TsnPlugfestApp

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Sven Meier
Reviewer(s)	-
Version	1.1
Date	20.06.2019

Copyright Notice

Copyright © 2025 NetTimeLogic GmbH, Switzerland. All rights reserved.
Unauthorized duplication of this document, in whole or in part, by any means, is prohibited without the prior written permission of NetTimeLogic GmbH, Switzerland.
All referenced registered marks and trademarks are the property of their respective owners

Disclaimer

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

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

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

Overview

NetTimeLogic's TSN Plugfest Application core is a full hardware (FPGA) only implementation of the IIC Plugfest Interoperability Application. It implements the Real-time Data Transmission (Talker), Real-time Data Reception (Listener), Analysis Application and Analysis Result Publication (Analyzer) according to the "IIC Interoperability Application v0.3" specification.
It also contains a LED Pattern Generator and a LED Application which generates the PWM brightness pattern as described in the "LED Strip Demo Application v0.3" specification.
In addition it allows to connect an external data sink and source which can receive or transmit real time data which is encapsulated in the IIC Plugfest frame (which is an OPC UA raw binary frame).
The core is meant to be connected to the TSN and Adjustable Counter Clock IP Cores from NetTimeLogic.
For maximum accuracy, the core is using additional Timestampers which are directly connected to the (R)(G)MII Interface Adapters to take timestamps as close to the cable as possible.
The core can be configured either by an AXI4Lite-Slave Register interface.

Key Features:

- Real-time Data Transmission (Talker)
- Real-time Data Reception (Listener)
- Analysis Application and Analysis Result Publication (Analyzer)
- LED Pattern Generator
- LED Application
- External Data Source interface
- External Data Sink interface
- HW Timestamping
- Which Features shall be available can be defined via Generics
- AXI4Lite register set
- Timestamp resolution with 50 MHz system clock: 10ns

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	26.04.2019	First draft
1.0	29.04.2019	First release
1.1	20.06.2019	Added Register descriptions and some minor clean-up

Table 1: Revision History

Content

Definitions

Definitions	
TSN Plugfest App	A defined Application for the IIC TSN Plugfest
Talker	A block that can send OPC/UA frames
Listener	A block that can receive OPC/UA frames
Analyzer	A block which analyses the received OPC/UA frames, does statistics and publishes the statistics

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
FPGA	Field Programmable Gate Array
IIC	Industrial Internet Consortium
TSN	Time Sensitive Networking
IRQ	Interrupt, Signaling to e.g. a CPU
TS	Timestamp
TB	Testbench
LUT	Look Up Table
FF	Flip Flop
RAM	Random Access Memory
ROM	Read Only Memory
VHDL	Hardware description Language for FPGA's

Table 3: Abbreviations

1 Introduction

1.1 Context Overview

The TSN Plugfest App is meant as a co-processor handling special OPC/UA frames as defined for the IIC TSN Plugfest. It takes the time form the Adjustable Counter Clock (which is synchronized by the TSN Network Core) and generates periodic OPC/UA Real-time frames containing either a Register Value, a LED Pattern, or Custom Data encapsulated in this specific frame. For Custom Applications an interface exists which allows to send and receive data via the OPC/UA Real-time frames but without the need of handling the actual frames.
It also receives and parses the received OPC/UA Real-time frames and does statistics on the received data which is periodical published via an OPC/UA Best-effort frame. Since the OPC/UA Realtime frames contain precise timestamps a special Timestamper module is used which timestamps the frames as close as possible at the PHYs (on transmission and reception).
In addition, the core encapsulated the LED App which listens for specific Data and converts the Data in 3 PWM brightness LED outputs and provides 15 binary LED outputs which represent a light curtain as defined in the specification. It can also generate its own LED pattern, for this an LED pattern generator is implemented.
All features can be enabled and disabled at synthesis time.
It contains an AXI4Lite slave for configuration and status supervision from a CPU.
Additional Status signals are provided for debugging.

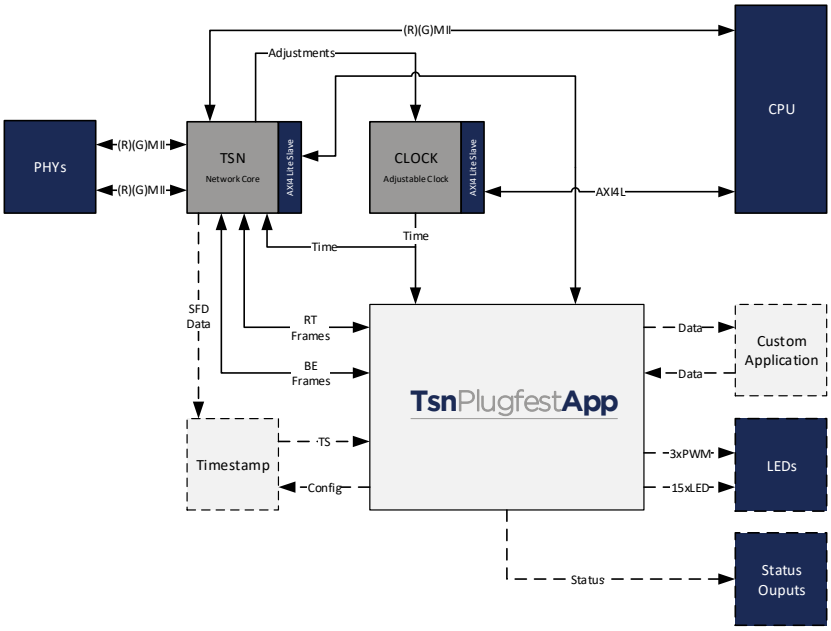

Figure 1: Context Block Diagram

1.2 Function

The TSN Plugfest App sends periodically OPC/UA Real-time frames for testing and measuring the timely behavior of the TSN core itself. The frames are sent as Layer2 multicast frames with different payload data (LED patter, Register Values, Custom App). At the same time, it is receiving the OPC/UA Real-time frames of all nodes, parses and extracts meta data, passes data to the Applications (LED PWM, Custom App) and does timely analysis (delays, offsets, etc.). The analysis result is published periodically via a OPC/UA Best-effort frames so a central node can do the analysis of the whole network.
Since for the analysis precise Timestampers are needed they are connected as close to the PHYs as possible an provide the timestamps to the Talker and Listener part.
All timestamps, scheduling etc. are based on the synchronized Clock.

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.

