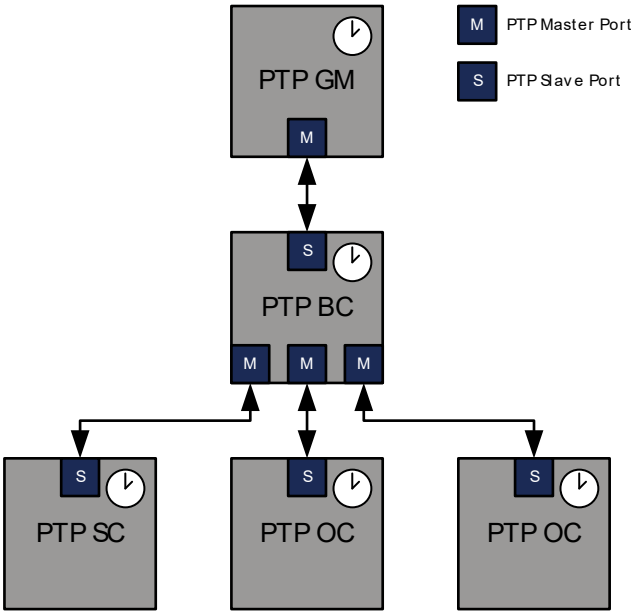

Figure 5: PTP network with Boundary Clock

2.2.1.5 Transparent Clock (TC)

A Transparent Clock (TC) is as the Boundary Clock a network element with PTP functionality. A Transparent Clock is normally an Ethernet Switch or another Network element with more than one port. In contrary to the Boundary Clock it is stateless, so no port is in a Master or Slave state. To overcome the mentioned problem of non-deterministic forwarding delays in the Switch it measure the residence time of a PTP message in the Switch and adds this value to a so called Correction Field within the PTP messages. So for the Slave a Transparent Clock is not visible, it just gets the correction values which it has to take into account in the Delay, Offset and Drift calculations. The Transparent clock comes in different flavors: E2E one-step or two-step TC and P2P one-step or two-step TC. To simplify the implementations of TCs only one-step TCs are considered in this description. A one-step TC can put the residence time of a PTP message on the fly into the message. This residence time has then be taken into account when calculating Delays and Offset so the residence time is falling out of the calculation and only the cable delays are remaining. An advantage of the TC over the BC is that no cascaded servo loops are introduced which makes deeper hierarchies possible without making the chain unstable Also reaction time of the system significantly increases since for each hierarchy you don't have to wait until the higher hierarchy has been synchronized.

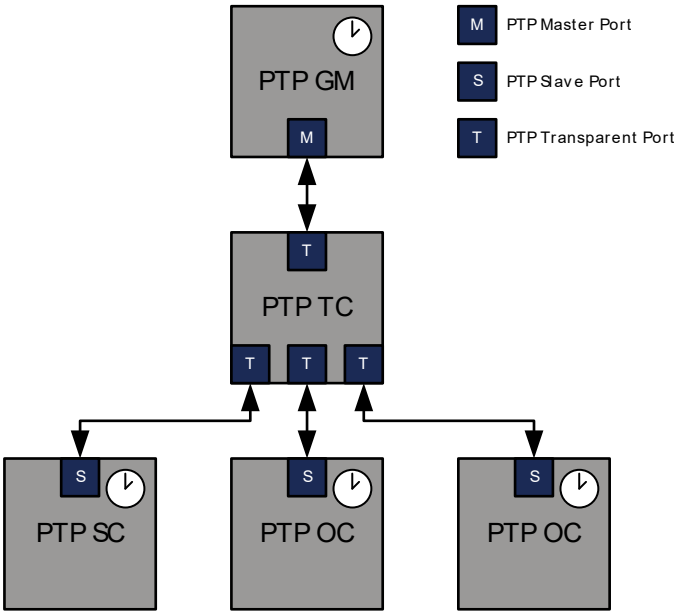

Figure 6: PTP network with Transparent Clock

2.2.1.6 Hybrid Clock (HC)

A Hybrid Clock (HC) is a combination of a Transparent and Ordinary Clock. Hybrid Clocks are often used in Daisy-Chains. In a Daisy-Chain a three port TC is used, two ports are used for the forwarding path on the Daisy-Chain and one is used as the uplink to the CPU and OC.

2.2.1.7 Management Node (MN)

A Management Node (MN) does not take part in the synchronization and BMC of the PTP network. It sends PTP Management messages to the nodes to supervise the state of the PTP network. All PTP nodes have to response to PTP Management messages according to the standard, however a lot of the PTP nodes don't support PTP Management anymore because of security reasons, therefore PTP Management nodes are not widely used.

2.2.2 Delay Mechanisms

Measuring the delay is one of the important mechanisms in PTP. In general all network nodes (Switches/Routers) shall be PTP aware (BC or TC) because of the mentioned non-determinism of message forwarding in Switches and Routers. However for E2E Delay measurements also standard Switches could be in the network, but this requires a lot of statistics and high message rates to achieve sub-microsecond accuracy (and is not always possible).
For the P2P Delay mechanism only PTP aware Switches/Routers are allowed, breaking this rule will break the synchronization! For the next chapters only PTP aware nodes are considered in the network. Only one delay mechanism per PTP segment is allowed and cannot be mixed. Special Boundary Clocks exist which can run different delay mechanisms per PTP segment (ports belonging to one network, in best case per port).

2.2.2.1 E2E

In End to End (E2E) Delay measurement the Slave measures the whole path delay to the next Master port.

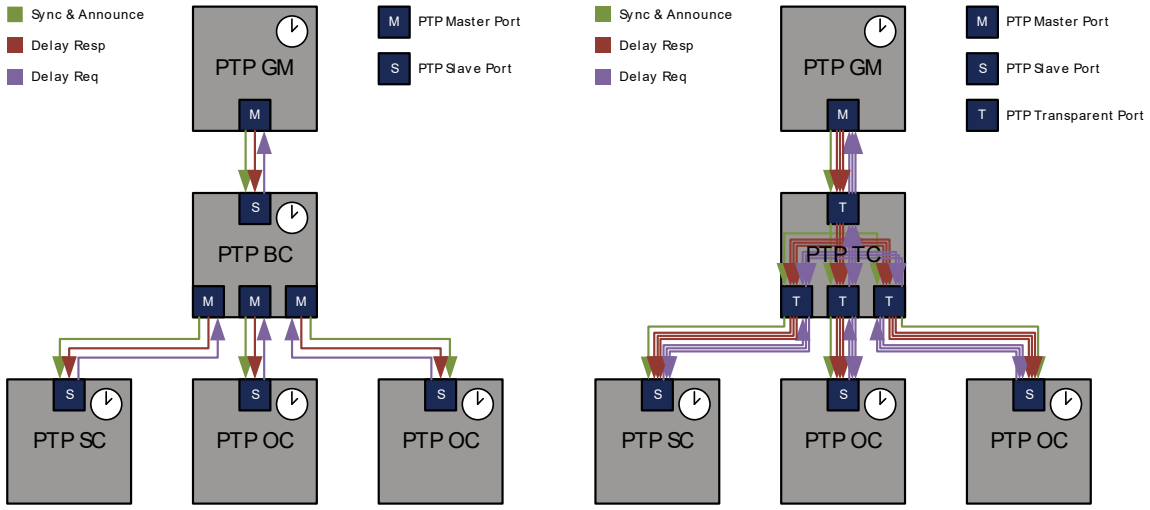

Figure 7: E2E Delay measurement with BC and TC

In the case of a network with a BC this means the Slaves measure the path directly to the BC. The BC itself measures the Delay to the Grandmaster and synchronizes to it. Since the Sync, Announce and Delay messages always end and are sourced at the BC, there is no correction needed in the Messages. Each Master port only receives the Delay Req messages from the Slave directly connected to it and each Slave port only receives Delay Resp for his Delay Req (this is not the case if non PTP aware Switches/Routers are used) from the BC.

In the case of a network with an E2E TC, things look differently. Since the TC is stateless it just forwards all PTP messages according to switching rules (all PTP messages in this case are L2 multicast messages) to all other ports except the port the message came from. This means that the Grandmaster receives the Delay Req messages from all three Slaves and has to answer all of the with a Delay Resp message. Also a Slave receives all Delay Req messages from all the other Slaves as well as all Delay Resp messages from the Master, means the Delay Resp message for his Delay Req message but also the Delay Resp messages as response to the other Slaves Delay Req messages. In a large-scale network this produces quite some unnecessary network load because of the other Slaves Delay messages at the Slaves and quite some CPU and network load on the Master because it has to answer all Slaves Delay Req message. The TC corrects the residence time of the Sync and Delay Req messages directly in the Correction Field of these messages. A Slave can subtract this correction value from its Delay so the only things that remain are the cable delays.

There is also a mixed multicast/unicast mode where PTP Sync and Announce messages are still L2 multicast messages, but PTP Delay Req and Delay Resp messages are sent as L2 unicast messages. This will allow to reduce the bandwidth usage in the network with E2E TCs so other PTP nodes don't receive any unnecessary PTP frames. Otherwise, everything is working the same way. The only requirement is that all PTP nodes also support this scheme.

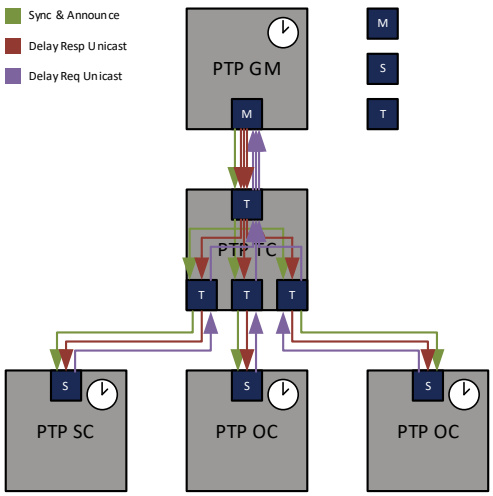

Figure 8: E2E Delay unicast measurement with TC

2.2.2.2 P2P

In Peer to Peer (P2P) Delay measurement each PTP node measures the delay only to its direct neighbor independent of it's the node type and port state. So the Slave never knows the whole delay to the master.

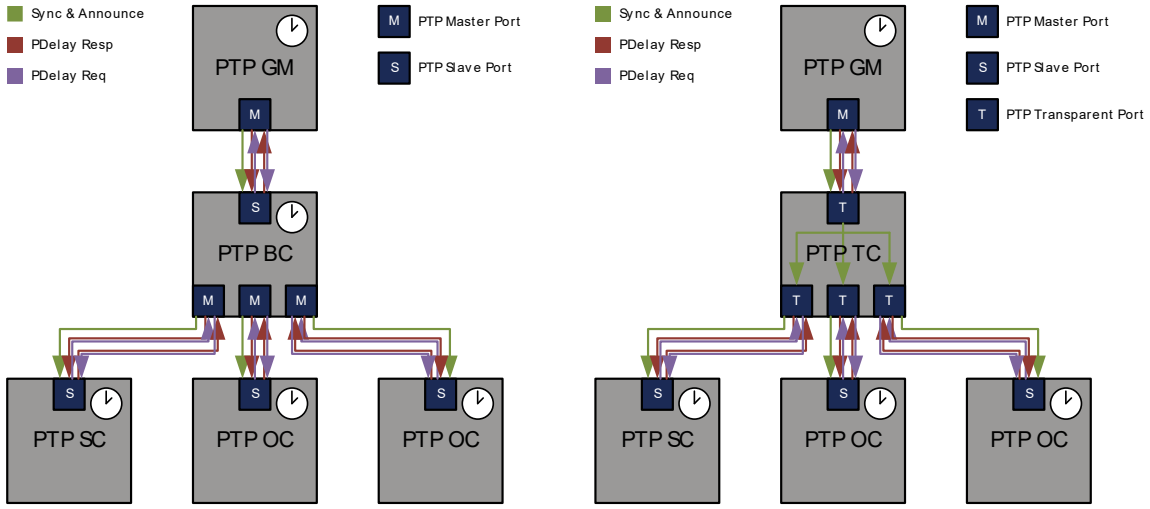

Figure 9: P2P Delay measurement with BC and TC

In the case of a network with a BC this means not only the Slaves but also the Grandmaster measure the path directly to the BC. The BC itself measures the Delay to the Grandmaster and synchronizes to it, and it also measures the Delay to all other nodes connected to it. Since the Sync, Announce and Delay messages always end and are sourced at the BC, there is no correction needed in the Messages. Each port only receives the PDelay Req and PDelay Resp messages from the node directly connected to it but this also means that PDelay Req messages have to be answered with PDelay Resp messages by each port in the network participating in PTP.

In the case of a network with a P2P TC, things work a bit differently. Unlike in the E2E case the TC measures in P2P like the BC the Delay to all other nodes (Slaves and Master) and answers all PDelay Req messages on each port. Since the PDelay messages have a Link-Local destination MAC address they will not be passed through a Switch or Router. Therefore similar as with a BC the PDelay messages end at and are sourced by the TC. The difference to E2E delay message handling gets clear when it comes to Sync messages. When a Sync message is received at the TC the Delay measured on this specific port is added to the Correction Field of this Sync message. When the frame is leaving the TC it adds the residence also to the Correction Field and forwards it to the Slave. So for a Slave the whole delay of

the Frame up to its last neighbor is summed up in the Correction field. Together with its own measured delay it gets the whole Delay that this frame has faced in the transmission from the Master to the Slave port.
A Slave can subtract this Correction together with its Delay value from its T2 timestamp and gets from there the Offset from the master.

2.2.2.3 E2E vs. P2P

They main advantage of E2E is that it works with non-PTP aware Switches (legacy or not feasible), which is often the case in Telecom or Office environments and that an E2E TC can be implemented very easily. Other than that E2E delay measurements has only drawbacks compared to P2P: E2E creates more network load and CPU load on the Master which means it doesn't scale well. E2E cannot react fast on Master switches, since it first has to measure the whole Delay chain again, where with P2P an immediate switchover can happen because of pre-measured delays and summing up of Delay and Residence values in the case of TCs. Also E2E measurement is not the preferred mechanism for Redundancy protocols like HSR and PRP and Ring topologies because this would require that Sync and Delay messages take the same way which is not always given.
So in general, whenever possible P2P delay measurement is the preferred mechanism.

2.2.3 Profiles

PTP comes in different flavors (Profiles), depending on the environment it shall be used in. Profiles define communication medium mappings (Ethernet, Profinet, and IR etc.), message rates, the delay mechanisms, default values of datasets and sometimes much more:

- Default Profile uses either Layer 2 or 3 with Multicast and either E2E or P2P Delay mechanism
- Power Profile uses Layer 2 with Multicast and P2P Delay mechanism and additional TLV
- Utility Profile uses also Layer 2 with Multicast and P2P Delay mechanism in combination with Redundancy Protocols like HSR or PRP
- TSN Profile uses Layer 2 with Multicast (with all PDelay MAC addresses) and P2P Delay mechanism, high Sync message rates and signaling messages and TC like BCs with syntonization
- ...

There are many other Profiles with other feature sets and mappings. Some Profiles are subsets of the Default Profile and compatible, some are supersets and therefor incompatible with other Profiles. This makes interoperability difficult.
So if you want to use PTP, first it is important to choose the right Profile for your application and second to make sure that all devices in the network support the chosen Profile

2.3 Accuracy

The accuracy of the synchronization depends highly on the precision of the timestamps.
They should reflect the send and receive time as precise as possible. The slave's Offset and Delay calculations are based on the difference of timestamps taken at two different places. Therefore, the two clocks should use the same scale, i.e. the same frequency.
This is achieved by Drift compensation: the Slave's clock rate is accelerated or slowed down by a control loop. A slightly different frequency will degrade the result.
It is assumed that the propagation Delay is the same for both directions. At a first glance, this is the case with an Ethernet link.
In the long run, conditions may change due to reconfiguration or environmental conditions (temperature).
How fast the clocks can react depends on the frequency of sync and delay measurement and the dynamic behavior of the servos controlling the Slave clock.

