TsnEndNode

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Sven Meier
Reviewer(s)	-
Version	1.0
Date	12.03.2021

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

The TSN End Node Core from NetTimeLogic is a standalone Time Sensitive Networking (TSN) core according to IEEE 802.1 and IEEE 1588 standards.
It allows to connect to a TSN network, supporting time synchronization, priorities, traffic scheduling, cyclic forwarding, frame preemption, credit-based shaping, etc.
The core is intercepting the path between an Ethernet PHYs and an Ethernet core that forwards or handles Ethernet frames (MAC, PHY or Switch). In addition, it has up to 8 streaming ports which allow to send to and receive frames directly from the specific priority queues and phases.
All tables, protocols and algorithms are implemented in the core, no CPU is required, except for configuration. This allows running TSN protocols completely independent and standalone from the user application. The core can be configured either by signals or by an AXI4Light-Slave Register interface.

Key Features:

- Up to 8 different priority queues, with freely definable VLAN priorities
- Up to 8 phases per cycle according to priority queues
- Cycle time and phase durations freely configurable (max 1 ms)
- Frame scheduling according to IEEE 802.1 Qbv
- Cyclic forwarding according to IEEE 802.1 Qch
- Credit based shaper according to IEEE 802.1 Qav
- Frame preemption according to IEEE 802.1 Qbu and IEEE 802.3 br can be enabled for the lowest priority to allow maximum bandwidth usage
- Registerset to configure according to IEEE 802.1 Qcc
- Synchronization with sub-microsecond accuracy according to IEEE 1588 Default-, Utility- or Power-Profile or according to IEEE 802.1 AS
- Intercepts path between MAC and PHY
- Supports up to 8 AXI streaming interfaces, one for each priority/phase
- Full line speed
- AXI4 Light register set or static configuration
- MII/RMII/GMII/RGMII Interface support
- Optional frame and error counters per Port
- Optional VLAN tagging and filtering
- Optional Tail Tagging mode
- Optional cut through frame processing

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	09.09.2019	First draft
0.2	31.01.2020	Added Preemption Verification and Hold Release
0.3	27.02.2020	Added cut through mode
0.4	20.03.2020	Added parameter description
0.5	20.01.2021	Allow to disable Frame input sheduling
1.0	12.03.2021	First release and added Load per Priority Parameter

Table 1: Revision History

Content

Definitions

Definitions	
End Node	A node that is source and sink for data

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
IRQ	Interrupt, Signaling to e.g. a CPU
PTP	Precision Time Protocol according to IEEE1588
TB	Testbench
LUT	Look Up Table
LSDU	Link Service Data Unit (Data in a frame)
LPDU	Link Protocol Data Unit (Data in a frame)
FF	Flip Flop
FCS	Frame Check Sequence also known as CRC
FIFO	First In First Out Buffer
RAM	Random Access Memory
RCT	Redundancy Control Trailer
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 TSN End Node Core is meant as a co-processor handling Time Sensitive Networking (TSN) protocols defined in IEEE 802.1 and IEEE 1588.
The core supports features like time synchronization, priority handling, traffic scheduling, shaping and cyclic forwarding and preemption. It intercepts the Media Independent Interface ((R)(G)MII) on the Ethernet path between a MAC/PHY/Switch and a PHY where it handles all TSN protocols. It also provides up to 8 AXI stream interfaces directly connected to the priority queues to receive frames from a specific priority or to send frames to a specific priority. In this case a VLAN tag associated with this priority is added together with a user VLAN id which can be changed on a per frame base. This can be used to handle frames directly within the FPGA, e.g. for very short cycles. Or it can be used to transfer frames between the core and a CPU, e.g. via a DMA core.
The uplink (R)(G)(MII) port is optional when AXI stream interfaces are used
The TSN End Node Core is designed to work in cooperation with the Counter Clock core from NetTimeLogic (not a requirement). It contains an AXI4Light slave for configuration from a CPU, this is however not required since the core can also be configured statically via signals/constants directly from within the FPGA.

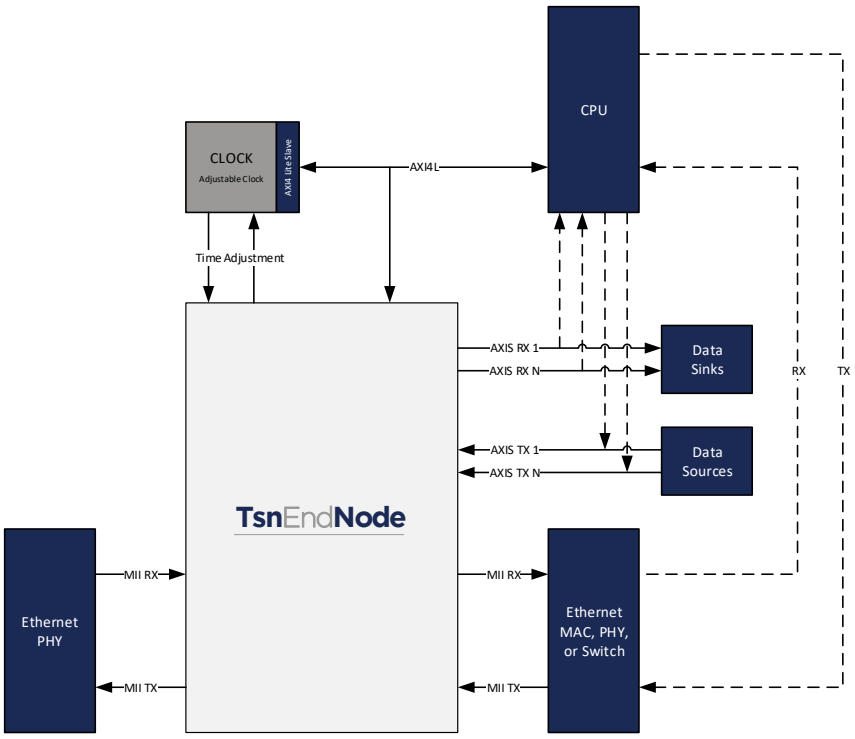


Figure 1: Context Block Diagram

1.2 Function

The TSN End Node Core handles the TSN protocols defined in IEEE 802.1 and IEEE 1588. These are standards which define time synchronization, priority handling, traffic scheduling, shaping and cyclic forwarding and preemption.
It synchronizes itself to the best time source in the network or synchronizes other nodes. This synchronized time is then used by the scheduler to divide the time into cycles and generate phases for different priorities.
VLAN tagged frames are handled according to their priority and forwarded in the correct phases and will additionally undergo traffic shaping and filtering. For frames not tagged or not in one of the high priority queue, best effort forwarding is done and frames can potentially preempted.

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.

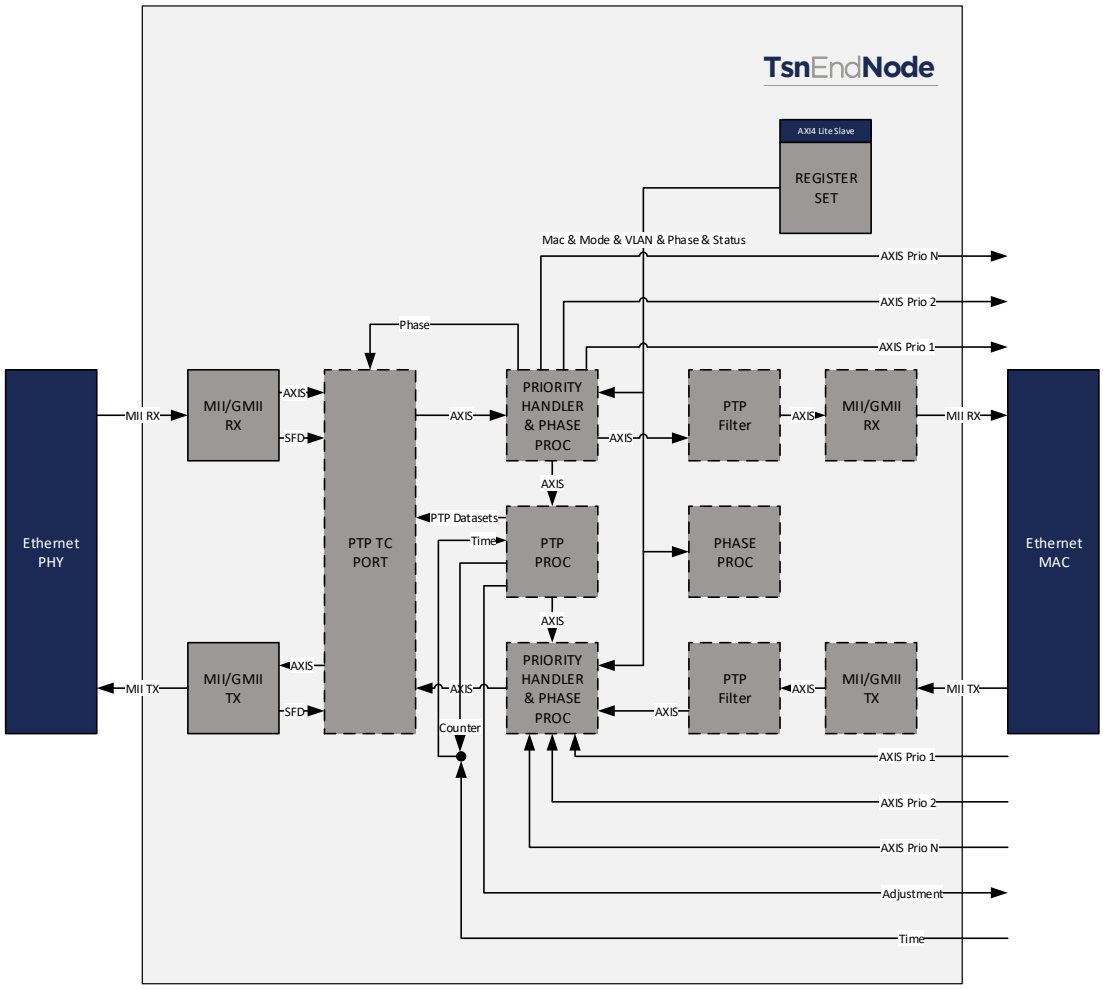


Figure 2: Architecture Block Diagram

Phase Processor

This module is the scheduler which in charge of generating cycles and phases for the different priority queues. It comes in to variants: A simple scheduler which just has a cycle time and a start and stop offset per priority to define the phase. An advance scheduler according to IEEE 802.1 Qbv which also has a cycle time and goes through a list of gate states which define the schedule within a cycle.
Both variants generate a cycle identifier which is used for cyclic forwarding and provide the remaining duration of a phase per priority in nanoseconds which is used by the arbiter to determine if a frame shall be forwarded or not.

PTP Processor

This module is a PTP Ordinary Clock (OC) combined with a PTP Transparent Clock (TC) port. It is in charge of handling all PTP messages and synchronization of the time reference. It receives PTP frames from all ports and can send PTP frames to all ports. The TC ports are in charge of handling the residence within the TSN core.

The PTP TC ports together with the PTP OC form a so called PTP Hybrid Clock (HC).

PTP TC Port

This module measures P2P delay and handles the residence time within the TSN core.

PTP Filter

This module filters out the PTP frames.

Priority Handler

This is the other core module when it comes to TSN related parts. It handles all priorities, phases, traffic shaping and cyclic forwarding as well as external streams.

(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 and also do SFD detection and parts of the preemption.

Register Set

This block allows reading status values and writing configuration.

1.4 Deviations from the Standards or Limitations

Deviations and limitations for IEEE 1588 and IEEE 802.1 AS:

- Hybrid Clock with 1 step Transparent Clock and Ordinary Clock also for IEEE 802.1 AS
- PTP messages are and shall always be sent in priority queue & phase 2
- Preemption not supported for PTP frames
- 2 Step can be disabled also for IEEE802.1 AS
- No Neighbor rates calculations and announcing
- No management objects and counters

Deviations and limitations for IEEE 802.1 Qbv and IEEE 802.1 Qch:

- Cycle start and phases shifted by the delay of the PHY and in transmit direction also optionally by the path delay measured by PTP to align with the cycle of the other node

- Frame sending is aware of the phase length and stops sending when a frame can not fit into the phase
- Option to choose between a simple scheduler and a advanced scheduler: The simple scheduler allows to set the cycle time and a start and stop offset within a cycle for each priority queue to define the phase. The advanced scheduler is the scheduler according to IEEE 802.1 Qbv.
- Max 32 entries in the scheduler list
- Max 1 ms cycle times
- No management objects and counters

Deviations and limitations for IEEE 802.1 Qav:

- Increment and Decrement done on a mixture of flags to get as close as possible to the link speed
- Waits 3 three clock cycles until it continues to count after the end of a frame to determine if no packet was pending
- No management objects and counters

Deviations and limitations for IEEE 802.3 br and IEEE 802.1 Qbu:

- Preemption only for the lowest priority
- No management objects and counters

General limitations:

- Cut Through only if interfaces have same link speed and maximum size filter for a specific queue is disabled and cyclic forwarding is disabled.

2 TSN Basics

TSN is not one standard, it consists of multiple standards:

- IEEE 1588, IEEE 802.1 AS: Time Synchronization (PTP)
- IEEE 802.1 Qbv, IEEE 802.1 Qch: Time Aware Scheduler, Cyclic Forwarding
- IEEE 802.1 Qav: Credit Based Shaper
- IEEE 802.3 br, IEEE 802.1 Qbu: Frame Preemption

Each feature can be enabled and disabled separately. However, some fratures depend on each other. E.g. Cyclic Forwarding depends on the Time Aware Scheduler which itself depends on Time Synchronization.

2.1 Time Synchronization (PTP) Basics

2.1.1 Protocol

PTP means Precision Time Protocol and is standardized in IEEE1588-2008 (or also IEC61588-Ed.2 and a IEEE802.1AS (Rev)). 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.

2.1.2 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:

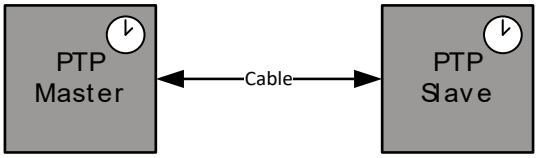


Figure 3: Simple setup

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.

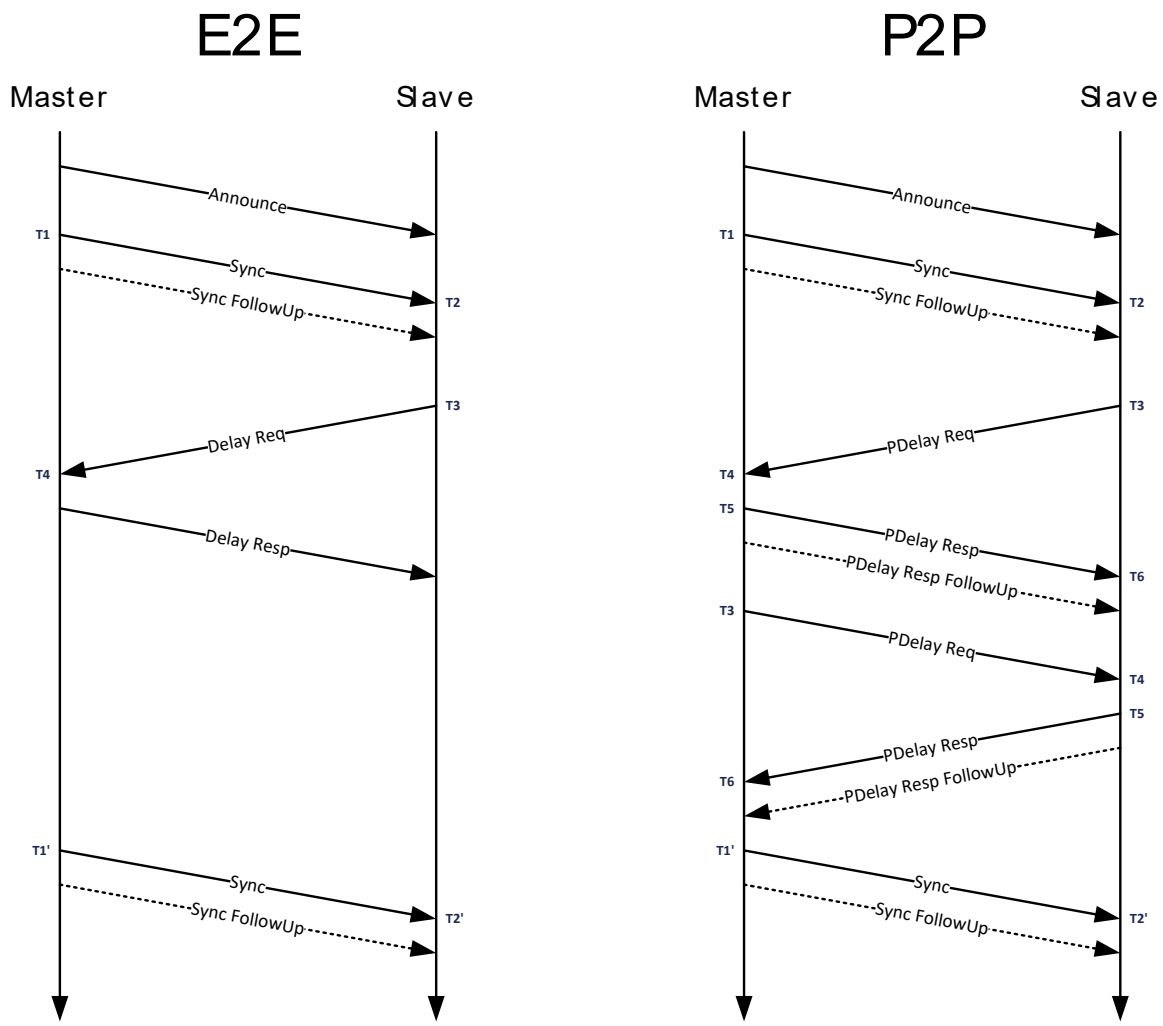


Figure 4: Message exchange simple setup

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

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

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

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.

2.1.3 PTP Nodes

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

2.1.3.1 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.

2.1.3.2 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.

2.1.3.3 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.

2.1.3.4 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.

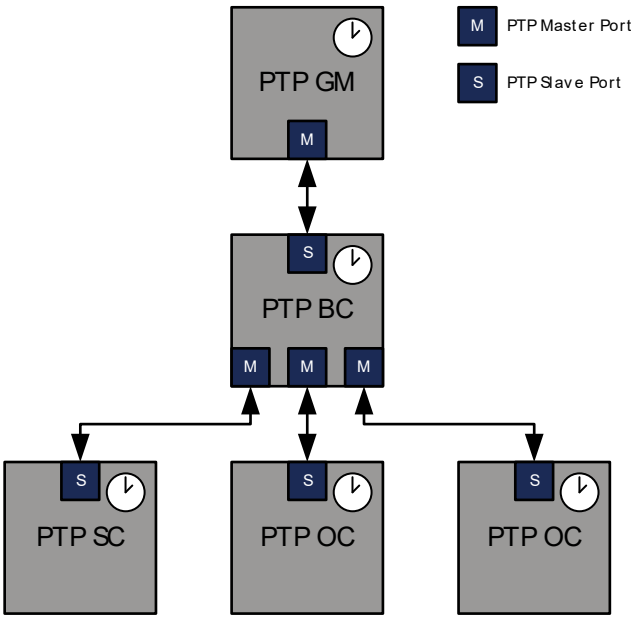


Figure 5: PTP network with Boundary Clock

2.1.3.5 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.

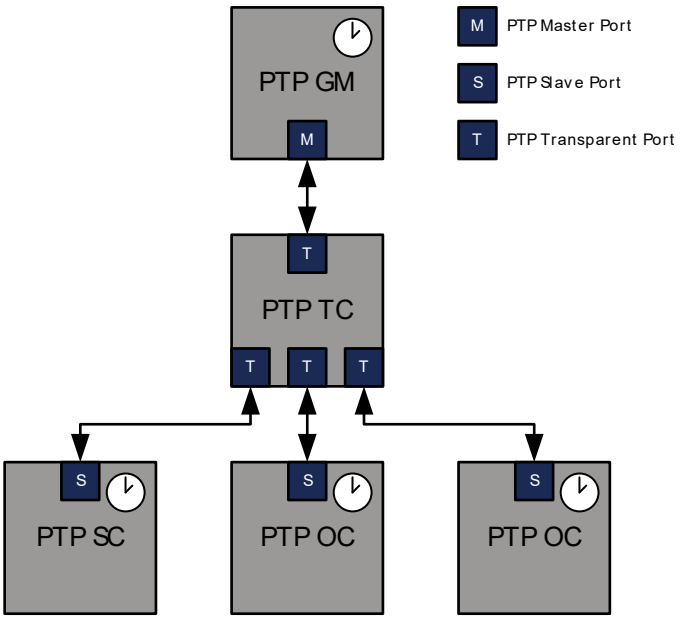


Figure 6: PTP network with Transparent Clock

2.1.3.6 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.

2.1.3.7 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.

2.1.4 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).

2.1.4.1 E2E

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

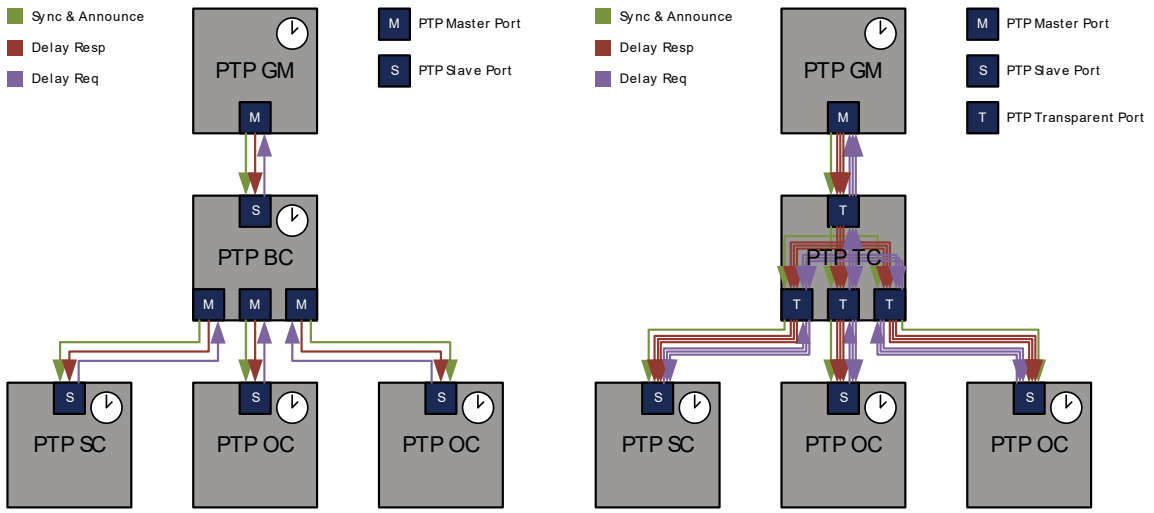


Figure 7: E2E Delay measurement with BC and TC

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.

2.1.4.2 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.

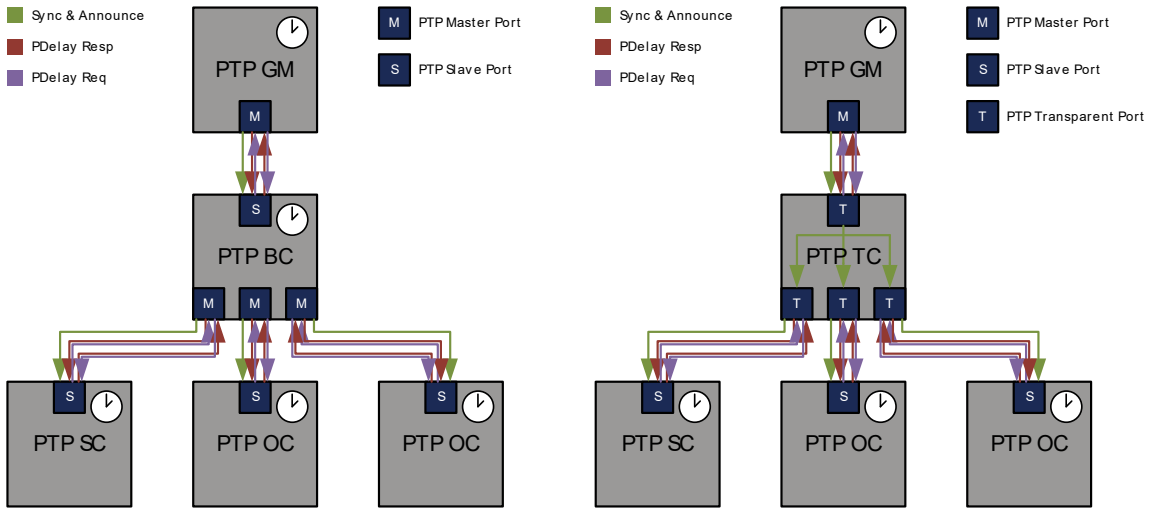


Figure 8: P2P Delay measurement with BC and TC

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.

2.1.4.3 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.

2.1.5 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

2.1.6 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)

2.1.6.1 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 inter-

face, 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.

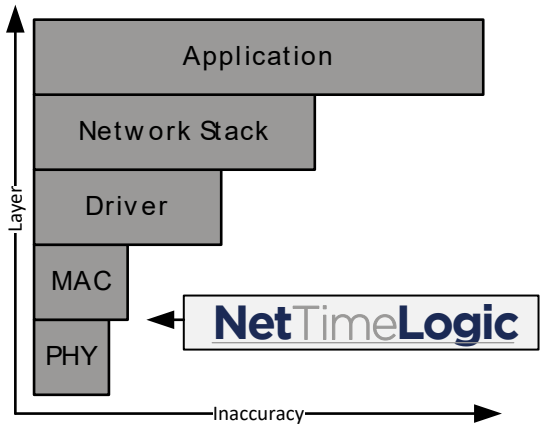


Figure 9: Timestamp Incauracy in the different Layers

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.

2.2 Scheduler (TAS) and Cyclic Forwarding (CQF) Basics

2.2.1 TAS

The Time Aware Scheduler (TAS) is defined in IEEE 802.1 Qbv. It divides the transmission into cycles and the cycles again are divided into phases. The TAS relies on a common time base in all devices which is provided via PTP so each node knows when a cycle starts and when which phase is active. These phases are referred also as priority phases and are representing the priorities in VLAN. It is common to have 3 priorities: Cyclic, Reserved and Best Effort priority. However, it is possible to have up to 8 priority queues representing the 8 priority levels in VLAN. For each priority queue a VLAN priority is assigned, except for the Best Effort priority where all

traffic which is either not VLAN tagged or with a VLAN priority not assigned to any other priority queue is going to.
It is common to define phases the same way as the priority queues and have only one priority queue active per phase and in the order highest priority to lowest priority.

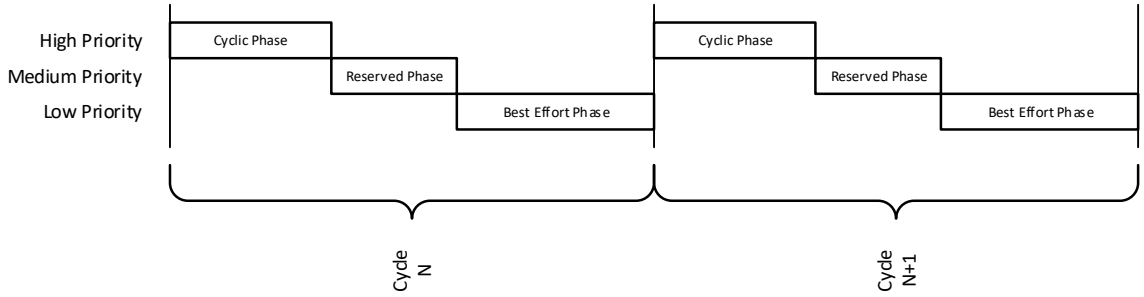


Figure 10: Simple Schedule

Having only one priority enabled per phase often wastes some bandwidth. With the TAS it is possible to have also overlapping phases. If multiple phases are active at the time and both priority queues are ready to send a frame the queue with the higher priority takes precedence over the lower. The arbitration between the different priorities in such a situation can further be influenced by other conditions like the credit shaper (see 2.3)

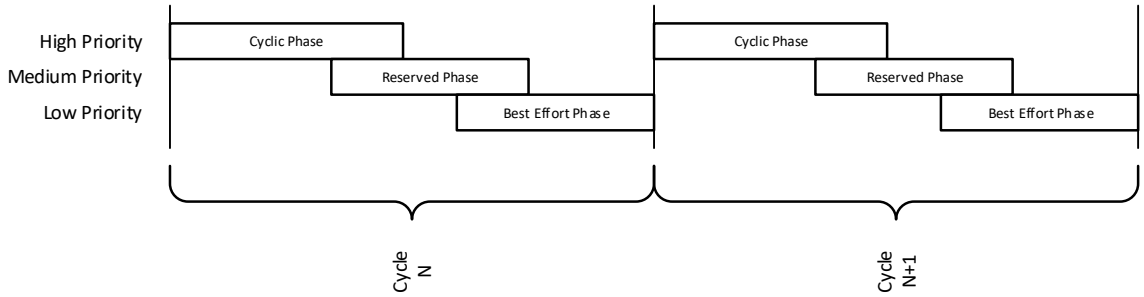


Figure 11: Overlapping Schedule

But in some cases, some priorities need a higher scheduling frequency than the cycle time. With the TAS you can have multiple phases within one cycle to handle this situation.