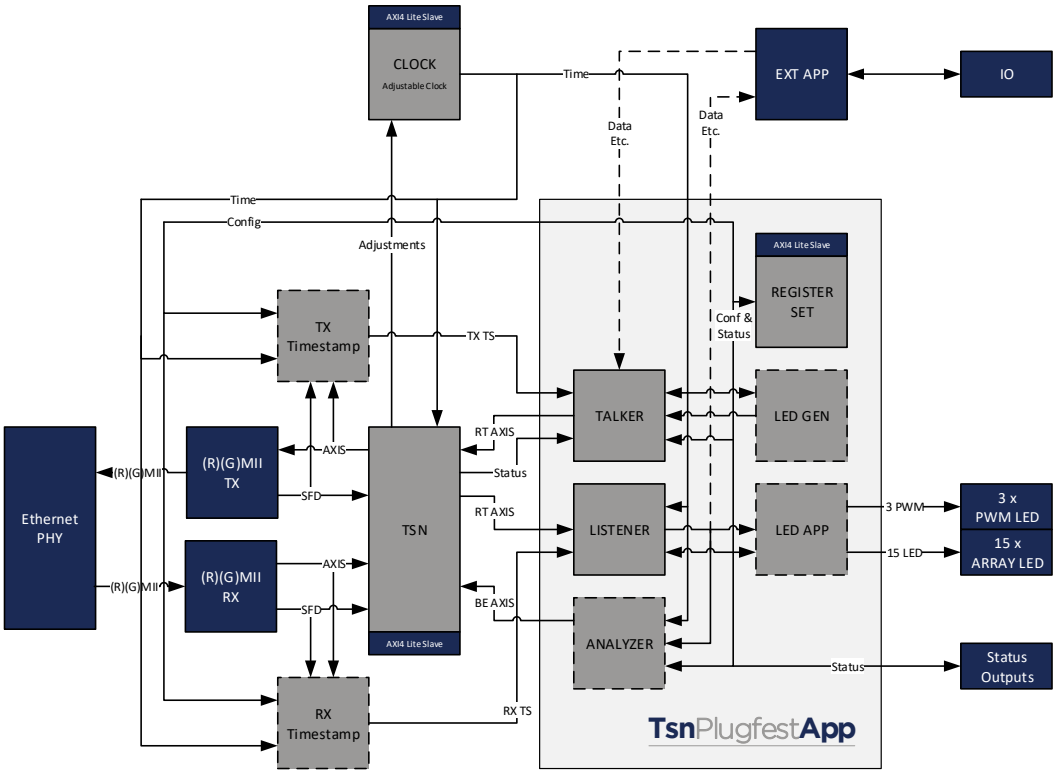

Figure 2: Architecture Block Diagram

The Block diagram shows only one PHY, for multiple PHYs there are also Timestamp Mergers which allow to get the timestamps from multiple instances.

Register Set

This block allows reading status values and writing configuration.

RX and TX Timestampers

These blocks take timestamps of the OPC/UA Real-time frames as close as possible to the PHY. They get the Data and the Start of Frame Delimiter (SFD) of the (R)(G)MII Interface Adapters which are part of the TSN core (but separated to get access to these signals). The block stores a timestamp based on the Talker Identifier and the Frame Sequence Number. The Talker and Listener can then request the timestamp for this specific frame.

Talker

This block sends periodic OPC/UA Real-time frames with definable interval, offset, delay etc. The payload, which can come from Registers, a LED Pattern Generator or a Custom External Application is encapsulated into the frame without the need of knowledge of the frame.
It also takes a timestamp when the frame was actually sent onto the cable and will put the sending timestamp into the next frame.

Listener

This block receives the OPC/UA Real-time frames, parses it and extracts meta data as well as associating a receive timestamp when the frame was actually received on the cable. The data and meta information is then provided to the LED PWM Application and to the Custom External Application which can decide to use the Data or not and also to the Analyzer for further analysis.

Analyzer

This block receives the data and meta information from the listener, analyzes it and creates statistics. The statics result is then periodically sent via a UDP/IP OPC/UA Best-effort frames so a centralized node can collect also statistics from all other nodes to get a picture of the Network.

LED Generator

This block is an LED Pattern Generator which generates a brightness pattern for 3 LED as a percentage between 0 and 100. It will create a pattern with slowly overlapping and fading LEDs.

LED PWM Application

This block converts the received LED brightness into a PWM signal which is fed to 3 LEDs. In addition, it also provides a 15bit Binary array representing the light curtain value.

2 TSN Plugfest App Basics

The IIC TSN Plugfest Application is built of several components which shall allow tests and measurements as well as data transfer as demo for the demo wall.
It can be also used to transfer custom data.

The specifications are available under the IIC terms. If you are not a Member of the IIC you can get one for free to access the documents by signing some NDAs.

The “IIC Interoperability Application v0.3” specification can be found here:
https://engage.iiconsortium.org/wg/TBTSN/document/11502
The “LED Strip Demo Application v0.3” specification can be found here:
https://engage.iiconsortium.org/wg/TBTSN/document/11373

2.1 Concept

The application is made of several sub-components (not all shown and not all implemented)

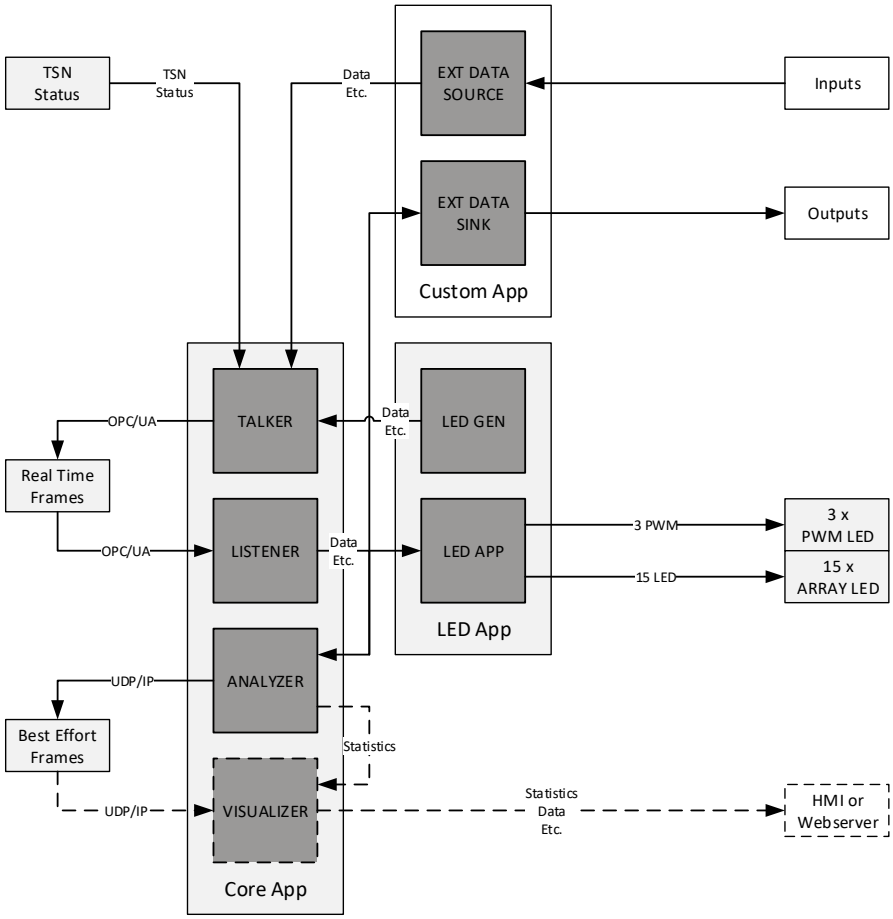

Figure 3: Concept

2.1.1 Core Application

The Core part of the IIC TSN Plugfest Application is formed from the following sub-components:

- Talker
- Listener
- Analyzer (Analysis and Result Publication)
- Visualizer

2.1.1.1 Talker

Transmit periodically OPC/UA frames at a specific offset (also as information in the frame) as Layer2 Multicast frames. The frames contain information about the TSN and Talker (802.1AS, Hardware Timestamping, Data Valid) status, a Talker ID, the vendor and device names, as well as sequence numbers and timestamps of the application, of each sent frame and a precise timestamp when the last frame was sent. In addition, it contains 32byte of data and an 8bit application identifier to determine what kind of data it is. The data can be provided either from registers, a LED Generator or any custom Application which wants to have the data encapsulated in this OPC/UA frame.
The idea is that every node sends these frames, and every node receives the frames from all other nodes, so it can do statistics.

2.1.1.2 Listener

Receives the frames from all other Talkers and associates the frame with a precise timestamp when the frame was received.
It parses the frame and extracts all meta data which it then provides to the other sub-components together with the receive timestamp.