To sum things up, the achievable accuracy depends on:

- Timestamp accuracy
- Clock stability
- Sync interval
- Clock control loop characteristics
- Drift compensated clocks (i.e. adjusted time base in Master and Slave clocks)
- The communication channel symmetry (i.e. same delay in both directions and constant over a longer period of time)

2.3.1 Timestamp accuracy

As just stated timestamp accuracy is the key to high accuracy. PTP timestamp support can be implemented at different layers with decrease in accuracy in the

higher layers. For this solution a timestamp point between MAC and PHY (on MII) was chosen to get the best possible accuracy without implementing PHY functionality This interface is perfect for the use of FPGAs since it is a strictly digital interface, standardized and has only a low frequency requirement. This interface can either be intercept if one-step support is desired or passively listened if two-step support is sufficient. For this implementation the FPGA is intercepting the Path between MAC and PHY to provide one-step support.

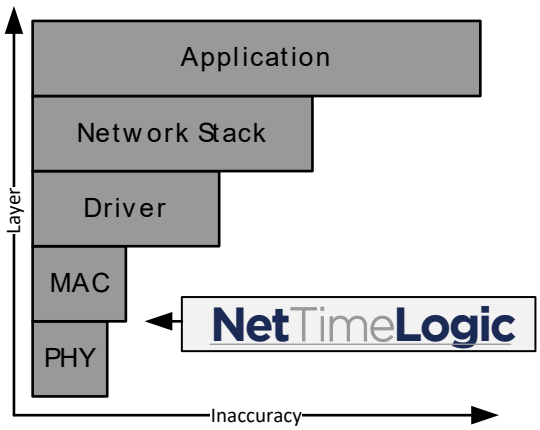

Figure 10: Timestamp Incauracy in the different Layers

Not only PTP timestamping but also frame generation, handling and servo loops can be done in different stages. Often a solution is to have the timestamping at a very low layer and all other things in the application layer which implies that a MAC, CPU and Operating System with Drivers a Network Stack and a PTP application is in place. NetTimeLogic's approach is different: NetTimeLogic's PTP cores are completely implemented in an FPGA means that all the upper layers after the PHY are not required. This decreases the complexity and dependencies between the different layers and allows running the system without CPU.

3 Register Set

This is the register set of the PTP Transparent Clock. It is accessible via AXI4Lite Memory Mapped. All registers are 32bit wide, no burst access, no unaligned access, no byte enables, no timeouts are supported. Register address space is not contiguous. Register addresses are only offsets in the memory area where the core is mapped in the AXI inter connects. Non existing register access in the mapped memory area is answered with a slave decoding error.

3.1 Register Overview

Registerset Overview			
Name	Description	Offset	Access
Ptp TcControl Reg	TC Enable Control Register	0x00000000	RW
Ptp TcStatus Reg	TC Error Status Register	0x00000004	WC
Ptp TcVersion Reg	TC Version Register	0x0000000C	RO
Ptp TcConfigControl Reg	TC Configuration Control Register	0x00000080	RW
Ptp TcConfigProfile Reg	TC Configuration Profile Register	0x00000084	RW
Ptp TcConfigVlanDscp Reg	TC Configuration VLAN Register	0x00000088	RW
Ptp TcConfigIp Reg	TC Configuration IPv6 0 and Ipv4 Register	0x0000008C	RW
Ptp TcConfigIpv61 Reg	TC Configuration IPv6 1 Register	0x00000090	RW
Ptp TcConfigIpv62 Reg	TC Configuration IPv6 2 Register	0x00000094	RW
Ptp TcConfigIpv63 Reg	TC Configuration IPv6 3 Register	0x00000098	RW
Ptp TcDefaultDsControl Reg	TC Default Dataset Control Register	0x00000100	RW
Ptp TcDefaultDs1 Reg	TC Default Dataset 1 Register	0x00000104	RW
Ptp TcDefaultDs2 Reg	TC Default Dataset 2 Register	0x00000108	RW
Ptp TcDefaultDs3 Reg	TC Default Dataset 3 Register	0x0000010C	RW
Ptp TcDefaultDs4 Reg	TC Default Dataset 4 Register	0x0000011C	RO
Ptp TcPortDsControl Reg	TC Port Dataset Control Register	0x00000200	RW

Ptp TcPortDs1 Reg	TC Port Dataset 1 Register	0x00000204	RO
Ptp TcPortDs2 Reg	TC Port Dataset 2 Register	0x00000208	RO
Ptp TcPortDs3 Reg	TC Port Dataset 3 Register	0x00000210	RW
Ptp TcPortDs4 Reg	TC Port Dataset 4 Register	0x0000021C	RW
Ptp TcPortDs5 Reg	TC Port Dataset 5 Register	0x00000220	RW
Ptp TcPortDs6 Reg	TC Port Dataset 6 Register	0x00000224	RW

Table 4: Register Set Overview

3.2 Register Descriptions

3.2.1 General

3.2.1.1 PTP TC Control Register

Used for general control over the PTP Transparent Clock, all configurations on the core shall only be done when disabled.

Ptp TcControl Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															ENABLE
RO																															RW
Reset: 0x00000000																															
Offset: 0x0000																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
ENABLE	Enable	Bit: 0	RW

3.2.1.2 PTP TC Status Register

Shows the current status of the PTP Transparent Clock.

Ptp TcStatus Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															ERROR
RO																															WC
Reset: 0x00000000																															
Offset: 0x0004																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
ENABLE	Error (sticky)	Bit: 0	WC

3.2.1.1 PTP TC Version Register

Version of the IP core, even though is seen as a 32bit value, bits 31 down to 24 represent the major, bits 23 down to 16 the minor and bits 15 down to 0 the build numbers.

Ptp TcVersion Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION																															
RO																															
Reset: 0xXXXXXXXX																															
Offset: 0x000C																															

Name	Description	Bits	Access
VERSION	Version of the core	Bit: 31:0	RO

3.2.1.2 PTP TC Config Control Register

Configuration valid bits, used to mark the corresponding fields as valid.

Ptp TcConfigControl Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RO																												DSCP_VAL	IP_VAL	VLAN_VAL	PROFILE_VAL
																												RW	RW	RW	RW
Reset: 0x00000000																															
Offset: 0x0080																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:4	RO
DSCP_VAL	DSCP valid (autocleared)	Bit: 3	RW
IP_VAL	IP valid (autocleared)	Bit: 2	RW
VLAN_VAL	VLAN valid (autocleared)	Bit: 1	RW
PROFILE_VAL	Profile valid (autocleared)	Bit: 0	RW

3.2.1.3 PTP TC Config Profile Register

PTP profile to run, changing this will automatically change clock parameters to the default values of the corresponding profile, these parameters can be overwritten afterwards. For the Default Profile also the layer mapping and delay mechanism can be chosen.

Ptp TcConfigProfile Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						DELAY_E2E_UNICAST	DELAY_MECHANISM	-							LAYER	-							TWO_STEP	PROFILE							
RO						RW	RW	RO							RW	RO							RW	RO						RW	
Reset: 0x00000000																															
Offset: 0x0084																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:26	RO
DELAY_E2E_UNICAST	Delay E2E Unicast (0 Multicast, 1 Unicast)	Bit:25	RW
DELAY_MECHANISM	Delay Mechanism (0 P2P, 1 E2E), only for Default Profile	Bit:24	RW
-	Reserved, read 0	Bit:23:17	RO
LAYER	Layer (0 802.3, 1 IPv4), only for Default Profile	Bit:16	RW

-	Reserved, read 0	Bit:15:9	RO
TWO_STEP	TwoStep for Sync and PDelay	Bit:8	RW
-	Reserved, read 0	Bit:7:2	RO
PROFILE	Profile (0 Default Profile, 1 Power Profile, 2 Utility Profile, 3 TSN Profile)	Bit:1:0	RW

3.2.1.4 PTP TC Config Vlan and DSCP Register

VLAN for 802.3q priority tagging or virtual networks. VLAN can be enabled or disabled for Power, Utility and TSN Profile. In Default Profile this is ignored.
Additionaly for Ipv4 the DSCP can be defined.

Ptp TcConfigVlanDscp Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						DSCP						-			VLAN_EN	VLAN															
RO						RW						RO			RW	RW															
Reset: 0x00000000																															
Offset: 0x0088																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:26	RO
DSCP	DSCP Classification	Bit: 25:20	RW
-	Reserved, read 0	Bit: 19:17	RO
VLAN_EN	VLAN enable (0 disabled, 1 enabled)	Bit: 16	RW
VLAN	VLAN (as in the TAG, combined Prio and VID)	Bit: 15:0	RW

3.2.1.5 PTP TC Config IP Register

IP address of the node. Used as source IP if in Default Profile and Layer 3 mappings, otherwise ignored. LSB is transferred first on the network. This is the full IP address for IPv4 and the highest part of IPv6

Ptp OcConfigIp Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP(3)								IP(2)								IP(1)								IP(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x008C																															

Name	Description	Bits	Access
IP(3)	IP Byte 3	Bit:31:24	RW
IP(2)	IP Byte 2	Bit:23:16	RW
IP(1)	IP Byte 1	Bit:15:8	RW
IP(0)	IP Byte 0	Bit:7:0	RW

3.2.1.6 PTP TC Config IP V6 1 Register

IPv6 address of the node. Used as source IP if in Default Profile and Layer 3 IPv6 mapping, otherwise ignored. LSB is transferred first on the network.

Ptp OcConfigIp Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP(7)								IP(6)								IP(5)								IP(4)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0090																															

Name	Description	Bits	Access
IP(7)	IP Byte 7 (Ipv6 only)	Bit:31:24	RW
IP(6)	IP Byte 6 (Ipv6 only)	Bit:23:16	RW
IP(5)	IP Byte 5 (Ipv6 only)	Bit:15:8	RW
IP(4)	IP Byte 4 (Ipv6 only)	Bit:7:0	RW

3.2.1.7 PTP TC Config IP V6 2 Register

IPv6 address of the node. Used as source IP if in Default Profile and Layer 3 IPv6 mapping, otherwise ignored. LSB is transferred first on the network.

Ptp OcConfigIp Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP(11)								IP(10)								IP(9)								IP(8)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0094																															

Name	Description	Bits	Access
IP(11)	IP Byte 11 (Ipv6 only)	Bit:31:24	RW
IP(10)	IP Byte 10 (Ipv6 only)	Bit:23:16	RW
IP(9)	IP Byte 9 (Ipv6 only)	Bit:15:8	RW
IP(8)	IP Byte 8 (Ipv6 only)	Bit:7:0	RW

3.2.1.8 PTP TC Config IP V6 3 Register

IPv6 address of the node. Used as source IP if in Default Profile and Layer 3 IPv6 mapping, otherwise ignored. LSB is transferred first on the network.

Ptp OcConfigIp Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP(15)								IP(14)								IP(13)								IP(12)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0098																															

Name	Description	Bits	Access
IP(15)	IP Byte 15 (Ipv6 only)	Bit:31:24	RW
IP(14)	IP Byte 14 (Ipv6 only)	Bit:23:16	RW
IP(13)	IP Byte 13 (Ipv6 only)	Bit:15:8	RW
IP(12)	IP Byte 12 (Ipv6 only)	Bit:7:0	RW

3.2.2 Default Dataset

Parameters from the PTP Default Dataset according to IEEE1588-2019/2008 Clause 8.2.1.

3.2.2.1 PTP TC Default Dataset Control Register

Configuration valid bits, used to mark the corresponding fields as valid. Additional flags to snapshot the current Default Dataset: set READ flag and check for READ_DONE flag set.

Ptp TcDefaultDsControl Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READ_DONE	READ	-																												DOMAIN_NR_VAL	CLOCK_ID_VAL
RO	RW	RO																												RW	RW
Reset: 0x00000000																															
Offset: 0x0100																															

Name	Description	Bits	Access
READ_DONE	Default Dataset was read	Bit: 31	RO
READ	Read Default Dataset (autocleared)	Bit: 30	RW
-	Reserved, read 0	Bit 29:2	RO

DOMAIN_NR_VAL	Domain Number valid (autocleared)	Bit: 1	RW
CLOCK_ID_VAL	Clock Identity valid (autocleared)	Bit: 0	RW

3.2.2.2PTP TC Default Dataset 1 Register

First 4 bytes of the Clock Identity of the node. CLOCK_ID(2..0) are also the first bytes of the MAC address of the node, LSB first. E.g. 0x01234567 => MAC: 67:45:32:XX:XX:XX. MAC48 to UID64 CLOCK_ID(3) should be set to FF.

Ptp TcDefaultDs1 Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLOCK_ID(3)								CLOCK_ID(2)								CLOCK_ID(1)								CLOCK_ID(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0104																															

Name	Description	Bits	Access
CLOCK_ID(3)	Clock ID Byte 3	Bit:31:24	RW
CLOCK_ID(2)	Clock ID Byte 2	Bit:23:16	RW
CLOCK_ID(1)	Clock ID Byte 1	Bit:15:8	RW
CLOCK_ID(0)	Clock ID Byte 0	Bit:7:0	RW

3.2.2.3PTP TC Default Dataset 2 Register

Second 4 bytes of the Clock Identity of the node. CLOCK_ID(7..5) are also the last bytes of the MAC address of the node, LSB first. E.g. 0x01234567 => MAC: XX:XX:XX:45:23:01. MAC48 to UID64 CLOCK_ID(4) should be set to FE.

Ptp TcDefaultDs2 Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLOCK_ID(7)								CLOCK_ID(6)								CLOCK_ID(5)								CLOCK_ID(4)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0108																															

Name	Description	Bits	Access
CLOCK_ID(7)	Clock ID Byte 7	Bit:31:24	RW
CLOCK_ID(6)	Clock ID Byte 6	Bit:23:16	RW
CLOCK_ID(5)	Clock ID Byte 5	Bit:15:8	RW
CLOCK_ID(4)	Clock ID Byte 4	Bit:7:0	RW

3.2.2.4 PTP TC Default Dataset 3 Register

Domain to run on, only one is supported.

Ptp TcDefaultDs3 Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								DOMAIN_NR							
RO																								RW							
Reset: 0x00000000																															
Offset: 0x010C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
DOMAIN_NR	Domain Number	Bit: 7:0	RW

3.2.2.5PTP TC Default Dataset 4 Register

Number of ports, for an TC this is defined by the number of ports used (3 for the reference design).

Ptp TcDefaultDs7 Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																NUMBER_OF_PORTS															
RO																RO															
Reset: 0x0000000X																															
Offset: 0x011C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
NUMBER_OF_PORTS	Number of ports of the TC	Bit: 15:0	RO

3.2.3 Port Dataset

Parameters from the PTP Port Dataset according to IEEE1588-2019/2008 Clause 8.2.5.

3.2.3.1 PTP TC Port Dataset Control Register

Flags to snapshot the current Port Dataset: set READ flag and check for READ_DONE flag set.

Ptp TcPortDsControl Reg																												
Reg Description																												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
READ_DONE	READ	-							PORT_SELECT							-											VLAN_MODE_VAL	MAX_PDELAY_VAL	ASYMMETRY_VAL	MSG_INTERVAL_VAL	DELAY_MECHANISM VAL
RO	RW	RO							RW							RO											RW	RW	RW	RW	RW
Reset: 0x00000000																															
Offset: 0x0200																															

Name	Description	Bits	Access
READ_DONE	Port Dataset was read	Bit: 31	RO
READ	Read Port Dataset (autocleared)	Bit: 30	RW
-	Reserved, read 0	Bit 29:24	RO
PORT_SELECT	Selected Port where the values shall go to or come from	Bit: 23:16	RW

-	Reserved, read 0	Bit 15:5	RO
VLAN_MODE_VAL	VLAN mode (insert/remove) valid (autocleared)	Bit 4	RW
MAX_PDELAY_VAL	Maximum Peer Delay valid (autocleared)	Bit 3	RW
ASYMMETRY_VAL	Asymmetry valid	Bit 2	RW
MSG_INTERVAL_VAL	Message Intervals valid	Bit 1	RW
DELAY_MECHANISM_VAL	Delay Mechanism valid	Bit 0	RW

3.2.3.2PTP TC Port Dataset 1 Register

Higher 32 bits of the current measured Peer Delay on this port. This is in scaled nanoseconds format.