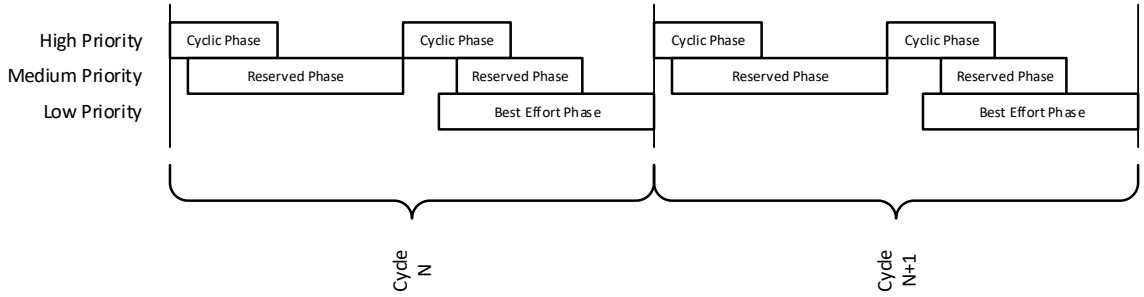


Figure 12: Advanced Schedule with repeating phases

The TAS is going through a list of scheduling entries. Each entry consists of the states of each priority queue (enabled/disabled) (gates) and a duration value which defines for how long this condition shall be valid until the next entry in the list will get active. The scheduler is handling one entry after the other until it reaches either the end of the list or until the next cycle starts which will reset the index to the first entry in the list.

The TAS has an active list which is defining the current schedule and a shadow list which will be copied to the active list when a different schedule shall be activated. There is an algorithm defined how and when the shadow list shall be copied so that all nodes in the system will do the switch simultaneously.

Cycle times are typically somewhere between 32.5 us and 2 ms. The core supports cycle times up to 1 ms due to buffering required.

The core determines when the gate will close and therefore knows if it can still fit a frame into the phase to make sure that it will not be sent partly in a phase where it is not allowed. Also the core takes the path delays into account (measured by PTP) to determine that the cycle and phase is aligned with the destination.

2.2.2 CQF

Cyclic Queuing and Forwarding (CQF) is defined in IEEE 802.1 Qch. CQF relies on the cycles coming from the TAS. The principal of the CQF is to forward a frame always in the next cycle than in which it was received. This guarantees a completely deterministic forwarding behavior (given that there is enough bandwidth) and the worst-case delay for a frame traversing the network can be calculated on the base of how many hops it will pass and the cycle time.

CQF is normally not done for Best Effort traffic, since this has anyhow no bound or guaranteed latency, it is forwarded whenever the Best Effort phase is active independent if in the same phase or not.
If latency is important (bound), the cycle time might have to be lowered to fulfill the latency requirement. Lowering the cycle time however has again other effects on bandwidth, max frame size etc.

This is an example where frames are received from a nonscheduled port and forwarded through two nodes supporting TAS with CQF:

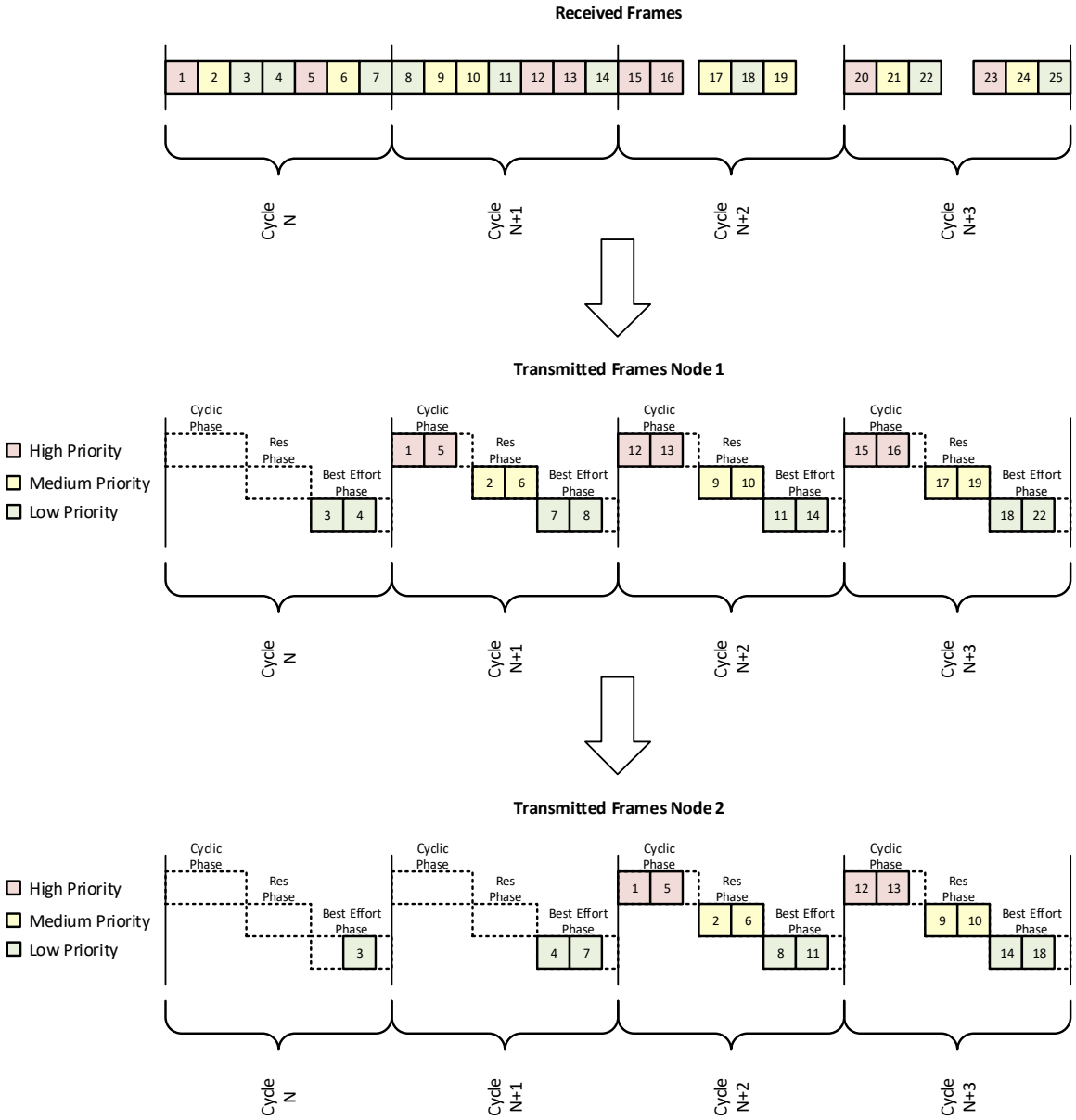


Figure 13: Advanced Schedule with repeating phases

2.3 Credit Based Shaper (CBS) Basics

2.3.1 CBS

The Credit Based Shaper (CBS) is defined in IEEE 802.1 Qav. It allows to shape the traffic per priority queue based on credits which are incremented during wait time and decremented during sending time.

The following conditions apply:

- Whenever the credit is larger or equal to zero the frame is allowed to start transmission.
- If the credit is larger or equal zero and no frame is ready for transmission the credit is set to zero
- If a frame has to wait for transmission the credit is incremented
- If the credit is below zero the credit is incremented
- If the credit reaches the maximum or minimum bound the credit is not further incremented or decremented

The increment and decrement value as well as the maximum and minimum bounds define the maximum burst sizes and maximum bandwidth usage. These values shall be configurable on a per priority queue base. In principal the bounds can be static and the increment and decrement can be adapted accordingly so they faster reach the maximum/minimum while having the same rate between increment and decrement.

The example below shows two priority queues where the priority queue 1 has a slower decrementing rate than priority queue 2, therefore the higher allowed bandwidth. In grey the transmitting phase of each queue and frame is marked.

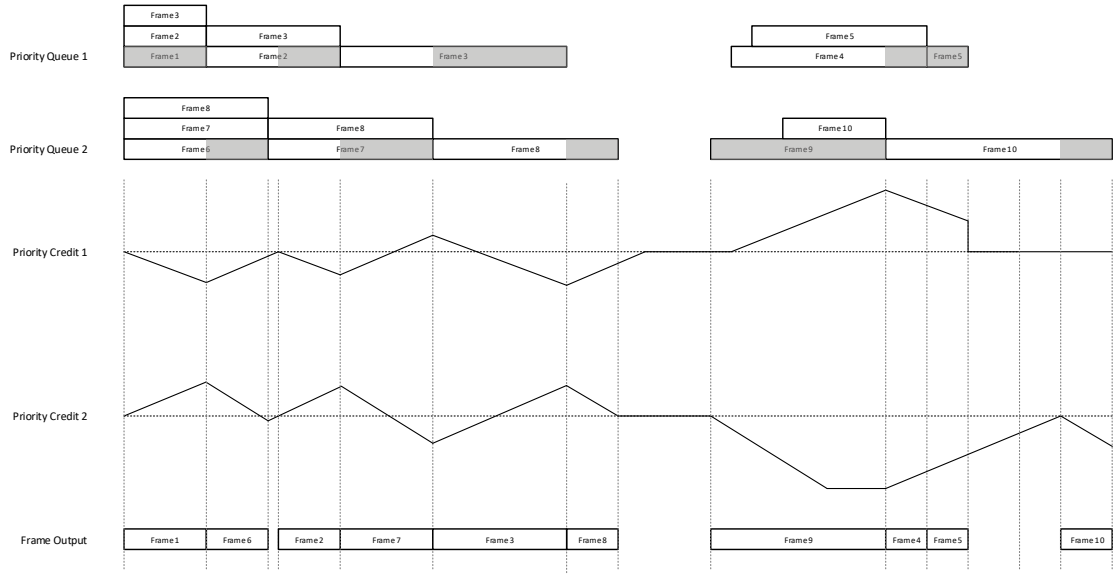


Figure 14: Credit Based Shaping

2.4 Frame Preemption Basics

2.4.1 Frame Preemption

Frame Preemption is defined in IEEE 802.3 br and IEEE 802.1 Qbu. The principal of preemption is that a high priority frame can intercept a low priority frame. A low priority frame can be split into multiple segments before completion.
Preemption has two purposes: The first is to minimize the time a high priority frame has to wait until it can be sent (in worst case 124 bytes). Second it minimizes the guard band in case of scheduled traffic, meaning the
To detect preemptable and preempted frames the last byte of the Preamble and the Start of Frame Delimiter (SFD) as well as the CRC are used.
A “normal” or high priority or non-preemptable frame has 0x55 as the last Preamble byte and 0x5D as the SFD.
Since it is not clear if a frame will be preempted or not the first part of a preemptable frame differs in the Preamble and SFD from fragments.
The first part of a preemptable frame has as a “normal” frame 0x55 as the last preamble byte. The SFD is replaced with a frame counter symbol which is incremented for every new preemptable frame (not fragments). The frame count goes from 0 to 3 and then wraps through 0 and has a special mapping:
Frame 0 => 0xE6
Frame 1 => 0x4C
Frame 2 => 0x7F

Frame 3 => 0xB3

When the frame doesn't get preempted all bytes (including CRC) are sent.
When the frame shall get preempted it will continue to the next 60 byte boundary and will add a new CRC xored with 0xFFFF0000. This is done only if the remaining data is larger or equal than 64bytes, otherwise the remaining data is sent, and no new CRC will be added.
If it got preempted the fragment will have the frame count it belongs to in the last Preamble byte but with a different encoding as in the first part of the preemptable frame:
Frame 0 => 0x61
Frame 1 => 0x52
Frame 2 => 0x9E
Frame 3 => 0x2A

The SFD is replaced with a fragment count starting with fragment 0. The fragment count also goes from 0 to 3 and then wraps through 0 and has as well a special mapping (same as the frame count in the first part):
Fragment 0 => 0xE6
Fragment 1 => 0x4C
Fragment 2 => 0x7F
Fragment 3 => 0xB3

Frames can get preempted multiple times until a frame is completed. E.g. A frame of 1530 bytes could be fragmented in 25 fragments: 24x60 and 1x90 bytes (the last). Frames smaller than 124 bytes will never be preempted since the second fragment would be smaller than the minimum MTU allowed by Ethernet which is 64 bytes.

The example below shows two priority queues where the priority queue 1 has a higher priority and non-preemptable frames and the lower priority queue 2 where the frames get preempted (Frame 5)

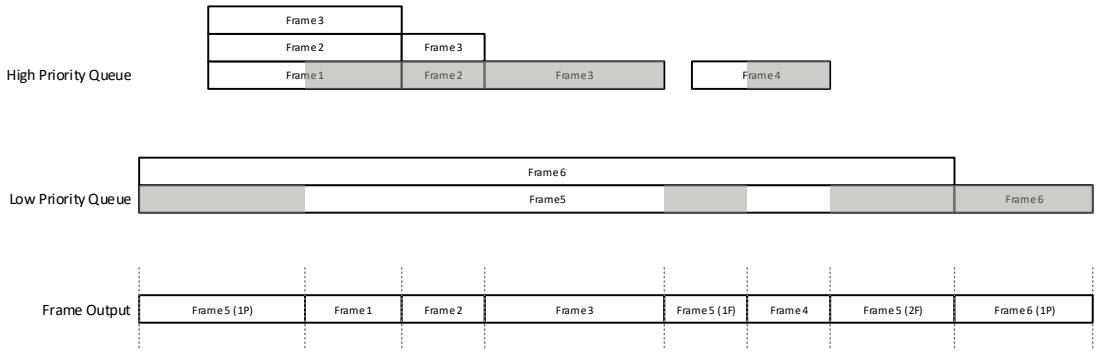


Figure 15: Preempting

The core implements preemption only on the lowest priority queue to have a good bandwidth usage and a minimal guard band. All other priority queues are non-preemptable. To get the maximum bandwidth usage the phase of the higher priority queues could overlap by one max sized frame per priority. If this is acceptable however heavily depends on the requirements.

3 Register Set

This is the register set of the HSR&PRP Core. It is accessible via AXI4 Light 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.

The Register set is split into two basic parts: The TSN and the PTP HC clock part

3.1 Register Overview

Registerset Overview			
Name	Description	Offset	Access
Red TsnControl Reg	Tsn Enable Control Register	0x00000000	RW
Red TsnStatus Reg	Tsn Error Status Register	0x00000004	WC
Red TsnVersion Reg	Tsn Version Register	0x0000000C	RO
Red TsnFrameCountControl Reg	Tsn Status Control Register	0x00000010	RW
Red TsnRxFrameCount Reg	Tsn Received Frames Count Register	0x00000060	RO
Red TsnRxErrCount Reg	Tsn Received Error Frames Count Register	0x00000064	RO
Red TsnTxFrameCount Reg	Tsn Transmitted Frames Count Register	0x00000070	RO
Red TsnTxErrCount Reg	Tsn Transmitted Error Frames Count Register	0x00000074	RO
Red TsnConfigControl Reg	Tsn Configuration Control Register	0x00000080	RW
Red TsnConfigMode Reg	Tsn Configuration Mode Register	0x00000084	RW
Red TsnMacControl Reg	Tsn MAC Control Register	0x00000100	RW
Red TsnMac1 Reg	Tsn MAC 0-3 Register	0x00000104	RW

Red TsnMac2 Reg	Tsn MAC 4-5 Register	0x00000108	RW
Red TsnCreditControl Reg	Tsn Credit Control Register	0x00000200	RW
Red TsnCredit0 Reg	Tsn Credit 0 Register	0x00000204	RW
Red TsnCredit1 Reg	Tsn Credit 1 Register	0x00000208	RW
Red TsnCredit2 Reg	Tsn Credit 2 Register	0x0000020C	RW
Red TsnCredit3 Reg	Tsn Credit 3 Register	0x00000210	RW
Red TsnCredit4 Reg	Tsn Credit 4 Register	0x00000214	RW
Red TsnCredit5 Reg	Tsn Credit 5 Register	0x00000218	RW
Red TsnCredit6 Reg	Tsn Credit 6 Register	0x0000021C	RW
Red TsnCredit7Reg	Tsn Credit 7 Register	0x00000220	RW
Red TsnCreditMin0 Reg	Tsn Credit Minimum 0 Register	0x00000224	RW
Red TsnCreditMin1 Reg	Tsn Credit Minimum 1 Register	0x00000228	RW
Red TsnCreditMin2 Reg	Tsn Credit Minimum 2 Register	0x0000022C	RW
Red TsnCreditMin3 Reg	Tsn Credit Minimum 3 Register	0x00000230	RW
Red TsnCreditMin4 Reg	Tsn Credit Minimum 4 Register	0x00000234	RW
Red TsnCreditMin5 Reg	Tsn Credit Minimum 5 Register	0x00000238	RW
Red TsnCreditMin6 Reg	Tsn Credit Minimum 6 Register	0x0000023C	RW
Red TsnCreditMin7Reg	Tsn Credit Minimum 7 Register	0x00000240	RW
Red TsnCreditMax0 Reg	Tsn Credit Maximum 0 Register	0x00000244	RW
Red TsnCreditMax1 Reg	Tsn Credit Maximum 1 Register	0x00000248	RW
Red TsnCreditMax2 Reg	Tsn Credit Maximum 2 Register	0x0000024C	RW
Red TsnCreditMax3 Reg	Tsn Credit Maximum 3 Register	0x00000250	RW
Red TsnCreditMax4 Reg	Tsn Credit Maximum 4 Register	0x00000254	RW
Red TsnCreditMax5 Reg	Tsn Credit Maximum 5 Register	0x00000258	RW
Red TsnCreditMax6 Reg	Tsn Credit Maximum 6 Register	0x0000025C	RW
Red TsnCreditMax7Reg	Tsn Credit Maximum 7 Register	0x00000260	RW

Red TsnCreditMin0 Reg	Tsn Credit Minimum 0 Register	0x00000224	RW
Red TsnCreditMin1 Reg	Tsn Credit Minimum 1 Register	0x00000228	RW
Red TsnCreditMin2 Reg	Tsn Credit Minimum 2 Register	0x0000022C	RW
Red TsnCreditMin3 Reg	Tsn Credit Minimum 3 Register	0x00000230	RW
Red TsnCreditMin4 Reg	Tsn Credit Minimum 4 Register	0x00000234	RW
Red TsnCreditMin5 Reg	Tsn Credit Minimum 5 Register	0x00000238	RW
Red TsnCreditMin6 Reg	Tsn Credit Minimum 6 Register	0x0000023C	RW
Red TsnCreditMin7Reg	Tsn Credit Minimum 7 Register	0x00000240	RW
Red TsnCreditMax0 Reg	Tsn Credit Maximum 0 Register	0x00000244	RW
Red TsnCreditMax1 Reg	Tsn Credit Maximum 1 Register	0x00000248	RW
Red TsnCreditMax2 Reg	Tsn Credit Maximum 2 Register	0x0000024C	RW
Red TsnCreditMax3 Reg	Tsn Credit Maximum 3 Register	0x00000250	RW
Red TsnCreditMax4 Reg	Tsn Credit Maximum 4 Register	0x00000254	RW
Red TsnCreditMax5 Reg	Tsn Credit Maximum 5 Register	0x00000258	RW
Red TsnCreditMax6 Reg	Tsn Credit Maximum 6 Register	0x0000025C	RW
Red TsnCreditMax7Reg	Tsn Credit Maximum 7 Register	0x00000260	RW
Red TsnMaxDataSizeControl Reg	Tsn Max Data Size Control Register	0x00000280	RW
Red TsnMaxDataSize0 Reg	Tsn Max Data Size 0 Register	0x00000284	RW
Red TsnMaxDataSize1 Reg	Tsn Max Data Size 1 Register	0x00000288	RW
Red TsnMaxDataSize2 Reg	Tsn Max Data Size 2 Register	0x0000028C	RW
Red TsnMaxDataSize3 Reg	Tsn Max Data Size 3 Register	0x00000290	RW
Red TsnMaxDataSize4 Reg	Tsn Max Data Size 4 Register	0x00000294	RW
Red TsnMaxDataSize5 Reg	Tsn Max Data Size 5 Register	0x00000298	RW
Red TsnMaxDataSize6 Reg	Tsn Max Data Size 6 Register	0x0000029C	RW
Red TsnMaxDataSize7Reg	Tsn Max Data Size 7 Register	0x000002A0	RW
Red TsnPreemptionControl Reg	Tsn Preemption Control Register	0x000002B0	RW

Red TsnPreemptionVerifyInterval Reg	Tsn Preemption Verify Interval Register	0x000002B4	RW
Red TsnPrioPhaseControl Reg	Tsn Prio and Phase Control Register	0x00000300	RW
Red TsnPrioVlan0 Reg	Tsn Prio VLAN 0 Register	0x00000310	RW
Red TsnPrioVlan1 Reg	Tsn Prio VLAN 1 Register	0x00000314	RW
Red TsnPrioVlan2 Reg	Tsn Prio VLAN 2 Register	0x00000318	RW
Red TsnPrioVlan3 Reg	Tsn Prio VLAN 3 Register	0x0000031C	RW
Red TsnPrioVlan4 Reg	Tsn Prio VLAN 4 Register	0x00000320	RW
Red TsnPrioVlan5 Reg	Tsn Prio VLAN 5 Register	0x00000324	RW
Red TsnPrioVlan6 Reg	Tsn Prio VLAN 6 Register	0x00000328	RW
Red TsnPhaseSplPeriod Reg	Tsn Simple Phase Period Register *	0x00000340	RW
Red TsnPhaseSplStart0 Reg	Tsn Simple Phase Start 0 Register *	0x00000380	RW
Red TsnPhaseSplStart1 Reg	Tsn Simple Phase Start 1 Register *	0x00000384	RW
Red TsnPhaseSplStart2 Reg	Tsn Simple Phase Start 2 Register *	0x00000388	RW
Red TsnPhaseSplStart3 Reg	Tsn Simple Phase Start 3 Register *	0x0000038C	RW
Red TsnPhaseSplStart4 Reg	Tsn Simple Phase Start 4 Register *	0x00000390	RW
Red TsnPhaseSplStart5 Reg	Tsn Simple Phase Start 5 Register *	0x00000394	RW
Red TsnPhaseSplStart6 Reg	Tsn Simple Phase Start 6 Register *	0x00000398	RW
Red TsnPhaseSplStart7 Reg	Tsn Simple Phase Start 7 Register *	0x0000039C	RW
Red TsnPhaseSplStop0 Reg	Tsn Simple Phase Stop 0 Register *	0x000003A0	RW
Red TsnPhaseSplStop1 Reg	Tsn Simple Phase Stop 1 Register *	0x000003A4	RW
Red TsnPhaseSplStop2 Reg	Tsn Simple Phase Stop 2 Register *	0x000003A8	RW
Red TsnPhaseSplStop3 Reg	Tsn Simple Phase Stop 3 Register *	0x000003AC	RW
Red TsnPhaseSplStop4 Reg	Tsn Simple Phase Stop 4 Register *	0x000003B0	RW
Red TsnPhaseSplStop5 Reg	Tsn Simple Phase Stop 5 Register *	0x000003B4	RW
Red TsnPhaseSplStop6 Reg	Tsn Simple Phase Stop 6 Register *	0x000003B8	RW
Red TsnPhaseSplStop7Reg	Tsn Simple Phase Stop 7 Register *	0x000003BC	RW

Red TsnPhaseAdvCycleTime Reg	Tsn Advanced Phase Cycle Time Register *	0x00000340	RW
Red TsnPhaseAdvCycleTimeExt Reg	Tsn Advanced Phase Cycle Time Extension Register *	0x00000344	RW
Red TsnPhaseAdvBaseTimeL Reg	Tsn Advanced Phase Base Time Nanoseconds Register *	0x00000348	RW
Red TsnPhaseAdvBaseTimeH Reg	Tsn Advanced Phase Base Time Seconds Register *	0x0000034C	RW
Red TsnPhaseAdvGateStates Reg	Tsn Advanced Phase Initial Gate States Register *	0x00000350	RW
Red TsnPhaseAdvCtrlListLength Reg	Tsn Advanced Phase List Length Register *	0x00000370	RW
Red TsnPhaseAdvCtrlListEntry0 Reg	Tsn Advanced Phase List Entry 0 Register *	0x00000380	RW
Red TsnPhaseAdvCtrlListEntry1 Reg	Tsn Advanced Phase List Entry 1 Register *	0x00000384	RW
Red TsnPhaseAdvCtrlListEntry2 Reg	Tsn Advanced Phase List Entry 2 Register *	0x00000388	RW
Red TsnPhaseAdvCtrlListEntry3 Reg	Tsn Advanced Phase List Entry 3 Register *	0x0000038C	RW
Red TsnPhaseAdvCtrlListEntry4 Reg	Tsn Advanced Phase List Entry 4 Register *	0x00000390	RW
Red TsnPhaseAdvCtrlListEntry5 Reg	Tsn Advanced Phase List Entry 5 Register *	0x00000394	RW
Red TsnPhaseAdvCtrlListEntry6 Reg	Tsn Advanced Phase List Entry 6 Register *	0x00000398	RW
Red TsnPhaseAdvCtrlListEntry7 Reg	Tsn Advanced Phase List Entry 7 Register *	0x0000039C	RW
Red TsnPhaseAdvCtrlListEntry8 Reg	Tsn Advanced Phase List Entry 8 Register *	0x000003A0	RW
Red TsnPhaseAdvCtrlListEntry9 Reg	Tsn Advanced Phase List Entry 9 Register *	0x000003A4	RW
Red TsnPhaseAdvCtrlListEntry10 Reg	Tsn Advanced Phase List Entry 10 Register *	0x000003A8	RW
Red TsnPhaseAdvCtrlListEntry11 Reg	Tsn Advanced Phase List Entry 11 Register *	0x000003AC	RW
Red TsnPhaseAdvCtrlListEntry12 Reg	Tsn Advanced Phase List Entry 12 Register *	0x000003B0	RW
Red TsnPhaseAdvCtrlListEntry13 Reg	Tsn Advanced Phase List Entry 13 Register *	0x000003B4	RW
Red TsnPhaseAdvCtrlListEntry14 Reg	Tsn Advanced Phase List Entry 14 Register *	0x000003B8	RW
Red TsnPhaseAdvCtrlListEntry15 Reg	Tsn Advanced Phase List Entry 15 Register *	0x000003BC	RW
Red TsnPhaseAdvCtrlListEntry16 Reg	Tsn Advanced Phase List Entry 16 Register *	0x000003C0	RW
Red TsnPhaseAdvCtrlListEntry17 Reg	Tsn Advanced Phase List Entry 17 Register *	0x000003C4	RW
Red TsnPhaseAdvCtrlListEntry18 Reg	Tsn Advanced Phase List Entry 18 Register *	0x000003C8	RW
Red TsnPhaseAdvCtrlListEntry19 Reg	Tsn Advanced Phase List Entry 19 Register *	0x000003CC	RW