2.1.1.3 Analyzer

This module makes statistics over the received Talker frames. It calculates the delay through the network, the send and receive offsets in the interval and for each value the min, max and current value. In addition, it checks if frames were missed or if an application updated was missed. All this is done on a per Talker base.
The results of these statistic are then sent periodically (1/s) via an UDP/IP Multicast frame (also OPC/UA UADP) to all other nodes, so each node gets the statistics of each other node. Since the maximum size of an Ethernet frame is limited to 1500bytes only 23 nodes can be sent with one frame. (this might change in the future). It shall also provide the local statistics to the Visualizer, if implemented.

2.1.1.4 Visualizer

This module shall receive the statistics of all other nodes and shall visualize them in some way (HMI, Webserver, ...). This is not defined yet.

2.1.2 LED Application

The IIC defined a specific Application which was implemented by most vendors to run a demo. The Demo consists of 3 LEDs which are adjusted in their brightness via a PWM generator and a 15 LED array which shows the state of the light curtain.
The source for the brightness and light curtain is a Sick device.
The Application Identifier for the LED Application is 0x01.
The first 3 bytes of the data payload represent the brightness of the 3 LEDs as a percentage (0-100). The next two bytes form a 16bit value which represent the light curtain of the Sick device. Only the lower 15bits are used since the Sick device has only 15 light beams.
The other Data bytes are don't cares.

The LED part is formed out of the following two sub-components:

- LED Gen
- LED App

2.1.2.1 LED Gen

This is a LED pattern generator which feeds the Talker with the Application Identifier, the actual Data, an Application Timestamp and an Application Sequence Number. What kind of pattern shall be generated and also the refresh rate is up to the implementer (our implementation is described later).

2.1.2.2 LED App

This is a PWM generator for the brightness of the LEDs. It receives the frame information from the Listener and checks if the Application Identifier is correct and optionally also if the Talker Identification matches a configured value. Whenever a new frame is received which matches the criteria it will calculate a pulse width value for the PWM per LED. The PWM period and also the refresh rate is up to the implementer.

2.1.3 Custom Application

A Custom Application is basically the same as the LED Application. It shall have a data sink connected to the Listener and a data source connected to the Talker.

There are many unused Application identifiers which can be used to send custom data with the same frame format.

3 Register Set

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

3.1 Register Overview

Registerset Overview			
Name	Description	Offset	Access
Tsn PlugfestAppControl Reg	Tsn PlugfestApp Control Register	0x00000000	RW
Tsn PlugfestAppStatus Reg	Tsn PlugfestApp Error Status Register	0x00000004	WC
Tsn PlugfestAppVersion Reg	Tsn PlugfestApp Version Register	0x0000000C	RO
Tsn PlugfestAppTalkerControl Reg	Tsn PlugfestApp Talker Control Register	0x00000100	RW
Tsn PlugfestAppTalkerMac1 Reg	Tsn PlugfestApp Talker Mac 1 Register	0x00000104	RW
Tsn PlugfestAppTalkerMac2 Reg	Tsn PlugfestApp Talker Mac 2 Register	0x00000108	RW
Tsn PlugfestAppTalkerId Reg	Tsn PlugfestApp Talker Id Register	0x0000010C	RW
Tsn PlugfestAppTalkerDelay Reg	Tsn PlugfestApp Talker Delay Register	0x00000110	RW
Tsn PlugfestAppTalkerOffset Reg	Tsn PlugfestApp Talker Offset Register	0x00000114	RW
Tsn PlugfestAppTalkerInterval Reg	Tsn PlugfestApp Talker Interval Register	0x00000118	RW
Tsn PlugfestAppTalkerVlan Reg	Tsn PlugfestApp Talker Configuration VLAN Register	0x0000011C	RW
Tsn PlugfestAppTalkerAppId Reg	Tsn PlugfestApp Talker Application Identifier Register	0x0000014C	RW
Tsn PlugfestAppTalkerAppData0to3 Reg	Tsn PlugfestApp Talker Application Data Bytes 0 to 3 Register	0x00000150	RW
Tsn PlugfestAppTalkerAppData4to7 Reg	Tsn PlugfestApp Talker Application Data Bytes 4 to 7 Register	0x00000154	RW

Tsn PlugfestAppTalkerAppData8to11 Reg	Tsn PlugfestApp Talker Application Data Bytes 8 to 11 Register	0x00000158	RW
Tsn PlugfestAppTalkerAppData12to15 Reg	Tsn PlugfestApp Talker Application Data Bytes 12 to 15 Register	0x0000015C	RW
Tsn PlugfestAppTalkerAppData16to19 Reg	Tsn PlugfestApp Talker Application Data Bytes 16 to 19 Register	0x00000160	RW
Tsn PlugfestAppTalkerAppData20to23 Reg	Tsn PlugfestApp Talker Application Data Bytes 20 to 23 Register	0x00000164	RW
Tsn PlugfestAppTalkerAppData24to27 Reg	Tsn PlugfestApp Talker Application Data Bytes 24 to 27 Register	0x00000168	RW
Tsn PlugfestAppTalkerAppData28to31 Reg	Tsn PlugfestApp Talker Application Data Bytes 28 to 31 Register	0x0000016C	RW
Tsn PlugfestAppListenerControl Reg	Tsn PlugfestApp Listener Control Register	0x00000200	RW
Tsn PlugfestAppListenerDelay Reg	Tsn PlugfestApp Listener Configuration Delay Register	0x00000210	RW
Tsn PlugfestAppListenerVlan Reg	Tsn PlugfestApp Listener Configuration VLAN Register	0x0000021C	RW
Tsn PlugfestAppAnalyzerControl Reg	Tsn PlugfestApp Analyzer Control Register	0x00000300	RW
Tsn PlugfestAppAnalyzerIp Reg	Tsn PlugfestApp Analyzer Configuration Source Ip Register	0x00000304	RW
Tsn PlugfestAppAnalyzerUdpPort Reg	Tsn PlugfestApp Analyzer Configuration UDP Source Port Register	0x00000308	RW
Tsn PlugfestAppAnalyzerVlan Reg	Tsn PlugfestApp Analyzer Configuration VLAN Register	0x0000031C	RW
Tsn PlugfestAppAppLedControl Reg	Tsn PlugfestApp AppLed Control Register	0x00001000	RW
Tsn PlugfestAppAppLedId Reg	Tsn PlugfestApp AppLed Configuration Talker Identifier Register	0x0000100C	RW
Tsn PlugfestAppAppLedVlan Reg	Tsn PlugfestApp AppLed Configuration VLAN Register	0x0000101C	RW

Table 4: Register Set Overview

3.2 Register Descriptions

3.2.1 General

3.2.1.1 TSN Plugfest App Control Register

Used for general control over the TSN Plugfest App core, all configurations on the core shall only be done when disabled.
The Enable acts as general enable and disable for all other components, meaning that only when this is enabled the other enabled sub-cores will be enabled as well, the same applies for disabling, it disables al other sub-cores. The APP_EXT bit signals if the external Application interface shall be used as Data source.