Ptp TcPortDs1 Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEER_DELAY(63:32)																														
RO																														
Reset: 0x00000000																														
Offset: 0x0204																														

Name	Description	Bits	Access
PEER_DELAY(63:32)	Peer Mean Path Delay higher 32bits	Bit: 31:0	RO

3.2.3.3PTP TC Port Dataset 2 Register

Lower 32 bits of the current measured Peer Delay on this port. This is in scaled nanoseconds format.

Ptp TcPortDs2 Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEER_DELAY(31:0)																														
RO																														
Reset: 0x00000000																														
Offset: 0x0208																														

Name	Description	Bits	Access
PEER_DELAY(31:0)	Peer Mean Path Delay lower 32bits	Bit: 31:0	RO

3.2.3.4 PTP TC Port Dataset 3 Register

Signed Peer Delay Request interval (only available when DynamicMessageRatesSupport_Gen is true)

Ptp TcPortDs3 Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																								PDELAYREQ_INTERVAL							
RO																								RW							
Reset: 0x00000000																															
Offset: 0x0210																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
PDELAYREQ_INTERVAL	Signed Peer Delay Request Log interval	Bit: 7:0	RW

3.2.3.5PTP TC Port Dataset 4 Register

Asymmetry as signed nanoseconds.

Ptp TcPortDs4 Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYMMETRY																														
RW																														
Reset: 0x00000000																														
Offset: 0x021C																														

Name	Description	Bits	Access
ASYMMETRY	Signed Asymmetry in Nanoseconds	Bit: 31:0	RW

3.2.3.6PTP TC Port Dataset 5 Register

Maximum Peer Delay which is still considered valid as nanoseconds. This is only used for IEEE802.1AS (TSN Profile)

Ptp TcPortDs5 Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAX_PDELAY																														
RW																														
Reset: 0x00000000																														
Offset: 0x0220																														

Name	Description	Bits	Access
MAX_PDELAY	Maximum Pdelay which is still valid (only for TSN)	Bit: 31:0	RW

3.2.3.7PTP TC Port Dataset 6 Register

Remove or Insert a VLAN tag on transmission or reception

Ptp TcPortDs6 Reg																														
Reg Description																														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																														INSERT_VLAN	REMOVE_VLAN
RO																														RW	RW
Reset: 0x00000000																															
Offset: 0x0224																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit 31:2	RO
INSERT_VLAN	Insert VLAN tag on reception	Bit: 1	RW
REMOVE_VLAN	Remove VLAN tag on sending	Bit: 0	RW

4 Design Description

The following chapters describe the internals of the PTP Transparent Clock: starting with the Top Level, which is a collection of subcores, followed by the description of all subcores.

4.1 Top Level – PTP Transparent Clock

4.1.1.1 Parameters

The core must be parametrized at synthesis time. There are a couple of parameters which define the final behavior and resource usage of the core.

Name	Type	Size	Description
DefaultProfileSupport_Gen	boolean	1	If the core shall support the PTP Default Profile
PowerProfileSupport_Gen	boolean	1	If the core shall support the PTP Power Profile
UtilityProfileSupport_Gen	boolean	1	If the core shall support the PTP Utility Profile
TsnProfileSupport_Gen	boolean	1	If the core shall support the PTP TSN Profile
E2eSupport_Gen	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
E2eUnicastSupport_Gen	boolean	1	If the core shall support E2E unicast delay mechanism (only valid with default profile support and E2E support)
P2pSupport_Gen	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
Layer2Support_Gen	boolean	1	If in Default Profile if Layer 2 shall be supported (Power and Utility Profile always use Layer 2)
Layer3v4Support_Gen	boolean	1	If in Default Profile if Layer 3 Ipv4 shall be supported (Pow-

			er and Utility Profile always use Layer 2)
Layer3v6Support_Gen	boolean	1	If in Default Profile if Layer 3 Ipv6 shall be supported (Power and Utility Profile always use Layer 2)
AsymmetrySupport_Gen	Boolean	1	If the core shall also support asymmetry corrections
TwoStepSupport_Gen	Boolean	1	If the core shall also support to generate TwoStep Sync and PDelay
StaticConfig_Gen	boolean	1	If Static Configuration or AXI is used: true = Static, false = AXI
ClockClkPeriod Nanosecond_Gen	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
ClockClkPeriod FractNum_Gen	natural	1	Fractional Clock Period Numerator (0 if integer)
ClockClkPeriod FractDeNum_Gen	natural	1	Fractional Clock Period Denumerator (0 if integer)
AverageWindow Nanosecond_Gen	natural	1	Averaging window for the delay measurement. If a new measurement is out of that range averaging is skipped
Port1RxDelayNanosecond10_Gen	integer	1	PHY receive delay (10Mbit)
Port1RxDelayNanosecond100_Gen	integer	1	PHY receive delay (100Mbit)
Port1RxDelayNanosecond1000_Gen	integer	1	PHY receive delay (1000Mbit)
Port1TxDelayNanosecond10_Gen	integer	1	PHY transmit delay (10Mbit)
Port1TxDelayNanosecond100_Gen	integer	1	PHY transmit delay (100Mbit)
Port1TxDelayNanosecond1000_Gen	integer	1	PHY transmit delay (1000Mbit)
Port2RxDelayNanosecond10_Gen	integer	1	PHY receive delay (10Mbit)

Port2RxDelayNanosecond100_Gen	integer	1	PHY receive delay (100Mbit)
Port2RxDelayNanosecond1000_Gen	integer	1	PHY receive delay (1000Mbit)
Port2TxDelayNanosecond10_Gen	integer	1	PHY transmit delay (10Mbit)
Port2TxDelayNanosecond100_Gen	integer	1	PHY transmit delay (100Mbit)
Port2TxDelayNanosecond1000_Gen	integer	1	PHY transmit delay (1000Mbit)
Port3RxDelayNanosecond10_Gen	integer	1	PHY receive delay (10Mbit)
Port3RxDelayNanosecond100_Gen	integer	1	PHY receive delay (100Mbit)
Port3RxDelayNanosecond1000_Gen	integer	1	PHY receive delay (1000Mbit)
Port3TxDelayNanosecond10_Gen	integer	1	PHY transmit delay (10Mbit)
Port3TxDelayNanosecond100_Gen	integer	1	PHY transmit delay (100Mbit)
Port3TxDelayNanosecond1000_Gen	integer	1	PHY transmit delay (1000Mbit)
Port1IoFf_Gen	boolean	1	If the interface adapter shall contain an IO flip flop
Port2IoFf_Gen	boolean	1	If the interface adapter shall contain an IO flip flop
Port3IoFf_Gen	boolean	1	If the interface adapter shall contain an IO flip flop
Port1PortIdModify_Gen	boolean	1	If the Port ID shall be modified on the received frames
Port2PortIdModify_Gen	boolean	1	If the Port ID shall be modified on the received frames
Port3PortIdModify_Gen	boolean	1	If the Port ID shall be modified on the received frames
AxiAddressRangLow_Gen	std_logic_vector	32	AXI Base Address
AxiAddressRangeHigh_Gen	std_logic_vector	32	AXI Base Address plus Register-set Size

			Default plus 0xFFFF
ManagementSup-port_Gen	boolean	1	If the core shall handle PTP Management messages
ManufacturerIdenti-ty_Gen	Common_ Byte_Type	3	3 byte array for the Manufac-turer Identity => NTL in hex 0x4E, 0x54, 0x4C
ProductDescrip-tion_Gen	string	32	String of the Products De-scription
RevisionData_Gen	string	32	String of the Products Revi-sion
UserDescription_Gen	string	32	String of the Products User Description
TimeInaccuracyNano-second_Gen	natural	1	The Inaccuracy that shall be added to the PTP frames when forwarded
Sim_Gen	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis

Table 5: Parameters

One of the three parameters DefaultSupport_Gen, PowerProfileSupport_Gen, UtilityProfileSupport_Gen and TsnProfileSupport_Gen has to be true.

4.1.1.2 Structured Types

4.1.1.2.1 Clk_Time_Type

Defined in Clk_Package.vhd of library ClkLib
Type represents the time used everywhere. For this type overloaded operators + and – with different parameters exist.

Field Name	Type	Size	Description
Second	std_logic_vector	32	Seconds of time
Nanosecond	std_logic_vector	32	Nanoseconds of time
Fraction	std_logic_vector	2	Fraction numerator (mostly not used)
Sign	std_logic	1	Positive or negative time, 1 =

			negative, 0 = positive.
TimeJump	std_logic	1	Marks when the clock makes a time jump (mostly not used)

Table 6: Clk_Time_Type

4.1.1.2.2 Clk_TimeAdjustment_Type

Defined in Clk_Package.vhd of library ClkLib
Type represents the time used everywhere. For this type overloaded operators + and – with different parameters exist.

Field Name	Type	Size	Description
TimeAdjustment	Clk_Time_Type	1	Time to adjust
Interval	std_logic_vector	32	Adjustment interval, for the drift correction this is the denumerator of the rate in nanoseconds (TimeAdjustment every Interval = drift rate), for offset correction this is the period in which the time shall be corrected(TimeAdjustment in Interval), for setting the time this has no mining.

Table 7: Clk_TimeAdjustment_Type

4.1.1.2.3 Ptp_TransparentClockStaticConfig_Type

Defined in Ptp_TransparentClockAddrPackage.vhd of library PtpLib
This is the type used for static configuration.

Field Name	Type	Size	Description
Profile	Ptp_Profile_Type	1	Which Profile the core shall be run in: DefaultProfile_E PowerProfile_E UtilityProfile_E

			TsnProfile_E
Layer	Ptp_Layer_Type	1	Which layer shall be used in Default Profile: Layer2_E Layer3v4_E Layer3v6_E
DelayMechanism	Ptp_Delay Mechanism_Type	1	Which layer shall be used in Default Profile: P2p_E E2e_E
DelayE2eUnicast	std_logic	1	If E2E unicast messages shall be handled
TwoStep	std_logic	1	If TwoStep shall be used
Vlan	Ptp_Vlan_Type	1	The Pcp,Dei and Vid of the VLAN
VlanEnable	std_logic	1	If VLAN shall be used
VlanRemove	std_logic	1	If on transmission removing of a VLAN shall be done
VlanInsert	std_logic	1	If on reception inserting of a VLAN shall be done
Ip	Com-mon_Byte_Type	16	The source IP to be used if in Layer3 mode, 4 bytes Ipv4, 16 bytes Ipv6, index 0 = MSB
DefaultDataset_ ClockIdentity	Ptp_ClockIdentit y_Type	1	The Clock Identity of the clock, also used for the MAC (with-out bytes 3&4)
DefaultDataset_ DomainNumber	std_logic_vector	8	Which PTP Domain the core shall run on

Table 8: Ptp_TransparentClockStaticConfig_Type

4.1.1.2.4 Ptp_TransparentClockStaticConfigVal_Type

Defined in Ptp_TransparentClockAddrPackage.vhd of library PtpLib
This is the type used for valid flags of the static configuration.

Field Name	Type	Size	Description
Enable_Val	std_logic	1	Enables the PTP Transparent

			Clock
Profile_Val	std_logic	1	If the Profile shall be set
Vlan_Val	std_logic	1	If the VLAN shall be set
VlanMode_Val	std_logic	1	If the VLAN Remove and Insert shall be set
Ip_Val	std_logic	1	If the IP shall be set
DefaultDataset_ ClockIdentity_Val	std_logic	1	If the ClockIdentity shall be set
DefaultDataset_ DomainNumber_Val	std_logic	1	If the DomainNumber shall be set

Table 9: Ptp_TransparentClockStaticConfigVal_Type

4.1.1.2.5 Ptp_TransparentClockStaticStatus_Type

Defined in Ptp_TransparentClockAddrPackage.vhd of library PtpLib
This is the type used for static status supervision.

Field Name	Type	Size	Description
CoreInfo	Clk_CoreInfo_ Type	1	Infor about the Cores state
DefaultDataset_Profile	Ptp_Profile_Type	1	Which Profile the Core runs in
DefaultDataset_Layer	Ptp_Layer_type	1	Which Transport Layer it uses
DefaultDataset_Vlan	Ptp_Vlan_Type	1	VLAN Tag that the Core uses
DefaultDataset_ VlanEnable	std_logic	1	If the Core uses VLAN Tags
DefaultDataset_Ip	Common_ Byte_Type	4	Which Source IP the Core uses
DefaultDataset_ TwoStepFlag	std_logic	1	If the Core runs in Twostep mode
DefaultDataset_ ClockIdentity	Ptp_Clock Identity_Type	1	Clockidentity of the Core
DefaultDataset_ NumberPorts	std_logic_vector	16	Number of Ports of the TC
DefaultDataset_ DomainNumber	std_logic_vector	8	Domain Number the Core runs on

Table 10: Ptp_TransparentClockStaticConfig_Type

4.1.1.2.6 Ptp_TransparentClockStaticStatusVal_Type

Defined in Ptp_TransparentClockAddrPackage.vhd of library PtpLib
This is the type used for valid flags of the static status supervision.

Field Name	Type	Size	Description
CoreInfo_Val	std_logic	1	Core Info valid

Table 11: Ptp_TransparentClockStaticConfigVal_Type

4.1.1.3 Entity Block Diagram

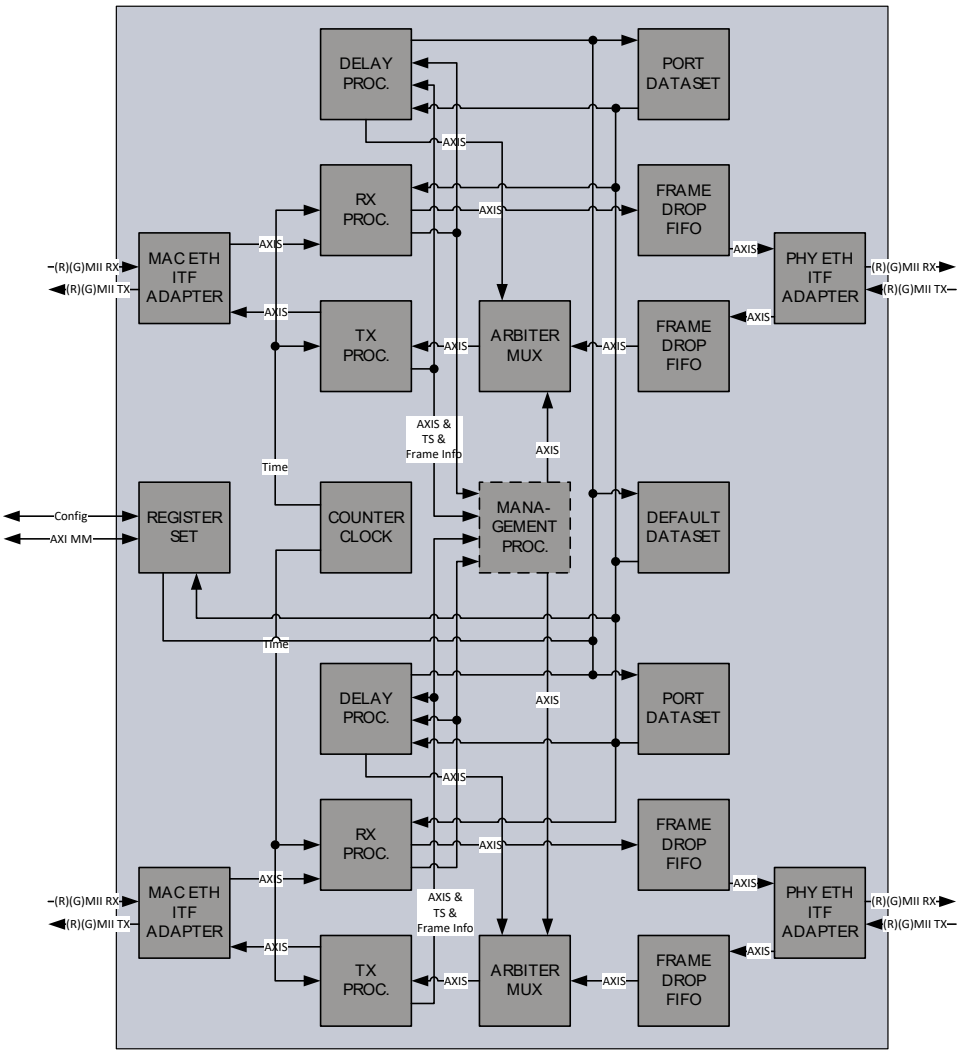

Figure 11: PTP Transparent Clock

4.1.1.4 Entity Description

Frame Arbiter & Mux

This module multiplexes multiple AXI stream inputs into one AXI stream output. Multiplexing is done on stream base, which means that once a stream has been started it will be the only stream forwarded until a TLAST is asserted, and then arbitration starts again. It handles two different priority levels. Each priority level is arbitrated in a round robin manner. Low priority streams can starve; high priority streams never starve.