Red TsnPhaseAdvCtrlListEntry20 Reg	Tsn Advanced Phase List Entry 20 Register *	0x000003D0	RW
Red TsnPhaseAdvCtrlListEntry21 Reg	Tsn Advanced Phase List Entry 21 Register *	0x000003D4	RW
Red TsnPhaseAdvCtrlListEntry22 Reg	Tsn Advanced Phase List Entry 22 Register *	0x000003D8	RW
Red TsnPhaseAdvCtrlListEntry23 Reg	Tsn Advanced Phase List Entry 23 Register *	0x000003DC	RW
Red TsnPhaseAdvCtrlListEntry24 Reg	Tsn Advanced Phase List Entry 24 Register *	0x000003E0	RW
Red TsnPhaseAdvCtrlListEntry25 Reg	Tsn Advanced Phase List Entry 25 Register *	0x000003E4	RW
Red TsnPhaseAdvCtrlListEntry26 Reg	Tsn Advanced Phase List Entry 26 Register *	0x000003E8	RW
Red TsnPhaseAdvCtrlListEntry27 Reg	Tsn Advanced Phase List Entry 27 Register *	0x000003EC	RW
Red TsnPhaseAdvCtrlListEntry28 Reg	Tsn Advanced Phase List Entry 28 Register *	0x000003F0	RW
Red TsnPhaseAdvCtrlListEntry29 Reg	Tsn Advanced Phase List Entry 29 Register *	0x000003F4	RW
Red TsnPhaseAdvCtrlListEntry30 Reg	Tsn Advanced Phase List Entry 30 Register *	0x000003F8	RW
Red TsnPhaseAdvCtrlListEntry31 Reg	Tsn Advanced Phase List Entry 31 Register *	0x000003FC	RW
Ptp OcControl Reg	OC Enable Control Register	0x00001000	RW
Ptp OcStatus Reg	OC Error Status Register	0x00001004	WC
Ptp OCVersion Reg	OC Version Register	0x0000100C	RO
Ptp OcConfigControl Reg	OC Configuration Control Register	0x00001080	RW
Ptp OcConfigProfile Reg	OC Configuration Profile Register	0x00001084	RW
Ptp OcConfigVlan Reg	OC Configuration VLAN Register	0x00001088	RW
Ptp OcConfigIp Reg	OC Configuration IP Register	0x0000108C	RW
Ptp OcDefaultDsControl Reg	OC Default Dataset Control Register	0x00001100	RW
Ptp OcDefaultDs1 Reg	OC Default Dataset 1 Register	0x00001104	RW
Ptp OcDefaultDs2 Reg	OC Default Dataset 2 Register	0x00001108	RW
Ptp OcDefaultDs3 Reg	OC Default Dataset 3 Register	0x0000110C	RW
Ptp OcDefaultDs4 Reg	OC Default Dataset 4 Register	0x00001110	RW
Ptp OcDefaultDs5 Reg	OC Default Dataset 5 Register	0x00001114	RW
Ptp OcDefaultDs6 Reg	OC Default Dataset 6 Register	0x00001118	RW

Ptp OcDefaultDs7 Reg	OC Default Dataset 7 Register	0x0000111C	RO
Ptp OcPortDsControl Reg	OC Port Dataset Control Register	0x00001200	RW
Ptp OcPortDs1 Reg	OC Port Dataset 1 Register	0x00001204	RO
Ptp OcPortDs2 Reg	OC Port Dataset 2 Register	0x00001208	RO
Ptp OcPortDs3 Reg	OC Port Dataset 3 Register	0x0000120C	RO
Ptp OcPortDs4 Reg	OC Port Dataset 4 Register	0x00001210	RW
Ptp OcPortDs5 Reg	OC Port Dataset 5 Register	0x00001214	RW
Ptp OcPortDs6 Reg	OC Port Dataset 6 Register	0x00001218	RW
Ptp OcPortDs7 Reg	OC Port Dataset 7 Register	0x0000121C	RW
Ptp OcPortDs8 Reg	OC Port Dataset 8 Register	0x00001220	RW
Ptp OcCurrentDsControl Reg	OC Current Dataset Control Register	0x00001300	RW
Ptp OcCurrentDs1 Reg	OC Current Dataset 1 Register	0x00001304	RO
Ptp OcCurrentDs2 Reg	OC Current Dataset 2 Register	0x00001308	RO
Ptp OcCurrentDs3 Reg	OC Current Dataset 3 Register	0x0000130C	RO
Ptp OcCurrentDs4 Reg	OC Current Dataset 4 Register	0x00001310	RO
Ptp OcCurrentDs5 Reg	OC Current Dataset 5 Register	0x00001314	RO
Ptp OcParentDsControl Reg	OC Parent Dataset Control Register	0x00001400	RW
Ptp OcParentDs1 Reg	OC Parent Dataset 1 Register	0x00001404	RO
Ptp OcParentDs2 Reg	OC Parent Dataset 2 Register	0x00001408	RO
Ptp OcParentDs3 Reg	OC Parent Dataset 3 Register	0x0000140C	RO
Ptp OcParentDs4 Reg	OC Parent Dataset 4 Register	0x00001410	RO
Ptp OcParentDs5 Reg	OC Parent Dataset 5 Register	0x00001414	RO
Ptp OcParentDs6 Reg	OC Parent Dataset 6 Register	0x00001418	RO
Ptp OcParentDs7 Reg	OC Parent Dataset 7 Register	0x0000141C	RO
Ptp OcParentDs8 Reg	OC Parent Dataset 8 Register	0x00001420	RO
Ptp OcParentDs9 Reg	OC Parent Dataset 9 Register	0x00001424	RO

Ptp OcTimePropertiesDsControl Reg	OC Time Properties Dataset Control Register	0x00001500	RW
Ptp OcTimePropertiesDs1 Reg	OC Time Properties Datasett 1 Register	0x00001504	RW
Ptp OcTimePropertiesDs2 Reg	OC Time Properties Datasett 2 Register	0x00001508	RW
Ptp OcTimePropertiesDs3 Reg	OC Time Properties Datasett 3 Register	0x0000150C	RW
Ptp OcTimePropertiesDs4 Reg	OC Time Properties Datasett 4 Register	0x00001510	RW
Ptp OcTimePropertiesDs5 Reg	OC Time Properties Datasett 5 Register	0x00001514	RW
Ptp OcTimePropertiesDs6 Reg	OC Time Properties Datasett 6 Register	0x00001518	RW
Ptp OcTimePropertiesDs7 Reg	OC Time Properties Datasett 7 Register	0x0000151C	RW
Ptp OcTimePropertiesDs8 Reg	OC Time Properties Datasett 8 Register	0x00001520	RW
Ptp OcTimePropertiesDs9 Reg	OC Time Properties Datasett 9 Register	0x00001524	RW
Ptp TcControl Reg	TC Enable Control Register	0x00001800	RW
Ptp TcStatus Reg	TC Error Status Register	0x00001804	WC
Ptp TcVersion Reg	TC Version Register	0x0000180C	RO
Ptp TcConfigControl Reg	TC Configuration Control Register	0x00001880	RW
Ptp TcConfigProfile Reg	TC Configuration Profile Register	0x00001884	RW
Ptp TcConfigVlan Reg	TC Configuration VLAN Register	0x00001888	RW
Ptp TcConfigIp Reg	TC Configuration IP Register	0x0000188C	RW
Ptp TcDefaultDsControl Reg	TC Default Dataset Control Register	0x00001900	RW
Ptp TcDefaultDs1 Reg	TC Default Dataset 1 Register	0x00001904	RW
Ptp TcDefaultDs2 Reg	TC Default Dataset 2 Register	0x00001908	RW
Ptp TcDefaultDs3 Reg	TC Default Dataset 3 Register	0x0000190C	RW
Ptp TcDefaultDs4 Reg	TC Default Dataset 4 Register	0x00001910	RO
Ptp TcPortDsControl Reg	TC Port Dataset Control Register	0x00001A00	RW
Ptp TcPortDs1 Reg	TC Port Dataset 1 Register	0x00001A04	RO
Ptp TcPortDs2 Reg	TC Port Dataset 2 Register	0x00001A08	RO
Ptp TcPortDs3 Reg	TC Port Dataset 3 Register	0x00001A10	RW

Ptp TcPortDs4 Reg	TC Port Dataset 4 Register	0x00001A1C	RW
Ptp TcPortDs5 Reg	TC Port Dataset 5 Register	0x00001A20	RW

Table 4: Register Set Overview

* Either the “Simple” or “Advanced” Phase generator is used, they are using partly the same address space

3.2 Register Descriptions

3.2.1 RED General

3.2.1.1 RED Tsn Control Register

Used for general control over the TSN Core, all configurations on the core shall only be done when disabled.

RED TsnControl 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 RED Tsn Status Register

Shows the current status and features of the RED Tsn core and status bits.

RED TsnStatus Reg																													
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
CUT_THROUGH_SUP	SIMPLE_SCHED_SUP	PREEMPTION_SUP	MAX_DATA_SIZE_SUP	CYCLE_SUP	CREDIT_SUP	PHASE_SUP	PRIO_SUP	-				NR_PRIOS				-					LINK_	-					PREEMPTION	-			
RO	RO	RO	RO	RO	RO	RO	RO	RO				RO				RO					RO	RO					RO	RO			
Reset: 0x00000000																															
Offset: 0x0004																															

Name	Description	Bits	Access
CUT_THROUGH_SUP	Cut Through support	Bit:31	RO
SIMPLE_SCHED_SUP	If the core supports the simple scheduler (one start and end point per cycle)	Bit: 30	RO
PREEMPTION_SUP	If the core supports frame preemption for best effort traffic	Bit: 29	RO
MAX_DATA_SIZE_SUP	If the core supports max data size filters	Bit: 28	RO
CYCLE_SUP	If the core supports cyclic forwarding	Bit: 27	RO

CREDIT_SUP	If the core supports credit shapping	Bit: 26	RO
PHASE_SUP	If the core supports phases	Bit: 25	RO
PRIO_SUP	If the core supports priority queues	Bit: 24	RO
-	Reserved, read 0	Bit:23:20	RO
NR_PRIOS	How many priorities the core supports (default 3)	Bit: 19:16	RO
-	Reserved, read 0	Bit:15:11	RO
LINK	Link state of Port	Bit: 10	RO
-	Reserved, read 0	Bit:9:5	RO
PREEMPTION	If Preemption is active (either hard set or via Verification)	Bit: 4	RO
-	Reserved, read 0	Bit:3:0	RO

3.2.1.3 RED Tsn 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.

RED TsnVersion 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																															
0xXXXXXXXX																															
Offset: 0x000C																															

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

3.2.1.4 RED Tsn Frame Count Control Register

Used for clearing all statistic counters

Only available if the generic PortStatusSupport_Gen is true.

RED TsnFrameCountControl 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
																															CLEAR
RO																															RW
Reset: 0x00000000																															
Offset: 0x0010																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
CLEAR	Clear all counters (self cleared)	Bit: 0	RW

3.2.1.5 RED Tsn RX Frame Count Register

Received number of frames. Includes also erroneous frames.
Only available if the generic PortStatusSupport_Gen is true.

Red TsnRxFrameCount 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
RX_FRAMES																															
RO																															
0x00000000																															
Offset: 0x0060																															

Name	Description	Bits	Access
RX_FRAMES	Received number of frames	Bit: 31:0	RO

3.2.1.6 RED Tsn RX Error Count Register

Received number of erroneous frames.

Only available if the generic PortStatusSupport_Gen is true.

Red TsnRxErrCount 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
RX_ERR																															
RO																															
0x00000000																															
Offset: 0x0064																															

Name	Description	Bits	Access
RX_ERR	Received number of erroneous frames on Port [X] X=A,B,C	Bit: 31:0	RO

3.2.1.7 RED Tsn TX Frame Count Register

Transmitted number of frames. Includes also erroneous frames.

Only available if the generic PortStatusSupport_Gen is true.

Red TsnTxFrameCount 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
RX_FRAMES																															
RO																															
0x00000000																															
Offset: 0x0070																															

Name	Description	Bits	Access
RX_FRAMES	Transmitted number of frames	Bit: 31:0	RO

3.2.1.8 RED Tsn TX Error Count Registers

Transmitted number of erroneous frames.

Only available if the generic PortStatusSupport_Gen is true.

Red TsnTxErrCount 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
RX_ERR																															
RO																															
0x00000000																															
Offset: 0x0074																															

Name	Description	Bits	Access
RX_ERR	Transmitted number of erroneous frames	Bit: 31:0	RO

3.2.1.9 RED Tsn Config Control Register

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

RED TsnConfigControl 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
																															MODE_VAL
RO																															RW
Reset: 0x00000000																															
Offset: 0x0080																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
MODE_VAL	Mode valid (autocleared)	Bit: 0	RW

3.2.1.10 RED Tsn Config Mode Register

If Promiscuous mode is enabled all traffic will be forwarded.

RED TsnConfigMode 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
-														CUT_THROUGH	PROMISCUOUS	-															
RO														RW	RW	RO															
Reset: 0x00000000																															
Offset: 0x0084																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:21	RO
CUT_THROUIGH	Enable cut through frame processing	Bit:20	RW
-	Reserved, read 0	Bit:19:17	RO
PROMISCUOUS	Promiscuous Mode	Bit:16	RW
-	Reserved, read 0	Bit:15:0	RO

3.2.2 RED Mac

3.2.2.1 RED Tsn Mac Control Register

Used for general control over the HSR&PRP Core, all configurations on the core shall only be done when disabled.

RED TsnMacControl 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
																															MAC_VAL
RO																															RW
Reset: 0x00000000																															
Offset: 0x0100																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
MAC_VAL	MAC address valid (selfclearing)	Bit: 0	RW

3.2.2.2 RED Tsn MAC 1 Register

MAC address of the node. LSB is transferred first on the network.
E.g. 0x01234567 => MAC: 67:45:32:01:XX:XX.

RED TsnMac1 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
MAC(3)								MAC(2)								MAC(1)								MAC(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0104																															

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

3.2.2.3 RED Tsn MAC 2 Register

MAC address of the node. LSB is transferred first on the network.

E.g. 0x0004567 => MAC: XX:XX:XX:67:45.

RED TsnMac2 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
-																MAC(5)								MAC(4)							
RO																RW								RW							
Reset: 0x00000000																															
Offset: 0x0108																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
MAC(5)	MAC Byte 5	Bit:15:8	RW
MAC(4)	MAC Byte 4	Bit:7:0	RW

3.2.3 RED Credit

These registers are only available when Credit support is enabled by generic.

3.2.3.1 RED Tsn Credit Control Register

Allows to control the credit shaper on a per priority queue level. Once all credit parameters are set the credit shaper can be enabled in general and then per priority

Red TsnCreditControl 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
-								PRIO_7	PRIO_6	PRIO_5	PRIO_4	PRIO_3	PRIO_2	PRIO_1	PRIO_0	-															CREDIT_ENABLE
RO								RW	RW	RW	RW	RW	RW	RW	RW	RO															RW
Reset: 0x00000000																															
Offset: 0x0200																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:24	RO
PRIO_X	Enable credit shaper for this priority queue (only up to the nr of queues)	Bit:23:16	RW

-	Reserved, read 0	Bit:15:1	RO
CREDIT_ENABLE	Enable credit shaper	Bit:0	RW

3.2.3.2 RED Tsn Credit Registers

Credit shaper increment and decrement on a per priority queue base.
Whenever a frame wants to send but has to wait the credit count is incremented and decremented when a frame is in the sending phase

Priority Queue 0 => 0x0204, Priority Queue 1 => 0x0208, etc.

Red TsnCredit[X] 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
CREDIT_DEC																CREDIT_INC															
RW																RW															
Reset: 0x00000000																															
Offset: 0x0204 - 0x0220																															

Name	Description	Bits	Access
CREDIT_DEC	Decrement counted when the frame is in sending phase	Bit:31:16	RW
CREDIT_INC	Increment counted when the frame has to wait	Bit:15:0	RW

3.2.3.3 RED Tsn Credit Minimum Registers

Credit shaper minimum, which the counter will ever be decremented to.

Priority Queue 0 => 0x0224, Priority Queue 1 => 0x0228, etc.

Red TsnCreditMin[X] 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
CREDIT_MIN																															
RW																															
Reset: -10^9																															
Offset: 0x0224 - 0x0240																															

Name	Description	Bits	Access
CREDIT_MIN	Credit Minimum	Bit:31:0	RW

3.2.3.4 RED Tsn Credit Maximum Registers

Credit shaper maximum, which the counter will ever be incremented to.

Priority Queue 0 => 0x0244, Priority Queue 1 => 0x0248, etc.

Red TsnCreditMax[X] 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
CREDIT_MAX																															
RW																															
Reset: 10^9																															
Offset: 0x0244 - 0x0260																															

Name	Description	Bits	Access
CREDIT_MAX	Credit Maximum	Bit:31:0	RW

3.2.4 RED Max Data Size

These registers are only available when Max Data Size filtering support is enabled by generic.

3.2.4.1 RED Tsn Max Data Size Control Register

Allows to control the max data size filter on a per priority queue level. Once all filter parameters are set the max data size filter can be enabled in general and then per priority

Red TsnMaxDataSizeControl 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
-								PRIO_7	PRIO_6	PRIO_5	PRIO_4	PRIO_3	PRIO_2	PRIO_1	PRIO_0	-															MAX_SIZE_ENABLE
RO								RW	RW	RW	RW	RW	RW	RW	RW	RO															RW

Reset: 0x00000000

Offset: 0x0280

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:24	RO

PRIO_X	Enable max size filter for this priority queue (only up to the nr of queues)	Bit:23:16	RW
-	Reserved, read 0	Bit:15:1	RO
MAX_SIZE_ENABLE	Enable max size filter	Bit:0	RW

3.2.4.2 RED Tsn Max Data Size Registers

Max data size on a per priority queue base.

Priority Queue 0 => 0x0284, Priority Queue 1 => 0x0288, etc.

Red TsnMaxDataSize[X] 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_SIZE											
RO																				RW											
Reset: 0x00000000																															
Offset: 0x0284 - 0x02A0																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:12	RO
MAX_SIZE	Max size of a frame for this priority queue	Bit:11:0	RW

3.2.5 RED Preemption

These registers are only available when Preemption support is enabled by generic.

3.2.5.1 RED Tsn Preemption Control Register

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

RED TsnConfigControl 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
																														HOLD_RELEASE	VERIFY_ENABLE
RO																														RW	RW
Reset: 0x00000000																															
Offset: 0x02B0																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:2	RO
HOLD_RELEASE	Use Hold Release Mechanism (802.1Qbu) or only Preemption (802.3br)	Bit:1	RW
VERIFY_ENABLE	Enable preemption verification	Bit:0	RW

3.2.5.2RED Tsn Preemption Verify Interval Register

Set the Verify Interval in Milliseconds (1-128ms, default 10), which happens when a Link Change happens or at Startup

Red TsnPreemptionVerifyInterval 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
																								INTERVAL							
RO																								RW							
Reset: 0x0000000A																															
Offset: 0x02B4																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
INTERVAL	Verification interval between the 3 attemps in milliseconds: Range 1-128 (default 10)	Bit:7:0	RW

3.2.6 RED Prio and Phase

3.2.6.1 RED Tsn Phase Control Register

Allows to enable the individual priority queues and phases as well as preemption, cyclic forwarding, phase shaping and priority queuing in general. Once a new phase configuration is done, the phase valid bit shall be set to apply the config.

Red TsnPhaseControl 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
-								PRIO_7	PRIO_6	PRIO_5	PRIO_4	PRIO_3	PRIO_2	PRIO_1	PRIO_0	-				PHASE_IN_LOW	-		PHASE_VAL	PHASE_IN	-			PREEMPT_ENABLE	CYCLE_ENABLE	PHASE_ENABLE	PRIO_ENABLE
RO								RW	RW	RW	RW	RW	RW	RW	RW	RO				RW	RO		RW	RW	RO			RW	RW	RW	RW
Reset: 0x00000000																															
Offset: 0x0300																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:24	RO
PRIO_[X]	Enable this priority queue 0 = highest prio, 7 lowest prio (only up to the nr of queues), lowest is always valid	Bit:23:16	RW

-	Reserved, read 0	Bit:15:12	RO
PHASE_IN_LOW	Phase enable handling on reception for lowest prio queue only	Bit:11	RW
-	Reserved, read 0	Bit:10:9	RO
PHASE_VAL	Phase configuration valid (autocleared)	Bit:8	RW
PHASE_IN	Phase enable handling on reception for all queues	Bit:7	RW
-	Reserved, read 0	Bit:7:4	RO
PREEMPT_ENABLE	Enable preemption	Bit:3	RW
CYCLE_ENABLE	Enable cyclic forwarding	Bit:2	RW
PHASE_ENABLE	Enable phase generation	Bit:1	RW
PRIO_ENABLE	Enable priority queuing	Bit:0	RW

3.2.6.2RED Tsn Priority Queue VLAN Registers

Defines the VLAN priority associated with the priority queue. For the lowest priority queue (7 or set by generic) no VLAN association exists, since all other frames will go there.

Red TsnPrioVlan[X] 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
																													PRIO_VLAN		
RO																													RW		
Reset: 0x00000000																															
Offset: 0x0310 - 0x0338																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:3	RO
PRIO_VLAN	VLAN associated with this priority queue	Bit:2:0	RW

3.2.6.3RED Tsn Phase Scheduler Simple Period Register

Defines the cycle period in nanoseconds for the simple scheduler. Shall not exceed 1 millisecond.

Red TsnPhaseSplPeriod 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
PERIOD																															
RW																															
Reset: 0x00000000																															
Offset: 0x0340																															

Name	Description	Bits	Access
PERIOD	Cycle Period in Nanoseconds	Bit:31:0	RW

3.2.6.4 RED Tsn Phase Scheduler Simple Start Registers

Defines the start time in nanoseconds in relation to a cycle start on a per priority queue base. Shall not exceed the cycle period and not 1 ms and shall be smaller than the stop time.

Red TsnPhaseSplStart[X] 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
PHASE_START																															
RW																															
Reset: 0x00000000																															
Offset: 0x0380 - 0x039C																															

Name	Description	Bits	Access
PHASE_START	Start of phase as offset to start of cycle in Nanoseconds	Bit:31:0	RW

3.2.6.5RED Tsn Phase Scheduler Simple Stop Registers

Defines the stop time in nanoseconds in relation to a cycle start on a per priority queue base. Shall not exceed the cycle period and not 1 ms and shall be larger than the start time.

Red TsnPhaseSplStop[X] 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
PHASE_STOP																															
RW																															
Reset: 0x00000000																															
Offset: 0x03A0 - 0x03BC																															

Name	Description	Bits	Access
PHASE_STOP	Stop of phase as offset to start of cycle in Nanoseconds	Bit:31:0	RW

3.2.6.6 RED Tsn Phase Scheduler Advanced Cycle Time Register

Defines the cycle period in nanoseconds for the simple scheduler. Shall not exceed 1 millisecond.

Red TsnPhaseAdvCycleTime 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
CYCLE_TIME																															
RW																															
Reset: 0x00000000																															
Offset: 0x0340																															

Name	Description	Bits	Access
CYCLE_TIME	Cycle Period in Nanoseconds	Bit:31:0	RW

3.2.6.7RED Tsn Phase Scheduler Advanced Cycle Time Extension Register

Defines the cycle period extension in nanoseconds for the simple scheduler. Shall not exceed 1 millisecond.

Red TsnPhaseAdvCycleTimeExt 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
CYCLE_TIME_EXT																															
RW																															
Reset: 0x00000000																															
Offset: 0x0344																															

Name	Description	Bits	Access
CYCLE_TIME_EXT	Cycle Period Extension in Nanoseconds	Bit:31:0	RW

3.2.6.8 RED Tsn Phase Scheduler Advanced Base Time Low Register

Defines the time where the cycle is aligned to, nanoseconds part. The alignment according to 802.1Qbv is done in the core.

Red TsnPhaseAdvBaseTimeL 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
BASE_TIME_NS																															
RW																															
Reset: 0x00000000																															
Offset: 0x0348																															

Name	Description	Bits	Access
BASE_TIME_NS	Base Time Nanosecond part	Bit:31:0	RW

3.2.6.9 RED Tsn Phase Scheduler Advanced Base Time High Register

Defines the time where the cycle is aligned to, seconds part. The alignment according to 802.1Qbv is done in the core.

Red TsnPhaseAdvBaseTimeH 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
BASE_TIME_S																															
RW																															
Reset: 0x00000000																															
Offset: 0x034C																															

Name	Description	Bits	Access
BASE_TIME_S	Base Time Second part	Bit:31:0	RW

3.2.6.10 RED Tsn Phase Scheduler Advanced Gate States Register

Initial gate state when the scheduler has not started. Each gate maps a priority queue: Gate 0 => Priority Queue 0

Red TsnPhaseAdvGateStates 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																								GATE_7	GATE_6	GATE_5	GATE_4	GATE_3	GATE_2	GATE_1	GATE_0
RO																								RW	RW	RW	RW	RW	RW	RW	RW
Reset: 0x00000000																															
Offset: 0x0350																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
GATE_[X]	Initial Gate states (only up to the nr of queues)	Bit:7:0	RW

3.2.6.11RED Tsn Phase Scheduler Advanced Control List Length Register

How many entries are in the control list defining the schedule. Max 32 entries (could be changed in the package).

Red TsnPhaseAdvCtrlListLength 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
																								CTRL_LIST_LENGTH							
RO																								RW							
Reset: 0x00000000																															
Offset: 0x0370																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
CTRL_LIST_LENGTH	How many entries will be valid in the control list	Bit:7:0	RW

3.2.6.12 RED Tsn Phase Scheduler Advanced Gate Control List Entry Register

A control list entry defines the state of the gates and duration of this configuration in nanoseconds. The 802.1Qbv scheduler goes automatically true the whole list.

Entry 0 => 0x0380, Entry 1 => 0x0384, etc.

Red TsnPhaseAdvCtrlListEntry[X] 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
GATE_7	GATE_6	GATE_5	GATE_4	GATE_3	GATE_2	GATE_1	GATE_0	TIME_INTERVAL		TIME_INTERVAL																					
RW	RW	RW	RW	RW	RW	RW	RW	RW		RW																					
Reset: 0x00000000																															
Offset: 0x0380 - 0x03FC																															

Name	Description	Bits	Access
GATE_[X]	Gate states (only up to the nr of queues)	Bit:31:24	RW
GATE_OP	Gate Operation: 0 => Set, 1 => unused, 2 => Set and Hold, 3 => Set and Release	Bit:23:22	RW
TIME_INTERVAL	Duration of Gate state in Nanoseconds	Bit:21:0	RW

3.2.7 PTP General OC

3.2.7.1 PTP OC Control Register

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

Ptp OcControl 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: 0x1000																															

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

3.2.7.2 PTP OC Status Register

Shows the current status of the PTP Hybrid Clock.

Ptp OcStatus 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: 0x1004																															

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

3.2.7.1 PTP OC 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 OcVersion 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: 0x100C																															

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

3.2.7.2 PTP OC Config Control Register

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

Ptp OcConfigControl 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_VAL	VLAN_VAL	PROFILE_VAL
RO																													RW	RW	RW
Reset: 0x00000000																															
Offset: 0x1080																															

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

3.2.7.3 PTP OC 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 OcConfigProfile 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_MECHANISM	-							LAYER	-						SIGNALING	TWO_STEP	-							PROFILE	
RO							RW	RO							RW	RO						RW	RW	RO						RW	
Reset: 0x00000000																															
Offset: 0x0084																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:25	RO
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:15:10	RO

SIGNALING	Signaling	Bit:9	RW
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

3.2.7.4 PTP OC Config Vlan 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.

Ptp OcConfigVlan 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
-															VLAN_EN	VLAN															
RO															RW	RW															
Reset: 0x00000000																															
Offset: 0x1088																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:17	RO
VLAN_EN	VLAN enable (0 disabled, 1 enabled)	Bit: 16	RW
VLAN	VLAN	Bit: 15:0	RW

3.2.7.5 PTP OC Config IP Register

IP address of the node. Used as source IP if in Default Profile and Layer 3 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(3)								IP(2)								IP(1)								IP(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x108C																															

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

3.2.8 PTP Default Dataset OC

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

3.2.8.1 PTP OC 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 OcDefaultDsControl 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	-																							GM_TIME_INAC_VAL	GM_ID_VAL	PRIORITY2_VAL	PRIORITY1_VAL	CLOCK_QUALITY_VAL	DOMAIN_NR_VAL	CLOCK_ID_VAL
RO	RW	RO																							RW	RW	RW	RW	RW	RW	RW
Reset: 0x00000000																															
Offset: 0x1100																															

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 29:7	RO

GM_TIME_INAC_VAL	Grandmaster Time Inacuracy valid (autocleared)	Bit: 6	RW
GM_ID_VAL	Grandmaster Short Identity valid (autocleared)	Bit: 5	RW
PRIORITY2_VAL	Priority2 valid (autocleared)	Bit: 4	RW
PRIORITY1_VAL	Priority1 valid (autocleared)	Bit: 3	RW
CLOCK_QUALITY_VAL	Clock Quality valid (autocleared)	Bit: 2	RW
DOMAIN_NR_VAL	Domain Number valid (autocleared)	Bit: 1	RW
CLOCK_ID_VAL	Clock Identity valid (autocleared)	Bit: 0	RW

3.2.8.2PTP OC 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 OcDefaultDs1 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: 0x1104																															

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

3.2.8.3PTP OC 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 OcDefaultDs2 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: 0x1108																															

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

3.2.8.4 PTP OC Default Dataset 3 Register

Domain to run on, only one is supported and Priorities.