Tsn PlugfestAppControl 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
-															APP_EXT	-															ENABLE
RO															R W	RO															R W
Reset: 0x00000000																															
Offset: 0x0000																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:17	RO
APP_EXT	Use External Application interface (else LedGen)	Bit: 16	RW
-	Reserved, read 0	Bit: 15:1	RO
ENABLE	Enable	Bit: 0	RW

3.2.1.2 Tsn Plugfest App Error Status Register

Shows the current status of the TSN Plugfest App. Each sub-core has its own error bit, all error bits are sticky and must be cleared by writing a 1 to the corresponding bit

Tsn PlugfestAppStatus 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
-												APPLED_ERROR	ANALYZER_ERROR	LISTENER_ERROR	TALKER_ERROR	-															ERROR
RO												WC	WC	WC	WC	RO															WC
Reset: 0x00000000																															
Offset: 0x0004																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:20	RO
APPLED_ERROR	Application LED Error (sticky)	Bit: 19	WC
ANALYZER_ERROR	Analyzer Error (sticky)	Bit: 18	WC
LISTENER_ERROR	Listener Error (sticky)	Bit: 17	WC
TALKER_ERROR	Talker Error (sticky)	Bit: 16	WC
-	Reserved, read 0	Bit: 15:1	RO

ERROR	Error (sticky)	Bit: 0	WC

3.2.1.3 TSN Plugfest App Version Register

BLABAL

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

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

3.2.2 Talker

3.2.2.1 TSN Plugfest App Talker Control Register

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

Tsn PlugfestAppTalkerControl 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	APP_REG	-														RESET_STAT	ENABLE
RO														RW	RW	RO														RW	RW
Reset: 0x00000000																															
Offset: 0x0100																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:18	RO
VLAN	Send VLAN as specified in Reg (else 0xABB8)	Bit: 17	RW
APP_REG	Use Register values as Application data (else LedGen or Ext)	Bit: 16	RW
-	Reserved, read 0	Bit: 15:2	RO
REST_STAT	Set the Reset Statistics bit (must be cleared again)	Bit: 1	RW

ENABLE	Enable	Bit: 0	RW

3.2.2.2 TSN Plugfest App Talker MAC 1 Register

Source MAC of the Talker frames.

Tsn PlugfestAppTalkerMac1 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)	Source MAC Byte 3	Bit:31:24	RW
MAC(2)	Source MAC Byte 2	Bit:23:16	RW
MAC(1)	Source MAC Byte 1	Bit:15:8	RW
MAC(0)	Source MAC Byte 0	Bit:7:0	RW

3.2.2.3TSN Plugfest App Talker MAC 2Register

Source MAC of the Talker frames.

Tsn PlugfestAppTalkerMac2 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)	Source MAC Byte 5	Bit:15:8	RW
MAC(4)	Source MAC Byte 4	Bit:7:0	RW

3.2.2.4 TSN Plugfest App Talker Id Register

Talker Id of the Talker frames. Based on this, the Destination MAC (lower 2 bytes) is calculated.

Tsn PlugfestAppTalkerId 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
-																TALKER_ID															
RO																RW															
Reset: 0x00000000																															
Offset: 0x010C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
TALKER_ID	Talker ID (use only 0-255)	Bit:15:0	RW

3.2.2.5 TSN Plugfest App Talker Delay Register

Delay between the core and the actual frame being sent by the PHY. This delay is taken into account to start sending of the frame early enough so it reaches the output port at the right time.

Tsn PlugfestAppTalkerDelay 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																															
RW																															
Reset: 0x00000000																															
Offset: 0x0110																															

Name	Description	Bits	Access
DELAY	TX Delay in Nanoseconds, used to send the frame early enough that it will be sent at the right moment (at OFF-SET), also needed for internal timestamp compensation when used	Bit: 31:0	RW

3.2.2.6TSN Plugfest App Talker Offset Register

Desired offset within a second when the frame sending shall start.

Tsn PlugfestAppTalkerOffset 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																															
RW																															
Reset: 0x00000000																															
Offset: 0x0114																															

Name	Description	Bits	Access
OFFSET	TX Offset from beginning of Second in Nanoseconds	Bit: 31:0	RW

3.2.2.7 TSN Plugfest App Talker Interval Register

Sending interval of the Talker frames..

Tsn PlugfestAppTalkerInterval 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																															
RW																															
Reset: 0x00000000																															
Offset: 0x0118																															

Name	Description	Bits	Access
INTERVAL	TX Interval in Nanoseconds	Bit: 31:0	RW

3.2.2.8TSN Plugfest App Talker VLAN Register

If VLAN is not enabled it will take the default VLAN tag defined in the specification (0xABB8), othwise it takes the VLAN from this register.

Tsn PlugfestAppTalkerVlan 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															
RO																RW															
Reset: 0x00000000																															
Offset: 0x011C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
VLAN	VLAN Tag to send if enabled	Bit:15:0	RW

3.2.2.9TSN Plugfest App Talker App IdRegister

Application ID of the Talker frames, only used when the Register mode is used

Tsn PlugfestAppTalkerAppId 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
-																								APP_ID							
RO																								RW							
Reset: 0x00000000																															
Offset: 0x014C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:8	RO
APP_ID	Application Identifier (0: undefined, 1: LedStrip, 2: Motion control, 3: Motion listener, 4-255 reserved)	Bit:7:0	RW

3.2.2.10 TSN Plugfest App Talker Data Registers

Data Bytes of the Talker frames. The Talker will always send 32 Bytes of Data.

Tsn PlugfestAppTalkerData[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
DATA(N)								DATA(N+1)								DATA(N+2)								DATA(N+3)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0150 - 0x16C																															

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

3.2.3 Listener

3.2.3.1 TSN Plugfest App Listener Control Register

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

Tsn PlugfestAppListenerControl 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
-													IGNORE_VLAN	VLAN	-															ENABLE
RO													RW	RW	RO															RW	
Reset: 0x00000000																															
Offset: 0x0200																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:19	RO
IGNORE_VLAN	Make no VLAN check at all (overrules VLAN)	Bit: 18	RW
VLAN	Check VLAN against specified in Reg (else 0xABB8)	Bit: 17	RW
-	Reserved, read 0	Bit: 16:1	RO
ENABLE	Enable	Bit: 0	RW

3.2.3.2TSN Plugfest App Listener Delay Register

Delay from the Input Port to the PHY to the core. This is only required for internal timestamping.

Tsn PlugfestAppListenerDelay 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																															
RW																															
Reset: 0x00000000																															
Offset: 0x0210																															

Name	Description	Bits	Access
DELAY	RX Delay in Nanoseconds, used needed for internal timestamp compensation when used	Bit: 31:0	RW

3.2.3.3TSN Plugfest App Listener VLAN Register

VLAN to check if VLAN check is enabled and not the default one shall be used or VLAN check shall be ignored.

Tsn PlugfestAppListenerVlan 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															
RO																RW															
Reset: 0x00000000																															
Offset: 0x021C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
VLAN	VLAN Tag to check against if enabled	Bit:15:0	RW

3.2.4 Analyzer

3.2.4.1 TSN Plugfest App Analyzer Control Register

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

Tsn PlugfestAppAnalyzerControl 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
-													IGNORE_VLAN	VLAN	-															ENABLE
RO													RW	RW	RO															RW	
Reset: 0x00000000																															
Offset: 0x0300																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:19	RO
IGNORE_VLAN	Make no VLAN check at all (overrules VLAN)	Bit: 18	RW
VLAN	Check VLAN against specified in Reg (else 0xABB8)	Bit: 17	RW
-	Reserved, read 0	Bit: 16:1	RO
ENABLE	Enable	Bit: 0	RW

3.2.4.2 TSN Plugfest App Analyzer IP Register

Source IP of the Analyzer frames.

Tsn PlugfestAppAnalyzerIp 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: 0x0304																															

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

3.2.4.3 TSN Plugfest App Analyzer UDP Port Register

Source UDP Port of the Analyzer frames.

Tsn PlugfestAppAnaylzerUdpPort 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
																UDP_PORT															
RO																RW															
Reset: 0x00000000																															
Offset: 0x0308																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
UDP_PORT	Source UDP Port	Bit:15:0	RW

3.2.4.4 TSN Plugfest App Analyzer VLAN Register

VLAN to check for the Talker frames received.