Frame Drop Fifo

This module is AXI stream frame FIFO. It is always ready to receive data, so no throttling of the stream is necessary. This on the other hand allows overflow conditions. The FIFO is large enough that it can contain at least two maximum sized frames. This is important, since when it runs into an overflow condition the current input frame shall be dropped. It can be configured either as cut-through (frame is forwarded immediately) or store-and-forward (whole frame is stored first before forwarding) FIFO. When run in store-and-forward mode it handles additionally a user drop-flag which allows explicitly dropping the current frame. In this design it is only used as store-and-forward FIFO, but with the drop-flag. The FIFO on the RX path has a couple of functionalities, it allows to drop PTP and corrupted frames (they shall not be forwarded), it allows a deterministic behavior of the RX Processor and it compensates the speed differences between incoming and outgoing data streams. The FIFO on the TX path has also similar functionalities, it buffers the incoming data stream for the case when an internal frame is processed (send DelayReq, DelayResp, Sync, Announce etc..), it allows a deterministic behavior (no starving) of the TX Processor and it compensates the speed differences between incoming and outgoing data streams.

Rx Processor

This module handles all incoming frames. It does the RX timestamping, frame parsing, on-the-fly CorrectionField processing (adding PortDelay and/or subtracting RX timestamps) and is the source of the drop-flag going to the RX Frame Drop Fifo for PTP frame filtering. In addition it realigns the 32bit stream so the PTP frames are always aligned the same way in case an IP header or HSR tag is inserted.
See 4.2.1 for more details.

Tx Processor

This module handles all outgoing frames. It does the TX timestamping; frame parsing, on-the-fly CorrectionField and TimestampField processing (adding or inserting TX timestamps). In addition it realigns the 32bit stream so the PTP frames are always aligned the same way in case an IP header or HSR tag is inserted.
See 4.2.2 for more details.

Delay Processor

This module handles all PTP Delay (P2P only, no E2E support) frames. It runs independent of the current state of the PTP Transparent Clock. It has a Client and Server functionality. As Client it periodically generates PeerDelayRequest frames based on the parameters of the Datasets and waits for the other peer to respond with a PeerDelayResponse. Based on the information and timestamps of the frames the Client calculates the PeerDelay averages it and stores it in a Dataset.
As Server it waits for incoming PeerDelayRequests and answers them with corresponding PeerDelayResponses.
See 4.2.3 for more details.

Management Processor

This module handles all PTP Management frames. It is optional. It runs independent of the current state of the PTP Transparent Clock. It acts as a Server, so it waits for incoming Management frames and answers them with corresponding Management frames. This allows remote configuration and supervision based on PTP.
See 4.2.4 for more details.

Datasets

This module contains the storage of all PTP Datasets: Default-, Port-, Parent-, Current-, TimeProperties- and ForeignMaster-Dataset. All values of the Datasets are writable.
See 4.2.5 for more details.

Clock Counter

This is a free-running counter with nanosecond resolution in a 32 bit second and 32 bit nanosecond format. It is used by the Delay and Residence time measurements.
See 4.2.6 for more details.

MAC & PHY Ethernet Interface Adapter

This module converts the Media Independent Interface (MII) to AXI stream and vice versa. In parallel it has a SFD detector for each path which generates an event

when a SFD is detected; this is used for timestamping in the RX/TX Processors. It is also in charge of generating correct Interframe Gaps and a Preamble with SFD. See 4.2.7 for more details.

Registerset

This module is an AXI4Lite Memory Mapped Slave. It provides access to all Datasets and allows to configure the PTP Transparent Clock. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the Datasets is done via signals and can be easily done from within the FPGA without CPU. If in AXI mode, a AXI Master has to configure the Datasets with AXI writes to the registers, which is typically done by a CPU
See 4.2.8 for more details.

4.1.1.5 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile
PowerProfile Support_Gen	-	boolean	1	Support for Power Profile
UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile
E2eSupport_Gen	-	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
P2pSupport_Gen	-	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
E2eUnicastSupport_G	-	boolean	1	E2E Unicast han-

en				dling support
Layer2Support_Gen	-	boolean	1	Support for Layer 2 Mapping
Layer3v4 Support_Gen	-	boolean	1	Support for Layer 3 Ipv4 Mapping
Layer3v6 Support_Gen	-	boolean	1	Support for Layer 3 Ipv6 Mapping
TwoStepSupport_Gen	-	Boolean	1	Support for TwoStep frames
StaticConfig_Gen	-	boolean	1	If Static Configura-tion or AXI is used
ClockClkPeriodNano-second_Gen	-	natural	1	Integer Clock Period
ClockClkPeriodFract-Num_Gen	-	natural	1	Fractional Clock Period Numerator
ClockClkPeriod-FractDeNum_Gen	-	natural	1	Fractional Clock Period Denumerator
AverageWindow Nanosecond_Gen	-	natural	1	Delay Averaging window, out of this window, delay is set hard
Port1RxDelayNano second10_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port1RxDelayNano second100_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port1RxDelayNano second1000_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port1TxDelayNano second10_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port1TxDelayNano second100_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port1TxDelayNano second1000_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port2RxDelayNano second10_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port2RxDelayNano second100_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port2RxDelayNano	-	integer	1	RX Delay of the PHY

second1000_Gen				in Nanosecond
Port2TxDelayNano second10_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port2TxDelayNano second100_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port2TxDelayNano second1000_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port3RxDelayNano second10_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port3RxDelayNano second100_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port3RxDelayNano second1000_Gen	-	integer	1	RX Delay of the PHY in Nanosecond
Port3TxDelayNano second10_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port3TxDelayNano second100_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port3TxDelayNano second1000_Gen	-	integer	1	TX Delay of the PHY in Nanosecond
Port1IoFf_Gen	-	boolean	1	Shall IO flip flops be instantiated
Port2IoFf_Gen	-	boolean	1	Shall IO flip flops be instantiated
Port3IoFf_Gen	-	boolean	1	Shall IO flip flops be instantiated
Port1PortId Modify_Gen	-	boolean	1	Shall the port Id be modified on the fly
Port2PortId Modify_Gen	-	boolean	1	Shall the port Id be modified on the fly
Port3PortId Modify_Gen	-	boolean	1	Shall the port Id be modified on the fly
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
ManagementSup-port_Gen	-	boolean	1	Should Management Messages be han-

				dled
ManufacturerIdenti-ty_Gen	-	Common_Byte_Type	3	Manufacturing Identity string
ProductDescrip-tion_Gen	-	string	32	Product Description string
RevisionData_Gen	-	string	32	Product Revision string
UserDescription_Gen	-	string	32	User Description string
TimeInaccuracy Nanosecond_Gen	-	natural	1	What shall be added to the Announce when in Power Profile
Sim_Gen	-	boolean	1	If in Testbench simulation mode
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Ptp_TransparentClock StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Ptp_TransparentClock StaticConfigVal_Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Ptp_TransparentClock StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Ptp_TransparentClock StaticStatusVal_Type	1	Static Status valid
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Unused for TC
ClockTime_ValIn	in	std_logic	1	Unused for TC
Timer				

Timer1ms_EvtIn	in	std_logic	1	1 millisecond Timer event
Redundancy Input				
RedMode_DatIn	in	Ptp_RedMode_Type	1	Redundancy Mode: Hsr_E Prp_E No_E
RedSeqResp_DatIn	in	Ptp_RedSeqResp Array_Type	6	Sequence Number Response
RedSeqResp_ValIn	in	Ptp_RedSeqRespVal Array_Type	6	Sequence Number Response valid
Redundancy Output				
RedSeqReq_DatOut	out	Ptp_RedSeqReq Array_Type	6	Sequence Number Request
RedSeqReq_ValOut	out	Ptp_RedSeqReqVal Array_Type	6	Sequence Number Request valid
Port1(R)(G)Mii RX Clk/Rst Input				
Port1(R)(G)MiiRxClk_ClkIn	in	std_logic	1	RX Clock
Port1(R)(G)MiiRxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
Port1(R)(G)Mii TX Clk/Rst Input				
Port1(R)(G)MiiTxClk_ClkIn	in	std_logic	1	RX Clock
Port1(R)(G)MiiTxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
Port1(R)(G)Mii RX Data Input/Output				
Port1(R)(G)MiiRxDv_Ena	in/ out	std_logic	1	RX Data valid
Port1(R)(G)MiiRxErr_Ena	in/ out	std_logic	1	RX Error
Port1(R)(G)MiiRxData_Dat	in/ out	std_logic_vector	2-8	RX Data MII:4, RMII:2, GMII:8, RGMII:4
Port1(R)(G)MiiCol_Dat	in/ out	std_logic	1	Collision
Port1(R)(G)MiiCrs_Dat	in/ out	std_logic	1	Carrier Sense
Port1(R)(G)Mii TX Data Input/Output				
Port1(R)(G)MiiTxEn_Ena	in/ out	std_logic	1	TX Data valid

Port1(R)(G)MiiTxErr_Ena	in/out	std_logic	1	TX Error
Port1(R)(G)MiiTxData_Dat	in/out	std_logic_vector	2-8	TX Data MII:4, RMII:2, GMII:8, RGMII:4
Port2(R)(G)Mii RX Clk/Rst Input				
Port2(R)(G)MiiRxClk_ClkIn	in	std_logic	1	RX Clock
Port2(R)(G)MiiRxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
Port2(R)(G)Mii TX Clk/Rst Input				
Port2(R)(G)MiiTxClk_ClkIn	in	std_logic	1	RX Clock
Port2(R)(G)MiiTxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
Port2(R)(G)Mii RX Data Input/Output				
Port2(R)(G)MiiRxDv_Ena	in/out	std_logic	1	RX Data valid
Port2(R)(G)MiiRxErr_Ena	in/out	std_logic	1	RX Error
Port2(R)(G)MiiRxData_Dat	in/out	std_logic_vector	2-8	RX Data MII:4, RMII:2, GMII:8, RGMII:4
Port2(R)(G)MiiCol_Dat	in/out	std_logic	1	Collision
Port2(R)(G)MiiCrs_Dat	in/out	std_logic	1	Carrier Sense
Port2(R)(G)Mii TX Data Input/Output				
Port2(R)(G)MiiTxEn_Ena	in/out	std_logic	1	TX Data valid
Port2(R)(G)MiiTxErr_Ena	in/out	std_logic	1	TX Error
Port2(R)(G)MiiTxData_Dat	in/out	std_logic_vector	2-8	TX Data MII:4, RMII:2, GMII:8, RGMII:4
Port3(R)(G)Mii RX Clk/Rst Input				
Port3(R)(G)MiiRxClk_ClkIn	in	std_logic	1	RX Clock
Port3(R)(G)MiiRxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
Port3(R)(G)Mii TX Clk/Rst Input				

Port3(R)(G)MiiTxClk_ClkIn	in	std_logic	1	RX Clock
Port3(R)(G)MiiTxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
Port3(R)(G)Mii RX Data Input/Output				
Port3(R)(G)MiiRxDv_Ena	in/ out	std_logic	1	RX Data valid
Port3(R)(G)MiiRxErr_Ena	in/ out	std_logic	1	RX Error
Port3(R)(G)MiiRxData_Dat	in/ out	std_logic_vector	2-8	RX Data MII:4, RMII:2, GMII:8, RGMII:4
Port3(R)(G)MiiCol_Dat	in/ out	std_logic	1	Collision
Port3(R)(G)MiiCrs_Dat	in/ out	std_logic	1	Carrier Sense
Port3(R)(G)Mii TX Data Input/Output				
Port3(R)(G)MiiTxEn_Ena	in/ out	std_logic	1	TX Data valid
Port3(R)(G)MiiTxErr_Ena	in/ out	std_logic	1	TX Error
Port3(R)(G)MiiTxData_Dat	in/ out	std_logic_vector	2-8	TX Data MII:4, RMII:2, GMII:8, RGMII:4
AXI4 Lite Slave				
AxiWriteAddrValid_ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady_RdyOut	out	std_logic	1	Write Address Ready
AxiWriteAddrAddress_AdrIn	in	std_logic_vector	32	Write Address
AxiWriteAddrProt_DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid_ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady_RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData_DatIn	in	std_logic_vector	32	Write Data
AxiWriteDataStrobe_DatIn	in	std_logic_vector	4	Write Data Strobe
AxiWriteRespValid_ValOut	out	std_logic	1	Write Response Valid

AxiWriteRespReady_RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid_ValIn	in	std_logic	1	Read Address Valid
AxiReadAddrReady_RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress_AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt_DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid_ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady_RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData_DatOut	out	std_logic_vector	32	Read Data Re-sponse

Table 12: PTP Transparent Clock

4.2 Design Parts

The PTP Transparent Clock core consists of a couple of subcores. Each of the subcores itself consist again of smaller function block. The following chapters describe these subcores and their functionality.

4.2.1 RX Processor

4.2.1.1 Entity Block Diagram

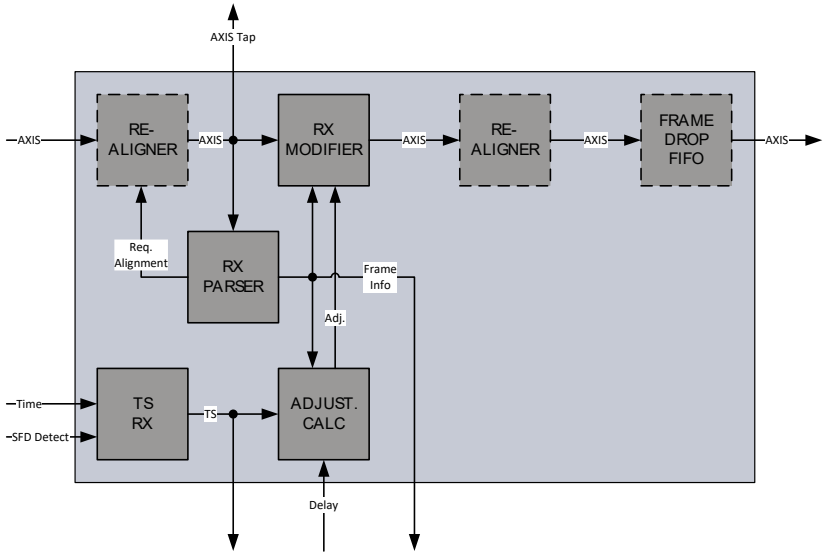

Figure 12: RX Processor

4.2.1.2 Entity Description

RX Timestamper

This module takes a snapshot of the free-running Counter Clock and of the PTP Adjusted Clock when the SFD detected event was asserted by the Interface Adapter. The timestamp is buffered and aligned with the incoming frame, so the Adjustment Calculator can calculate the correct adjustments.