Ptp OcDefaultDs3 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
PRIORITY1								PRIORITY2								-								DOMAIN_NR							
RW								RW								RO								RW							
Reset: 0x00000000																															
Offset: 0x110C																															

Name	Description	Bits	Access
PRIORITY1	Priority 1	Bit:31:24	RW
PRIORITY2	Priority 2	Bit:23:16	RW
-	Reserved, read 0	Bit:31:8	RO
DOMAIN_NR	Domain Number	Bit: 7:0	RW

3.2.8.5PTP OC Default Dataset 4 Register

PTP Clock Parameters, these are set to the corresponding default parameters according to the used Profile.

Ptp OcDefaultDs4 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_CLASS								CLOCK_ACCURACY								CLOCK_VARIANCE															
RW								RW								RW															
Reset: 0x00000000																															
Offset: 0x1110																															

Name	Description	Bits	Access
CLOCK_CLASS	Clock Quality, Clock Class	Bit:31:24	RW
CLOCK_ACCURACY	Clock Quality, Clock Accuracy	Bit:23:16	RW
CLOCK_VARIANCE	Clock Quality, Clock Variance	Bit:15:0	RW

3.2.8.6 PTP OC Default Dataset 5Register

Grandmaster Short Identity of the Node. Only used for Power Profile, else ignored.

Ptp OcDefaultDs5 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
																GM_ID															
RO																RW															
Reset: 0x00000000																															
Offset: 0x1114																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
GM_ID	Grandmaster Short ID according to PowerProfile	Bit: 15:0	RW

3.2.8.7PTP OC Default Dataset 6 Register

Grandmaster Time Inaccuracy, known inaccuracy to the norm second. Only used for Power Profile, else ignored.

Ptp OcDefaultDs6 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
GM_TIME_INAC																													
RW																													
Reset: 0x00000000																													
Offset: 0x1118																													

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
GM_TIME_INAC	Grandmaster Time Inaccuracy according to PowerProfile	Bit: 31:0	RW

3.2.8.8PTP OC Default Dataset 7 Register

Number of ports, for an OC this is always 1.

Ptp OcDefaultDs7 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: 0x00000001																															
Offset: 0x111C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
NUMBER_OF_PORTS	Number of ports of the OC (always 1)	Bit: 15:0	RO

3.2.9 PTP Port Dataset OC

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

3.2.9.1 PTP OC Port Dataset Control Register

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

Ptp OcPortDsControl 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																												ASYMMETRY_VAL	MSG_INTERVAL_VAL	DELAY_MECHANISM VAL
RO	RW	RO																											RW	RW	RW
Reset: 0x00000000																															
Offset: 0x1200																															

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 29:3	RO
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

3.2.9.2PTP OC Port Dataset 1 Register

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

Ptp OcPortDs1 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: 0x1204																														

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

3.2.9.3PTP OC Port Dataset 2 Register

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

Ptp OcPortDs2 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: 0x1208																														

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

3.2.9.4 PTP OC Port Dataset 3 Register

Port State as defined in IEEE1588

Ptp OcPortDs3 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
																								PORT STATE							
RO																								RO							
Reset: 0x00000000																															
Offset: 0x120C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
PORT_STATE	Port state as defined in IEEE1588	Bit: 7:0	RO

3.2.9.5PTP OC Port Dataset 4 Register

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

Ptp OcPortDs4 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
-																DELAYREQ_INTERVAL								PDELAYREQ_INTERVAL							
RO																RW								RW							
Reset: 0x00000000																															
Offset: 0x1210																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
DELAYREQ_INTERVAL	Signed Delay Request Log interval	Bit: 15:8	RW
PDELAYREQ_INTERVAL	Signed Peer Delay Request Log interval	Bit: 7:0	RW

3.2.9.6 PTP OC Port Dataset 5 Register

Signed Announce interval and Announce timeout (only available when DynamicMessageRatesSupport_Gen is true)

Ptp OcPortDs5 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
-																ANNOUNCE_TIMEOUT								ANNOUNCE_INTERVAL							
RO																RW								RW							
Reset: 0x00000000																															
Offset: 0x1214																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
ANNOUNCE TIMEOUT	Announce timeout	Bit: 15:8	RW
ANNOUNCE_INTERVAL	Signed Announce Log interval	Bit: 7:0	RW

3.2.9.7PTP OC Port Dataset 6 Register

Signed Sync interval (only available when DynamicMessageRatesSupport_Gen is true)

Ptp OcPortDs6 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
																								SYNC_INTERVAL							
RO																								RW							
Reset: 0x00000000																															
Offset: 0x1218																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
SYNC_INTERVAL	Signed Sync Log interval	Bit: 7:0	RW

3.2.9.8 PTP OC Port Dataset 7 Register

Asymmetry as signed nanoseconds.

Ptp OcPortDs7 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: 0x121C																														

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

3.2.9.9 PTP OC Port Dataset 8 Register

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

Ptp OcPortDs7 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

3.2.10 PTP Current Dataset OC

Parameters from the PTP Current Dataset according to IEEE1588-2008 Clause 8.2.2.

3.2.10.1 PTP OC Current Dataset Control Register

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

Ptp OcCurrentDsControl 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																														
RO	RW	RO																													
Reset: 0x00000000																														
Offset: 0x1300																														

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

3.2.10.2 PTP OC Current Dataset 1 Register

Number of network hops (BCs) between the Master and the Slave. If OC is Master this value is 0.

Ptp OcCurrentDs1 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
																STEPS_REMOVED															
RO																RO															
Reset: 0x00000000																															
Offset: 0x1304																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
STEPS_REMOVED	How many steps were removed between master and slave	Bit: 15:0	RO

3.2.10.3 PTP OC Current Dataset 2 Register

Higher 32 bits of the current measured/calculated Offset to the Master. This is in scaled nanoseconds format.

Ptp OcCurrentDs2 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
OFFSET(63:32)																														
RO																														
Reset: 0x00000000																														
Offset: 0x1308																														

Name	Description	Bits	Access
OFFSET(63:32)	Offset from Master higher 32bits	Bit: 31:0	RO

3.2.10.4 PTP OC Current Dataset 3 Register

Lower 32 bits of the current measured/calculated Offset to the Master. This is in scaled nanoseconds format.

Ptp OcCurrentDs3 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
OFFSET(31:0)																													
RO																													
Reset: 0x00000000																													
Offset: 0x130C																													

Name	Description	Bits	Access
OFFSET(31:0)	Offset from Master lower 32bits	Bit: 31:0	RO

3.2.10.5 PTP OC Current Dataset 4 Register

Higher 32 bits of the current measured E2E Delay. This is in scaled nanoseconds format.

Ptp OcCurrentDs4 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(63:32)																														
RO																														
Reset: 0x00000000																														
Offset: 0x1310																														

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

3.2.10.6 PTP OC Current Dataset 5 Register

Lower 32 bits of the current measured E2E Delay. This is in scaled nanoseconds format.

Ptp OcCurrentDs5 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(31:0)																														
RO																														
Reset: 0x00000000																														
Offset: 0x1314																														

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

3.2.11PTP Parent Dataset OC

Parameters from the PTP Parent Dataset according to IEEE1588-2008 Clause 8.2.3.

3.2.11.1PTP OC Parent Dataset Control Register

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

Ptp OcParentDsControl 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																														
RO	RW	RO																													
Reset: 0x00000000																														
Offset: 0x1400																														

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

3.2.11.2PTP OC Parent Dataset 1 Register

First 4 bytes of the Clock Identity of the current Master. If the OC is Master this is the same as the CLOCK_ID from the Default Dataset.

Ptp OcParentDs1 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
PARENT_CLOCK_ID(3)								PARENT_CLOCK_ID(2)								PARENT_CLOCK_ID(1)								PARENT_CLOCK_ID(0)							
RO								RO								RO								RO							
Reset: 0x00000000																															
Offset: 0x1404																															

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

3.2.11.3PTP OC Parent Dataset 2 Register

Second 4 bytes of the Clock Identity of the current Master. If the OC is Master this is the same as the CLOCK_ID from the Default Dataset.

Ptp OcParentDs2 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
PARENT_CLOCK_ID(7)								PARENT_CLOCK_ID(6)								PARENT_CLOCK_ID(5)								PARENT_CLOCK_ID(4)							
RO								RO								RO								RO							
Reset: 0x00000000																															
Offset: 0x1408																															

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

3.2.11.4 PTP OC Parent Dataset 3 Register

Port Number part of the Clock Identity of the current Master. If the OC is master this is 0. Priorities of the GM.

Ptp OcParentDs3 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
GM_PRIORITY1								GM_PRIORITY2								PARENT_PORT_NR															
RO								RO								RO															
Reset: 0x00000000																															
Offset: 0x140C																															

Name	Description	Bits	Access
GM_PRIORITY1	Grandmaster Priority 1	Bit:31:24	RO
GM_PRIORITY2	Grandmaster Priority 2	Bit:23:16	RO
PORT_NR	Parent Port Number	Bit: 15:0	RO

3.2.11.5PTP OC Parent Dataset 4 Register

First 4 bytes of the Grandmaster Clock Identity of the current Grandmaster. If the OC is Master this is the same as the CLOCK_ID from the Default Dataset.

Ptp OcParentDs4 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
GM_CLOCK_ID(3)								GM_CLOCK_ID(2)								GM_CLOCK_ID(1)								GM_CLOCK_ID(0)							
RO								RO								RO								RO							
Reset: 0x00000000																															
Offset: 0x1410																															

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

3.2.11.6PTP OC Parent Dataset 5 Register

Second 4 bytes of the Grandmaster Clock Identity of the current Grandmaster. If the OC is Master this is the same as the CLOCK_ID from the Default Dataset.

Ptp OcParentDs5 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
GM_CLOCK_ID(7)								GM_CLOCK_ID(6)								GM_CLOCK_ID(5)								GM_CLOCK_ID(4)							
RO								RO								RO								RO							
Reset: 0x00000000																															
Offset: 0x1414																															

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

3.2.11.7 PTP OC Parent Dataset 6 Register

Clock Parameters of the current Grandmaster. If the OC is Master this is the same as the Clock Parameters from the Default Dataset.

Ptp OcParentDs6 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
GM_CLOCK_CLASS								GM_CLOCK_ACCURACY								GM_CLOCK_VARIANCE															
RO								RO								RO															
Reset: 0x00000000																															
Offset: 0x1418																															

Name	Description	Bits	Access
GM_CLOCK_CLASS	Grandmaster Clock Quality, Clock Class	Bit:31:24	RO
GM_CLOCK_ACCURACY	Grandmaster Clock Quality, Clock Accuracy	Bit:23:16	RO
GM_CLOCK_VARIANCE	Grandmaster Clock Quality, Clock Variance	Bit:15:0	RO

3.2.11.8PTP OC Parent Dataset 7 Register

Grandmaster Short Identity of the current Grandmaster. If the OC is Master this is the same as the GM_ID from the Default Dataset.

Ptp OcParentDs7 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
																GM_ID															
RO																RO															
Reset: 0x00000000																															
Offset: 0x141C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
GM_ID	Grandmaster Short ID according to PowerProfile	Bit: 15:0	RO

3.2.11.9PTP OC Parent Dataset 8 Register

Grandmaster Time Inaccuracy of the current Grandmaster. If the OC is Master this is the TIME_INAC from the Default Dataset.

Ptp OcParentDs8 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
GM_TIME_INAC																															
RO																															
Reset: 0x00000000																															
Offset: 0x1420																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
GM_TIME_INAC	Grandmaster Time Inacuracy according to PowerProfile	Bit: 31:0	RO

3.2.11.10 PTP OC Parent Dataset 9 Register

Accumulated received Network Time Inaccuracy between the Slave and the Grandmaster. If the OC is Master this is the 0.

Ptp OcParentDs9 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
NETWORK_TIME_INAC																															
RO																															
Reset: 0x00000000																															
Offset: 0x1424																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
NETWORK_TIME_INAC	Network Time Inacuracy according to PowerProfile	Bit: 31:0	RO

3.2.12 PTP Time Properties Dataset OC

Parameters from the PTP Time Properties Dataset according to IEEE1588-2008 Clause 8.2.4

3.2.12.1PTP OC Time Properties Dataset Control Register

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

Ptp OcTimePropertiesDsControl 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																		DISP_NAME_VAL	DISP_NAME_LENG_VAL	TIME_OF_NEXT_JMP_VAL	JUMP_SECOND_VAL	CURRENT_OFFSET_VAL	TIME_SOURCE_VAL	PTP_TIMESCALE_VAL	FREQ_TRACEABLE_VAL	TIME_TRACEABLE_VAL	LEAP61_VAL	LEAP59_VAL	UTC_OFFSET_VAL_VAL	UTC_OFFSET_VAL
RO	RW	RO																	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset: 0x00000000																															
Offset: 0x1500																															

Name	Description	Bits	Access
READ_DONE	Time Properties Dataset was read	Bit: 31	RO
READ	Read Time Properties Dataset (autocleared)	Bit: 30	RW
-	Reserved, read 0	Bit 29:13	RO

DISP_NAME_VAL	Display Name Valid (autocleared) (power profile)	Bit:12	RW
DISP_NAME_LENG_VAL	Diplay Name Length Valid (autocleared) (power profile)	Bit:11	RW
TIME_OF_NEXT_JMP_VAL	Time of Next Jump Valid (autocleared) (power profile)	Bit:10	RW
JUMP_SECOND_VAL	Jump in Seconds Valid (autocleared) (power profile)	Bit:9	RW
CURRENT_OFFSET_VAL	Current Offset Valid (autocleared) (power profile)	Bit:8	RW
TIME_SOURCE_VAL	Time Source Valid (autocleared)	Bit:7	RW
PTP_TIMESCALE_VAL	PTP Timescale Valid (autocleared)	Bit:6	RW
FREQ_TRACEABLE_VAL	Frequency Traceable Valid (autocleared)	Bit:5	RW
TIME_TRACEABLE_VAL	Time Traceable Valid (autocleared)	Bit:4	RW
LEAP61_VAL	Leap Second 61 Valid (autocleared)	Bit:3	RW
LEAP59_VAL	Leap Second 59 Valid (autocleared)	Bit:2	RW
UTC_OFFSET_VAL_VAL	UTC Offset Valid Valid (autocleared)	Bit:1	RW
UTC_OFFSET_VAL	UTC Offset Valid (autocleared)	Bit:0	RW

3.2.12.2 PTP OC Time Properties Dataset 1 Register

Quality bits of the current Grandmaster. If the OC is Master and external synchronization is disabled, the values are set to the unknown default values according to IEEE1588-2008 Clause 9.4 else to the values provided by the external synchronization source.

Ptp OcTimePropertiesDs1 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
UTC_OFFSET																-		UTC_OFFSET_VAL	LEAP59	LEAP61	TIME_TRACEABLE	FREQ_TRACEABLE	PTP_TIMESCALE	TIME_SOURCE							
RW																RO		RW	RW	RW	RW	RW	RW	RW							
Reset: 0x00000000																															
Offset: 0x1504																															

Name	Description	Bits	Access
UTC_OFFSET	Current UTC offset in seconds	Bit:31:16	RO
-	Reserved, read 0	Bit:15:14	RO
UTC_OFFSET_VAL	Current UTC offset is valid	Bit: 13	RO
LEAP59	Leap second 59 and next midnight change	Bit: 12	RO
LEAP61	Leap second 61 and next midnight change	Bit: 11	RO
TIME_TRACEABLE	Time is also phase traceable	Bit: 10	RO
FREQ_TRACEABLE	Time is Frequency traceable	Bit: 9	RO

PTP_TIMESCALE	Time is in PTP Timescale	Bit: 8	RO
TIME_SOURCE	Time Source	Bit: 7:0	RO

3.2.12.3 PTP OC Time Properties Dataset 2 Register

Alternate Timescale Offset of the current Grandmaster. Only used for Power Profile. If the OC gets Master this is reset to 0, it can be changed afterwards. This is not directly used by the PTP core. Software has to calculate its Alternate Timescale from the current time.

Ptp OcTimePropertiesDs2 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
CURRENT_OFFSET																														
RW																														
Reset: 0x00000000																														
Offset: 0x1508																														

Name	Description	Bits	Access
CURRENT_OFFSET	Current Offset in Seconds	Bit: 31:0	RO

3.2.12.4 PTP OC Time Properties Dataset 3 Register

Alternate Timescale Increment/Decrement of the next Jump in Seconds. Only used for Power Profile. If the OC gets Master this is reset to 0, it can be changed afterwards. This is not directly used by the PTP core. Software has to calculate its Alternate Timescale from the current time.

Ptp OcTimePropertiesDs3 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
JUMP_SECONDS																														
RW																														
Reset: 0x00000000																														
Offset: 0x150C																														

Name	Description	Bits	Access
JUMP_SECONDS	Jump in Seconds	Bit: 31:0	RW

3.2.12.5 PTP OC Time Properties Dataset 4 Register

Higher 2 bytes of the Alternate Timescale Time of the next Jump in Seconds. Only used for Power Profile. If the OC gets Master this is reset to 0, it can be changed afterwards. This is not directly used by the PTP core. Software has to calculate its Alternate Time-scale from the current time.

Ptp OcTimePropertiesDs4 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
																TIME_OF_NEXT_JUMP(47:32)															
RO																RW															
Reset: 0x00000000																															
Offset: 0x1514																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:1	RO
TIME_OF_NEXT_JUMP(47:32)	Time of next Jump higher Second part	Bit: 31:0	RW

3.2.12.6 PTP OC Time Properties Dataset 5 Register

Lower 4 bytes of the Alternate Timescale Time of the next Jump in Seconds. Only used for Power Profile. If the OC gets Master this is reset to 0, it can be changed afterwards. This is not directly used by the PTP core. Software has to calculate its Alternate Time-scale from the current time.

Ptp OcTimePropertiesDs5 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
TIME_OF_NEXT_JUMP(31:0)																														
RW																														
Reset: 0x00000000																														
Offset: 0x1510																														

Name	Description	Bits	Access
TIME_OF_NEXT_JUMP(31:0)	Time of next Jump lower Second part	Bit: 31:0	RW

3.2.12.7 PTP OC Time Properties Dataset 6 Register

Display Name Length in bytes,without \0 termination of the Alternate Timescale. Only used for Power Profile. If the OC gets Master this is reset to 3 (“PTP”), it can be changed afterwards.

Ptp OcTimePropertiesDs6 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
																								DISPLAY_NAME_LENG							
RO																								RW							
Reset: 0x00000000																															
Offset: 0x1518																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
DISPLAY_NAME_LENG	Diplay Name Lenght (max 12)	Bit: 7:0	RO

3.2.12.8 PTP OC Time Properties Dataset 7 Register

First 4 characters of Display Name in ASCII text of the Alternate Timescale. Only used for Power Profile. If the OC gets Master this is reset to “PTP”, it can be changed afterwards.

Ptp OcTimePropertiesDs7 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
DISPLAY_NAME(3)								DISPLAY_NAME(2)								DISPLAY_NAME(1)								DISPLAY_NAME(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x151C																															

Name	Description	Bits	Access
DISPLAY_NAME(3:0)	Display Name Byte 0 to 3	Bit: 31:0	RW

3.2.12.9 PTP OC Time Properties Dataset 8 Register

Second 4 characters of Display Name in ASCII text of the Alternate Timescale. Only used for Power Profile. If the OC gets Master this is reset to \0, it can be changed afterwards.

Ptp OcTimePropertiesDs8 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
DISPLAY_NAME(7)								DISPLAY_NAME(6)								DISPLAY_NAME(5)								DISPLAY_NAME(4)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x1520																															

Name	Description	Bits	Access
DISPLAY_NAME(3:0)	Display Name Byte 4 to 7	Bit: 31:0	RW

3.2.12.10 PTP OC Time Properties Dataset 9 Register

Third 4 characters of Display Name in ASCII text of the Alternate Timescale. Only used for Power Profile. If the OC gets Master this is reset to \0, it can be changed afterwards.

Ptp OcTimePropertiesDs9 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
DISPLAY_NAME(11)								DISPLAY_NAME(10)								DISPLAY_NAME(9)								DISPLAY_NAME(8)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x1524																															

Name	Description	Bits	Access
DISPLAY_NAME(3:0)	Display Name Byte 8 to 11	Bit: 31:0	RW

3.2.13 PTP General TC

3.2.13.1PTP 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: 0x1800																															

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

3.2.13.2 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: 0x1804																															

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

3.2.13.3 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: 0x180C																															

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

3.2.13.4 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
																													IP_VAL	VLAN_VAL	PROFILE_VAL
RO																													RW	RW	RW
Reset: 0x00000000																															
Offset: 0x1880																															

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

3.2.13.5 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_MECHANISM	-							LAYER	-							TWO_STEP	-						PROFILE	
RO							RW	RO							RW	RO							RW	RO						RW	
Reset: 0x00000000																															
Offset: 0x1884																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:25	RO
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

3.2.13.6 PTP TC Config Vlan 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.

Ptp TcConfigVlan 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
-															VLAN_EN	VLAN															
RO															RW	RW															
Reset: 0x00000000																															
Offset: 0x1888																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:17	RO
VLAN_EN	VLAN enable (0 disabled, 1 enabled)	Bit: 16	RW
VLAN	VLAN	Bit: 15:0	RW

3.2.13.7 PTP TC Config IP Register

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

Ptp TcConfigIp 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: 0x188C																															

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

3.2.14 PTP Default Dataset TC

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

3.2.14.1 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: 0x1900																															

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

3.2.14.2 PTP 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: 0x1904																															

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

3.2.14.3 PTP 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: 0x1908																															

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

3.2.14.4 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: 0x190C																															

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

3.2.14.5 PTP 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: 0x191C																															

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

3.2.15 PTP Port Dataset TC

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

3.2.15.1PTP 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							-													ASYMMETRY_VAL	MSG_INTERVAL_VAL	DELAY_MECHANISM VAL
RO	RW	RO							RW							RO													RW	RW	RW
Reset: 0x00000000																															
Offset: 0x1A00																															

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

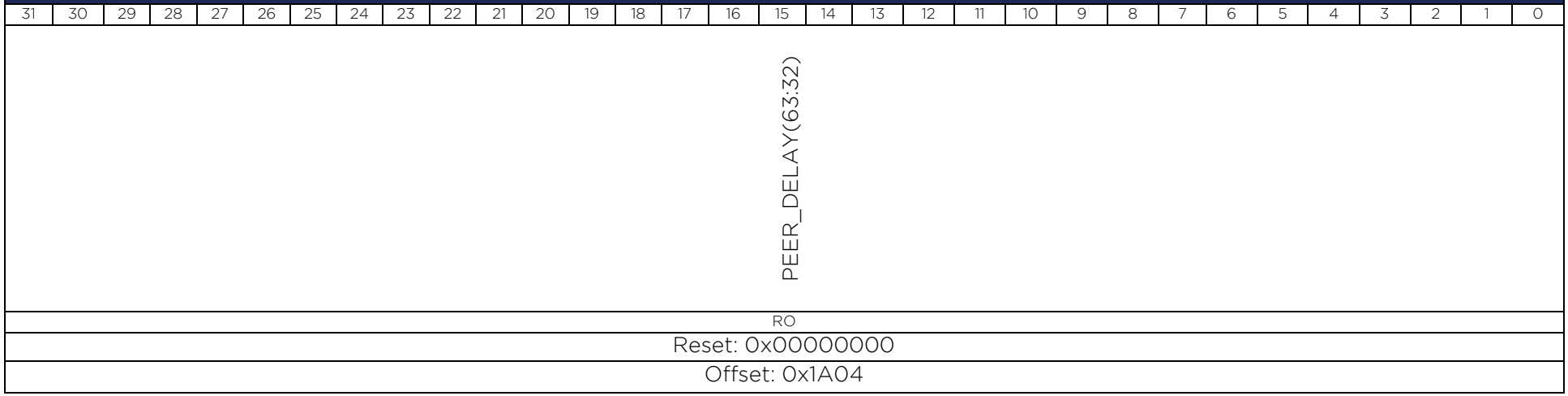
-	Reserved, read 0	Bit 15:3	RO
ASYMMETRY_VAL	Asymmetry valid	Bit 2	RW
MSG_INTERVAL_VAL	Message Intervals valid	Bit 1	RW
DELAY_MECHANISM_VAL	Delay Mechanism valid	Bit 0	RW

3.2.15.2 PTP 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



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

3.2.15.3 PTP 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: 0x1A08																														

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

3.2.15.4 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: 0x1A10																															

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

3.2.15.5 PTP 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: 0x1A1C																														

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

3.2.15.6 PTP 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 OcPortDs7 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: 0x1A20																														

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

4 Design Description

The following chapters describe the internals of the RED Tsn Core: starting with the Top Level, which is a collection of subcores, followed by the description of all subcores. The core is an extension to the HSR&PRP redundancy core which integrates the PTP Hybrid Clock time synchronization core. The internals of the PTP HC core are only briefly described where they differ.
For more details on the PTP Hybrid Clock core check the documentation:
www.nettimelogic.com/resources/Ptp_HybridClock_ReferenceManual.pdf

4.1 Top Level – RED Tsn

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
CutThrough_Gen	boolean	1	Support for Cut Through frame handling
PrioritySupport_Gen	boolean	1	Support for priority queues
NrOfPriorities_Gen	natural	1	How many priorities queues shall be supported (3-8)
PriorityQueue-Load_Gen	Common_ Natural_Type	Red_MaxPriori-ties_Con	These numbers in percent (0-100) together with the parameters MaxCyleDuration-Nanoseconds_Gen and Link-SpeedSupport_Gen define the size of the required buffers. It defines in percent how much of the traffic is expected to be traffic. This basically defines how many frames must be storable within a cycle for this.
PhaseSupport_Gen	boolean	1	Support for phases per priority and scheduler
PhasePortSup-port_Gen	boolean	1	If a scheduler per port shall be used

PhaseInSupport_Gen	boolean	1	Support for phases per priority and scheduler also on input queues (use only in spezial cases)
CreditSupport_Gen	boolean	1	Support for the credit based shaper
CycleSupport_Gen	boolean	1	Support for cyclic forwarding
Preemption Support_Gen	boolean	1	Support for preemption on the lowest priority
PreemptionVerify Support_Gen	boolean	1	If the Preemption Verify mechanism schall be used
PreemptionHold Release_Gen	boolean	1	If the Hold Release mechanism for Preemption shall be used
MaxCyleDuration-Nanoseconds_Gen	natural	1	Maximum configurable cycle time in nanoseconds
MaxDataSizeSupport_Gen	boolean	1	Support for the max data size filters per priority
PortStatus Support_Gen	boolean	1	If frame and error counters shall be available in registers
SimpleScheduler_Gen	boolean	1	If the simple scheduler shall be used for phases
ExtAxisIn Support_Gen	boolean	1	Support for external input AXI stream interfaces
ExtAxisOut Support_Gen	boolean	1	Support for external output AXI stream interfaces
ExtPortSupport_Gen	boolean	1	If the uplink also has an (R)(G)MII interface
UntaggerSupport_Gen	boolean	1	If the core shall be able to handle Redundncy tags
DstMacCheck_Gen	boolean	1	If only frames matching the configured MAC (plus multi and broadcast) will be forwarded
LinkSpeed Support_Gen	natural	1	Shall be either 100 or 1000. For 1000 duplication is parallelized to achieve the required throughput.