Tsn PlugfestAppAnalyzerVlan 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															
RO																RW															
Reset: 0x00000000																															
Offset: 0x031C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
VLAN	VLAN Tag to check against if enabled	Bit:15:0	RW

3.2.5 LED App

3.2.5.1 TSN Plugfest App LED App Control Register

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

Tsn PlugfestAppAppLedControl 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
-													IGNORE_VLAN	VLAN	-															ENABLE
RO													RW	RW	RO															RW	
Reset: 0x00000000																															
Offset: 0x1000																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit: 31:19	RO
IGNORE_VLAN	Make no VLAN check at all (overrules VLAN)	Bit: 18	RW
VLAN	Check VLAN against specified in Reg (else 0xABB8)	Bit: 17	RW
-	Reserved, read 0	Bit: 16:1	RO
ENABLE	Enable	Bit: 0	RW

3.2.5.2TSN Plugfest App LED App Talker Id Register

Talker ID for which the LED App shall extract the Data.

Tsn PlugfestAppAppLedId 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
-																TALKER_ID															
RO																RW															
Reset: 0x00000000																															
Offset: 0x100C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
TALKER_ID	Talker ID to listen to (0xFFFF means all)	Bit:15:0	RW

3.2.5.3TSN Plugfest App LED App VLAN Register

VLAN to check if VLAN check is enabled and not the default one shall be used or VLAN check shall be ignored.

Tsn PlugfestAppAppLedVlan 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															
RO																RW															
Reset: 0x00000000																															
Offset: 0x101C																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:16	RO
VLAN	VLAN Tag to check against if enabled	Bit:15:0	RW

4 Design Description

The following chapters describe the internals of the TSN Plugfest App: starting with the Top Level, which is a collection of subcores, followed by the description of all subcores.

4.1 Interface Definition for a Custom Application

To access the TSN Plugfest App from the outside an interface is defined:
A Source and a Sink Interface.

4.1.1 Custom Application Data Source Interface

The Data Source interface consists of three basic parts:

- Clock
- Valid/New
- Data

The Clock is the same as the System Clock of the Core

The Valid/New signal is a single clock cycle signal which is indicating when Data is valid (Event Mode) or when new Data is provided (Buffered Mode). For a detailed description of the modes see 4.1.3.

The Data is encapsulated in a record called “Tsn_AppExtSource_Type” together with Meta Data. The record is defined in “Tsn_Package.vhd” of library TsnLib

Field Name	Type	Size	Description
AppId	std_logic_vector	8	Application Identifier which type of Data shall be transmit-ted to determine for which Application it is intended
AppData	Common_Byte_ Type	32	Application Data as Byte array. Byte 0 is the first to transmit and Byte 31 the last to transmit
AppDataLength	std_logic_vector	8	IGNORED always 32 How many Bytes of the trans-mit Data are valid, the others shall be ignored and shall be

			not transmitted. This shall always be 32 for the TSN Plugfest App (however it might be, that in the future this is a dynamic value, therefore we define this already)
AppDataValid	std_logic	1	Indication if the Application Data is valid or not
AppSequenceNr	std_logic_vector	16	Application Sequence number is incremented by 1 for every time the Valid signal is asserted
AppTimestamp	std_logic_vector	64	Application Timestamp in TAI Nanoseconds, when the Application Data was updated (e.g. sampling time of an ADC, therefore this might be earlier then the assertion of the Valid signal). Must change for every time the Valid signal is asserted

Table 5: Tsn_AppExtSource_Type

4.1.2 Custom Application Data Sink Interface

The Data Sink interface also consists of three basic parts:

- Clock
- Valid/New
- Data

The Clock is the same as the System Clock of the Core

The Valid/New signal is a single clock cycle signal which is indicating when Data is valid (Event Mode) or when new Data has arrived (Buffered Mode). For a detailed description of the modes see 4.1.3.

The Data is encapsulated in a record called “Tsn_AppExtSink_Type” together with Meta Data. The record is defined in “Tsn_Package.vhd” of library TsnLib

Field Name	Type	Size	Description
AppId	std_logic_vector	8	Application Identifier which type of Data was received to determine for which Application it is intended
AppData	Common_Byte_Type	32	Application Data as Byte array. Byte 0 is the first received and Byte 31 the last received
AppDataLength	std_logic_vector	8	How many Bytes of the received Data are valid, the others shall be ignored. This shall be 32 for the TSN Plugfest App (however it might be that some other Talker uses a different amount of Data (which is not according to spec))
AppDataValid	std_logic	1	Indication if the Application Data is valid or not
AppSequenceNr	std_logic_vector	16	Application Sequence number is incremented by 1 by the Node who sent the frame when it updates (also with same Data) the Application Data (this might not change every time a frame is received because Data was maybe not updated)
AppTimestamp	std_logic_vector	64	Application Timestamp in TAI Nanoseconds, when the Node who sent the frame last updated (also with same Data) the Application Data (this might not change every time a frame is received because Data was maybe not updated)
AppTalkerId	std_logic_vector	16	Talker Identifier of the Node

			who sent the frame, can be used to filter Data for specific a specific Talker (with the current version of the TSN Plugfest App, only Ids 0-255 are supported)

Table 6: Tsn_AppExtSink_Type

4.1.3 Interface Modes

There are two modes how the interface can be used:

- Event Mode
- Buffered Mode

4.1.3.1 Event Mode

In this mode, Data is only valid during that one Clock cycle when Valid is asserted. This means the Data Sink must process the Data at the moment when Valid is asserted. When Valid is not asserted Data also has to be considered invalid and must not be used.
As a Data Source the Data can be changed until it asserts the Valid signal to mark that this Data shall be used. When it resets Valid it can change Data again.

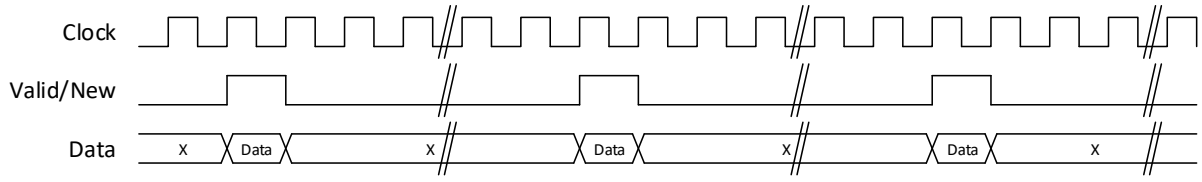

Figure 4: Ext App Interface Event Mode

4.1.3.2 Buffered Mode

In this mode, Data is valid all the time and the Valid signal acts as an indication that new Data has arrived.
This means the Data Sink can start processing the Data at the moment when Valid is asserted, which indicates that new Data is there. And it can use the Data until the Valid signal is asserted again (new Data). When Valid is not asserted Data is still valid. After Reset until first ready Data, there is an indication in the Data itself that Data is not valid yet.

As a Data Source the Data can only be changed together with the assertion of the Valid signal to mark that new Data is there. When it resets Valid it must not change Data.