RX Parser

This module parses all incoming Ethernet frames. It extracts the frame length, in the case of PTP frames also the CorrectionField and NetworkTimeInaccuracy and it checks the CRC. If Layer 3 mapping or the Utility or TSN Profile is used it request a specific alignment from the Realigner so the PTP frame is starting 32 bit aligned again.

RX Modifier

This module is the heart of the RX processor. It also parses the frame, waits for the CorrectionField in the frame. It then replaces the CorrectionField with the one from the Adjustment Calculation. In the case of Utility Profile it also modifies the Source Port Identity of PTP frames according to the Stateless Redbox defined in IEC62439. Since the frame is modified on-the-fly the CRC and UDP checksum (IPv6) is recalculated and overwritten if the incoming CRC was correct otherwise the frames are dropped. PTP frames shall not be forwarded to the MAC, therefore it drops the frames. This is the same Modifier as in the Transparent Clock but with a different configuration. If no PeerDelay was calculated yet, all PTP frames are dropped.

Adjustment Calculation

This module calculates the new CorrectionField out of the incoming Correction-Field, the receive timestamp (subtracted) and the PeerDelay (added). For the TransparentClock this is not used, but as with the Modifier this is the same module for the Transparent Clock.

Re-Aligner

This module allows aligning the AXI Data stream as required. It is only instantiated if Layer 3 or Utility or TSN mapping is used. The input can be 32/24/16/8 bit aligned and the output can also be 32/24/16/8 bit aligned. From an external input, the output alignment can be requested. This is used for alignment of the PTP start to a 32 bit boundary (Some mappings cause that the header is not 32 bit aligned anymore) again, and for realignment to a constant 32bit wide stream again.

Frame Drop Fifo

This is an optional Frame Drop Fifo to overcome additional data speed differences. It is not used in the Transparent Clock.

4.2.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile
PowerProfile	-	boolean	1	Support for Power

Support_Gen				Profile
UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile
Layer2Support_Gen	-	boolean	1	Support for Layer 2 Mapping
Layer3v4 Support_Gen	-	boolean	1	Support for Layer 3 Ipv4 Mapping
Layer3v6 Support_Gen	-	boolean	1	Support for Layer 3 Ipv6 Mapping
UnicastSupport_Gen	-	boolean	1	Support for Unicast Messages
E2eSupport_Gen	-	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
P2pSupport_Gen	-	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
TwoStepSupport_Gen	-	Boolean	1	Support for TwoStep frames
Asymmetry Support_Gen	-	Boolean	1	Support for Asymmetry corrections
InsertVlanSupport_Gen	-	Boolean	1	If on reception inserting of a VLAN shall be supported
ClockType_Gen	-	Ptp_ClockType_Typ	1	What Kind of Clock this is (TC)
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond
RX Processor				
DelayNanosecond10_Gen	-	integer	1	Input Delay of the PHY in Nanosecond
DelayNanosecond100	-	integer	1	Input Delay of the

_Gen				PHY in Nanosecond
DelayNano second1000_Gen	-	integer	1	Input Delay of the PHY in Nanosecond
PortIdModify_Gen	-	boolean	1	If the ID shall be modified on the fly
PortId_Gen	-	std_logic_vector	4	Which port id shall be set
Buffer_Gen	-	boolean	1	If a frame drop fifo shall be instantiated
TimeInaccuracy Nanosecond_Gen	-	natural	1	Inaccuracy to add in Power Profile
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Unused for TC
ClockTime_ValIn	in	std_logic	1	Unused for TC
Counter Input				
CounterTime_DatIn	in	Clk_Time_Type	1	Freerun Counter Clock Time
CounterTime_ValIn	in	std_logic	1	Freerun Counter Clock Time valid
Link Speed Input				
LinkSpeed_DatIn	in	Common_ LinkSpeed_Type	1	Link Speed of the interface
Frame Input				
SfdDetected_EvtIn	in	std_logic	1	Start of Frame Delimiter detected
Timestamp Output				
ClockTimestamp _DatOut	out	Clk_Time_Type	1	Unused for TC
ClockTimestamp _ValOut	out	std_logic	1	Unused for TC
Timestamp Output				
CounterTimestamp _DatOut	out	Clk_Time_Type	1	Freerun Counter Clock Timestamp
CounterTimestamp _ValOut	out	std_logic	1	Freerun Counter Clock Timestamp valid
Dataset Input				

PortDataset_DatIn	in	Ptp_PortDataset_Type	1	PTP Port Dataset input for this port, contains Delay
DefaultDataset_DatIn	in	Ptp_DefaultDataset_Type	1	PTP Default Dataset input for this clock
Drop Input				
Drop_ValIn	in	std_logic	1	If input frame shall be dropped
Axi Input				
AxisValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisReady_ValOut	out	std_logic	1	
AxisData_DatIn	in	std_logic_vector	32	
AxisStrobe_ValIn	in	std_logic_vector	4	
AxisKeep_ValIn	in	std_logic_vector	4	
AxisLast_ValIn	in	std_logic	1	
AxisUser_DatIn	in	std_logic_vector	3	
Axi Tap Output				
AxisTapValid_ValOut	out	std_logic	1	AXI Stream frame tap output
AxisTapReady_ValOut	out	std_logic	1	
AxisTapData_DatOut	out	std_logic_vector	32	
AxisTapStrobe_ValOut	out	std_logic_vector	4	
AxisTapKeep_ValOut	out	std_logic_vector	4	
AxisTapLast_ValOut	out	std_logic	1	
AxisTapUser_DatOut	out	std_logic_vector	3	
Drop Output				
Drop_ValOut	out	std_logic	1	If output frame shall be dropped
Axi Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	
AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	
Frame Info Output				
FrameInfo_DatOut	out	Ptp_FrameInfo_Type	1	Frame Information

FrameInfo_ValOut	out	Ptp_FrameInfoVal_Type	1	Frame Information valid

Table 13: RX Processor

4.2.2 TX Processor

4.2.2.1 Entity Block Diagram

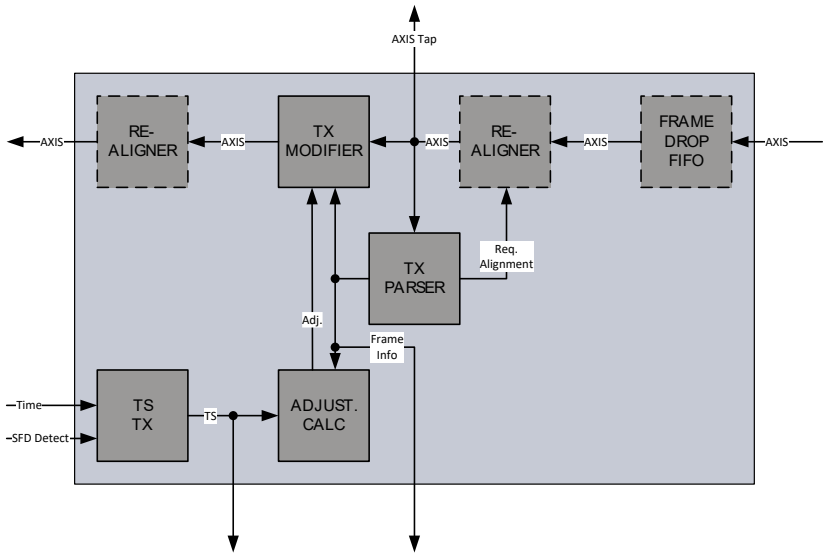

Figure 13: TX Processor

4.2.2.2 Entity Description

TX Timestamper

This module takes a snapshot of the free-running Counter Clock and of the PTP Adjusted Clock when the SFD detected event was asserted by the Interface Adapter. The timestamp is buffered and aligned with the incoming frame, so the Adjustment Calculator can calculate the correct adjustments.

TX Parser

This module parses all outgoing Ethernet frames. It extracts the frame length, in the case of PTP frames also the CorrectionField and NetworkTimeInaccuracy and it checks the CRC. If Layer 3 mapping or the Utility or TSN Profile is used it request a specific alignment from the Realigner so the PTP frame is starting 32 bit aligned again.

TX Modifier

This module is the heart of the TX processor. It also parses the frame, waits for the CorrectionField in the frame. It then replaces the CorrectionField with the one from the Adjustment Calculation. For local generated Sync messages it does the same thing with the TimestampField. In the case of Utility Profile it also modifies (zeros

out) the Source Port Identity of PTP frames according to the Stateless Redbox defined in IEC62439. Since the frame is modified on-the-fly the CRC and UDP checksum (IPv6) is recalculated and overwritten if the incoming CRC was correct or the frame was locally generated, otherwise the frames are marked with a wrong CRC. This is the same Modifier as in the Transparent Clock but with a different configuration.

Adjustment Calculation

This module calculates the new CorrectionField out of the outgoing Correction-Field and the transmit timestamp (added). It also provides the TimestampField for local generated Sync frames.

Re-Aligner

This module allows aligning the AXI Data stream as required. It is only instantiated if Layer 3 or Utility or TSN mapping is used. The input can be 32/24/16/8 bit aligned and the output can also be 32/24/16/8 bit aligned. From an external input, the output alignment can be requested. This is used for alignment of the PTP start to a 32 bit boundary (Some mappings cause that the header is not 32 bit aligned anymore) again, and for realignment to a constant 32bit wide stream again.

Frame Drop Fifo

This is an optional Frame Drop Fifo to overcome additional data speed differences. It is not used in the Transparent Clock.

4.2.2.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile
PowerProfile Support_Gen	-	boolean	1	Support for Power Profile
UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile

E2eSupport_Gen	-	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
P2pSupport_Gen	-	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
E2eUnicastSupport_Gen	-	boolean	1	E2E Unicast handling support
Layer2Support_Gen	-	boolean	1	Support for Layer 2 Mapping
Layer3v4 Support_Gen	-	boolean	1	Support for Layer 3 Ipv4 Mapping
Layer3v6 Support_Gen	-	boolean	1	Support for Layer 3 Ipv6 Mapping
UnicastSupport_Gen	-	boolean	1	Support for Unicast Messages
TwoStepSupport_Gen	-	Boolean	1	Support for TwoStep frames
Asymmetry Support_Gen	-	Boolean	1	Support for Asymmetry corrections
RemoveVlanSupport_Gen	-	Boolean	1	If on transmission removing of a VLAN shall be supported
ClockType_Gen	-	Ptp_ClockType_Typ	1	What Kind of Clock this is (TC)
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond
TX Processor				
DelayNanosecond10_Gen	-	integer	1	Input Delay of the PHY in Nanosecond
DelayNanosecond100_Gen	-	integer	1	Input Delay of the PHY in Nanosecond
DelayNano second1000_Gen	-	integer	1	Input Delay of the PHY in Nanosecond

PortIdModify_Gen	-	boolean	1	NA
PortId_Gen	-	std_logic_vector	4	NA
Buffer_Gen	-	boolean	1	If a frame drop fifo shall be instantiated
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Unused for TC
ClockTime_ValIn	in	std_logic	1	Unused for TC
Counter Input				
CounterTime_DatIn	in	Clk_Time_Type	1	Freerun Counter Clock Time
CounterTime_ValIn	in	std_logic	1	Freerun Counter Clock Time valid
Link Speed Input				
LinkSpeed_DatIn	in	Common_ LinkSpeed_Type	1	Link Speed of the interface
Frame Input				
SfdDetected_EvtIn	in	std_logic	1	Start of Frame Delimiter detected
Timestamp Output				
ClockTimestamp _DatOut	out	Clk_Time_Type	1	Unused for TC
ClockTimestamp _ValOut	out	std_logic	1	Unused for TC
Timestamp Output				
CounterTimestamp _DatOut	out	Clk_Time_Type	1	Freerun Counter Clock Timestamp
CounterTimestamp _ValOut	out	std_logic	1	Freerun Counter Clock Timestamp valid
Dataset Input				
PortDataset_DatIn	in	Ptp_PortDataset _Type	1	PTP Port Dataset input for this port
DefaultDataset_DatIn	in	Ptp_DefaultDataset _Type	1	PTP Default Dataset input for this clock
Drop Input				
Drop_ValIn	in	std_logic	1	If input frame shall be dropped
Axi Input				

AxisValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisReady_ValOut	out	std_logic	1	
AxisData_DatIn	in	std_logic_vector	32	
AxisStrobe_ValIn	in	std_logic_vector	4	
AxisKeep_ValIn	in	std_logic_vector	4	
AxisLast_ValIn	in	std_logic	1	
AxisUser_DatIn	in	std_logic_vector	3	
Axi Tap Output				
AxisTapValid_ValOut	out	std_logic	1	AXI Stream frame tap output
AxisTapReady _ValOut	out	std_logic	1	
AxisTapData_DatOut	out	std_logic_vector	32	
AxisTapStrobe _ValOut	out	std_logic_vector	4	
AxisTapKeep_ValOut	out	std_logic_vector	4	
AxisTapLast_ValOut	out	std_logic	1	
AxisTapUser_DatOut	out	std_logic_vector	3	
Drop Output				
Drop_ValOut	out	std_logic	1	If output frame shall be dropped
Axi Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	
AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	
Frame Info Output				
FrameInfo_DatOut	out	Ptp_FrameInfo _Type	1	Frame Information
FrameInfo_ValOut	out	Ptp_FrameInfoVal_T ype	1	Frame Information valid

Table 14: TX Processor

4.2.3 Delay Processor

This is one of the two (Delay and Management) core parts, handling all Delay message related things.

4.2.3.1 Entity Block Diagram

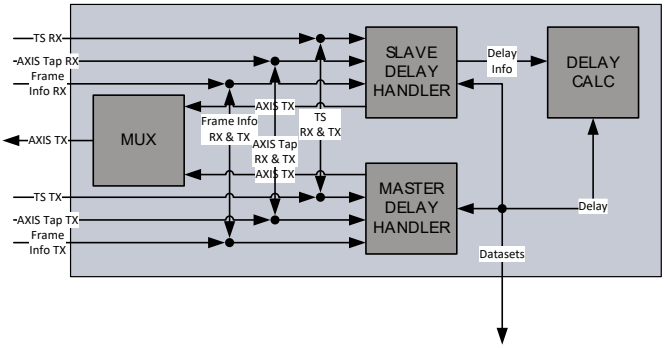

Figure 14: Delay Processor

4.2.3.2 Entity Description

Slave Delay Handler

This module periodically sends PDelay Requests messages stores the transmit timestamp and waits for the PDelay response (and PDelay Response FollowUp) messages from the other nodes. When a PDelay Response message is received it extracts and merges the frame information needed for the delay calculation from the Delay Response and Delay Response FollowUp and passes the information to the Delay Calculator block. No CRC is inserted and calculated, since this is done in the TX Processor for internally generated frames. It shares it's transmit path with the other Processors (Master Delay and Management).

Master Delay Handler

This module detects and parses incoming PDelay Request messages and sends PDelay Response messages. It extracts information from the PDelay Requests which is used for generating the PDelay Responses. It acts in one-step mode, means the residence time will be stored in the correction field and no PDelay Response FollowUp will be sent. For that it subtracts the RX timestamp from the correction field. The TX Processor will add the TX timestamp to the correction field in a later stage and calculates the CRC. No CRC is inserted and calculated, since this is done in the TX Processor for internally generated frames. It shares it's transmit path with the other Processors (Slave Delay and Management).

Delay Calculator

This module calculates the delay based on the RX and TX timestamps, correction fields and information extracted from the frame by the Slave Delay Handler. It averages the Delay over the last 4 messages if they are all within a certain range or hard sets the delay if they are out of the window. This allows smooth measurements for the case no topology change has happened. Finally it stores the calculated delay in the Port Dataset

Multiplexer

This module multiplexes the two transmit paths from the Slave Delay Handler and Master Delay handler. Arbitration is round robin.