ClockClkPeriod Nanosecond_Gen	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
HighResSupport_Gen	boolean	1	If a high-resolution clock SysClkNx with alignment to SysClk is used
HighResFreq Multiply_Gen	natural	1	Multiplication factor of the high-resolution clock compared to SysClk
PtpPortSupport_Gen	boolean	1	If PTP Hybrid Clock shall be instantiated
PtpDefaultProfile Support_Gen	boolean	1	If PTP shall support the PTP default profile
PtpUtilityProfile Support_Gen	boolean	1	If PTP shall support the PTP utility profile
PtpSlaveOnly_Gen	boolean	1	If set to true, the Core can only be Slave
PtpMasterOnly_Gen	boolean	1	If set to true, the Core can only be Master
StaticConfig_Gen	boolean	1	If Static Configuration or AXI is used
RxDelayNano second10_Gen	integer	1	Input delay (10Mbit)
RxDelayNano second100_Gen	integer	1	Input delay (100Mbit)
RxDelayNano second1000_Gen	integer	1	Input delay (1000Mbit)
TxDelayNano second10_Gen	integer	1	Output delay (10Mbit)
TxDelayNano second100_Gen	integer	1	Output delay (100Mbit)
TxDelayNano second1000_Gen	integer	1	Output delay (1000Mbit)
IoFf_Gen	boolean	1	If Port A shall contain a IO Flip Flop.
AxiAddressRange Low_Gen	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	std_logic_vector	32	AXI Base Address plus Registerset Size

			Default plus 0xFFFF
Sim_Gen	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis

Table 5: Parameters

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.
Valid	std_logic	1	Whether the Adjustment is valid or not

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 Layer3_E
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
Ip	Common_Byte_Type	4	The source IP to be used if in Layer3 mode, 4 bytes, index 0 = MSB
DefaultDataset_ClockIdentity	Ptp_ClockIdentity_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 Tod_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
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_OrdinaryClockStaticConfig_Type

Defined in Ptp_OrdinaryClockAddrPackage.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

			Layer3_E
TwoStep	std_logic	1	If TwoStep shall be used
Signaling	std_logic	1	If Signaling 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
Ip	Com-mon_Byte_Type	4	The source IP to be used if in Layer3 mode, 4 bytes, 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
DefaultDataset_ ClockQuality	Ptp_ClockQuality _Type;		ClockClass, ClockAccuracy and OffsetScaledLogVariance of the clock
DefaultDataset_ Priority1	std_logic_vector	8	Priority 1 of the clock
DefaultDataset_ Priority2	std_logic_vector	8	Priority 2 of the clock
DefaultDataset_ GrandmasterId	std_logic_vector	16	Grandmaster ID for the Power Profile to be announced
DefaultDataset_ Grandmaster TimeInaccuracy	std_logic_vector	32	Grandmaster Inaccuracy to be Announce
TimeProperties Dataset_ CurrentUtcOffset	std_logic_vector	16	UTC offset from TAI to UTC
TimeProperties Dataset_CurrentUtc OffsetValid	std_logic	1	If the UTC offset is valid
TimeProperties Dataset_Leap59	std_logic	1	If a leap second 59 shall be announced
TimeProperties Dataset_Leap61	std_logic	1	If a leap second 61 shall be announced
TimeProperties Dataset_	std_logic	1	If the time is traceable to a primary source

TimeTraceable			
TimeProperties Dataset_ FrequencyTraceable	std_logic	1	If the frequency is traceable to a primary source
TimeProperties Dataset_PtpTimescale	std_logic	1	If the clock runs in PTP time-scale (TAI)
TimePropertiesDa-taset_TimeSource	std_logic_vector	8	What the clock source for the time to distribute is
TimeProperties Dataset_CurrentOffset	std_logic_vector	32	The current Offset of the PTP time in an alternative timescale in Power Profile
TimePropertiesDa-taset_JumpSeconds	std_logic_vector	32	The offset in seconds to jump when announce in power Profile (not used internally, only for distribution)
TimeProperties Dataset_Time OfNextJumpSeconds	std_logic_vector	48	When the next timejump will happen
TimeProperties Dataset_ DisplayNameLength	std_logic_vector	8	Display name length shall be 3
TimeProperties Dataset_DisplayName	Com-mon_Byte_Type	12	Display name shall be “PTP” in hex

Table 10: Ptp_OrdinaryClockStaticConfig_Type

4.1.1.2.6 Ptp_OrdinaryClockStaticConfigVal_Type

Defined in Tod_OrdinaryClockAddrPackage.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 Ordinary Clock
Profile_Val	std_logic	1	If the Profile shall be set
Vlan_Val	std_logic	1	If the VLAN shall be set
Ip_Val	std_logic	1	If the IP shall be set
DefaultDataset_	std_logic	1	If the ClockIdentity shall be set

ClockIdentity_Val			
DefaultDataset_ DomainNumber_Val	std_logic	1	If the DomainNumber shall be set
DefaultDataset_ ClockQuality_Val	std_logic	1	If the ClockQuality shall be set
DefaultDataset_ Priority1_Val	std_logic	1	If the Priority1 shall be set
DefaultDataset_ Priority2_Val	std_logic	1	If the Priority2 shall be set
DefaultDataset_ GrandmasterId_Val	std_logic	1	If the GrandmasterId shall be set
DefaultDataset_ Grandmaster TimeInaccuracy_Val	std_logic	1	If the GrandmasterTimeInaccu-racy shall be set
TimeProperties Dataset_ CurrentUtcOffset_Val	std_logic	1	If the CurrentUtcOffset shall be set
TimeProperties Dataset_ CurrentUtcOffset Valid_Val	std_logic	1	If the CurrentUtcOffsetValid shall be set
TimeProperties Dataset_Leap59_Val	std_logic	1	If the Leap59 shall be set
TimeProperties Dataset_Leap61_Val	std_logic	1	If the Leap61 shall be set
TimeProperties Dataset_ TimeTraceable_Val	std_logic	1	If the TimeTraceable shall be set
TimePropertiesDa-ta-set_FrequencyTraceable_Val	std_logic	1	If the FrequencyTraceable shall be set
TimeProperties Dataset_ PtpTimescale_Val	std_logic	1	If the PtpTimescale shall be set
TimeProperties Dataset_ TimeSource_Val	std_logic	1	If the TimeSource shall be set

TimeProperties Dataset_ CurrentOffset_Val	std_logic	1	If the CurrentOffset shall be set
TimeProperties Dataset_ JumpSeconds_Val	std_logic	1	If the JumpSeconds shall be set
TimeProperties Dataset_Time OfNextJumpSeconds _Val	std_logic	1	If the TimeOfNextJumpSec-onds shall be set
TimeProperties Dataset_DisplayName Length_Val	std_logic	1	If the DisplayNameLength shall be set
TimeProperties Dataset_ DisplayName_Val	std_logic	1	If the DisplayName shall be set

Table 11: Ptp_OrdinaryClockStaticConfigVal_Type

4.1.1.2.7 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 12: Ptp_TransparentClockStaticConfig_Type

4.1.1.2.8 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 13: Ptp_TransparentClockStaticConfigVal_Type

4.1.1.2.9 Ptp_OrdinaryClockStaticStatus_Type

Defined in Ptp_OrdinaryClockAddrPackage.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_ Signaling	std_logic	1	If the Core uses Signaling
DefaultDataset_ TwoStepFlag	std_logic	1	If the Core runs in Twostep mode
DefaultDataset_ ClockIdentity	Ptp_Clock Identity_Type	1	Clockidentity of the Core
DefaultDataset_	std_logic_vector	16	Number of Ports (1 for OC)

NumberPorts			
DefaultDataset_ ClockQuality	Ptp_ClockQuality _Type	1	Clock Quality of the Core
DefaultDataset_ Priority1	std_logic_vector	8	Priority 1 of the Core
DefaultDataset_ Priority2	std_logic_vector	8	Priority 2 of the Core
DefaultDataset_ DomainNumber	std_logic_vector	8	Domain Number the Core runs on
DefaultDataset_ SlaveOnly	std_logic	1	If it is running in slave only mode
DefaultDataset_ GrandmasterId	std_logic_vector	16	Power Profile Grandmaster Id
DefaultDataset_ Grandmaster TimeInaccuracy	std_logic_vector	32	Power Profile Grandmaster Inaccuracy
TimeProperties Dataset_ CurrentUtcOffset	std_logic_vector	16	UTC offset from TAI to UTC
TimeProperties Dataset_CurrentUtc OffsetValid	std_logic	1	If the UTC offset is valid
TimeProperties Dataset_Leap59	std_logic	1	If a leap second 59 shall be announced
TimeProperties Dataset_Leap61	std_logic	1	If a leap second 61 shall be announced
TimeProperties Dataset_ TimeTraceable	std_logic	1	If the time is traceable to a primary source
TimeProperties Dataset_ FrequencyTraceable	std_logic	1	If the frequency is traceable to a primary source
TimeProperties Dataset_PtpTimescale	std_logic	1	If the clock runs in PTP time-scale (TAI)
TimePropertiesDa-taset_TimeSource	std_logic_vector	8	What the clock source for the time to distribute is
TimeProperties Dataset_CurrentOffset	std_logic_vector	32	The current Offset of the PTP time in an alternative timescale

			in Power Profile
TimePropertiesDa-taset_JumpSeconds	std_logic_vector	32	The offset in seconds to jump when announce in power Profile (not used internally, only for distribution)
TimeProperties Dataset_Time OfNextJumpSeconds	std_logic_vector	48	When the next timejump will happen
TimeProperties Dataset_ DisplayNameLength	std_logic_vector	8	Display name length shall be 3
TimeProperties Dataset_DisplayName	Common_ Byte_Type	12	Display name shall be “PTP” in hex
CurrentDataset_ StepsRemoved	std_logic_vector	16	Number of Steps between Grandmaster and Slave
CurrentDataset_ OffsetFromMaster	std_logic_vector	64	Offset from Master
CurrentDataset_ MeanPathDelay	std_logic_vector	64	E2E Delay
ParentDataset_ ParentPortIdentity	Ptp_Port Identity_Type;	1	Parent Port Indentity
ParentDataset_ ParentStats	std_logic	1	Parent Statistics (always 0)
ParentDataset_ ObsParentOffset ScaledLogVaria	std_logic_vector	16	Observed Parent Offser Vari-ance
ParentDataset_ ObsParentClock PhaseChangeRate	std_logic_vector	32	Parent Phase Change Rate
ParentDataset_ GrandmasterIdentity	Ptp_Clock Identity_Type	1	Granmaster Clock Identity
ParentDataset_ Grandmaster ClockQuality	Ptp_Clock Quality_Type	1	Grandmaster Clock Quality
ParentDataset_ GrandmasterPriority1	std_logic_vector	8	Grandmaster Priority 1
ParentDataset_ GrandmasterPriority2	std_logic_vector	8	Grandmaster Priority 2

ParentDataset_ GrandmasterId	std_logic_vector	16	Grandmaster Short ID
ParentDataset_ Grandmaster TimeInaccuracy	std_logic_vector	32	Granmaster Inaccuracy
ParentDataset_ Network TimeInaccuracy	std_logic_vector	32	Network Time Inacucuracy to Grandmaster
PortDataset_ PortIdentity	Ptp_Port Identity_Type	1	Port Idenity
PortDataset_ PortState	std_logic_vector	8	Port State
PortDataset_ LogMinDelayReq Interval	std_logic_vector	8	Delay Request Message Interval
PortDataset_ PeerMeanPathDelay Valid	std_logic	1	P2P Delay valid
PortDataset_ PeerMeanPathDelay	std_logic_vector	64	P2P Delay
PortDataset_ LogAnnounceInterval	std_logic_vector	8	Announce Message Interval
PortDataset_ AnnounceReceipt Timeout	std_logic_vector	8	Announce Receipt Timeout
PortDataset_ LogSyncInterval	std_logic_vector	8	Sync Message Interval
PortDataset_ LogPtpCapable SignalingInterval	std_logic_vector	8	802.1 Signaling Message Interval
PortDataset_ PtpCapableTimeout	std_logic_vector	8	Timeout for 802.1 PTP Capable signaling messages
PortDataset_ DelayMechanism	std_logic_vector	8	Delay Mechanism
PortDataset_ LogMinPdelayReq Interval	std_logic_vector	8	Peer Delay Request Message Interval
PortDataset_	std_logic_vector	8	Timeout for Peer Dealy An-

DelayTimeout			swers
PortDataset_ MaxPdelay	std_logic_vector	32	Maximum allowed Peer Delay in TSN mode
PortDataset_ VersionNumber	std_logic_vector	4	PTP Version
PortDataset_ Asymmetry	std_logic_vector	32	Port Asymmetry

Table 14: Ptp_OrdinaryClockStaticConfig_Type

4.1.1.2.10 Ptp_OrdinaryClockStaticStatusVal_Type

Defined in Ptp_OrdinaryClockAddrPackage.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 15: Ptp_OrdinaryClockStaticConfigVal_Type

4.1.1.2.11 Red_SimpleControlListEntry_Type

Defined in Red_Package.vhd of library RedLib
This is the type used for the phase generation when the simple scheduler is used.

Field Name	Type	Size	Description
Start	std_logic_vector	32	Offset in nanoseconds to cycle when the phase shall start
Stop	std_logic_vector	32	Offset in nanoseconds to cycle when the phase shall stop

Table 16: Red_SimpleControlListEntry_Type

4.1.1.2.12 Red_TsnStaticConfig_Type

Defined in Red_TsnAddrPackage.vhd of library RedLib
This is the type used for static configuration.

Field Name	Type	Size	Description
PrioVlan	Red_VlanPrios_Type	Red_MaxPriori-ties_Con-1	Which VLAN priority shall be used for each queue (index 0 = highest prio)
PrioEnable	std_logic	1	If the priority handling shall be enabled
PrioPhaseEnable	std_logic_vector	Red_MaxPriori-ties_Con	If the priority queue shall be enabled
Phase	Red_Simple ControlList_ Type	Red_MaxPriori-ties_Con	Definition of the phase per priority queue (simple sched-uler)
PhaseEnable	std_logic	1	If phase generation shall be enabled
PhaseCycleTime	std_logic_vector	32	Cyle time in nanoseconds
CreditInc	Red_Credit_Type	Red_MaxPriori-ties_Con	Credit shaper increment per priority queue
CreditDec	Red_Credit_Type	Red_MaxPriori-ties_Con	Credit shaper decrement per priority queue
CreditEnable	std_logic;	1	If credit shaping shall be enabled
CreditPhaseEnable	std_logic_vector	Red_MaxPriori-ties_Con	If the credit shaping shall be enabled for this priority queue
CreditPortCInc	Red_Credit_Type	1	Credit shaper increment for Port C (unused)
CreditPortCDec	Red_Credit_Type	1	Credit shaper decrement for Port C (unused)
CreditPortCEnable	std_logic_vector	1	If credit shaping shall be enabled for Port C (unused)
MaxDataSize	Red_MaxData Size_Type	Red_MaxPriori-ties_Con	Max data size per priority queue
MaxDataSizeEnable	std_logic	1	If max data size filter shall be enabled
MaxDataSizePhase Enable	std_logic_vector	Red_MaxPriori-ties_Con	If max data size filter shall be enabled for this priority queue
CycleEnable	std_logic	1	If cyclic forwarding shall be enabled
PreemptionEnable	std_logic	1	If preemption shall be enabled

			for the lowest priority
CutThough	std_logic	1	If frames shall be processed in cut through mode
Vlan	Red_Vlan_Type	1	The Pcp,Dei and Vid of the VLAN (unused)
VlanEnable	std_logic	1	If VLAN shall be used (unused)
OwnMac	Common_ Byte_Type	6	The MAC of the node. Used for supervision frames and dis-carding
RedMode	Red_Mode_Type	1	Redundancy Mode (unused): Hsr_E Prp_E No_E Tsn_E
NoForward	std_logic	1	If forwarding between Ports A & B shall be disabled (unused)
TailTagging	std_logic	1	If tail tagging shall be used (unused)
PrpUntagging	std_logic	1	If frames shall be PRP un-tagged (unused)
PromiscuousMode	std_logic	1	If Promiscuous mode shall be active

Table 17: Red_TsnStaticConfig_Type

4.1.1.2.13 Red_TsnStaticConfigVal_Type

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

Field Name	Type	Size	Description
Prio_Val	std_logic	1	If the priority related values shall be set
Phase_Val	std_logic	1	If the phase and sheduler related values shall be set
Credit_Val	std_logic	1	If the credit shaper related values shall be set
MaxDataSize_Val	std_logic	1	If the max data size related

			values shall be set
Cycle_Val	std_logic	1	If the cyclic forwarding related values shall be set
Preemption_Val	std_logic	1	If the preemption related values shall be set
Vlan_Val	std_logic	1	If the VLAN shall be set
Enable_Val	std_logic	1	Enables the RED Tsn

Table 18: Red_TsnStaticConfigVal_Type

4.1.1.2.14 Red_TsnStaticStatus_Type

Defined in Red_TsnAddrPackage.vhd of library RedLib
This is the type used for static status supervision.

Field Name	Type	Size	Description
Enabled	std_logic	1	If the core is enabled
RedMode	Red_Mode_Type	1	Redundancy Mode: Hsr_E Prp_E Tsn_E No_E
SupervisionTimeout PortA	std_logic	1	No supervision frames received on Port A
SupervisionTimeout PortA	std_logic	1	No supervision frames received on Port B
Phase	std_logic_vector	Red_MaxPriorities_Con	Phase states
Cycle Start	std_logic	1	Set on the cycle start

Table 19: Red_TsnStaticStatus_Type

4.1.1.2.15 Red_TsnStaticStatusVal_Type

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

Field Name	Type	Size	Description
Phase_Val	std_logic	1	If the phase is valid

Table 20: Red_TsnStaticStatusVal_Type

4.1.1.3 Entity Block Diagram

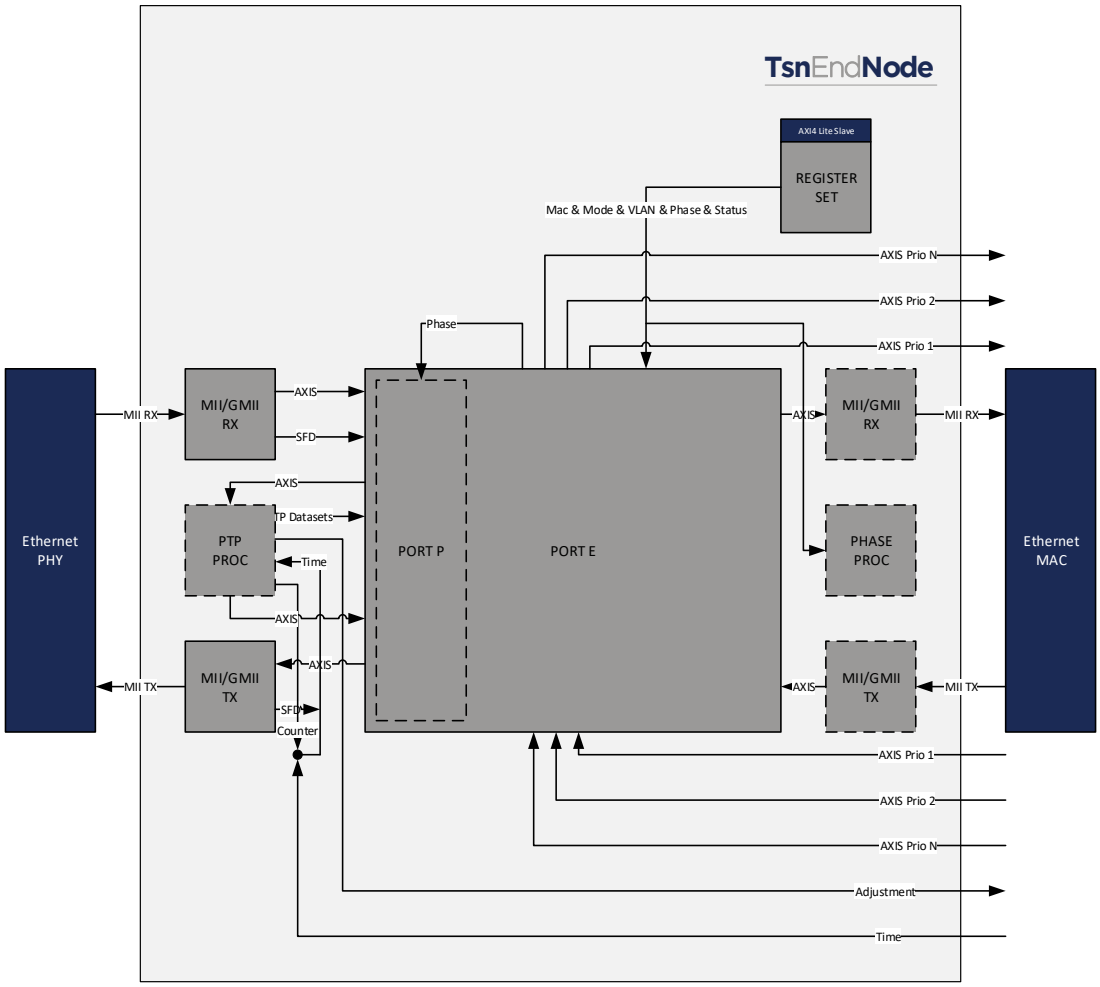


Figure 16: RED Tsn Core

4.1.1.4 Entity Description

Port E

This module combines all TSN related submodules, like scheduler, priority queues, preemption etc.
See 4.2.1 for more details.

Port P

This module is part of Port E and represents a PTP Transparent Clock Port handling delay measurements and residence time corrections of frames.
See 4.2.2 for more details.

Priority Handler

This module is part of each Port and is instanciated twice per Port: one for the input and one for the output side. Depending on the configuration of this module it will do quite different things. It is the core module for TSN since it handles all TSN related parts like priority queuing, cyclic forwarding, scheduled gates, preemption etc.
See 4.2.3 for more details

Phase Processor

This module generates cycles and phases on a per priority queue base. If no port specific Phase Processor is enabled, this is the only Phase Processor in the system. It can run in a simple scheduler mode where just start and stop times per phase need to be set or in an advanced scheduler mode where it goes through a list of gate states to generate phases according to IEEE 802.3 Qbv.
See 4.2.4 for more details.

PTP Processor

This module is basically a PTP Ordinary Clock which can synchronize the system clock or can provide time for other nodes. Together with the PTP TC port in Port A&B and C it forms a PTP Hybrid Clock. It contains an AXI interconnect and two individual registersets: one for the OC and one for the TC part
See 4.2.4.1 for more details.

MAC & PHY Ethernet Interface Adapter

This module converts the Media Independent Interface (MII) to AXI stream and vice versa. It is also in charge of generating correct Interframe Gaps and a Preamble with SFD.
See 0 for more details.

Registerset

This module is an AXI Light Memory Mapped Slave. It provides access to all Registers and allows to configure the TSN Core. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the Registers 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 4.2.7 for more details.

4.1.1.5 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
CutThrough_Gen	-	boolean	1	Support for Cut Through frame handling
PrioritySupport_Gen	-	boolean	1	Support for priority queues
NrOfPriorities_Gen	-	natural	1	How many priorities queues shall be supported (3-8)
PriorityQueue-Load_Gen	-	Common_ Natural_Type	Red_MaxPriori-ties_Con	These numbers in percent (0-100) together with the parameters MaxCyleDuration-Nanoseconds_Gen and LinkSpeedSup-port_Gen define the size of the required buffers. It defines in percent how much of the traffic is expected to be traffic. This basically defines how many frames must be storable within a cycle for this.
PhaseSupport_Gen	-	boolean	1	Support for phases per priority and scheduler
PhasePortSup-port_Gen	-	boolean	1	If a scheduler per port shall be used
PhaseInSupport_Gen	-	boolean	1	Support for phases per priority and

				scheduler also on input queues (use only in spezial cases)
CreditSupport_Gen	-	boolean	1	Support for the credit based shaper
CycleSupport_Gen	-	boolean	1	Support for cyclic forwarding
Preemption Support_Gen	-	boolean	1	Support for preemption on the lowest priority
MaxCyleDuration-Nanoseconds_Gen	-	natural	1	Maximum configu-rable cycle time in nanoseconds
MaxDataSizeSup-port_Gen	-	boolean	1	Support for the max data size filters per priority
PortStatus Support_Gen	-	boolean	1	If frame and error counters shall be available in registers
SimpleScheduler_Gen	-	boolean	1	If the simple sched-uler shall be used for phases
ExtAxisIn Support_Gen	-	boolean	1	Support for external input AXI stream interfaces
ExtAxisOut Support_Gen	-	boolean	1	Support for external output AXI stream interfaces
ExtPortSupport_Gen	-	boolean	1	If Port C also has an (R)(G)MII interface
UntaggerSup-port_Gen	-	boolean	1	If the core shall be able to handle Redundncy tags
DstMacCheck_Gen	-	boolean	1	If only frames matching the con-figured MAC (plus multi and broad-

				cast) will be for-warded
LinkSpeed Support_Gen	-	natural	1	Shall be either 100 or 1000. For 1000 duplication is paral-lelized to achieve the required throughput.
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
StaticConfig_Gen	-	boolean	1	If Static Configura-tion or AXI is used
PtpDefaultProfile Support_Gen	-	boolean	1	If PTP shall support the PTP default profile
PtpUtilityProfile Support_Gen	-	boolean	1	If PTP shall support the PTP utility profile
StaticConfig_Gen	-	boolean	1	If Static Configura-tion or AXI is used
UntaggerSup-port_Gen	-	boolean	1	If an untagger shall remove HSR and TSN tags on recep-tion
RxDelayNano second10_Gen	-	integer	1	Input delay (10Mbit)
RxDelayNano second100_Gen	-	integer	1	Input delay (100Mbit)
RxDelayNano second1000_Gen	-	integer	1	Input delay (1000Mbit)
TxDelayNano second10_Gen	-	integer	1	Output delay (10Mbit)
TxDelayNano second100_Gen	-	integer	1	Output delay (100Mbit)
TxDelayNano second1000_Gen	-	integer	1	Output delay (1000Mbit)

PtpDefaultProfile Support_Gen	-	boolean	1	If PTP shall support the PTP default profile
PtpUtilityProfile Support_Gen	-	boolean	1	If PTP shall support the PTP utility profile
HighResSupport_Gen	-	boolean	1	If a high resolution Timestamper shall be used
HighResFreq Multiply_Gen	-	natural	1	Multiply factor of the System Clock to the High Resolution Clock frequency
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size Default plus 0xFFFF
Sim_Gen	-	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysClkNx_ClkIn	in	std_logic	1	High resolution Timestamping Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig Tsn_DatIn	in	Red_Tsn StaticConfig_Type	1	Static Configuration for TSN and Redun-dancy
StaticConfig Tsn_ValIn	in	Red_Tsn StaticConfigVal _Type	1	Static Configuration valid for TSN and Redundancy
StaticConfig PtpOc_DatIn	in	Ptp_OrdinaryClock StaticConfig_Type	1	Static Configuration for PTP OC

StaticConfig PtpOc_ValIn	in	Ptp_OrdinaryClock StaticConfigVal _Type	1	Static Configuration valid for PTP OC
StaticConfig PtpTc_DatIn	in	Ptp_Transparent ClockStaticConfig _Type	1	Static Configuration for PTP TC
StaticConfig PtpTc_ValIn	in	Ptp_Transparent ClockStaticConfigVal _Type	1	Static Configuration valid for PTP TC
Status				
StaticStatus Tsn_DatOut	out	Red_Tsn StaticStatus_Type	1	Static Status for TSN and Redundan-cy
StaticStatus Tsn_ValOut	out	Red_Tsn StaticStatusVal _Type	1	Static Status valid for TSN and Redun-dancy
StaticStatus PtpOc_DatOut	out	Red_Tsn StaticStatus_Type	1	Static Status for PTP OC
StaticStatus PtpOc_ValOut	out	Red_Tsn StaticStatusVal _Type	1	Static Status valid for PTP OC
StaticStatus PtpTc_DatOut	out	Red_Tsn StaticStatus_Type	1	Static Status for PTP TC
StaticStatus PtpTc_ValOut	out	Red_Tsn StaticStatusVal _Type	1	Static Status valid for PTP TC
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Drift Input				
DriftCountAdjust-ment_DatIn	in	Clk_DriftCount Adjustment_Type	1	Drift Adjustment of the Adjusted Clock
AXIS Inputs				