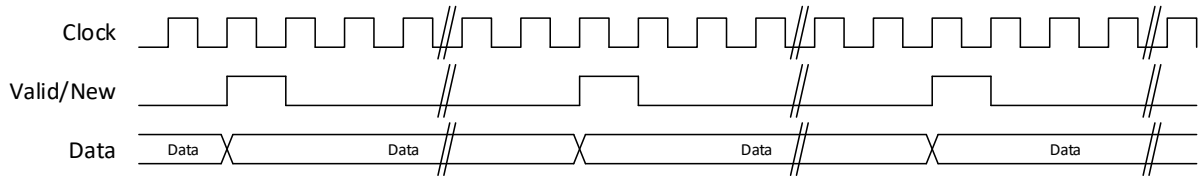

Figure 5: Ext App Interface Buffered Mode

4.2 Top Level – TSN Plugfest App

4.2.1.1 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
ClockClkPeriod Nanosecond_Gen	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
TalkerSupport_Gen	boolean	1	If the Talker shall be instantiated
ListenerSupport_Gen	boolean	1	If the Listener shall be instantiated
AnalyzerSupport_Gen	boolean	1	If the Analyzer shall be instantiated
AppLedSupport_Gen	boolean	1	If the LED Application (PWM brightness generator) shall be instantiated
AppLedGen Support_Gen	boolean	1	If the LED Pattern Generator shall be instantiated
AppRegSupport_Gen	boolean	1	If the Talker Data can also come from Register Values
AppExtSource Support_Gen	boolean	1	If an external Application Source shall be connected (to the Talker)
AppExtSink	boolean	1	If an external Application Sink

Support_Gen			shall be connected (to the Listener)
AppExtBuffered Mode_Gen	boolean	1	If the External Application interface shall run in Buffered Mode or Evenet Mode: true = Buffered, false = Event
LocalTimestamp_Gen	boolean	1	DO NOT USE If timestamps shall be taken internally in the core, this shall only be used when the exter-nal Timestampers are not used or e.g. no TSN core is in place.
TalkerTimeout Nanosecond_Gen	natural	1	How long the Talker shall wait in Nanoseconds to get a timestamp for the frame (this can take a while until the frame is processed by the TSN core, shall be less than the interval time)
ListenerTimeout Nanosecond_Gen	natural	1	How long the Listener shall wait in Nanoseconds to get a timestamp for the frame (can be short since timestamp shall be taken already)
VendorName_Gen	string	32	Vendor Name as ASCII String (must be filled with spaces)
DeviceName_Gen	string	10	Device Name as ASCII String (must be filled with spaces)
AxiAddress RangeLow_Gen	std_logic_vector	32	AXI Base Address
AxiAddress RangeHigh_Gen	std_logic_vector	32	AXI Base Address plus Regis-terset Size Default plus 0xFFFF
Sim_Gen	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis

Table 7: Parameters

4.2.1.2 Structured Types

4.2.1.2.1 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 8: Clk_Time_Type

4.2.1.2.2 Tsn_TalkerConfiglic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the configuration vector for the Talker

Field Name	Type	Size	Description
TalkerEnable	std_logic	1	Enable the Talker
TalkerResetStat	std_logic	1	Indicate on Sending that the other Nodes shall reset their statistics for this Node
TalkerVlanReg	std_logic	1	If the VLAN Tag from the Register shall be used or the default one
TalkerVlan	std_logic_vector	16	Talker VLAN Tag
TalkerMac	Common_ Byte_Type	6	Talker Source MAC

TalkerId	std_logic_vector	16	Talker Identifier
TalkerDelay	std_logic_vector	32	Talker Delay (Delay from sending to on the Cable) in Nanoseconds
TalkerOffset	std_logic_vector	32	Talker Offset in Nanoseconds
TalkerInterval	std_logic_vector	32	Talker Interval in Nanoseconds
TalkerAppReg	std_logic	1	If the Talker shall send the Application Data from the Register
TalkerAppId	std_logic_vector	8	The Register Application Identifier to be sent by the Talker
TalkerAppData	Common_ Byte_Type	32	The Register Application Data to be sent by the Talker

Table 9: Tsn_TalkerConfiglic_Type

4.2.1.2.3 Tsn_ListenerConfiglic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the configuration vector for the Listener

Field Name	Type	Size	Description
ListenerEnable	std_logic	1	Enable the Listener
ListenerVlanReg	std_logic	1	If the VLAN Tag from the Register shall be used or the default one
ListenerIgnoreVlan	std_logic	1	Ignore the VLAN
ListenerVlan	std_logic_vector	16	Listener VLAN Tag
ListenerDelay	std_logic_vector	32	Listener Delay (Delay from receiving on the Cable) in Nanoseconds

Table 10: Tsn_ListenerConfiglic_Type

4.2.1.2.4 Tsn_AnalyzerConfiglic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the configuration vector for the Analyzer

Field Name	Type	Size	Description
AnalyzerEnable	std_logic	1	Enable the Analyzer
AnalyzerVlanReg	std_logic	1	If the VLAN Tag from the Register shall be used or the default one
AnalyzerIgnoreVlan	std_logic	1	Ignore the VLAN
AnalyzerVlanReg	std_logic	1	If the VLAN Tag from the Register shall be used or the default one
AnalyzerVlan	std_logic_vector	16	Analyzer VLAN Tag
AnalyzerIp	Common_ Byte_Type	4	Source IP of the Analyzer
AnalyzerUdpPort	Common_ Byte_Type	16	Source UDP Port of the Ana- lyzer

Table 11: Tsn_AnalyzerConfiglic_Type

4.2.1.2.5 Tsn_AppLedConfiglic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the configuration vector for the LED App

Field Name	Type	Size	Description
AppEnable	std_logic	1	Enable the App
AppVlanReg	std_logic	1	If the VLAN Tag from the Register shall be used for the check or the default one
AppIgnoreVlan	std_logic	1	Ignore the VLAN
AppVlan	std_logic_vector	16	App VLAN Tag to check
AppTalkerId	std_logic_vector	16	The Talker Identifier we ant to listen to

Table 12: Tsn_AppLedConfiglic_Type

4.2.1.2.6 Tsn_Configlic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the configuration vector for all sub-cores

Field Name	Type	Size	Description
Enable	std_logic	1	Enable the System
AppExt	std_logic	1	If the external Application shall be used
TalkerConfig	Tsn_Talker Configlic_Type	1	Configuration for the Talker
ListenerConfig	Tsn_Listener Configlic_Type	1	Configuration for the Listener
AnalyzerConfig	Tsn_Analyzer Configlic_Type	1	Configuration for the Analyzer
AppLedConfig	Tsn_AppLed Configlic_Type	1	Configuration for the LED Application

Table 13: Tsn_TalkerConfiglic_Type

4.2.1.2.7 Tsn_TalkerStatuslic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the status vector of the Talker

Field Name	Type	Size	Description
TalkerEnabled	std_logic	1	If the Talker is enabled
TalkerError	std_logic	1	If an error occurred in the Talker
TalkerSendBusy	std_logic	1	If the Talker is busy with sending

Table 14: Tsn_TalkerStatuslic_Type

4.2.1.2.8 Tsn_ListenerStatuslic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the status vector of the Listener

Field Name	Type	Size	Description
ListenerEnabled	std_logic	1	If the Listener is enabled
ListenerError	std_logic	1	If an error occurred in the Listener

ListenerReceiveBusy	std_logic	1	If the Listener is busy with receiving

Table 15: Tsn_ListenerStatuslic_Type

4.2.1.2.9 Tsn_AnalyzerStatuslic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the status vector of the Analyzer

Field Name	Type	Size	Description
AnalyzerEnabled	std_logic	1	If the Analyzer is enabled
AnalyzerError	std_logic	1	If an error occurred in the Analyzer

Table 16: Tsn_AnalyzerStatuslic_Type

4.2.1.2.10 Tsn_AppStatuslic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the status vector of the LED App

Field Name	Type	Size	Description
AppEnabled	std_logic	1	If the App is enabled
AppError	std_logic	1	If an error occurred in the App
AppLed0	std_logic_vector	8	Current brightness in percent for LED 0
AppLed1	std_logic_vector	8	Current brightness in percent for LED 1
AppLed2	std_logic_vector	8	Current brightness in percent for LED 2
AppCurtain	std_logic_vector	15	Current State of the Curtain