4.2.3.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
SlaveOnly_Gen	-	boolean	1	Slave Only Clock
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile
PowerProfile Support_Gen	-	boolean	1	Support for Power Profile
UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile
E2eSupport_Gen	-	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
P2pSupport_Gen	-	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
Layer2Support_Gen	-	boolean	1	Support for Layer 2

				Mapping
Layer3v4 Support_Gen	-	boolean	1	Support for Layer 3 Ipv4 Mapping
Layer3v6 Support_Gen	-	boolean	1	Support for Layer 3 Ipv6 Mapping
TwoStepSupport_Gen	-	Boolean	1	Support for TwoStep frames
ClockType_Gen	-	Ptp_ClockType_Typ	1	What Kind of Clock this is (TC)
Sim_Gen	-	boolean	1	If in Testbench simulation mode
Delay Processor				
AverageWindow Nanosecond_Gen	-	natural	1	Delay Averaging window, out of this window, delay is set hard
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Enable Input				
Enable_EnaIn	in	std_logic	1	Clock Enabled
Timer				
Timer1ms_EvtIn	in	std_logic	1	Adjusted PTP Clock aligned 1 millisecond Timer event
Axi RX Input				
AxisValidRx_ValIn	in	std_logic	1	AXI Stream RX frame input
AxisReadyRx_ValIn	in	std_logic	1	
AxisDataRx_DatIn	in	std_logic_vector	32	
AxisStrobeRx_ValIn	in	std_logic_vector	4	
AxisKeepRx_ValIn	in	std_logic_vector	4	
AxisLastRx_ValIn	in	std_logic	1	
AxisUserRx_DatIn	in	std_logic_vector	3	
Axi TX Input				
AxisValidTx_ValIn	in	std_logic	1	AXI Stream TX frame input
AxisReadyTx_ValIn	in	std_logic	1	
AxisDataTx_DatIn	in	std_logic_vector	32	
AxisStrobeTx_ValIn	in	std_logic_vector	4	

AxisKeepTx_ValIn	in	std_logic_vector	4	
AxisLastTx_ValIn	in	std_logic	1	
AxisUserTx_DatIn	in	std_logic_vector	3	
Axi Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	
AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	
PTP Statemachine Input				
Ptp_State_StaIn	in	Ptp_State_Type	1	Unused for TC
Frame Info RX Input				
FrameInfoRx_DatIn	in	Ptp_FrameInfo_Type	1	Frame Information
FrameInfoRx_ValIn	in	Ptp_FrameInfoVal_Type	1	Frame Information valid
Frame Info TX Input				
FrameInfoTx_DatIn	in	Ptp_FrameInfo_Type	1	Frame Information
FrameInfoTx_ValIn	in	Ptp_FrameInfoVal_Type	1	Frame Information valid
Timestamp RX Input				
CounterTimestampRx_DatIn	in	Clk_Time_Type	1	Freerun Counter Clock Timestamp
CounterTimestampRx_ValIn	in	std_logic	1	Freerun Counter Clock Timestamp valid
ClockTimestampRx_DatIn	in	Clk_Time_Type	1	Freerun Counter Clock Timestamp
ClockTimestampRx_ValIn	in	std_logic	1	Freerun Counter Clock Timestamp valid
Timestamp TX Input				
CounterTimestampTx_DatIn	in	Clk_Time_Type	1	Freerun Counter Clock Timestamp
CounterTimestampTx_ValIn	in	std_logic	1	Freerun Counter Clock Timestamp

				valid
ClockTimestampTx_DatIn	in	Clk_Time_Type	1	Freerun Counter Clock Timestamp
ClockTimestampTx_ValIn	in	std_logic	1	Freerun Counter Clock Timestamp valid
Redundancy Input				
RedMode_DatIn	in	Ptp_RedMode_Type	1	Redundancy Mode: Hsr_E Prp_E No_E
RedSeqResp_DatIn	in	Ptp_RedSeqResp Array_Type	2	Sequence Number Response
RedSeqResp_ValIn	in	Ptp_RedSeqRespVal Array_Type	2	Sequence Number Response valid
Redundancy Output				
RedSeqReq_DatOut	out	Ptp_RedSeqReq Array_Type	2	Sequence Number Request
RedSeqReq_ValOut	out	Ptp_RedSeqReqVal Array_Type	2	Sequence Number Request valid
Sync Info Input				
SyncReceived_ValIn	in	std_logic	1	Sync Message was received
CorrectionField SyncRx_DatIn	in	std_logic_vector	64	Correction field of received Sync
Timestamp SyncRx_DatIn	in	std_logic_vector	80	Timestamp when Sync was received
Timestamp SyncTx_DatIn	in	std_logic_vector	80	Timestamp when Sync was sent
Dataset Input				
ParentDataset_DatIn	in	Ptp_Parent Dataset_Type	1	PTP Parent Dataset input for this clock
PortDataset_DatIn	in	Ptp_Port Dataset_Type	1	PTP Port Dataset input for this clock
DefaultDataset_DatIn	in	Ptp_Default Dataset_Type	1	PTP Default Dataset input for this clock
Dataset Output				
PortDataset_DatOut	out	Ptp_Port Dataset_Type	1	PTP Port Dataset data output

PortDataset_ValOut	out	Ptp_Port DatasetVal_Type	1	PTP Port Dataset valid output valid output valid output
Current Dataset_DatOut	out	Ptp_Current Dataset_Type	1	PTP Current Dataset data output
Current Dataset_ValOut	out	Ptp_Current DatasetVal_Type	1	PTP Current Dataset valid output valid output valid output

Table 15: Delay Processor

4.2.4 Management Processor

This is one of the two (Delay and Management) core parts, handling all Management message related things. This module is optional since a lot of the PTP Profiles removed PTP Management from the scope by purpose for security reasons.

4.2.4.1 Entity Block Diagram

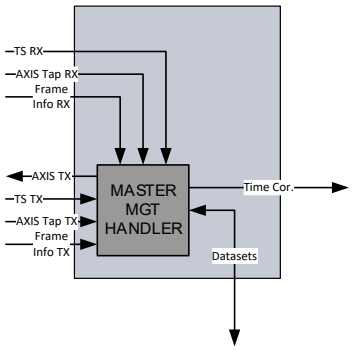

Figure 15: Management Processor

4.2.4.2 Entity Description

Master Management Handler

This module detects and parses incoming Management messages and sends Management Messages (Responses). It supports most of the PTP Management messages except some which were disabled because of security reasons or because they are PTP Profile specific. It extracts the command and does the corresponding action: For a GET it gets the requested value from the corresponding Datasets and sends the value with a Response Management message; For a SET it takes the value from the Management message, sets the requested value in the corresponding Dataset and sends an Acknowledge Management message. Also the time can be set, this via the time adjustment interface similar to the Drift and Offset.

4.2.4.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
SlaveOnly_Gen	-	boolean	1	Slave Only Clock
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile

PowerProfile Support_Gen	-	boolean	1	Support for Power Profile
UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile
Layer2Support_Gen	-	boolean	1	Support for Layer 2 Mapping
Layer3v4 Support_Gen	-	boolean	1	Support for Layer 3 Ipv4 Mapping
Layer3v6 Support_Gen	-	boolean	1	Support for Layer 3 Ipv6 Mapping
E2eSupport_Gen	-	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
P2pSupport_Gen	-	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
ClockType_Gen	-	Ptp_ClockType_Typ	1	What Kind of Clock this is (TC)
Sim_Gen	-	boolean	1	If in Testbench simulation mode
Management Processor				
NumberOfPorts_Gen	-	natural	1	Number of Ports of that Clock
ManufacturerIdentity_Gen	-	Common_Byte_Type	3	Manufacturing Identity string
ProductDescription_Gen	-	string	32	Product Description string
RevisionData_Gen	-	string	32	Product Revision string
UserDescription_Gen	-	string	32	User Description string
Ports				

System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Enable Input				
Enable_EnaIn	in	std_logic	1	Clock Enabled
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Unused for TC
ClockTime_ValIn	in	std_logic	1	Unused for TC
Timer				
Timer1ms_EvtIn	in	std_logic	1	Adjusted PTP Clock aligned 1 millisecond Timer event
Axi RX Input				
AxisValidRx_ValIn	in	std_logic	1	AXI Stream RX frame input
AxisReadyRx_ValIn	in	std_logic	1	
AxisDataRx_DatIn	in	std_logic_vector	32	
AxisStrobeRx_ValIn	in	std_logic_vector	4	
AxisKeepRx_ValIn	in	std_logic_vector	4	
AxisLastRx_ValIn	in	std_logic	1	
AxisUserRx_DatIn	in	std_logic_vector	3	
Axi TX Input				
AxisValidTx_ValIn	in	std_logic	1	AXI Stream TX frame input
AxisReadyTx_ValIn	in	std_logic	1	
AxisDataTx_DatIn	in	std_logic_vector	32	
AxisStrobeTx_ValIn	in	std_logic_vector	4	
AxisKeepTx_ValIn	in	std_logic_vector	4	
AxisLastTx_ValIn	in	std_logic	1	
AxisUserTx_DatIn	in	std_logic_vector	3	
Axi Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	
AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	
PTP Statemachine Input				
Ptp_State_StaIn	in	Ptp_State_Type	1	Unused for TC
Frame Info RX Input				
FrameInfoRx_DatIn	in	Ptp_FrameInfo	1	Frame Information

		_Type		
FrameInfoRx_ValIn	in	Ptp_FrameInfoVal_Type	1	Frame Information valid
Frame Info TX Input				
FrameInfoTx_DatIn	in	Ptp_FrameInfo_Type	1	Frame Information
FrameInfoTx_ValIn	in	Ptp_FrameInfoVal_Type	1	Frame Information valid
Dataset Input				
CurrentDataset_DatIn	in	Ptp_Current Dataset_Type	1	Unused for TC
ParentDataset_DatIn	in	Ptp_Parent Dataset_Type	1	Unused for TC
ForeignMasterDataset_DatIn	in	Ptp_ForeignMaster Dataset_Type	1	Unused for TC
TimeProperties Dataset_DatIn	in	Ptp_TimeProperties Dataset_Type	1	Unused for TC
PortDataset_DatIn	in	Ptp_Port Dataset_Type	N	PTP Port Dataset input for this clock
DefaultDataset_DatIn	in	Ptp_Default Dataset_Type	1	PTP Default Dataset input for this clock
Dataset Output				
PortDataset _DatOut	out	Ptp_Port Dataset_Type	N	PTP Port Dataset data output
PortDataset _ValOut	out	Ptp_Port DatasetVal_Type	N	PTP Port Dataset valid output
CurrentDataset _DatOut	out	Ptp_Current Dataset_Type	1	Unused for TC
CurrentDataset _ValOut	out	Ptp_Current DatasetVal_Type	1	Unused for TC
DefaultDataset _DatOut	out	Ptp_Default Dataset_Type	1	PTP Default Dataset data output
DefaultDataset _ValOut	out	Ptp_Default DatasetVal_Type	1	PTP Default Dataset valid output
ParentDataset _DatOut	out	Ptp_Parent Dataset_Type	1	Unused for TC
ParentDataset _ValOut	out	Ptp_Parent DatasetVal_Type	1	Unused for TC

ForeignMasterDataset_DatOut	out	Ptp_ForeignMaster Dataset_Type	1	Unused for TC
ForeignMasterDataset_ValOut	out	Ptp_ForeignMaster DatasetVal_Type	1	Unused for TC
TimeProperties Dataset_DatOut	out	Ptp_TimeProperties Dataset_Type	1	Unused for TC
TimeProperties Dataset_ValOut	out	Ptp_TimeProperties DatasetVal_Type	1	Unused for TC
Time Adjustment Output				
TimeAdjustment _DatOut	out	Clk_TimeAdjustment _Type	1	Unused for TC
TimeAdjustment _ValOut	out	std_logic	1	Unused for TC

Table 16: Management Processor

4.2.5 Datasets

4.2.5.1 Entity Block Diagram

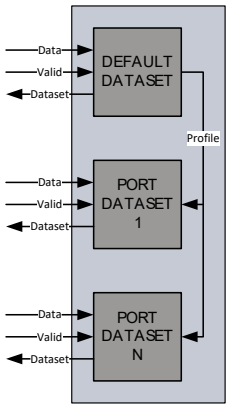

Figure 16: Datasets

4.2.5.2 Entity Description

Default Dataset

This module contains the storage for the Dataset members of the Default Dataset according to IEEE1588-2019/2008 clause 8.3.2:

```
Profile                    : Ptp_Profile_Type;
Layer                      : Ptp_Layer_type;
Vlan                       : Ptp_Vlan_Type;
VlanEnable                 : std_logic;
Dscp                       : std_logic_vector(5 downto 0);
Ip                         : Common_Byte_Type(0 to 15);
Signaling                  : std_logic;
TwoStepFlag                : std_logic;
ClockIdentity              : Ptp_ClockIdentity_Type;
NumberPorts                : std_logic_vector(15 downto 0);
DomainNumber               : std_logic_vector(7 downto 0);
```

Port Dataset

This module contains the storage for the Dataset members of the Port Dataset according to IEEE1588-2019/2008 clause 8.3.3:

```
PortIdentity               : Ptp_PortIdentity_Type;
PortState                  : std_logic_vector(7 downto 0);
PeerMeanPathDelayValid     : std_logic;
PeerMeanPathDelay          : std_logic_vector(63 downto 0);
LogPtpCapableSignalingInterval : std_logic_vector(7 downto 0);
PtpCapableTimeout          : std_logic_vector(7 downto 0);
PtpCapable                 : std_logic;
DelayMechanism             : std_logic_vector(7 downto 0);
```

```
LogMinPdelayReqInterval        : std_logic_vector(7 downto 0);
DelayReceiptTimeout            : std_logic_vector(7 downto 0);
MaxPdelay                      : std_logic_vector(31 downto 0);
VersionNumber                  : std_logic_vector(3 downto 0);
Asymmetry                      : std_logic_vector(31 downto 0);
VlanRemove                     : std_logic;
VlanInsert                     : std_logic;
```

4.2.5.3 Entity Declarations

Name	Dir	Type	Size	Description
Generics				
General				
SlaveOnly_Gen	-	boolean	1	Slave Only Clock
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile
PowerProfile Support_Gen	-	boolean	1	Support for Power Profile
UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile
Layer2Support_Gen	-	Boolean	1	Support for Layer 2 Mapping
Layer3v4 Support_Gen	-	boolean	1	Support for Layer 3 Ipv4 Mapping
Layer3v6 Support_Gen	-	boolean	1	Support for Layer 3 Ipv6 Mapping
InsertVlanSup-port_Gen	-	Boolean	1	If on reception inserting of a VLAN shall be supported
RemoveVlanSup-port_Gen	-	Boolean	1	If on transmission removing of a VLAN shall be supported
TwoStepSupport_Gen	-	Boolean	1	Support for TwoStep frames
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock

SysRstN_RstIn	in	std_logic	1	System Reset
Dataset Input				
DefaultDataset_DatIn	in	Ptp_Default Dataset_Type	1	PTP Default Dataset input for this clock
DefaultDataset_ValIn	in	Ptp_Default DatasetVal_Type	1	PTP Default Dataset input for this clock
Dataset Output				
DefaultDataset _DatOut	in	Ptp_Default Dataset_Type	1	PTP Default Dataset input for this clock

Table 17: Default Dataset

Name	Dir	Type	Size	Description
Generics				
General				
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile
PowerProfile Support_Gen	-	boolean	1	Support for Power Profile
UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile
E2eSupport_Gen	-	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
P2pSupport_Gen	-	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
E2eUnicastSupport_Gen	-	boolean	1	E2E Unicast handling support
PortNumber_Gen	-	std_logic_vector	16	Portnumber
Asymmetry Support_Gen	-	Boolean	1	Support for Asymmetry corrections

Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Profile				
Profile_DatIn	in	Ptp_Profile_Type	1	Profile active
Profile_ValIn	in	std_logic	1	Profile valid
Dataset Input				
PortDataset_DatIn	in	Ptp_Port Dataset_Type	1	PTP Port Dataset input for this clock
PortDataset_ValIn	in	Ptp_Port DatasetVal_Type	1	PTP Port Dataset input for this clock
Dataset Output				
PortDataset _DatOut	in	Ptp_Port Dataset_Type	1	PTP Port Dataset input for this clock

Table 18: Port Datasets

4.2.6 Clock Counter

4.2.6.1 Entity Block Diagram

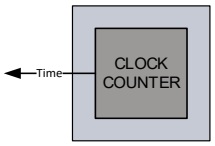

Figure 17: Clock Counter

4.2.6.2 Entity Description

Clock Counter

This is a free-running counter with nanosecond resolution in a 32 bit second and 32 bit nanosecond format. It is used by the Delay and Residence time measurements. The reason for this is mainly based on the fact that this is also used for the PTP Transparent clock and no cascading PI servo loops shall be introduced. It can take any input frequency, also non integer values with fractions. For the PTP Transparent Clock this is run on the system clock. The time domain is TAI, so after every reset the clock starts from 1.1.1970 at midnight.