VlanExt_DatIn	in	Red_VlanArray_Type	NrOfPriorities_Gen	VLAN ID associated with the frame to send and priority class
AxisExtValid_ValIn	in	Axis32_Itf Valid_Type	NrOfPriorities_Gen	AXI Stream frame inputs per priority queue (0 = highest priority)
AxisExtReady_ValOut	out	Axis32_Itf Ready_Type	NrOfPriorities_Gen	
AxisExtData_DatIn	in	Axis32_Itf Data_Type	NrOfPriorities_Gen	
AxisExtStrobe_ValIn	in	Axis32_Itf Strobe_Type	NrOfPriorities_Gen	
AxisExtKeep_ValIn	in	Axis32_Itf Keep_Type	NrOfPriorities_Gen	
AxisExtLast_ValIn	in	Axis32_Itf Last_Type	NrOfPriorities_Gen	
AxisExtUser_DatIn	in	Axis32_Itf User_Type	NrOfPriorities_Gen	
AXIS Outputs				
AxisExtValid_ValOut	out	Axis32_Itf Valid_Type	NrOfPriorities_Gen	AXI Stream frame outputs per priority queue (0 = highest priority)
AxisExtReady_ValIn	in	Axis32_Itf Ready_Type	NrOfPriorities_Gen	
AxisExtData_DatOut	out	Axis32_Itf Data_Type	NrOfPriorities_Gen	
AxisExtStrobe _ValOut	out	Axis32_Itf Strobe_Type	NrOfPriorities_Gen	
AxisExtKeep_ValOut	out	Axis32_Itf Keep_Type	NrOfPriorities_Gen	
AxisExtLast_ValOut	out	Axis32_Itf Last_Type	NrOfPriorities_Gen	
AxisExtUser_DatOut	out	Axis32_Itf User_Type	NrOfPriorities_Gen	
Port Link Input				
Port Link_DatIn	in	std_logic		Link state
Port(R)(G)Mii RX Clk/Rst Input				
Port (R)(G)MiiRxClk_ClkIn	in	std_logic	1	RX Clock
Port	in	std_logic	1	Reset aligned with

(R)(G)MiiRxRstN_RstIn				RX Clock
Port(R)(G)Mii TX Clk/Rst Input				
Port (R)(G)MiiTxClk_ClkIn	in	std_logic	1	TX Clock
Port (R)(G)MiiTxRstN_RstIn	in	std_logic	1	Reset aligned with TX Clock
Port(R)(G)Mii RX Data Input				
Port (R)(G)MiiRxDv_EnaIn	in	std_logic	1	RX Data valid
Port (R)(G)MiiRxErr_EnaIn	in	std_logic	1	RX Error
Port (R)(G)MiiRxData_DatIn	in	std_logic_vector	2-8	RX Data MII:4, RMII:2, GMII:8, RGMII:4
Port (R)(G)MiiCol_DatIn	in	std_logic	1	Collision
Port (R)(G)MiiCrs_DatIn	in	std_logic	1	Carrier Sense
Port(R)(G)Mii TX Data Output				
Port (R)(G)MiiTxEn_EnaOut	out	std_logic	1	TX Data valid
Port (R)(G)MiiTxErr_EnaOut	out	std_logic	1	TX Error
Port (R)(G)MiiTxData_DatOut	out	std_logic_vector	2-8	TX Data MII:4, RMII:2, GMII:8, RGMII:4
Port(R)(G)Mii RX Clk/Rst Output				
Port (R)(G)MiiRxClk_ClkOut	out	std_logic	1	RX Clock
Port (R)(G)MiiRxRstN_RstOut	out	std_logic	1	Reset aligned with RX Clock
Port(R)(G)Mii TX Clk/Rst Output				
Port (R)(G)MiiTxClk_ClkOut	out	std_logic	1	TX Clock
Port (R)(G)MiiTxRstN_RstOut	out	std_logic	1	Reset aligned with TX Clock
Port(R)(G)Mii RX Data Output				
Port (R)(G)MiiRxDv_EnaOut	out	std_logic	1	RX Data valid
Port (R)(G)MiiRxErr_EnaOut	out	std_logic	1	RX Error
Port (R)(G)MiiRxData_Dat	out	std_logic_vector	2-8	RX Data

Out				MII:4, RMII:2, GMII:8, RGMII:4
Port (R)(G)MiiCol_DatOut	out	std_logic	1	Collision
Port (R)(G)MiiCrs_DatOut	out	std_logic	1	Carrier Sense
Port(R)(G)Mii TX Data Input				
Port (R)(G)MiiTxEn_Ena In	in	std_logic	1	TX Data valid
Port (R)(G)MiiTxErr_Ena In	in	std_logic	1	TX Error
Port (R)(G)MiiTxData_Dat In	in	std_logic_vector	2-8	TX Data MII:4, RMII:2, GMII:8, RGMII:4
AXI4 Light 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
Time Adjustment Output				
TimeAdjustment_DatOut	out	Clk_TimeAdjustment_Type	1	Time to set hard
TimeAdjustment_ValOut	out	std_logic	1	Time valid
Offset Adjustment Output				
OffsetAdjustment_DatOut	out	Clk_TimeAdjustment_Type	1	Calculated new Offset between Master and Slave
OffsetAdjustment_ValOut	out	std_logic;	1	Calculated new Offset valid
Drift Adjustment Output				
DriftAdjustment_DatOut	out	Clk_TimeAdjustment_Type	1	Calculated new Drift between Master and Slave
DriftAdjustment_ValOut	out	std_logic;	1	Calculated new Drift valid
Offset Adjustment Input				
OffsetAdjustment_DatIn	in	Clk_TimeAdjustment_Type	1	Calculated new Offset after the PI Servo loop
OffsetAdjustment_ValIn	in	std_logic;	1	Calculated new Offset after the PI Servo loop valid
Drift Adjustment Input				
DriftAdjustment_DatIn	in	Clk_TimeAdjustment_Type	1	Calculated new Drift after the PI Servo loop
DriftAdjustment_ValIn	in	std_logic	1	Calculated new Drift after the PI Servo loop valid

Table 21: RED Tsn Core

4.2 Design Parts

The RED Tsn Core 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 Port E

4.2.1.1 Entity Block Diagram

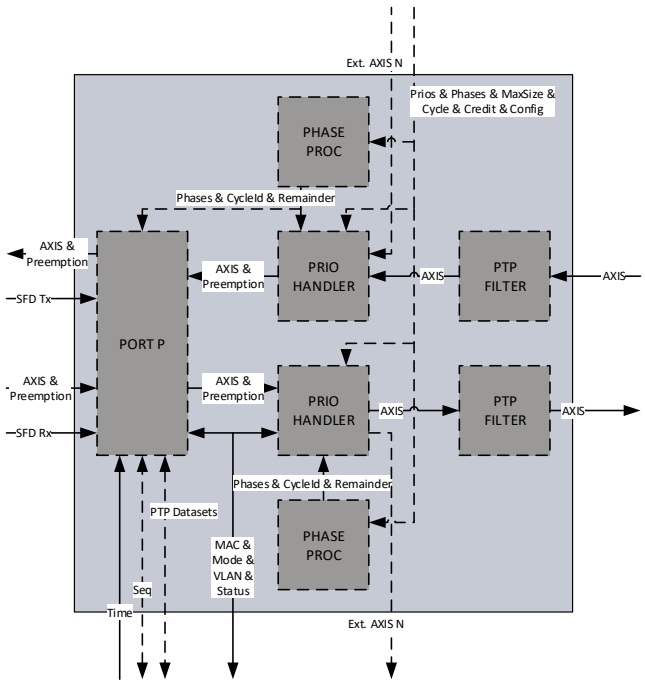


Figure 17: Port E

4.2.1.2 Entity Description

PTP Filter

This module filters out all PTP traffic from and to the uplink..

Port P

This module is a PTP Transparent Clock Port which handles Delay measurement and residence time modifications. It is not Preemption aware for PTP frames but will forward Preemption information. It also uses the phase to know when it is allowed to send Delay messages.

See 4.2.2 for more details

Priority Handler

This module handles all TSN related parts like priority queueing, scheduling, cyclic forwarding, preemption etc. There are two instantiates per Port. One handling the output path and one handling the input path. The functionalities and tasks of the two instantiates differ quite a bit but are only defined via generics.
See 4.2.3 for more details

Phase Processor

This module creates the cycle and phases based on the reference clock and input and output delays as well as path delays on the output path. Because input and output paths are not identically in regard to delays it is instantiated twice.
See 4.2.4 for more details

4.2.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
CutThrough_Gen	-	boolean	1	Support for Cut Through frame handling
PtpSupport_Gen	-	boolean	1	Support for PTP in forwarding (also in general)
PtpDefaultProfile Support_Gen	-	boolean	1	If PTP shall support the PTP default profile
PtpUtilityProfile Support_Gen	-	boolean	1	If PTP shall support the PTP utility profile
PrioritySupport_Gen	-	boolean	1	Support for priority queues
NrOfPriorities_Gen	-	natural	1	How many priorities queues shall be supported (3-8)

PriorityQueue-Load_Gen	-	Common_ Natural_Type	Red_MaxPriori-ties_Con	These numbers in percent (0-100) together with the parameters MaxCyleDuration-Nanoseconds_Gen and LinkSpeedSup-port_Gen define the size of the required buffers. It defines in percent how much of the traffic is expected to be traffic. This basically defines how many frames must be storable within a cycle for this.
PhaseSupport_Gen	-	boolean	1	Support for phases per priority and scheduler
CreditSupport_Gen	-	boolean	1	Support for the credit based shaper
CycleSupport_Gen	-	boolean	1	Support for cyclic forwarding
MaxCyleDuration-Nanoseconds_Gen	-	natural	1	Maximum configu-rable cycle time in nanoseconds
MaxDataSizeSup-port_Gen	-	boolean	1	Support for the max data size filters per priority
Preemption Support_Gen	-	boolean	1	Support for preemption on the lowest priority
UntaggerSup-port_Gen	-	boolean	1	If an untagger shall remove HSR and TSN tags on recep-tion

LinkSpeed Support_Gen	-	natural	1	Shall be either 100 or 1000. For 1000 duplication is parallelized to achieve the required throughput.
SimpleScheduler_Gen	-	boolean	1	If the simple scheduler shall be used for phases
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
HighResSupport_Gen	-	boolean	1	If a high resolution Timestamper shall be used
HighResFreq Multiply_Gen	-	natural	1	Multiply factor of the System Clock to the High Resolution Clock frequency
Sim_Gen	-	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis
Ports				
PtpPortSupport_Gen	-	boolean	1	Support for PTP OC and PTP TC Ports A&B
PortStatus Support_Gen	-	boolean	1	If frame and error counters shall be available in registers
ExtAxisIn Support_Gen	-	boolean	1	Support for external input AXI stream interfaces
ExtAxisOut Support_Gen	-	boolean	1	Support for external output AXI stream interfaces
RxDelayNano	-	integer	1	Input delay (10Mbit)

second10_Gen				
RxDelayNano second100_Gen	-	integer	1	Input delay (100Mbit)
RxDelayNano second1000_Gen	-	integer	1	Input delay (1000Mbit)
TxDelayNano second10_Gen	-	integer	1	Output delay (10Mbit)
TxDelayNano second100_Gen	-	integer	1	Output delay (100Mbit)
TxDelayNano second1000_Gen	-	integer	1	Output delay (1000Mbit)
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysClkNx_ClkIn	in	std_logic	1	High resolution Timestamping Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Own MAC Input				
OwnMac_DatIn	in	Common_Byte_Type	6	MAC address of the node
Mode Input				
Promiscuous-Mode_DatIn	in	std_logic	1	If in Promiscuous mode
CutThrough_ValIn	in	std_logic	1	If cut through shall be enabled
Link Input				
Link_DatIn	in	std_logic	1	Link state of the Port
LinkSpeed_DatIn	in	Com-mon_LinkSpeed_Typ	6	Link spped of the node

		e		
TSN Input				
PhaseControl_DatIn	in	Red_PhaseControl_Type	1	Phase control structure defining cycles and phases
PhaseControl_ValIn	in	std_logic	1	Phase control structure valid
PrioPhase_ValIn	in	std_logic_vector	NrOfPriorities_Gen	Priority and Phase valid
CycleId_DatIn	in	std_logic	1	Cycle identifier
CycleEnable_ValIn	in	std_logic	1	Cyclic forwarding enabled
PrioVlan_DatIn	in	Red_VlanPrios_Type	NrOfPriorities_Gen	VLAN identifier per priority queue
Prio_ValIn	in	std_logic_vector	NrOfPriorities_Gen	Priority valid
Phase_ValIn	in	std_logic_vector	NrOfPriorities_Gen	Phase ready
PhaseRemainder_DatIn	in	Red_Phase Remainder_Type	NrOfPriorities_Gen	Remaining time in phase in nanoseconds
PhaseEnable_ValIn	in	std_logic	1	Scheduler enabled
PrioEnable_ValIn	in	std_logic	1	Priorities enabled
Preemption Enable_ValIn	in	std_logic	1	Preemption enabled
CreditInc_DatIn	in	Red_Credit_Type	NrOfPriorities_Gen	Credit shaper increment
CreditDec_DatIn	in	Red_Credit_Type	NrOfPriorities_Gen	Credit shaper decrement
CreditMin_DatIn	in	Red_CreditMinMax_Type	NrOfPriorities_Gen	Credit shaper min threshold
CreditMax_DatIn	in	Red_CreditMinMax_Type	NrOfPriorities_Gen	Credit shaper max threshold
CreditEnable_ValIn	in	std_logic_vector	NrOfPriorities_Gen	Credit shaping enabled per priority queue
MaxDataSize_DatIn	in	Red_MaxDataSize_Type	NrOfPriorities_Gen	Max data size per priority queue
MaxDataSizeEnable_DatIn	in	std_logic_vector	NrOfPriorities_Gen	Max data size per priority queue

				enable (this will set the priority queue to store and for-ward)
PTP Interface				
PtpCounterTime_DatIn	in	Clk_Time_Type	1	Frequency Adjusted Clock Time
PtpCounterTime_ValIn	in	std_logic	1	Frequency Adjusted Clock Time valid
PtpProfile_DatIn	in	Ptp_Profile_Type	1	PTP Profile
PtpProfile_ValIn	in	std_logic	1	PTP Profile valid
PtpDefaultDataset_DatIn	in	Ptp_Default Dataset_Type	1	PTP Default Dataset
PtpPortDataset_DatIn	in	Ptp_Port Dataset_Type	1	PTP Port Dataset
PtpPortDataset_ValIn	in	Ptp_Port DatasetVal_Type	1	PTP Port Dataset valid
PtpPortDataset_DatOut	out	Ptp_Port Dataset_Type	1	PTP Port Dataset
PtpEnable_EnaIn	in	std_logic	1	Enable PTP
Port Status Output				
PortStatus_DatOut	out	Red_Port Status_Type	1	Port Status
SFD Input				
SfdDetectedRx_EvtIn	in	std_logic	1	SFD RX detected
Preemption Info Input				
PreemptionInfo_DatIn	in	std_logic_vector	16	Preemption Info
PreemptionInfo_ValIn	in	std_logic	1	Preemption Info valid
Axi Input				
AxisRxValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisRxReady_ValOut	out	std_logic	1	
AxisRxData_DatIn	in	std_logic_vector	32	
AxisRxStrobe_ValIn	in	std_logic_vector	4	
AxisRxKeep_ValIn	in	std_logic_vector	4	
AxisRxLast_ValIn	in	std_logic	1	
AxisRxUser_DatIn	in	std_logic_vector	3	
SFD Input				
SfdDetectedTx_EvtIn	in	std_logic	1	SFD TX detected
Preemption Info Output				

PreemptionInfo_DatOut	out	std_logic_vector	16	Preemption Info
PreemptionInfo_ValOut	out	std_logic	1	Preemption Info valid
Axi Output				
AxisTxValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisTxReady_ValIn	in	std_logic	1	
AxisTxData_DatOut	out	std_logic_vector	32	
AxisTxStrobe_ValOut	out	std_logic_vector	4	
AxisTxKeep_ValOut	out	std_logic_vector	4	
AxisTxLast_ValOut	out	std_logic	1	
AxisTxUser_DatOut	out	std_logic_vector	3	
Port Status Output				
PortStatus_DatOut	in	Red_Port Status_Type	1	Port Status
AXIS Inputs				
VlanExt_DatIn	in	Red_VlanArray_Type	NrOfPrior-ities_Gen	VLAN ID associated with the frame to send and priority class
AxisExtValid_ValIn	in	Axis32_Itf Valid_Type	NrOfPrior-ities_Gen	AXI Stream frame inputs per priority queue (0 = highest priority)
AxisExtReady_ValOut	out	Axis32_Itf Ready_Type	NrOfPrior-ities_Gen	
AxisExtData_DatIn	in	Axis32_Itf Data_Type	NrOfPrior-ities_Gen	
AxisExtStrobe_ValIn	in	Axis32_Itf Strobe_Type	NrOfPrior-ities_Gen	
AxisExtKeep_ValIn	in	Axis32_Itf Keep_Type	NrOfPrior-ities_Gen	
AxisExtLast_ValIn	in	Axis32_Itf Last_Type	NrOfPrior-ities_Gen	
AxisExtUser_DatIn	in	Axis32_Itf User_Type	NrOfPrior-ities_Gen	
AXIS Outputs				
AxisExtValid_ValOut	out	Axis32_Itf Valid_Type	NrOfPrior-ities_Gen	AXI Stream frame outputs per priority

				queue (0 = highest priority)
AxisExtReady_ValIn	in	Axis32_Itf Ready_Type	NrOfPrior-ities_Gen	
AxisExtData_DatOut	out	Axis32_Itf Data_Type	NrOfPrior-ities_Gen	
AxisExtStrobe _ValOut	out	Axis32_Itf Strobe_Type	NrOfPrior-ities_Gen	
AxisExtKeep_ValOut	out	Axis32_Itf Keep_Type	NrOfPrior-ities_Gen	
AxisExtLast_ValOut	out	Axis32_Itf Last_Type	NrOfPrior-ities_Gen	
AxisExtUser_DatOut	out	Axis32_Itf User_Type	NrOfPrior-ities_Gen	
Axi Input				
AxisTxValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisTxReady_ValOut	out	std_logic	1	
AxisTxData_DatIn	in	std_logic_vector	32	
AxisTxStrobe_ValIn	in	std_logic_vector	4	
AxisTxKeep_ValIn	in	std_logic_vector	4	
AxisTxLast_ValIn	in	std_logic	1	
AxisTxUser_DatIn	in	std_logic_vector	3	
Axi Output				
AxisRxValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisRxReady_ValIn	in	std_logic	1	
AxisRxData_DatOut	out	std_logic_vector	32	
AxisRxStrobe_ValOut	out	std_logic_vector	4	
AxisRxKeep_ValOut	out	std_logic_vector	4	
AxisRxLast_ValOut	out	std_logic	1	
AxisRxUser_DatOut	out	std_logic_vector	3	
Axi TX Port P Input				
AxisTxPortP Valid_ValIn	in	std_logic	1	AXI Stream frame input
AxisTxPortP Ready_ValOut	out	std_logic	1	
AxisTxPortP Data_DatIn	in	std_logic_vector	32	
AxisTxPortP Strobe_ValIn	in	std_logic_vector	4	
AxisTxPortP Keep_ValIn	in	std_logic_vector	4	
AxisTxPortP	in	std_logic	1	

Last_ValIn				
AxisTxPortP User_DatIn	in	std_logic_vector	3	
Axi RX Port P Output				
AxisRxPortP Valid_ValOut	out	std_logic	1	AXI Stream frame output
AxisRxPortP Ready_ValIn	in	std_logic	1	
AxisRxPortP Data_DatOut	out	std_logic_vector	32	
AxisRxPortP BStrobe_ValOut	out	std_logic_vector	4	
AxisRxPortP Keep_ValOut	out	std_logic_vector	4	
AxisRxPortP Last_ValOut	out	std_logic	1	
AxisRxPortP User_DatOut	out	std_logic_vector	3	
Enable Input				
Enable_EnaIn	in	std_logic	1	Enable core

Table 22: Port E

4.2.2 Port P

This entity is only available if PTP Port support is enabled

4.2.2.1 Entity Block Diagram

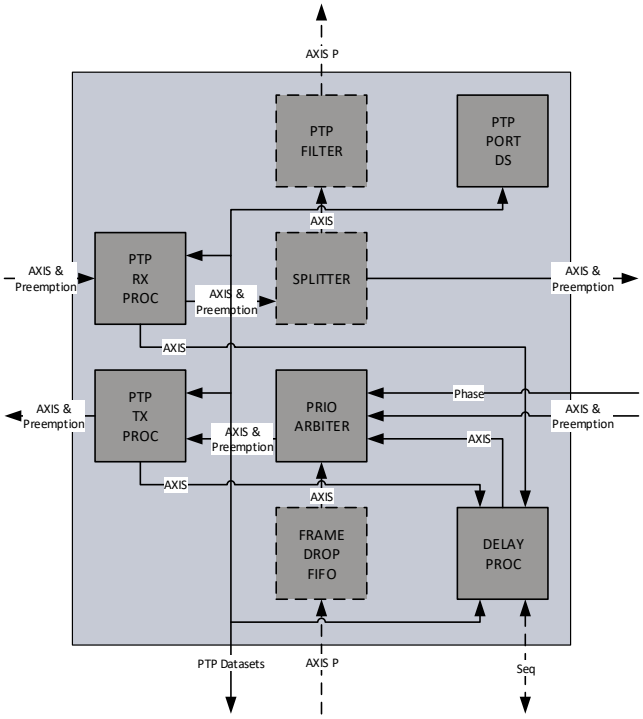


Figure 18: Port P

4.2.2.2 Entity Description

Priority Arbiter

This module merges the AXIS paths from the previous block with the AXIS from the Delay Processor and the AXIS from the PTP Processor based on the active phase, so PTP frames are only sent in phase/prio 2. The merged stream is then forwarded to the PTP TX Processor.

Splitter

This module splits one AXIS into two AXIS. One is forwarded to the next block in Port A&B and C and the other one is forwarded to the PTP Filter.

PTP Filter

This module filters out all other traffic than PTP frame which are then forwarded to the PTP Processor.

PTP Port Dataset

This module represents the PTP Port Dataset as defined in IEEE 1588. It contains the PTP configuration and status information for this PTP port. The Dataset can be written from the PTP Processor's TC Registerset as well as the Delay Processor and provides information to the PTP RX Processor, PTP TX Processor and Delay Processor as well as back to the Registerset.

PTP 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/FRER tag is inserted.

PTP TX Processor

This module handles all outgoing frames. It does the TX timestamping; frame parsing and on-the-fly CorrectionField processing. In addition it realigns the 32bit stream so the PTP frames are always aligned the same way in case an IP header or HSR/FRER tag is inserted.

Delay Processor

This module handles all PTP Delay (P2P only, no E2E support) frames. It runs independent of the current state of the PTP Hybrid 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 the Port Dataset.
As Server it waits for incoming PeerDelayRequests and answers them with corresponding PeerDelayResponses.

Frame Drop Fifo

This is an optional Frame Drop Fifo to overcome additional data speed differences for frames sent from the PTP Processor.

Priority Handler

This module handles all TSN related parts like priority queueing, scheduling, cyclic forwarding, etc. There are two instantiates per Port. One in the top layer handling the output path and one in the TX Processor handling the input path. The functionalities and tasks of the two instantiates differ quite a bit but are only defined via generics. Compared to Ports A&B it doesn't support preemption but support external AXI streams

See 4.2.3 for more details

4.2.2.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
NoSupport_Gen	-	boolean	1	Support for NO (default profile)
HsrSupport_Gen	-	boolean	1	Support for HSR (utility profile)
PrpSupport_Gen	-	boolean	1	Support for PRP (utility profile)
TsnSupport_Gen	-	boolean	1	Support for FRER (tsn profile)
VlanSupport_Gen	-	boolean	1	Support for VLAN
PtpSupport_Gen	-	boolean	1	Support for PTP in forwarding (also in general)
PhaseSupport_Gen	-	boolean	1	Support for phases per priority and scheduler
Preemption Support_Gen	-	boolean	1	Support for preemption on the lowest priority
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
HighResSupport_Gen	-	boolean	1	If a high resolution

				Timestamper shall be used
HighResFreq Multiply_Gen	-	natural	1	Multiply factor of the System Clock to the High Resolution Clock frequency
Sim_Gen	-	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis
Ports				
RxDelayNano second10_Gen	-	integer	1	Output delay (10Mbit)
RxDelayNano second100_Gen	-	integer	1	Output delay (100Mbit)
RxDelayNano second1000_Gen	-	integer	1	Output delay (1000Mbit)
TxDelayNano second10_Gen	-	integer	1	Input delay (10Mbit)
TxDelayNano second100_Gen	-	integer	1	Input delay (100Mbit)
TxDelayNano second1000_Gen	-	integer	1	Input delay (1000Mbit)
ExtPtpPort_Gen	-	boolean	1	Support for external AXI stream interfac-es
PortIdModify_Gen	-	boolean	1	If Port Identities shall be modified
PortNumber_Gen	-	std_logic_vector	16	Which port this is for PTP
SrcId_Gen	-	std_logic_vector	4	Port identifier: 0xA for Port A 0xB for Port B
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	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Mode Input				
RedMode_DatIn	in	Red_Mode_Type	1	Redundancy Mode: Hsr_E Prp_E No_E
Link Input				
LinkSpeed_DatIn	in	Com-mon_LinkSpeed_Type	6	Link spped of the node
TSN Input				
Phase_ValIn	in	std_logic	1	Phase ready
PhaseEnable_ValIn	in	std_logic	1	Scheduler enabled
SFD Input				
SfdDetectedRx_EvtIn	in	std_logic	1	SFD RX detected
Preemption Info Input				
PreemptionInfoRx_DatIn	in	std_logic_vector	16	Preemption Info
PreemptionInfoRx_ValIn	in	std_logic	1	Preemption Info valid
Axi Input				
AxisRxValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisRxReady_ValOut	out	std_logic	1	
AxisRxData_DatIn	in	std_logic_vector	32	
AxisRxStrobe_ValIn	in	std_logic_vector	4	
AxisRxKeep_ValIn	in	std_logic_vector	4	
AxisRxLast_ValIn	in	std_logic	1	
AxisRxUser_DatIn	in	std_logic_vector	3	
Preemption Info Output				
PreemptionInfoRx_DatOut	out	std_logic_vector	16	Preemption Info
PreemptionInfoRx_ValOut	out	std_logic	1	Preemption Info valid
Axi Output				
AxisRxValid_ValOut	out	std_logic	1	AXI Stream frame

AxisRxReady_ValIn	in	std_logic	1	output
AxisRxData_DatOut	out	std_logic_vector	32	
AxisRxStrobe_ValOut	out	std_logic_vector	4	
AxisRxKeep_ValOut	out	std_logic_vector	4	
AxisRxLast_ValOut	out	std_logic	1	
AxisRxUser_DatOut	out	std_logic_vector	3	
SFD Input				
SfdDetectedTx_EvtIn	in	std_logic	1	SFD TX detected
Preemption Info Input				
PreemptionInfoTx_DatIn	in	std_logic_vector	16	Preemption Info
PreemptionInfoTx_ValIn	in	std_logic	1	Preemption Info valid
Axi Input				
AxisTxValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisTxReady_ValOut	out	std_logic	1	
AxisTxData_DatIn	in	std_logic_vector	32	
AxisTxStrobe_ValIn	in	std_logic_vector	4	
AxisTxKeep_ValIn	in	std_logic_vector	4	
AxisTxLast_ValIn	in	std_logic	1	
AxisTxUser_DatIn	in	std_logic_vector	3	
Preemption Info Output				
PreemptionInfoTx_DatOut	out	std_logic_vector	16	Preemption Info
PreemptionInfoTx_ValOut	out	std_logic	1	Preemption Info valid
Axi Output				
AxisTxValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisTxReady_ValIn	in	std_logic	1	
AxisTxData_DatOut	out	std_logic_vector	32	
AxisTxStrobe_ValOut	out	std_logic_vector	4	
AxisTxKeep_ValOut	out	std_logic_vector	4	
AxisTxLast_ValOut	out	std_logic	1	
AxisTxUser_DatOut	out	std_logic_vector	3	
Axi Port POutput				
AxisRxPortPValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisRxPortPReady_ValIn	in	std_logic	1	
AxisRxPortPData_DatOut	out	std_logic_vector	32	
AxisRxPortPStrobe_ValOut	out	std_logic_vector	4	

AxisRxPortP Keep_ValOut	out	std_logic_vector	4	
AxisRxPortP Last_ValOut	out	std_logic	1	
AxisRxPortP User_DatOut	out	std_logic_vector	3	
Axi Port P Input				
AxisTxPortP Valid_ValIn	in	std_logic	1	AXI Stream frame input
AxisTxPortP Ready_ValOut	out	std_logic	1	
AxisTxPortP Data_DatIn	in	std_logic_vector	32	
AxisTxPortP Strobe_ValIn	in	std_logic_vector	4	
AxisTxPortP Keep_ValIn	in	std_logic_vector	4	
AxisTxPortP Last_ValIn	in	std_logic	1	
AxisTxPortP User_DatIn	in	std_logic_vector	3	
PTP Interface				
CounterTime _DatIn	in	Clk_Time_Type	1	Frequency Adjusted Clock Time
CounterTime _ValIn	in	std_logic	1	Frequency Adjusted Clock Time valid
Profile_DatIn	in	Ptp_Profile_Type	1	PTP Profile
Profile_ValIn	in	std_logic	1	PTP Profile valid
DefaultDataset _DatIn	in	Ptp_Default Dataset_Type	1	PTP Default Dataset
PortDataset_DatIn	in	Ptp_Port Dataset_Type	1	PTP Port Dataset
PortDataset_ValIn	in	Ptp_Port DatasetVal_Type	1	PTP Port Dataset valid
PortDataset_DatOut	out	Ptp_Port Dataset_Type	1	PTP Port Dataset
Enable_EnaIn	in	std_logic	1	Enable PTP
SeqNr Info Output				
SeqReq_DatOut	out	Red_SeqReq_Type	2	SeqNr request for Delay
SeqReq_ValOut	out	Red_SeqReqVal_ Type	2	SeqNr request for Delay valid
SeqNr Info Input				
SeqResp_DatIn	in	Red_SeqResp_ Type	2	SeqNr response for Delay

SeqResp_ValIn	in	Red_SeqRespVal_Type	2	SeqNr response for Delay valid

Table 23: Port P

4.2.3 Priority Handler

This module is highly configurable. It depens that priority queues are supported. In the core it is used in 2 different configurations: RX Port, TX Port. The different configurations differ in their functionality: E.g. RX ports will never do cycle checks, credit shaping or preemption, TX ports on the other hands will never assemble preempted streams. Having one module for all differne configurations however makes sense to avoid code duplication.