Table 17: Tsn_AppStatuslic_Type

4.2.1.2.11 Tsn_Statuslic_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the status vector of all sub-cores

Field Name	Type	Size	Description
Enabled	std_logic	1	If the System is enabled
AppExt	std_logic	1	If the external Application is used
TalkerStatus	Tsn_Talker Statuslic_Type	1	Status of the Talker
ListenerStatus	Tsn_Listener Statuslic_Type	1	Status of the Listener
AnalyzerStatus	Tsn_Analyzer Statuslic_Type	1	Status of the Analyzer
AppLedStatus	Tsn_AppLed Statuslic_Type	1	Status of the LED Application

Table 18: Tsn_TalkerStatuslic_Type

4.2.1.2.12 Tsn_AppExtSource_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the External Interface Source vector

Field Name	Type	Size	Description
AppId	std_logic_vector	8	Application Identifier which type of Data shall be transmitted to determine for which Application it is intended
AppData	Common_Byte_ Type	32	Application Data as Byte array. Byte 0 is the first to transmit and Byte 31 the last to transmit
AppDataLength	std_logic_vector	8	IGNORED always 32 How many Bytes of the transmit Data are valid, the others shall be ignored and shall be not transmitted. This shall always be 32 for the TSN Plugfest App (however it might be, that in the future this is a dynamic value, therefore we define this already)

AppDataValid	std_logic	1	Indication if the Application Data is valid or not
AppSequenceNr	std_logic_vector	16	Application Sequence number is incremented by 1 for every time the Valid signal is asserted
AppTimestamp	std_logic_vector	64	Application Timestamp in TAI Nanoseconds, when the Application Data was updated (e.g. sampling time of an ADC, therefore this might be earlier then the assertion of the Valid signal). Must change for every time the Valid signal is asserted

Table 19: Tsn_AppExtSource_Type

4.2.1.2.13 Tsn_AppExtSink_Type

Defined in Tsn_Package.vhd of library TsnLib
Type represents the External Interface Sink vector

Field Name	Type	Size	Description
AppId	std_logic_vector	8	Application Identifier which type of Data was received to determine for which Application it is intended
AppData	Common_Byte_Type	32	Application Data as Byte array. Byte 0 is the first received and Byte 31 the last received
AppDataLength	std_logic_vector	8	How many Bytes of the received Data are valid, the others shall be ignored. This shall be 32 for the TSN Plugfest App (however it might be that some other Talker uses a different amount

			of Data (which is not according to spec))
AppDataValid	std_logic	1	Indication if the Application Data is valid or not
AppSequenceNr	std_logic_vector	16	Application Sequence number is incremented by 1 by the Node who sent the frame when it updates (also with same Data) the Application Data (this might not change every time a frame is received because Data was maybe not updated)
AppTimestamp	std_logic_vector	64	Application Timestamp in TAI Nanoseconds, when the Node who sent the frame last updated (also with same Data) the Application Data (this might not change every time a frame is received because Data was maybe not updated)
AppTalkerId	std_logic_vector	16	Talker Identifier of the Node who sent the frame, can be used to filter Data for specific a specific Talker (with the current version of the TSN Plugfest App, only Ids 0-255 are supported)

Table 20: Tsn_AppExtSink_Type

4.2.1.2.14 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_	1	Infor about the Cores state

	Type		
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 21: Ptp_TransparentClockStaticConfig_Type

4.2.1.2.15 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 22: Ptp_TransparentClockStaticConfigVal_Type

4.2.1.2.16 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_ NumberPorts	std_logic_vector	16	Number of Ports (1 for OC)
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;		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;		Granmaster Clock Identity
ParentDataset_ Grandmaster ClockQuality	Ptp_Clock Quality_Type;		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;		Port Idenity
PortDataset_ PortState	std_logic_vector	8	Port State
PortDataset_ LogMinDelayReq Interval	std_logic_vector	8	Delay Request Message Inter-val
PortDataset_ PeerMeanPathDelay Valid	std_logic;		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_	std_logic_vector	8	Sync Message Interval

LogSyncInterval			
PortDataset_ LogPtpCapable SignalingInterval	std_logic_vector	8	802.1 Signaling Message Interval
PortDataset_ DelayMechanism	std_logic_vector	8	Delay Mechanism
PortDataset_ LogMinPdelayReq Interval	std_logic_vector	8	Peer Delay Request Message Interval
PortDataset_ VersionNumber	std_logic_vector	4	PTP Version
PortDataset_ Asymmetry	std_logic_vector	32	Port Asymmetry

Table 23: Ptp_OrdinaryClockStaticConfig_Type

4.2.1.2.17 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 24: Ptp_OrdinaryClockStaticConfigVal_Type

4.2.1.2.18 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	std_logic	1	No supervision frames re-

PortA			ceived 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 25: Red_TsnStaticStatus_Type

4.2.1.2.19 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 26: Red_TsnStaticStatusVal_Type

4.2.1.3 Entity Block Diagram

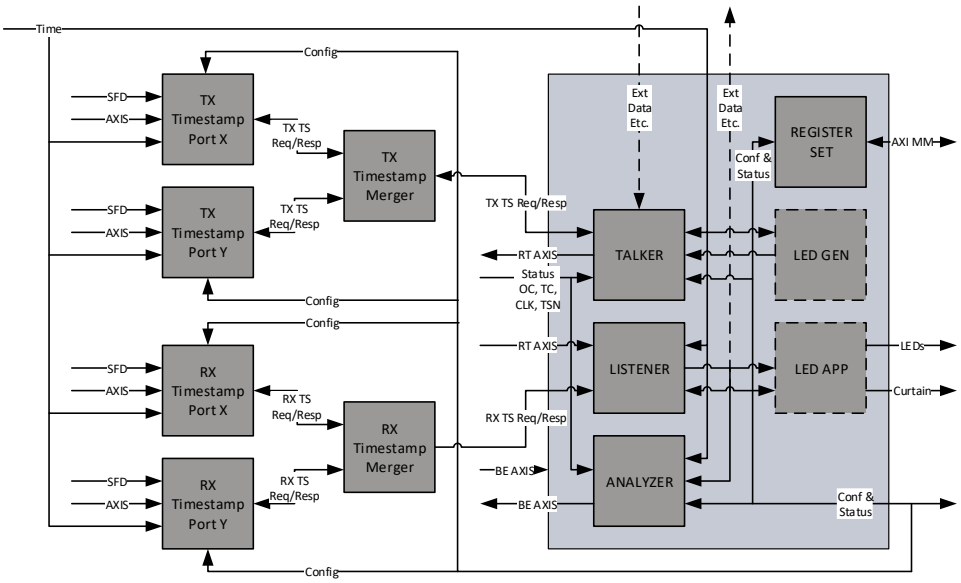

Figure 6: TSN Plugfest App

4.2.1.4 Entity Description

Talker

This module handles the timely and repeated sending of OPC/UA Real-time frames. It encodes Data from either the LED Pattern Generator, the External Data Source or Registers into the frames, increments counters and waits and inserts Timestamps. See Fehler! Verweisquelle konnte nicht gefunden werden. for more details.

Listener

This module handles the receiving of OPC/UA Real-time frames. It parses Data from the frames and waits and provides Timestamps together with Meta Data for further Processing by the LED App, the External Data Sink and Analyzer.
See 4.3.2 for more details.

Analyzer

This module calculates statistics over the received OPC/UA Real-time frames of all other nodes and provides this information by periodically sending an OPC/UA Best-effort frame.
See 4.3.3 for more details.