4.2.6.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
Counter Clock				
ClockClkPeriodNano-second_Gen	-	natural	1	Integer Clock Period
ClockClkPeriodFract-Num_Gen	-	natural	1	Fractional Clock Period Numerator
ClockClkPeriod-FractDeNum_Gen	-	natural	1	Fractional Clock Period Denumerator
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Time Output				
ClockTime_DatIn	in	Clk_Time_Type	1	Freerun Counter Clock Time

ClockTime_ValIn	in	std_logic	1	Freerun Counter Clock Time valid

Table 19: Clock Counter

4.2.7 Ethernet Interface Adapter

4.2.7.1 Entity Block Diagram

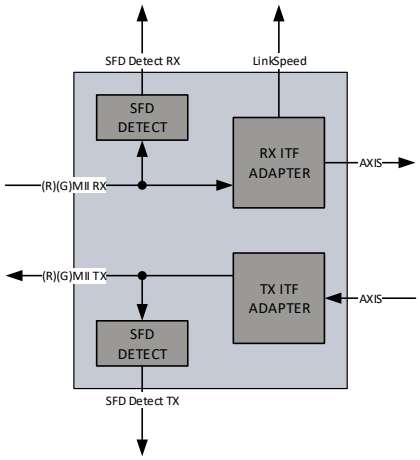

Figure 18: Ethernet Interface Adapter

4.2.7.2Entity Description

SFD Detector

This module detects the Start of Frame Delimiter (SFD) on the (R)(G)MII stream. It runs directly on the (R)(G)MII clock domain for minimal jitter on the timestamp point detection. Once the SFD is detected, an event is signaled which is used by the timestamper.

RX Interface Adapter

This module convert the Media Independent Interface (R)(G)MII data stream (2/4/8bit) into a 32bit AXI stream. First bytes on the cable are mapped to the AXI MSB of the data array. It contains an asynchronous Fifo to on one hand do clock domain crossing from the external clock to the system clock and on the other hand also to minimal buffer data for speed differences. The Fifo size is kept quite small to assure correct timestamp alignment with the frame. It converts the different data widths into a 32bit block AXI stream. The Preamble and SFD are removed on reception. Also, it detects the link speed based on the interface clock.

TX Interface Adapter

This module convert the 32bit AXI stream into a Media Independent Interface (R)(G)MII data stream (2/4/8bit) which is continuous. The MSB of the AXI data array is mapped to the first byte on the cable. It contains an asynchronous Fifo to

on one hand do clock domain crossing from the system clock to the external clock and on the other hand also to minimal buffer data for speed differences. The Fifo size is kept quite small to assure correct timestamp alignment with the frame. It converts the 32bit block AXI stream into the different data widths. The Preamble and SFD are added before transmission. It also assures the correct interframe gap between frames.

4.2.7.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
Interface Adapter				
ClockClkPeriodNano-second_Gen	-	natural	1	Integer Clock Period
IoFf_Gen	-	boolean	1	Shall IO flip flops be instantiated
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
(R)(G)Mii RX Clk/Rst Input				
(R)(G)MiiRxClk_ClkIn	in	std_logic	1	RX Clock
(R)(G)MiiRxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
(R)(G)Mii TX Clk/Rst Input				
(R)(G)MiiTxClk_ClkIn	in	std_logic	1	RX Clock
(R)(G)MiiTxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
(R)(G)Mii RX Data Input/Output				
(R)(G)MiiRxDv_Ena	In/ out	std_logic	1	RX Data valid
(R)(G)MiiRxErr_Ena	In/ out	std_logic	1	RX Error
(R)(G)MiiRxData_Dat	In/ out	std_logic_vector	2-8	RX Data MII:4, RMII:2, GMII:8, RGMII:4
(R)(G)MiiCol_Dat	In/ out	std_logic	1	Collision

(R)(G)MiiCrs_Dat	In/ out	std_logic	1	Carrier Sense
(R)(G)Mii TX Data Input				
(R)(G)MiiTxEn_Ena	In/ out	std_logic	1	TX Data valid
(R)(G)MiiTxErr_Ena	In/ out	std_logic	1	TX Error
(R)(G)MiiTxData_Dat	In/ out	std_logic_vector	2-8	TX Data MII:4, RMII:2, GMII:8, RGMII:4
Link Speed Output				
LinkSpeed_DatOut	out	Common_ LinkSpeed_Type	1	Link Speed of the interface
SfdDetected Output				
(R)(G)MiiInSfdDetected_EvtOut	out	std_logic	1	Start of Frame Delimiter detected
(R)(G)MiiOutSfdDetected_EvtOut	out	std_logic	1	Start of Frame Delimiter detected
Axi Input				
AxisValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisReady_ValOut	out	std_logic	1	
AxisData_DatIn	in	std_logic_vector	32	
AxisStrobe_ValIn	in	std_logic_vector	4	
AxisKeep_ValIn	in	std_logic_vector	4	
AxisLast_ValIn	in	std_logic	1	
AxisUser_DatIn	in	std_logic_vector	3	
Axi Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	
AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	

Table 20: Ethernet Interface Adapter

4.2.8 Registerset

4.2.8.1 Entity Block Diagram

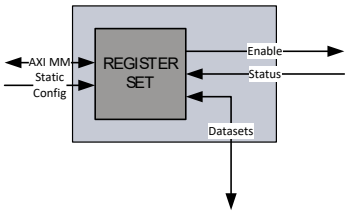

Figure 19: Registerset

4.2.8.2 Entity Description

Register Set

This module is an AXI4Lite Memory Mapped Slave. It provides access to all Datasets and allows configuring the PTP Transparent Clock. AXI4Lite only supports 32 bit wide data access, no byte enables, no burst, no simultaneous read and writes and no unaligned access. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the Datasets is done via signals and can be easily done from within the FPGA without CPU. For each parameter a valid signal is available, the enable signal shall be set last (or simultaneously). To change parameters the clock has to be disabled and enabled again. If in AXI mode, a AXI Master has to configure the Datasets with AXI writes to the registers, which is typically done by a CPU. Parameters can in this case also be changed at runtime. For the PortDataset the each port has to be addressed first and values from the default dataset validated again since the clock identity used by the PortDatasets come from the DefaultDataset; see configuration example for details.

4.2.8.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
SlaveOnly_Gen	-	boolean	1	Slave Only Clock
DefaultProfile Support_Gen	-	boolean	1	Support for Default Profile
PowerProfile Support_Gen	-	boolean	1	Support for Power Profile

UtilityProfile Support_Gen	-	boolean	1	Support for Utility Profile
TsnProfile Support_Gen	-	boolean	1	Support for TSN Profile
E2eSupport_Gen	-	boolean	1	If the core shall support E2E delay mechanism (only valid with default profile support)
P2pSupport_Gen	-	boolean	1	If the core shall support P2P delay mechanism (mandatory for Power and Utility profile)
E2eUnicastSupport_Gen	-	boolean	1	E2E Unicast handling support
Layer2Support_Gen	-	Boolean	1	Support for Layer 2 Mapping
Layer3v4 Support_Gen	-	boolean	1	Support for Layer 3 Ipv4 Mapping
Layer3v6 Support_Gen	-	boolean	1	Support for Layer 3 Ipv6 Mapping
TwoStepSupport_Gen	-	Boolean	1	Support for TwoStep frames
Asymmetry Support_Gen	-	Boolean	1	Support for Asymmetry corrections
InsertVlanSupport_Gen	-	Boolean	1	If on reception inserting of a VLAN shall be supported
RemoveVlanSupport_Gen	-	Boolean	1	If on transmission removing of a VLAN shall be supported
NrOfPorts_Gen	-	boolean	1	Number of Ports of the TC
Register Set				
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used
AxiAddressRange	-	std_logic_vector	32	AXI Base Address

Low_Gen				
AxiAddressRangeHigh_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Ptp_TransparentClock StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Ptp_TransparentClock StaticConfigVal_Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Ptp_TransparentClock StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Ptp_TransparentClock StaticStatusVal_Type	1	Static Status valid
Error Input				
Ptp_ErrIn	in	std_logic	1	An error happened
AXI4 Lite Slave				
AxiWriteAddrValid_ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady_RdyOut	out	std_logic	1	Write Address Ready
AxiWriteAddrAddress_AdrIn	in	std_logic_vector	32	Write Address
AxiWriteAddrProt_DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid_ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady_RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData_DatIn	in	std_logic_vector	32	Write Data
AxiWriteDataStrobe_DatIn	in	std_logic_vector	4	Write Data Strobe

AxiWriteRespValid_ValOut	out	std_logic	1	Write Response Valid
AxiWriteRespReady_RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid_ValIn	in	std_logic	1	Read Address Valid
AxiReadAddrReady_RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress_AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt_DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid_ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady_RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData_DatOut	out	std_logic_vector	32	Read Data Response
Port Select				
PortSelect_DatOut	in	std_logic_vector	8	Which Port shall be selected for the Dataset
Dataset Input				
PortDataset_DatIn	in	Ptp_Port Dataset_Type	1	PTP Port Dataset input for this clock
DefaultDataset_DatIn	in	Ptp_Default Dataset_Type	1	PTP Default Dataset input for this clock
Dataset Output				
PortDataset_DatOut	out	Ptp_Port Dataset_Type	1	PTP Port Dataset data output
PortDataset_ValOut	out	Ptp_Port DatasetVal_Type	1	PTP Port Dataset valid output
DefaultDataset_DatOut	out	Ptp_Default Dataset_Type	1	PTP Default Dataset data output
DefaultDataset_ValOut	out	Ptp_Default DatasetVal_Type	1	PTP Default Dataset valid output
Enable Output				
PtpTransparentClock Enable_DatOut	out	std_logic	1	Enable PTP Transparent Clock

Table 21: Registerset

4.3 Configuration example

4.3.1 Static Configuration

```
constant PtpStaticConfigTc_Con : Ptp_TransparentClockStaticConfig_Type := (
    Profile                                     => DefaultProfile_E,
    Layer                                       => Layer2_E,
    DelayMechanism                              => P2p_E,
    DelayE2eUnicast                             => '0',
    TwoStep                                     => '0',
    Vlan                                        => Ptp_Vlan_Type_Rst_Con,
    VlanEnable                                  => '1',
    Ip                                          => (
        0                                       => x"C0",
        1                                       => x"A8",
        2                                       => x"00",
        3                                       => x"30",
        others                                  => x"00"),
    DefaultDataset_ClockIdentity                => (
        0                                       => x"4E",
        1                                       => x"54",
        2                                       => x"4C",
        3                                       => x"FF",
        4                                       => x"FE",
        5                                       => x"00",
        6                                       => x"00",
        7                                       => x"30"),
    DefaultDataset_DomainNumber                 => x"00"
);

constant PtpStaticConfigValTc_Con : Ptp_TransparentClockStaticConfigVal_Type := (
    Enable_Val                                  => '1',
    Profile_Val                                 => '1',
    Vlan_Val                                    => '1',
    Ip_Val                                      => '1',
    DefaultDataset_ClockIdentity_Val            => '1',
    DefaultDataset_DomainNumber_Val             => '1'
);
```

Figure 20: Static Configuration

4.3.2 AXI Configuration

The following code is a simplified pseudocode from the testbench: The base address of the Transparent Clock is 0x10000000.

```
-- PTP TC
-- Config
-- profile: 0 default profile
AXI WRITE 10000084 00000000
-- VLAN 0x4000 (unused)
AXI WRITE 10000088 00014000
-- set config valid bits
AXI WRITE 10000080 00000003

-- Default Dataset TC
-- clock id: 00:01:02:FF:FE:03:04:06
AXI WRITE 10000104 FF020100
AXI WRITE 10000108 060403FE
-- domain: 0
AXI WRITE 1000010C 00000000
-- set default dataset valid bits
AXI WRITE 10000100 00000003

-- Port Dataset
-- set portnr 0
AXI WRITE 10000200 00000000
-- set default dataset valid bits
AXI WRITE 10000100 00000001
-- set portnr 1
AXI WRITE 10000200 00010000
-- set default dataset valid bits
AXI WRITE 10000100 00000001
-- set portnr 2
AXI WRITE 10000200 00020000
-- set default dataset valid bits
AXI WRITE 10000100 00000001

-- enable PTP TC
AXI WRITE 10000000 00000001
```

Figure 21: AXI Configuration

4.4 Clocking and Reset Concept

4.4.1 Clocking

To keep the design as robust and simple as possible, the whole Transparent Clock, including the Counter Clock and all other cores from NetTimeLogic are run in one clock domain. This is considered to be the system clock. Per Default this clock is 50MHz. Where possible also the interfaces are run synchronous to this clock. For clock domain crossing asynchronous Fifos with gray counters or message patterns with meta-stability flip-flops are used. Clock domain crossings for the AXI interface is moved from the AXI slave to the AXI interconnect.

Clock	Frequency	Description
System		
System Clock	50MHz (Default)	System clock where the TC runs on as well as the counter clock etc.
(R)(G)MII Interfaces		
PHY (R)(G)MII RX Clocks	2.5/25/125MHz	Asynchronous, external receive clocks from the PHYs also used for the MAC. Depending on the interface not all frequencies apply.
PHY (R)(G)MII TX Clocks	2.5/25/125MHz	Asynchronous, external transmit clocks to/from the PHYs also used for the MAC. Depending on the interface not all frequencies apply.
AXI Interface		
AXI Clock	50MHz (Default)	Internal AXI bus clock, same as the system clock

Table 22: Clocks

4.4.2 Reset

In connection with the clocks, there is a reset signal for each clock domain. All resets are active low. All resets can be asynchronously set and shall be synchronously released with the corresponding clock domain. All resets shall be asserted for the first couple (around 8) clock cycles. All resets shall be set simultaneously and released simultaneously to avoid overflow conditions in the core. See the reference designs top file for an example of how the reset shall be handled.

Reset	Polarity	Description
System		
System Reset	Active low	Asynchronous set, synchronous release with the system clock
(R)(G)MII Interface		
PHY (R)(G)MII RX Reset	Active low	Asynchronous set, synchronous release with the (R)(G)MII RX clock
PHY (R)(G)MII TX Reset	Active low	Asynchronous set, synchronous release with the (R)(G)MII TX clock
AXI Interface		
AXI Reset	Active low	Asynchronous set, synchronous release with the AXI clock, which is the same as the system clock

Table 23: Resets

5 Resource Usage

Since the FPGA Architecture between vendors and FPGA families differ there is a split up into the two major FPGA vendors.