4.2.3.1 Entity Block Diagram

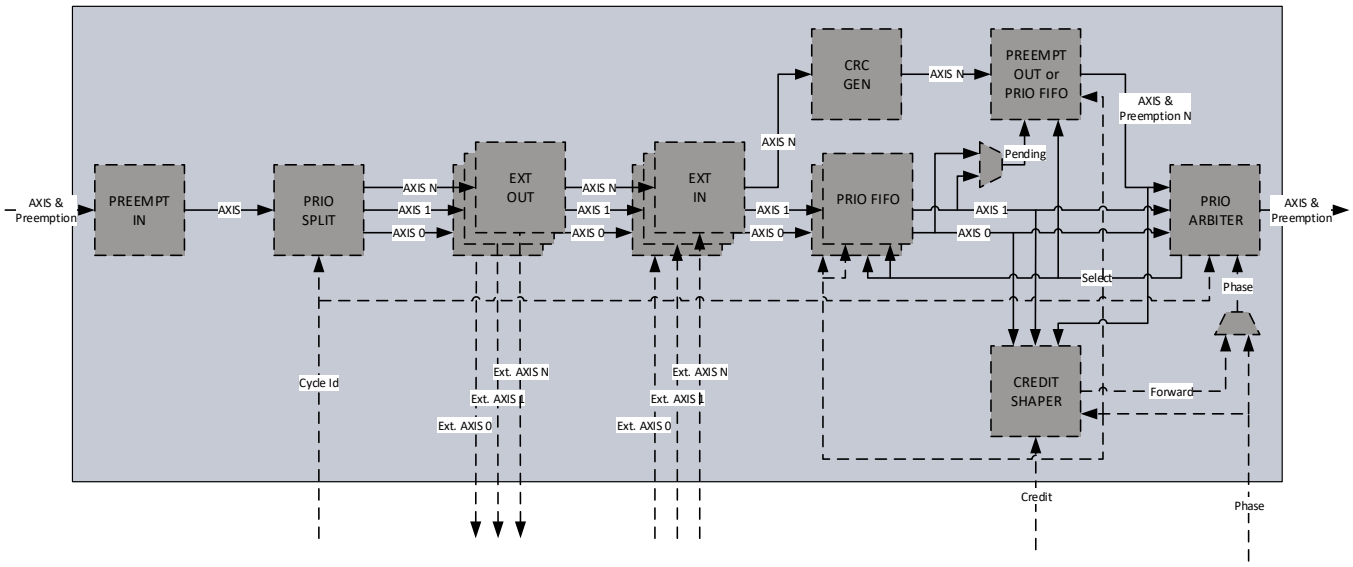


Figure 19: Priority Handler

4.2.3.2 Entity Description

Preemption Input

This module detects and reasambles preempted frames. It forwards “normal” frames directly and waits for the preempted frames until a frame is completely assembled before forwarding the frame. There are buffers for both types and an arbiter between them, since when a preempted frame is completed it will be forwarded at once which means that the “normal” frames have to wait. Over time this levels out and no buffer overrun should happen.

Priority Splitter

This module parses the frames for VLAN tags and extracts the priorities. Depending on the configuration it then forwards the frame to the corresponding priority

queue. It also inserts the Cycle identifier to mark in which phase the frame was received. Optionally it has a CRC checker which drops incoming frames when the CRC is wrong.

External Ouput

This module splits the AXI streams into two streams one as output stream and one to the Priorty FIFO. This means frames to the external ports are forwarded as soon as they arrive independent of phases and cycles

Priority FIFO

This module is a frame drop fifo which has checks for minimum and maximum frame sizes and drops the frame if it is not in bounds. Also it aligns the sending with phases and checks if it has enough time to finish the frame before the phase ends. It is the main buffer in the system which is also in charge of buffering frames for cyclic forwarding. The buffersize has to be chosen depending on the max cycle time and link speed supported

CRC Generator

This module generates a CRC for the lowest priority frame in preemption is used and external AXI streams are used and CRC generation is enabled. Normaly the CRC calculation is done in the tagging part, but for an End Node the tagger does not exist.

Preemption Output

This module is a store and forward FIFO which splits a frame in preemption fragments. It checks if a higher priority queue wants to send a frame or if the whole frame will not fit into the phase anymore and will split it if these conditions are met. It makes sure that fragments will have the xored crc and that min Ethernet frame sizes are always assured.

Priority Arbiter

This module merges the AXIS paths from the previous blocks based on phases and cycle identifiers. The phases are generated by the credit based shaper together with the scheduler. Optionally it also contains a CRC Generator for “normal” frames. Normaly the CRC calculation is done in the tagging part, but for an End Node the tagger does not exist.

Credit Shaper

This module decrements a credit whenever a frame is sent and increments the credit whenever it is either idle or it must wait for another frame to be sent.
This allows to shape the bandwidth usage and minimal gap between two consecutive frame per priority class

4.2.3.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
NrOfPriorities_Gen	-	natural	1	How many priorities queues shall be supported (3-8)
Depth_Gen	-	natural	1	Depth of the FIFOs in Words
ExtAxisIn_Gen	-	boolean	1	If external input AXI stream interfaces shall be used
ExtAxisOut_Gen	-	boolean	1	If external output AXI stream interfaces shall be used
ExtPortIn_Gen	-	boolean	1	If the module is fed by an external port (e.g. (R)(G)MII port)
ExtPortOut_Gen	-	boolean	1	If the module is feeding an external port (e.g. (R)(G)MII port)
Ready_Gen	-	boolean	1	If the FIFO generates a Ready signal when it becomes full or just drops the incoming frame
CutThrough_Gen	-	boolean	1	If the FIFO shall run in cut through mode (this is only supported if Cyclic

				forwarding and MaxSize filtering is not enabled)
CycleIdCheck_Gen	-	boolean	1	If it shall check the Cycle identifier in ther Priority Arbiter when sending
CycleIdInsert_Gen	-	boolean	1	If it shall insert the Cycle identifier in the Priority Splitter on receiving
Phase_Gen	-	boolean	1	Support for phases per priority and scheduler
Credit_Gen	-	boolean	1	Support for the credit based shaper
MaxDataSize_Gen	-	boolean	1	Support for the max data size filters per priority
CrcCheck_Gen	-	boolean	1	If a CRC check shall be done on recepe-tion
CrcGen_Gen	-	natural	1	If a CRC generation shall be done on sending
PreemptionIn_Gen	-	boolean	1	If preempted frame assembling shall be supported
PreemptionOut_Gen	-	boolean	1	If preemption shall be done for the lowest priority
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Link Input				
Link_DatIn	in	std_logic	1	Link state of the Port

LinkSpeed_DatIn	in	Com-mon_LinkSpeed_Type	6	Link spped of the node
TSN Input				
PhaseEnable_ValIn	in	std_logic	1	Scheduler enabled
CycleEnable_ValIn	in	std_logic	1	Cyclic forwarding enabled
PrioVlan_DatIn	in	Red_VlanPrios_Type	NrOfPrior-ities_Gen	VLAN identifier per priority queue
Prio_ValIn	in	std_logic_vector	NrOfPrior-ities_Gen	Priority valid
Preemption Enable_ValIn	in	std_logic	1	Preemption enabled
CycleIdExt_DatIn	in	std_logic	1	Cycle identifier for external AXI streams
CycleIdIn_DatIn	in	std_logic	1	Cycle identifier to add
CycleIdOut_DatIn	in	std_logic	1	Cycle identifier to check
PhaseRemainder _DatIn	in	Red_Phase Remainder_Type	NrOfPrior-ities_Gen	Remaining time in phase in nanosec-onds
PhaseOut_ValIn	in	std_logic_vector	NrOfPrior-ities_Gen	Phase ready
CreditInc_DatIn	in	Red_Credit_Type	NrOfPrior-ities_Gen	Credit shaper in-crement
CreditDec_DatIn	in	Red_Credit_Type	NrOfPrior-ities_Gen	Credit shaper dec-rement
CreditEnable_ValIn	in	std_logic_vector	NrOfPrior-ities_Gen	Credit shaping enabled per priority queue
MaxDataSize_DatIn	in	Red_MaxDataSize_Type	NrOfPrior-ities_Gen	Max data size per priority queue
MaxDataSizeEnable _DatIn	in	std_logic_vector	NrOfPrior-ities_Gen	Max data size per priority queue enable (this will set the priority queue to store and for-ward)
AXIS Input				

VlanExt_DatIn	in	Red_VlanArray_Type	NrOfPriorities_Gen	VLAN ID associated with the frame to send and priority class
AxisExtValid_ValIn	in	Axis32_Itf Valid_Type	NrOfPriorities_Gen	AXI Stream frame inputs per priority queue (0 = highest priority)
AxisExtReady_ValOut	out	Axis32_Itf Ready_Type	NrOfPriorities_Gen	
AxisExtData_DatIn	in	Axis32_Itf Data_Type	NrOfPriorities_Gen	
AxisExtStrobe_ValIn	in	Axis32_Itf Strobe_Type	NrOfPriorities_Gen	
AxisExtKeep_ValIn	in	Axis32_Itf Keep_Type	NrOfPriorities_Gen	
AxisExtLast_ValIn	in	Axis32_Itf Last_Type	NrOfPriorities_Gen	
AxisExtUser_DatIn	in	Axis32_Itf User_Type	NrOfPriorities_Gen	
AXIS Outputs				
AxisExtValid_ValOut	out	Axis32_Itf Valid_Type	NrOfPriorities_Gen	AXI Stream frame outputs per priority queue (0 = highest priority)
AxisExtReady_ValIn	in	Axis32_Itf Ready_Type	NrOfPriorities_Gen	
AxisExtData_DatOut	out	Axis32_Itf Data_Type	NrOfPriorities_Gen	
AxisExtStrobe _ValOut	out	Axis32_Itf Strobe_Type	NrOfPriorities_Gen	
AxisExtKeep_ValOut	out	Axis32_Itf Keep_Type	NrOfPriorities_Gen	
AxisExtLast_ValOut	out	Axis32_Itf Last_Type	NrOfPriorities_Gen	
AxisExtUser_DatOut	out	Axis32_Itf User_Type	NrOfPriorities_Gen	
Preemption Info Input				

PreemptionInfo_DatIn	in	std_logic_vector	16	Preemption Info
PreemptionInfo_ValIn	in	std_logic	1	Preemption Info valid
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	
Preemption Info Output				
PreemptionInfo_DatOut	out	std_logic_vector	16	Preemption Info
PreemptionInfo_ValOut	out	std_logic	1	Preemption Info valid
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 24: Priority Handler

4.2.4 Phase Processor

This entity comes in two versions a simple version where just a cycle period and start and stop offset in regards to the cycle start are provided and an advanced version where the scheduler according to IEEE 802.1 Qbv is implemented. Here only the advanced scheduler is described.
In some configurations the phases are not relevant but only the cycle start and cycle id, therefore the Phase Process is optional.

4.2.4.1 Entity Block Diagram

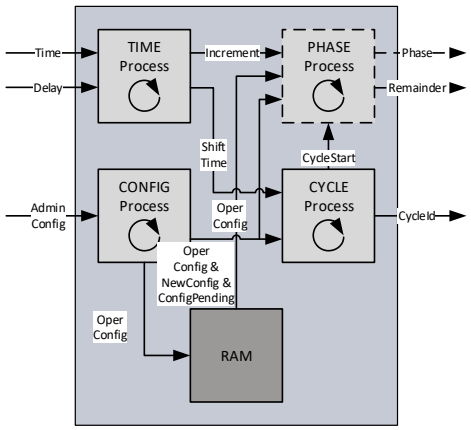


Figure 20: Phase Processor

4.2.4.2 Entity Description

RAM

This module stores the operational config which is a doublebuffer which can be switched between the new config at the right point in time.

Time Process

This process adds or substracts the input ot output delay from the current time to have a correct alignment for this port. This shifted time is provided to the Cycle Prosses. It also calculates the clock increment so the frequency difference to the ascillator frequency is taken account of for the duration of the phases.

Config Process

This process aligns configuration changes with the cycle. It signals whenever a new configuration change is pending or when a new configuration gets valid.

When a configuration gets valid it copies the Admin Configuration to the Operational Configuration.

Cycle Process

This process generates the Cycles and Cycle Identifier. It signals to the Phase Process whenever a new cycle will start, which will triger the phase generation.

Phase Process

This process is in charge of generating the phases. It is triggered by the start of a new cycle an goes then through a list of gate states. Each entry in the list contains the gate state (phase) and a duration which defines for how long this configuration will be valid. It will assert the gate states and will then wait until the time for this configuration has passed (time is measured with the frequency adjusted clock). It also calculates the phase remainder time per priority over the next 3 valid entries which is used to determine if a frame can be completely sent in the phase or not and also if a frame shall be preempted.

4.2.4.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriodNanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
GenRemainder_Gen	-	boolean	1	If a remainder shall be generated per priority (requires that phases are generated)
GenCycle_Gen	-	boolean	1	If a cycle identifier shall be generated
GenPhase_Gen	-	boolean	1	If phases shall be generated
NrOfPriorities_Gen	-	natural	1	How many priorities queues shall be supported (2-8)

MaxDelay_Gen	-	natural	1	Maximum of the Delay
AddSubDelay_Gen	-	boolean	1	If the Delay shall be added or subtracted from the Time (false = to generate earlier)
SimpleScheduler_Gen	-	boolean	1	If the simple or advanced shall be used (true = simple)
Sim_Gen	-	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Delay Input				
DelayNanosecond_DatIn	in	integer	1	Delay in nanoseconds
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
TSN Input				
PhaseControl_DatIn	in	Red_PhaseControl_Type	1	Phase control structure defining cycles and phases
PhaseControl_ValIn	in	std_logic	1	Phase control structure valid
PrioPhase_ValIn	in	std_logic_vector	NrOfPriorities_Gen	Priority and Phase valid
PhaseEnable_ValIn	in	std_logic	1	Scheduler enabled
TSN Output				
CycleId_DatOut	out	std_logic	1	Cycle identifier
Prio_ValOut	out	std_logic_vector	NrOfPriorities_Gen	Priority valid
PhaseRemainder_DatOut	out	Red_Phase	NrOfPriorities_Gen	Remaining time in

		Remainder_Type		phase in nanosec-onds

Table 25: Phase Processor

4.2.5 PTP Processor

This entity is only available if PTP Port support is enabled

4.2.5.1 Entity Block Diagram

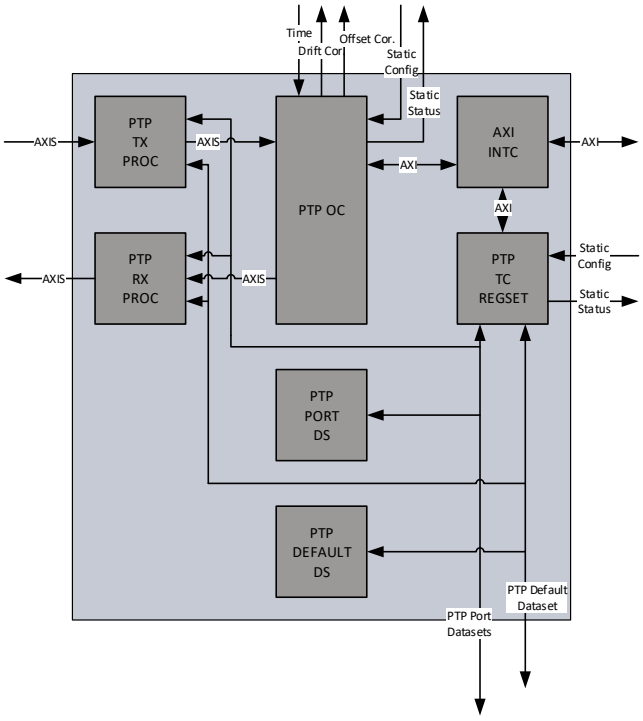


Figure 21: PTP Processor

4.2.5.2 Entity Description

PTP Default Dataset

This module represents the PTP Default Dataset as defined in IEEE 1588. It contains the PTP configuration (e.g. MAC, profile etc) and status information for all PTP TC ports (Port P) and the PTP RX Processor, PTP TX Processor. The Dataset can be written from the PTP TC Registerset and provides information back to the Registerset.

PTP Port Dataset

This module represents the PTP Port Dataset as defined in IEEE 1588. It contains the PTP configuration and status information for this PTP port. The Dataset can be written from the PTP TC Registerset as well as and provides information to the PTP RX Processor, PTP TX Processor as well as back to the Registerset.

PTP Transparent Clock Registerset

This module is an AXI Light Memory Mapped Slave. It provides access to all registers and allows configuring the PTP TC part. It allows to read and write the PTP Default Dataset and PTP Port Datasets associated with the TC part of the PTP Hybrid Clock.
AXI4 Light 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 registers 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 configuration parameters the clock has to be disabled and enabled again. If in AXI mode, an AXI Master has to configure the registers with AXI writes to the registers, which is typically done by a CPU. Parameters can in this case also be changed at runtime.

AXI Interconnect

This module is a simple AXI Interconnect allowing to access multiple AXI Light Memory Mapped Slaves from one Master. It does no width conversion, clock domain crossing, doesn't support bursts etc. It is simply to map multiple AXI Light Memory Mapped Slave into the same address range so from the core level only one AXI Light Memory Mapped interface is visible.

PTP 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/FRER tag is inserted.

PTP TX Processor

This module handles all outgoing frames. It does the TX timestamping; frame parsing and on-the-fly CorrectionField processing. In addition it realigns the 32bit stream so the PTP frames are always aligned the same way in case an IP header or HSR/FRER tag is inserted.

PTP Ordinary Clock

This module is a complete FPGA only PTP Ordinary Clock according to IEEE 1588.

It can be Master or Slave which is determined in the Best Master Clock Algorithm (BMCA). When in Slave mode it calculates and adjusts the drift and offset against the Master which is the time reference. All calculations, frame handling and algorithms are done in the core, no CPU is required.
For more details on the PTP Ordinary Clock core check the documentation:
www.nettimelogic.com/resources/Ptp_OrdinaryClock_ReferenceManual.pdf

4.2.5.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used
NoSupport_Gen	-	boolean	1	Support for NO (default profile)
HsrSupport_Gen	-	boolean	1	Support for HSR (utility profile)
PrpSupport_Gen	-	boolean	1	Support for PRP (utility profile)
TsnSupport_Gen	-	boolean	1	Support for FRER (tsn profile)
NoSupport_Gen	-	boolean	1	Support for NO (just forward no dropping, but duplication)
PtpSupport_Gen	-	boolean	1	Support for PTP in forwarding (also in general)
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
NrOfPtpPorts_Gen	-	natural	1	Nr of PTP Ports (always 1)
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address

AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size Default plus 0xFFFF
Sim_Gen	-	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig PtpOc_DatIn	in	Ptp_OrdinaryClock StaticConfig_Type	1	Static Configuration for PTP OC
StaticConfig PtpOc_ValIn	in	Ptp_OrdinaryClock StaticConfigVal _Type	1	Static Configuration valid for PTP OC
StaticConfig PtpTc_DatIn	in	Ptp_Transparent ClockStaticConfig _Type	1	Static Configuration for PTP TC
StaticConfig PtpTc_ValIn	in	Ptp_Transparent ClockStaticConfigVal _Type	1	Static Configuration valid for PTP TC
Status				
StaticStatus PtpOc_DatOut	out	Red_Tsn StaticStatus_Type	1	Static Status for PTP OC
StaticStatus PtpOc_ValOut	out	Red_Tsn StaticStatusVal _Type	1	Static Status valid for PTP OC
StaticStatus PtpTc_DatOut	out	Red_Tsn StaticStatus_Type	1	Static Status for PTP TC
StaticStatus PtpTc_ValOut	out	Red_Tsn StaticStatusVal _Type	1	Static Status valid for PTP TC
TSN Input				
CycleId_DatIn	in	std_logic	1	Cycle identifier
CycleEnable_ValIn	in	std_logic	1	Cyclic forwarding enabled

Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Own MAC Input				
OwnMac_DatIn	in	Common_Byte_Type	6	MAC address of the node
Mode Input				
RedMode_DatIn	in	Red_Mode_Type	1	Redundancy Mode: Hsr_E Prp_E No_E
Drift Input				
DriftCountAdjustment_DatIn	in	Clk_DriftCount Adjustment_Type	1	Drift Adjustment of the Adjusted Clock
PTP Interface				
PtpCounterTime_DatOut	out	Clk_Time_Type	1	Adjusted Counter Time
PtpCounterTime_ValOut	out	std_logic	1	Adjusted Counter Time valid
PtpProfile_DatOut	out	Ptp_Profile_Type	1	PTP Profile
PtpProfile_ValOut	out	std_logic	1	PTP Profile valid
PtpDefaultDataset_DatOut	out	Ptp_Default Dataset_Type	1	PTP Default Dataset
PtpPortDataset_DatOut	out	Ptp_Port Dataset_Type	1	PTP Port Dataset
PtpPortDataset_ValOut	out	Ptp_Port DatasetVal_Type	1	PTP Port Dataset valid
PtpPortDataset_DatIn	in	Ptp_Port Dataset_Type	1	PTP Port Dataset
PtpEnable_EnaIn	in	std_logic	1	Enable PTP
AXI4 Light 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
Axi TX Port P Input				
AxisTxPortP Valid_ValIn	in	std_logic	1	AXI Stream frame input
AxisTxPortP Ready_ValOut	out	std_logic	1	
AxisTxPortP Data_DatIn	in	std_logic_vector	32	
AxisTxPortP Strobe_ValIn	in	std_logic_vector	4	
AxisTxPortP Keep_ValIn	in	std_logic_vector	4	
AxisTxPortP Last_ValIn	in	std_logic	1	
AxisTxPortP User_DatIn	in	std_logic_vector	3	

Axi TX Port P Output				
AxisTxPortP Valid_ValOut	out	std_logic	1	AXI Stream frame output
AxisTxPortP Ready_ValIn	in	std_logic	1	
AxisTxPortP Data_DatOut	out	std_logic_vector	32	
AxisTxPortP BStrobe_ValOut	out	std_logic_vector	4	
AxisTxPortP Keep_ValOut	out	std_logic_vector	4	
AxisTxPortP Last_ValOut	out	std_logic	1	
AxisTxPortP User_DatOut	out	std_logic_vector	3	
Time Adjustment Output				
TimeAdjustment _DatOut	out	Clk_TimeAdjustment _Type	1	Time to set hard
TimeAdjustment _ValOut	out	std_logic	1	Time valid
Offset Adjustment Output				
OffsetAdjustment _DatOut	out	Clk_TimeAdjustment _Type	1	Calculated new Offset between Master and Slave
OffsetAdjustment _ValOut	out	std_logic;	1	Calculated new Offset valid
Drift Adjustment Output				
DriftAdjustment _DatOut	out	Clk_TimeAdjustment _Type	1	Calculated new Drift between Master and Slave
DriftAdjustment _ValOut	out	std_logic;	1	Calculated new Drift valid
Offset Adjustment Input				
OffsetAdjustment _DatIn	in	Clk_TimeAdjustment _Type	1	Calculated new Offset after the PI Servo loop
OffsetAdjustment _ValIn	in	std_logic;	1	Calculated new Offset after the PI Servo loop valid
Drift Adjustment Input				
DriftAdjustment _DatIn	in	Clk_TimeAdjustment _Type	1	Calculated new Drift after the PI Servo loop
DriftAdjustment _ValIn	in	std_logic	1	Calculated new Drift after the PI Servo

				loop valid

Table 26: PTP Processor

4.2.6 Ethernet Interface Adapter

4.2.6.1 Entity Block Diagram

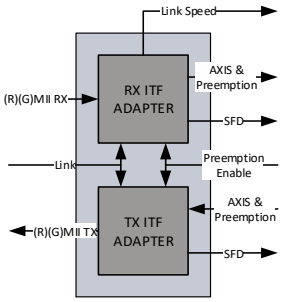


Figure 22: Ethernet Interface Adapter

4.2.6.2 Entity Description

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.
It also detects the SFD and signals this to the previous module.
When Preemption is supported and enabled it extracts the Preemption information and forwards it aligned with the AXI stream interface.
It also detects the link speed automatically and provides it to the other modules.
When link is down it will not push any more frames to the asynchronous FIFO, so no frames are received.

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.
It also detects the SFD and signals this to the previous module.
When Preemption is supported and enabled it inserts the Preemption information in the last Premable byte and the SFD. It expects the Preemption onformation aligned with the AXI stream interface.
When link is down it will not push any more frames to the asynchronous FIFO, so no frames are sent.

4.2.6.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
Interface Adapter				
PreemptionSupport_Gen	-	boolean	1	If Preemption shall be supported
IoFf_Gen	-	boolean	1	Shall IO flip flops be instantiated
ClockClkPeriodNano-second_Gen	-	natural	1	Integer Clock Period
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Preemption Input				
PreemptionEnable_ValIn	in	std_logic	1	If Preemption is enabled
Link Input				
Link_DatIn	in	std_logic	1	Link state of the Port
Link Output				
LinkSpeed_DatOut	in	std_logic	1	Link speed of the Port
SFD Output				
(R)(G)MiiOutSfd Detected_EvtIn	in	std_logic	1	SFD on output to (R)(G)MII detected
(R)(G)MiiInSfd Detected_EvtIn	in	std_logic	1	SFD on input to (R)(G)MII detected
(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	TX Clock
(R)(G)MiiTxRstN_RstIn	in	std_logic	1	Reset aligned with TX 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
Preemption Info Input				
PreemptionInfo_DatIn	in	std_logic_vector	16	Preemption Info
PreemptionInfo_ValIn	in	std_logic	1	Preemption Info valid
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	
Preemption Info Output				
PreemptionInfo _DatOut	out	std_logic_vector	16	Preemption Info
PreemptionInfo _ValOut	out	std_logic	1	Preemption Info valid
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 27: Ethernet Interface Adapter

4.2.7 Registerset

4.2.7.1 Entity Block Diagram

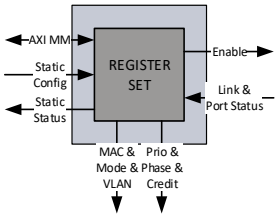


Figure 23: Registerset

4.2.7.2Entity Description

Register Set

This module is an AXI Light Memory Mapped Slave. It provides access to the TSN related registers and allows configuring the RED Tsn Core. AXI4 Light 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 registers 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 configuration parameters the core has to be disabled and enabled again. If in AXI mode, an AXI Master has to configure the registers with AXI writes to the registers, which is typically done by a CPU. Parameters can in this case also be changed at runtime.
Eventhough configuration is possible via a signals, then only the simple scheduler can be used. This core is made to be configured by a CPU, since LLDP and discovery functions etc. shall run on the CPU.
In addition it counts frames and errors of all ports to allow status supervision. This only when the generic PortStatusSupport_Gen is true

4.2.7.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
NoSupport_Gen	-	boolean	1	Support for NO

				(default profile)
HsrSupport_Gen	-	boolean	1	Support for HSR (utility profile)
PrpSupport_Gen	-	boolean	1	Support for PRP (utility profile)
TsnSupport_Gen	-	boolean	1	Support for FRER (tsn profile)
CutThrough_Gen	-	boolean	1	Support for Cut Through frame handling
PrioritySupport_Gen	-	boolean	1	Support for priority queues
NrOfPriorities_Gen	-	natural	1	How many priorities queues shall be supported (3-8)
PhaseSupport_Gen	-	boolean	1	Support for phases per priority and scheduler
CreditSupport_Gen	-	boolean	1	Support for the credit based shaper
CycleSupport_Gen	-	boolean	1	Support for cyclic forwarding
Preemption Support_Gen	-	boolean	1	Support for preemption on the lowest priority
MaxDataSizeSup-port_Gen	-	boolean	1	Support for the max data size filters per priority
PortStatus Support_Gen	-	boolean	1	If frame and error counters shall be available in registers
SimpleScheduler_Gen	-	boolean	1	If the simple sched-uler shall be used for phases
Register Set				
StaticConfig_Gen	-	boolean	1	If Static Configura-tion or AXI is used
PortStatus	-	boolean	1	If frames and error

Support_Gen				counters shall be available in the registerset
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_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	Red_Tsn StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Red_Tsn StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Red_Tsn StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Red_Tsn StaticStatusVal _Type	1	Static Status valid
Port Status Input				
PortStatus PortA_DatIn	in	Red_Port Status_Type	1	Port Status
PortStatus PortB_DatIn	in	Red_Port Status_Type	1	Port Status
PortStatus PortC_DatIn	in	Red_Port Status_Type	1	Port Status
AXI4 Light 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
Timeout Input				
SupervisionTimeout PortA_DatIn	in	std_logic	1	Whether Port A has a Supervision timeout
SupervisionTimeout PortB_DatIn	in	std_logic	1	Whether Port B has a Supervision timeout
TSN Input				
CycleId_DatIn	in	std_logic	1	What the current Cycle Id is
Phase_ValIn	in	std_logic_vector	NrOfPriorities_Gen	Which phase is active
Own MAC Output				
OwnMac_DatOut	out	Common_Byte_Type	6	MAC address of the node

VLAN Output				
Vlan_DatOut	out	Red_Vlan_Type	6	VLAN
VlanEnable_DatOut	out	std_logic	1	VLAN mode enabled
Mode Output				
Promiscuous-Mode_DatOut	out	std_logic	1	If in Promiscuous mode
CutThrough_ValOut	out	std_logic	1	If cut through shall be enabled
Enable Output				
RedTsn Enable_DatOut	out	std_logic	1	Enables the core
TSN Output				
PhaseControl_DatOut	out	Red_PhaseControl _Type	1	Phase control structure defining cycles and phases
PhaseControl_ValOut	out	std_logic	1	Phase control structure valid
PrioPhase_ValOut	out	std_logic_vector	NrOfPriorities_Gen	Priority and Phase valid
PrioVlan_DatOut	out	Red_VlanPrios_Type	NrOfPriorities_Gen	VLAN Priority per priority queue
PhaseEnable_ValOut	out	std_logic	1	Scheduler and phases enabled
PrioEnable_ValOut	out	std_logic	1	Priority Queues enabled
CycleEnable_ValOut	out	std_logic	1	Cyclic Forwarding enabled
PreemptionEnable _ValOut	out	std_logic	1	Preemption enabled
CreditInc_DatOut	out	Red_Credit_Type	NrOfPriorities_Gen	Credit shaper increment
CreditDec_DatOut	out	Red_Credit_Type	NrOfPriorities_Gen	Credit shaper decrement
CreditMin_DatOut	out	Red_CreditMinMax _Type	NrOfPriorities_Gen	Credit shaper minimum threshold
CreditMax_DatOut	out	Red_CreditMinMax _Type	NrOfPriorities_Gen	Credit shaper maximum threshold
CreditEnable_ValOut	out	std_logic_vector	NrOfPriorities_Gen	Credit shaping enabled per priority

				queue
MaxDataSize_DatOut	out	std_logic_vector	NrOfPrior-ities_Gen	Max data size per priority queue
MaxDataSizeEnable_DatOut	out	std_logic_vector	NrOfPrior-ities_Gen	Max data size per priority queue enable (this will set the priority queue to store and for-ward)

Table 28: Registerset

4.3 Configuration example

In both cases the enabling of the core shall be done last, after or together with the configuration.

4.3.1 Static Configuration

```
constant RedStaticConfigTsn_Con : Red_TsnStaticConfig_Type := (
    PrioVlan                                        => (
        0                                           => "010",
        1                                           => "001",
        others                                      => "000"),
    PrioEnable                                      => '1',
    PrioPhaseEnable                                 => (
        0                                           => '1'
        1                                           => '1'
        2                                           => '1'
        others                                      => '0'),
    Phase                                           => (
        0                                           => (
            Start                                   => x" 00000000",
            Stop                                    => x" 00004E20"),
        3                                           => (
            Start                                   => x" 00004E20",
            Stop                                    => x" 00009C40"),
        2                                           => (
            Start                                   => x" 00009C40",
            Stop                                    => x" 000186A0"),
        others                                      => (
            Start                                   => x" 00000000",
            Stop                                    => x" 00000000")),
    PhaseEnable                                     => '1'
    PhaseCycleTime                                  => x" 000186A0"
    CreditInc                                       => (others => x"0001"),
    CreditDec                                       => (others => x"0002"),
    CreditMin                                       => (others => x”C4653600”),
    CreditMax                                       => (others => x”3B9ACA00”),
    CreditEnable                                    => '1'
    CreditPhaseEnable                               => (
        0                                           => '1'
        1                                           => '1'
        2                                           => '1'
        others                                      => '0'),
    CreditPortCInc                                  => (others => '0'),
    CreditPortCDec                                  => (others => '0'),
    CreditPortCMin                                  => x”C4653600”,
    CreditPortCMax                                  => x”3B9ACA00”,
    CreditPortCEnable                               => '0',
    MaxDataSize                                     => (others => (others => '0')),
```

```
    MaxDataSizeEnable                                => '0',
    MaxDataSizePhaseEnable                           => (others => '0'),
    CycleEnable                                      => '1',
    PreemptionEnable                                 => '1',
    CutThrough                                       => '1',
    Vlan                                             => (
        Pcp                                          => "001",
        Dei                                          => '0',
        Vid                                          => x"004"),
    VlanEnable                                       => '1',
    OwnMac                                           => (
        0                                            => x"00",
        1                                            => x"01",
        2                                            => x"02",
        3                                            => x"03",
        4                                            => x"04",
        5                                            => x"05"),
    RedMode                                          => Hsr_E,
    NoForward                                        => '0',
    PromiscuousMode                                  => '0'
);

constant RedStaticConfigValTsn_Con : Red_TsnStaticConfigVal_Type := (
    Prio_Val                                         => '1',
    Phase_Val                                        => '1',
    Credit_Val                                       => '1',
    MaxDataSize_Val                                  => '1',
    Cycle_Val                                        => '1',
    Preemption_Val                                   => '1',
    Vlan_Val                                         => '1',
    Enable_Val                                       => '1',
);

constant PtpStaticConfigOc_Con : Ptp_OrdinaryClockStaticConfig_Type := (
    Profile                                          => TsnProfile_E,
    Layer                                            => Layer2_E,
    TwoStep                                          => '0',
    Signaling                                        => '0',
    Vlan                                             => (
        Pcp                                          => "001",
        Dei                                          => '0',
        Vid                                          => x"004"),
    Ip                                               => (
        0                                            => x"C0",
        1                                            => x"A8",
        2                                            => x"00",
        3                                            => x"10"),
    DefaultDataset_ClockIdentity                     => (
        0                                            => x"00",
```

```
        1                                       => x"01",
        2                                       => x"02",
        3                                       => x"FF",
        4                                       => x"FE",
        5                                       => x"03",
        6                                       => x"04",
        7                                       => x"05"),
    DefaultDataset_DomainNumber                 => x"00",
    DefaultDataset_ClockQuality                 => Ptp_ClockQuality_Type_Rst_Con,
    DefaultDataset_Priority1                    => x"80",
    DefaultDataset_Priority2                    => x"80",
    DefaultDataset_GrandmasterId                => x"0003",
    DefaultDataset_GrandmasterTimeInaccuracy    => x"00000032",
    TimePropertiesDataset_CurrentUtcOffset      => x"0019",
    TimePropertiesDataset_CurrentUtcOffsetValid => '0',
    TimePropertiesDataset_Leap59                => '0',
    TimePropertiesDataset_Leap61                => '0',
    TimePropertiesDataset_TimeTraceable         => '0',
    TimePropertiesDataset_FrequencyTraceable    => '0',
    TimePropertiesDataset_PtpTimescale          => '1',
    TimePropertiesDataset_TimeSource            => x"A0",
    TimePropertiesDataset_CurrentOffset         => (others => '0'),
    TimePropertiesDataset_JumpSeconds           => (others => '0'),
    TimePropertiesDataset_TimeOfNextJumpSeconds => (others => '0'),
    TimePropertiesDataset_DisplayNameLength     => x"03",
    TimePropertiesDataset_DisplayName => (
        0                           => Common_CharacterToStdLogic_Func('P'),
        1                           => Common_CharacterToStdLogic_Func('T'),
        2                           => Common_CharacterToStdLogic_Func('P'),
        others                      => (others => '0'))
);

constant PtpStaticConfigValOc_Con : Ptp_OrdinaryClockStaticConfigVal_Type := (
    Enable_Val                                  => '1',
    Profile_Val                                 => '1',
    Vlan_Val                                    => '1',
    Ip_Val                                      => '1',
    DefaultDataset_ClockIdentity_Val            => '1',
    DefaultDataset_DomainNumber_Val             => '1',
    DefaultDataset_ClockQuality_Val             => '1',
    DefaultDataset_Priority1_Val                => '1',
    DefaultDataset_Priority2_Val                => '1',
    DefaultDataset_GrandmasterId_Val            => '1',
    DefaultDataset_GrandmasterTimeInaccuracy_Val => '1',
    TimePropertiesDataset_CurrentUtcOffset_Val  => '1',
    TimePropertiesDataset_CurrentUtcOffsetValid_Val => '1',
    TimePropertiesDataset_Leap59_Val            => '1',
    TimePropertiesDataset_Leap61_Val            => '1',
    TimePropertiesDataset_TimeTraceable_Val     => '1',
```

```
    TimePropertiesDataset_FrequencyTraceable_Val    => '1',
    TimePropertiesDataset_PtpTimescale_Val          => '1',
    TimePropertiesDataset_TimeSource_Val            => '1',
    TimePropertiesDataset_CurrentOffset_Val         => '1',
    TimePropertiesDataset_JumpSeconds_Val           => '1',
    TimePropertiesDataset_TimeOfNextJumpSeconds_Val => '1',
    TimePropertiesDataset_DisplayNameLength_Val     => '1',
    TimePropertiesDataset_DisplayName_Val           => '1'
);

constant PtpStaticConfigTc_Con : Ptp_TransparentClockStaticConfig_Type := (
    Profile                                         => TsnProfile_E,
    Layer                                           => Layer2_E,
    TwoStep                                         => '0',
    Vlan                                            => Ptp_Vlan_Type_Rst_Con,
    VlanEnable                                      => '1',
    Ip                                              => (
        0                                           => x"C0",
        1                                           => x"A8",
        2                                           => x"00",
        3                                           => x"30"),
    DefaultDataset_ClockIdentity                    => (
        0                                           => x"00",
        1                                           => x"01",
        2                                           => x"02",
        3                                           => x"FF",
        4                                           => x"FE",
        5                                           => x"03",
        6                                           => x"04",
        7                                           => x"05"),
    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 24: Static Configuration

4.3.2 AXI Configuration

The following code is a simplified pseudocode from the testbench: The base address of the RED Tsn Core is 0x10000000. The base address of the PTP part is with the RED Tsn range 0x10001000

```
-- PTP OC
-- Config
-- profile: 2 utility profile
AXI WRITE 10001084 00000002
-- VLAN 0x4000 = prio2
AXI WRITE 10001088 00014000
-- set config valid bits
AXI WRITE 10001080 00000003

-- Default Dataset OC
-- clock id: 00:01:02:FF:FE:03:04:05
AXI WRITE 10001104 FF020100
AXI WRITE 10001108 050403FE
-- domain: 0
AXI WRITE 1000110C 00000000
-- clock class: BB, clock accuracy FE, priority1 80, Priority 80
AXI WRITE 10001110 BBFE8080
-- set default dataset valid bits
AXI WRITE 10001100 0000007F

-- TimeProperties Dataset OC
-- current utc offset & "00" & current utc offset valid &
-- leap59 & leap61 & time traceable & frequency traceable & ptp timescale &
-- time source
AXI WRITE 10001504 001901A0
-- set timeproperties dataset valid bits
AXI WRITE 10001500 00001FFF

-- PTP TC
-- Config
-- profile: 2 utility profile
AXI WRITE 10001884 00000002
-- VLAN 0x4000 = prio2
AXI WRITE 10001888 00014000
-- set config valid bits
AXI WRITE 10001880 00000003

-- Default Dataset TC
-- clock id: 00:01:02:FF:FE:03:04:05
AXI WRITE 10001904 FF020100
AXI WRITE 10001908 050403FE
-- domain: 0
```

```
AXI WRITE 1000190C 00000000
-- set default dataset valid bits
AXI WRITE 10001900 00000003

-- set portnr 0
AXI WRITE 10001A00 00000000
-- set default dataset valid bits
AXI WRITE 10001900 00000001
-- set portnr 1
AXI WRITE 10001A00 00010000
-- set default dataset valid bits
AXI WRITE 10001900 00000001
-- set portnr 2
AXI WRITE 10001A00 00020000
-- set default dataset valid bits
AXI WRITE 10001900 00000001

-- enable PTP OC
AXI WRITE 10001000 00000001

-- enable PTP TC
AXI WRITE 10001800 00000001

-- RED TSN
-- Config
-- Mode No Promiscous
AXI WRITE 10000084 00000000
-- Mode valid
AXI WRITE 10000080 00000001
-- MAC 3..0
AXI WRITE 10000104 03020100
-- MAC 5..4
AXI WRITE 10000108 00000504
-- MAC valid
AXI WRITE 10000100 00000001
-- Enable
AXI WRITE 10000000 00000001

-- PRIO 1: VLAN 1
AXI AXI0 WRITE 10000310 00000001
-- PRIO 2: VLAN 2
AXI AXI0 WRITE 10000314 00000002

--------------------------------
-- SIMPLE SHEDULER
--------------------------------
-- PERIOD : 100000 ns
AXI WRITE 10000340 000186A0
```

```
-- PHASE START PRIO1 : 0 ns
AXI WRITE 10000380 00000000
-- PHASE STOP PRIO1 : 20000 ns
AXI WRITE 100003A0 00004E20

-- PHASE START PRIO2 : 20000 ns
AXI WRITE 10000384 00004E20
-- PHASE STOP PRIO2 : 40000 ns
AXI WRITE 100003A4 00009C40

-- PHASE START ALL : 40000 ns
AXI WRITE 10000388 00009C40
-- PHASE STOP ALL : 100000 ns
AXI WRITE 100003A8 000186A0

-- Enable Prio and Phase and Preemption
AXI WRITE 10000300 7007011B

--------------------------------
-- ADVANCED SHEDULER
--------------------------------
-- CYCLE : 100000 ns
AXI WRITE 10000340 000186A0

-- CYCLE EXTENSION : 10000 ns
AXI WRITE 10000344 00002710

-- BASE TIME NS : 0 ns
AXI AXI0 WRITE 10000348 00000000

-- BASE TIME S : 0 s
AXI WRITE 1000034C 00000000

-- GATE STATES : All enabled
AXI WRITE 10000350 000000FF

-- CONTROL LENGTH : 3
AXI WRITE 10000370 00000003

-- CONTROL 1 : 78800 ns: PRIO 1
AXI WRITE 10000380 010133D0
-- CONTROL 2 : 20000 ns: PRIO 2§
AXI WRITE 10000384 02004E20
-- CONTROL 3 :  1200 ns: PRIO 2 & 3
AXI WRITE 10000388 060004B0

-- Enable Prio, Phase and Preemption
AXI WRITE 10000300 7007011B
```

Figure 25: AXI Configuration

In the example the Promiscuous Mode is disabled.

4.4 Clocking and Reset Concept

4.4.1 Clocking

To keep the design as robust and simple as possible, the whole RED Tsn Core, 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 RED Tsn runs on as well as the counter clock etc.
(R)(G)MII Interface		
PHY (R)(G)MII RX Clock	2.5/25/125MHz	Asynchronous, external receive clock from the PHY also used for the MAC. Depending on the interface not all frequencies apply.
PHY (R)(G)MII TX Clock	2.5/25/125MHz	Asynchronous, external transmit clock to/from the PHY 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 29: Clocks

4.4.2Reset

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 30: 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 Altera (Cyclone V)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (3 priority queue, simple scheduler, PTP, no cyclic forwarding, no credit shaper, no preemption, no max size filter, NO&PRP, DAN, static config, 100Mbit)	xxx	xxx	xxx	xxx
Maximal (8 priority queue, advanced scheduler, PTP, cyclic forwarding, credit shaper, preemption, max size filter, NO&HSR&PRP&FRER, RedBox, 64 proxy, AXI, 1000Mbit)	xxx	xxx	xxx	xxx

Table 31: Resource Usage Altera

5.2 Xilinx (Kintex 7)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (3 priority queue, simple scheduler, PTP, no cyclic forwarding, no credit shaper, no preemption, no max size filter, NO&PRP, DAN, static config, 100Mbit)	xxx	xxx	xxx	xxx
Maximal (8 priority queue, advanced scheduler, PTP, cyclic forwarding, credit shaper, preemption, max size filter, NO&HSR&PRP&FRER, RedBox, 64 proxy, AXI, 1000Mbit)	xxx	xxx	xxx	xxx

Table 32: Resource Usage Xilinx

6 Delivery Structure

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

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

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

RED                                 -- RED library folder
|-Core                              -- RED library cores
 |-Doc                              -- RED library cores documentations
 |-Library                          -- RED library component sources
 |-Package                          -- RED library package sources
 |-Refdesign                        -- RED library cores reference designs
 |-Testbench                        -- RED 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 TSN Core testbench consist of 4 parse/port types: AXI, ETH (AXIS, MII or GMII), CLK and PTP (MII or GMII). Multiple instances exist. The TSN core is configured with all features supported and 3 priority queues, depending on the test case the advanced or simple scheduler is used. The PHY0 ETH port is multiplexed with the PTP port and connected to the port going to the PHY from the DUT (which acts like a MAC). MAC0 is connected to the port going to the MAC from the DUT (which acts like a PHY). The PTP port has its own CLK port as reference time. A Clock instance provides the time and 1ms timer event for the DUT and receives the adjustments from the PTP OC which is part of the DUT.
Three AXIS ETH ports are used for the external streaming ports of the DUT.
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.

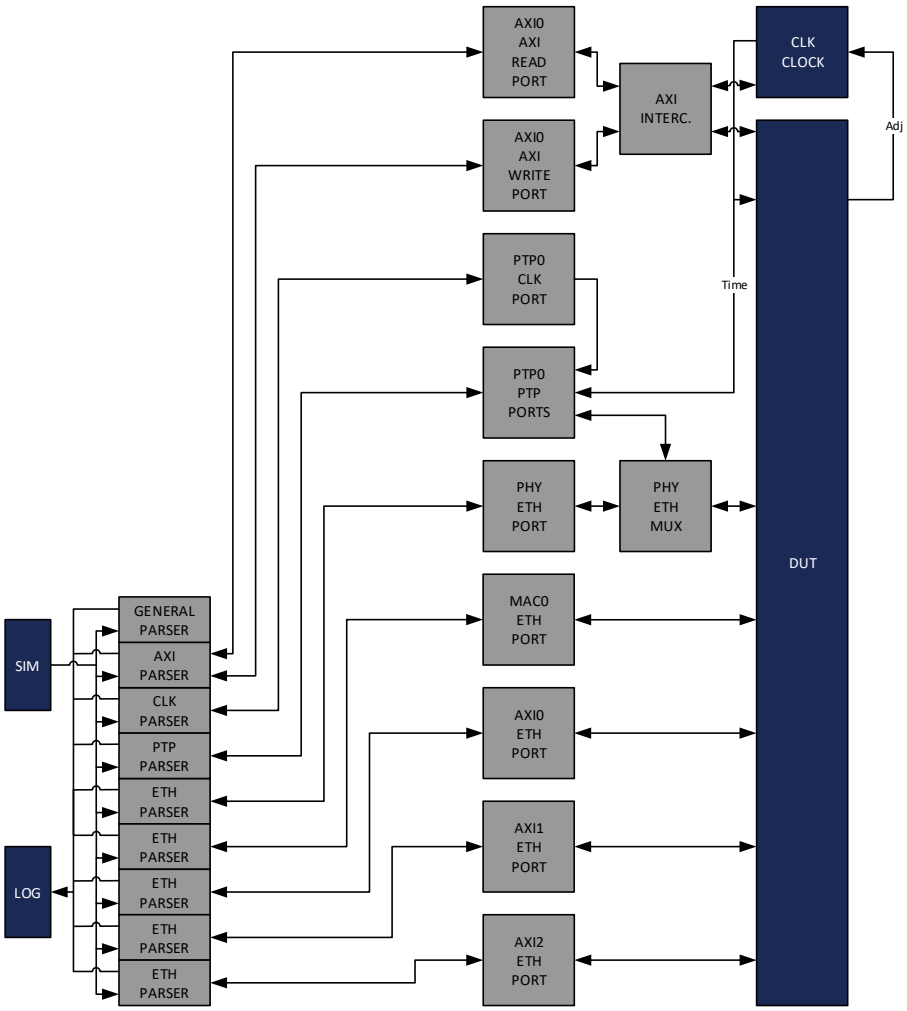


Figure 26: Testbench Framework

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/RED/Testbench/Core/RedTsnEndNode/Script/run_Red_TsnEndNodeMii_Tb.tcl
```

3. Check the log file LogFile1.txt in the XXX/RED/Testbench/Core/RedTsnEndNode/Log/ folder for simulation results.

8 Reference Designs

The TSN End Node Core reference design contains a PLL to generate all necessary clocks (cores are run at 50 MHz), an instance of the RED Tsn End Node Core 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 a TSN network which supports scheduling, cyclic forwarding, and preemption.
All TSN features can be enabled and disabled according to the capabilities or requirement of the system connected.
All generics can be adapted to the specific needs.

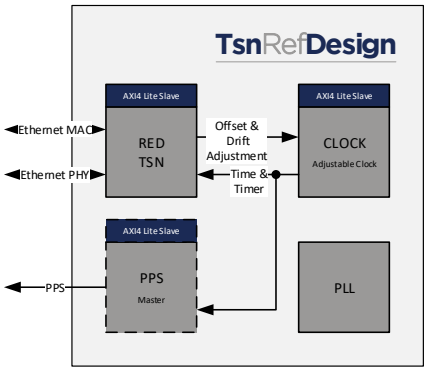


Figure 27: Reference Design

8.1 Xilinx: Digilent NetFpga

The NetFpga board is an FPGA board from Digilent Inc. with a Xintex7 FPGA from Xilinx. (http://store.digilentinc.com/netfpga-1g-cml-kintex-7-fpga-development-board)

1. Open Vivado 2017.4
2. Run TCL script /RED/Refdesign/Xilinx/NetFpga/RedTsnEndNodeMii/RedTsnEndNode.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. Rerun implementation
5. Download to FPGA via JTAG

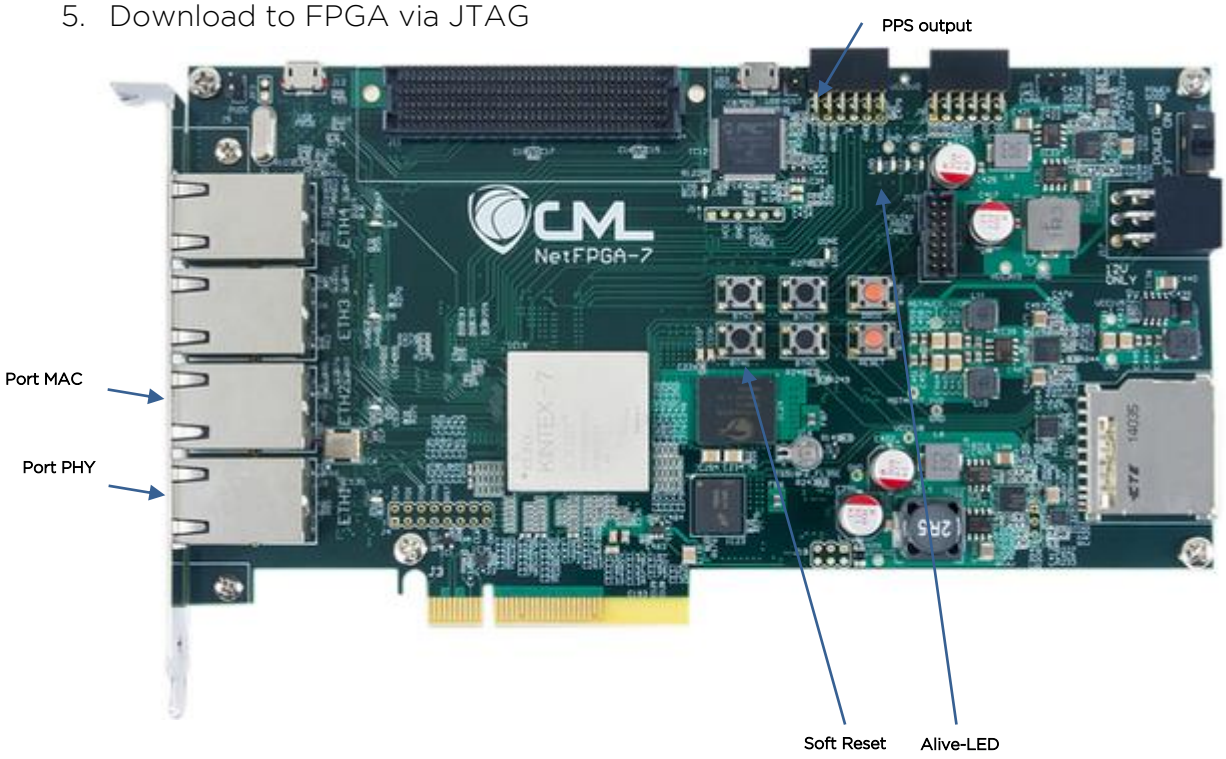


Figure 28: NetFPGA (source Digilent Inc)

A List of tables

B List of figures