5.1 Intel/Altera (Cyclone V)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (Only Default Profile, L2, No Management)	11351	24187	60	0
Maximal (All Profiles, L2 & L4, Management)	14732	32285	100	0

Table 24: Resource Usage Intel/Altera

5.2 AMD/Xilinx (Kintex 7)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (Only Default Profile, L2, No Management)	11122	19968	15	0
Maximal (All Profiles, L2 & L4, Management)	15023	32797	25	0

Table 25: Resource Usage AMD/Xilinx

6 Delivery Structure

```
AXI                                 -- AXI library folder
 |-Library                          -- AXI library component sources
 |-Package                          -- AXI library package sources

CLK                                 -- CLK library folder
 |-Library                          -- CLK library component sources
 |-Package                          -- CLK library package sources

COMMON                              -- COMMON library folder
 |-Library                          -- COMMON library component sources
 |-Package                          -- COMMON library package sources

PPS                                 -- PPS library folder
|-Package                           -- PPS library package sources

PTP                                 -- PTP library folder
|-Core                              -- PTP library cores
 |-Doc                              -- PTP library cores documentations
 |-Library                          -- PTP library component sources
 |-Package                          -- PTP library package sources
 |-Refdesign                        -- PTP library cores reference designs
 |-Testbench                        -- PTP library cores testbench sources and sim/log

SIM                                 -- SIM library folder
 |-Doc                              -- SIM library command documentation
 |-Package                          -- SIM library package sources
 |-Testbench                        -- SIM library testbench template sources
 |-Tools                            -- SIM simulation tools
```

7 Testbench

The Ptp Transparent Clock testbench consist of 4 parse/port types: AXI, CLK, ETH and PTP. Multiple instances exist. PTP0 CLK, PTP0 PTP and MAC0 ETH ports are all multiplexed with the MAC0 ETH MUX to one Ethernet channel connected to the port going to the PHY Port 1 of the DUT (which acts like a MAC). PTP1 CLK, PTP1 PTP MAC1 ETH are all multiplexed with the MAC1 ETH MUX to the port going to the PHY Port 2 of the DUT (which acts like a MAC). Port1 and Port2 are interconnected on the inner side of the Transparent Clock. The third Port is unconnected as in the reference design.
The PTP Master ports take the CLK ports times as reference and sets the timestamps aligned with the time from the CLK ports. In Master mode they send Announce and Sync messages in one and two-step mode. The PTP Slave ports checks the reference time and checks it with the times in the frames from the DUT. The PTP ports are answering PDelay Request messages and also measure the Delay themselves with their own PDelay Request messages. In addition for configuration and result checks an AXI read and write port is used in the testbench and for accessing more than one AXI slave also an AXI interconnect is required.
With this Setup Master and Slave scenarios as well as multiple PTP nodes can be simulated with different PTP Profiles.

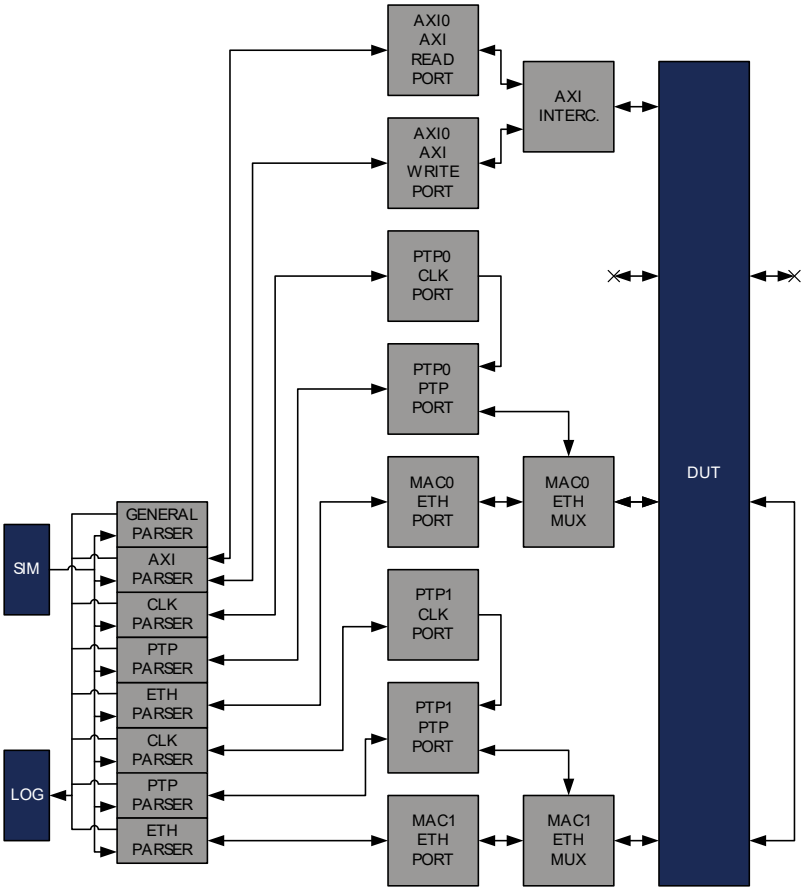

Figure 22: Testbench Framework

For more information on the testbench framework check the Sim_ReferenceManual documentation.

With the Sim parameter set the time base for timeouts are divided by 1000 to 100000 to speed up simulation time.

7.1 Run Testbench

1. Run the general script first

```
source XXX/SIM/Tools/source_with_args.tcl
```

2. Start the testbench with all test cases

```
src XXX/PTP/Testbench/Core/PtpTransparentClock/Script/run_Ptp_TransparentClock_Tb.tcl
```

3. Check the log files LogFileX.txt in the XXX/PTP/Testbench/Core/PtpTransparentClock/Log/ folder for simulation results.

8 Reference Designs

The PTP Transparent Clock reference design contains a PLL to generate all necessary clocks (cores are run at 50 MHz), an instance of the PTP Transparent Clock IP core and an instance of the Adjustable Counter Clock IP core (needs to be purchased separately). The Reference Design is intended to be connected to any PTP Master or Slave device which can run in Layer 2, P2P mode either in Default or Power Profile. The Reference Design is using MII in 100Mbit full duplex as Ethernet link. The reference design uses two Ethernet ports for forwarding between each other via frame drop FIFOs which mimic another core introducing delay (Switch or Redundancy Core). The Transparent Clock reference design therefore acts like a node in a Daisy Chain. Multiple instances of the reference design can be chained up to test the inaccuracy introduced by multiple hops.
For the implementation on the NetFpga CML board 2 additional ports exist which just directly forward the traffic between the two ports also via frame drop FIFOs but without TC functionality.
All generics can be adapted to the specific needs.

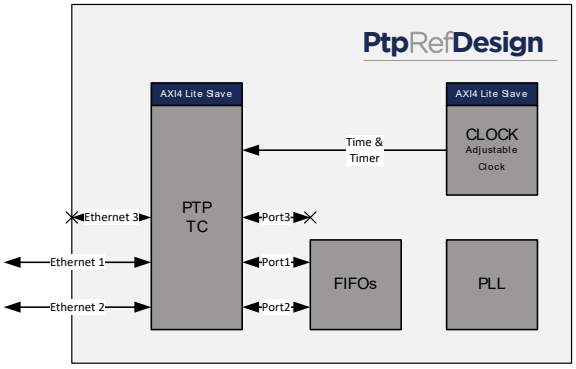

Figure 23: Reference Design

8.1 Intel/Altera: Terasic SocKit

The SocKit board is an FPGA board from Terasic Inc. with a Cyclone V SoC FPGA from Intel/Altera. (http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=816) and an Ethernet adapter in HSMC form factor HSMC-NET also from Terasic: (http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=71&No=355)

1. Open Quartus 16.x

2. Open Project /PTP/Refdesign/Altera/SocKit/PtpTransparentClockMii/PtpTransparentClock.qpf
3. If the optional core PPS Master Clock is available add the files from the corresponding folders (PPS/Core, PPS/Library and PPS/Package)
4. Change the generics (PpsMasterAvailable_Gen) in Quartus (in the settings menu, not in VHDL) to true for the optional cores that are available.
5. Rerun implementation
6. Download to FPGA via JTAG

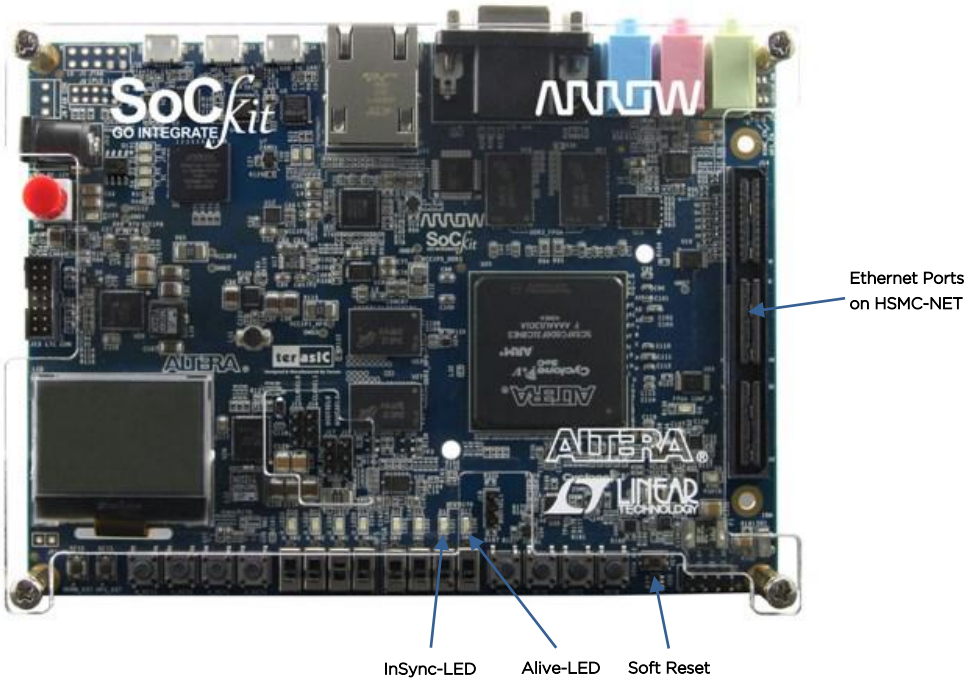

Figure 24: SocKit (source Terasic Inc)

For the ports on the HSMC connector the Ethernet to HSMC adapter from Terasic Inc. was used.

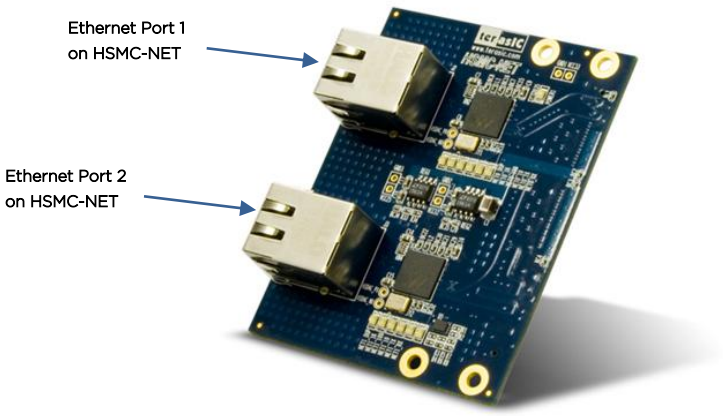

Figure 25: HSMC-NET (source Terasic Inc)

8.2 AMD/Xilinx: Digilent NetFpga CML

The NetFpga CML board is an FPGA board from Digilent Inc. with a Kintex7 FPGA from AMD/Xilinx. (http://store.digilentinc.com/netfpga-1g-cml-kintex-7-fpga-development-board/)

1. Open Vivado 2019.1.
   Note: If a different Vivado version is used, see chapter 8.3.
2. Run TCL script
   /PTP/Refdesign/Xilinx/NetFpga/PtpTransparentClockMii/PtpTransparentClock.tcl
   a. This has to be run only the first time and will create a new Vivado Project
3. If the project has been created before open the project and do not rerun the project TCL
4. If the optional core PPS Master Clock is available add the files from the corresponding folders (PPS/Core, PPS/Library and PPS/Package) to the corresponding Library (PpsLib).
5. Change the generics (PpsMasterAvailable_Gen) in Vivado (in the settings menu, not in VHDL) to true for the optional cores that are available.
6. Rerun implementation
7. Download to FPGA via JTAG

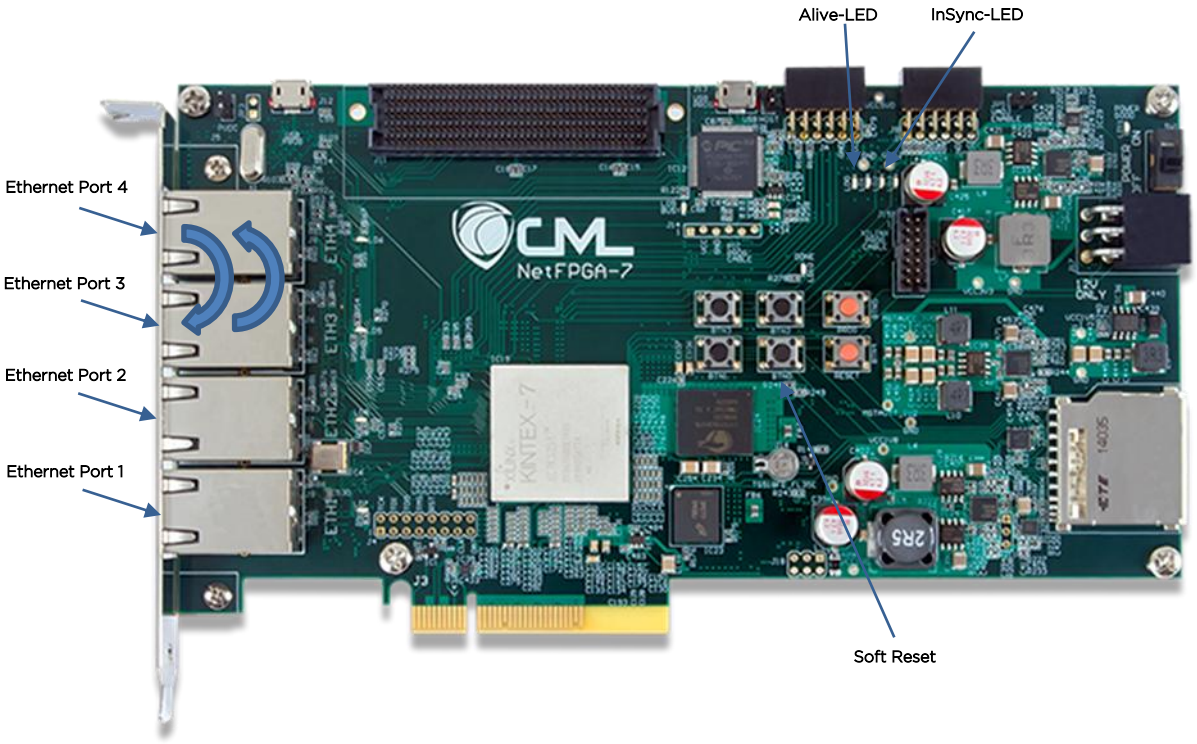

Figure 26: NetFpga CML (source Digilent Inc)

8.3 AMD/Xilinx : Vivado version

The provided TCL script for creation of the reference-design project is targeting AMD/Xilinx Vivado 2019.1.

If a lower Vivado version is used, it is recommended to upgrade to Vivado 2019.1 or higher.

If a higher Vivado version is used, the following steps are recommended:

- Before executing the project creation TCL script, the script's references of Vivado 2019 should be manually replaced to the current Vivado version. For example, if version Vivado 2022 is used, then:
  - The statement occurrences:

    ```
    set_property flow "Vivado Synthesis 2019" $obj
    ```

    shall be replaced by:

    ```
    set_property flow "Vivado Synthesis 2022 $obj
    ```

  - The statement occurrences:

    ```
    set_property flow "Vivado Implementation 2019" $obj
    ```

    shall be replaced by:

    ```
    set_property flow "Vivado Implementation 2022" $obj
    ```

- After executing the project creation TCL script, the AMD/Xilinx IP cores, such as the Clocking Wizard core, might be locked and a version upgrade might be required. To do so:

1. At "Reports" menu, select "Report IP Status".
2. At the opened "IP Status" window, select "Upgrade Selected". The tool will upgrade the version of the selected IP cores.

A List of tables

B List of figures