App LED

This module converts the received brightness values from the Listener if it is LED Application Data into 3 PWM signals for 3 LEDs. Additionally, it provides the raw Light Curtain value.
The LED App can listen to a specific Talker or any Talker.
See 4.3.4 for more details.

App LED Gen

This module creates an LED brightness Pattern for 3 LEDs. It is the counterpart to the App LED module. In this core it creates a slowly fading overlapped LED pattern. The Data it creates is feed to the Talker for sending.
See 4.3.5 for more details.

Timestamper

This module takes timestamps based on the SFD signals and merges it with Meta Data from the frames and stores them in RAM so the Talker or Listener can request a specific timestamp later (it indicates when the timestamp is there). For this it parses the Frames and extracts some Mata Data from the OPC/UA Real-time frames. Timestamping is only done for OPC/UA Real-time frames, on RX side for all OPC/UA Real-time frames on the TX side only for frames sourced by the local Talker.
See 4.3.3 for more details.

Timestamp Merger

This module allows to connect multiple Timestampers to the TSN Plugfest App. This is required where the TSN Plugfest App is talking to TSN core with multiple ports.
See 4.3.3 for more details.

Registerset

This module is an AXI4Lite Memory Mapped Slave. It provides access to all registers and allows configuring the TSN Plugfest App. An AXI Master has to configure the registers with AXI writes to the registers, which is typically done by a CPU
See 4.3.8 for more details.

4.2.1.5 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
TalkerSupport_Gen	-	boolean	1	If the Talker shall be instantiated
ListenerSupport_Gen	-	boolean	1	If the Listener shall be instantiated
AnalyzerSupport_Gen	-	boolean	1	If the Analyzer shall be instantiated
AppLedSupport_Gen	-	boolean	1	If the LED Application (PWM brightness generator) shall be instantiated
AppLedGen Support_Gen	-	boolean	1	If the LED Pattern Generator shall be instantiated
AppRegSupport_Gen	-	boolean	1	If the Talker Data can also come from Register Values
AppExtSource Support_Gen	-	boolean	1	If an external Application Source shall be connected (to the Talker)
AppExtSink Support_Gen	-	boolean	1	If an external Application Sink shall be connected (to the Listener)
AppExtBuffered Mode_Gen	-	boolean	1	If the External Application interface shall run in Buffered Mode or Evenet Mode: true =

				Buffered, false = Event
LocalTimestamp_Gen	-	boolean	1	DO NOT USE If timestamps shall be taken internally in the core, this shall only be used when the external Timestampers are not used or e.g. no TSN core is in place.
TalkerTimeout Nanosecond_Gen	-	natural	1	How long the Talker shall wait in Nano-seconds to get a timestamp for the frame (this can take a while until the frame is processed by the TSN core, shall be less than the interval time)
ListenerTimeout Nanosecond_Gen	-	natural	1	How long the Lis-tener shall wait in Nanoseconds to get a timestamp for the frame (can be short since timestamp shall be taken al-ready)
VendorName_Gen	-	string	32	Vendor Name as ASCII String (must be filled with spac-es)
DeviceName_Gen	-	string	10	Device Name as ASCII String (must be filled with spac-es)
AxiAddress	-	std_logic_vector	32	AXI Base Address

RangeLow_Gen				
AxiAddress RangeHigh_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
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
LED App Output				
Curtain_DatOut	out	std_logic_vector	15	Adjusted PTP Clock Time
Led_DatOut	out	std_logic_vector	3	Adjusted PTP Clock Time valid
Config				
Config_DatOut	out	Tsn_ConfigIIC_Type	1	Static Configuration
Status				
Status_DatOut	out	Tsn_StatusIIC_Type	1	Static Status
Status Inputs				
StaticStatusClk_DatIn	in	Clk_ClockStatic Status_Type	1	Static Status Data of the Clock Core
StaticStatusClk_ValIn	in	Clk_ClockStatic StatusVal_Type	1	Static Status Valid of the Clock Core
StaticStatusTsn_DatIn	in	Red_TsnStatic Status_Type;	1	Static Status Data of the TSN Core
StaticStatusTsn_ValIn	in	Red_TsnStatic StatusVal_Type;	1	Static Status Valid of the TSN Core

StaticStatus PtpOc_DatIn	in	Ptp_OrdinaryClock StaticStatus_Type;	1	Static Status Data of the PTP OC Core
StaticStatus PtpOc_ValIn	in	Ptp_OrdinaryClock StaticStatusVal _Type;	1	Static Status Valid of the PTP OC Core
StaticStatus PtpTc_DatIn	in	Ptp_Transparent ClockStatic Status_Type;	1	Static Status Data of the PTP TC Core
StaticStatus PtpTc_ValIn	in	Ptp_Transparent ClockStatic StatusVal_Type;	1	Static Status Valid of the PTP TC Core
RX Timestamp				
RxTimestamp_DatIn	in	Clk_Time_Type	1	Timestamp Value
RxTimestamp Ok_ValIn	in	std_logic	1	Timestamp Ok (found)
RxTimestamp Resp_ValIn	in	std_logic	1	Timestamp searched
RxTimestamp Info_DatOut	out	std_logic_vector	32	Timestamp Info to find the correct one
RxTimestamp Req_ValOut	out	std_logic	1	Search Timestamp
Axi Real Time Input				
AxisRtValid_ValIn	in	std_logic	1	AXI Stream Real Time frame input
AxisRtReady_ValOut	out	std_logic	1	
AxisRtData_DatIn	in	std_logic_vector	32	
AxisRtStrobe_ValIn	in	std_logic_vector	4	
AxisRtKeep_ValIn	in	std_logic_vector	4	
AxisRtLast_ValIn	in	std_logic	1	
AxisRtUser_DatIn	in	std_logic_vector	3	
TX Timestamp				
TxTimestamp_DatIn	in	Clk_Time_Type	1	Timestamp Value
TxTimestamp Ok_ValIn	in	std_logic	1	Timestamp Ok (found)
TxTimestamp Resp_ValIn	in	std_logic	1	Timestamp searched
TxTimestamp Info_DatOut	out	std_logic_vector	32	Timestamp Info to find the correct one
TxTimestamp Req_ValOut	out	std_logic	1	Search Timestamp
Axi Real Time Output				

AxisRtValid_ValOut	out	std_logic	1	AXI Stream Real Time frame output
AxisRtReady_ValIn	in	std_logic	1	
AxisRtData_DatOut	out	std_logic_vector	32	
AxisRtStrobe_ValOut	out	std_logic_vector	4	
AxisRtKeep_ValOut	out	std_logic_vector	4	
AxisRtLast_ValOut	out	std_logic	1	
AxisRtUser_DatOut	out	std_logic_vector	3	
Axi Best Effort Input				
AxisBeValid_ValIn	in	std_logic	1	AXI Stream Best Effort frame input
AxisBeReady_ValOut	out	std_logic	1	
AxisBeData_DatIn	in	std_logic_vector	32	
AxisBeStrobe_ValIn	in	std_logic_vector	4	
AxisBeKeep_ValIn	in	std_logic_vector	4	
AxisBeLast_ValIn	in	std_logic	1	
AxisBeUser_DatIn	in	std_logic_vector	3	
Axi Best EffortOutput				
AxisBeValid_ValOut	out	std_logic	1	AXI Stream Best Effort frame output
AxisBeReady_ValIn	in	std_logic	1	
AxisBeData_DatOut	out	std_logic_vector	32	
AxisBeStrobe_ValOut	out	std_logic_vector	4	
AxisBeKeep_ValOut	out	std_logic_vector	4	
AxisBeLast_ValOut	out	std_logic	1	
AxisBeUser_DatOut	out	std_logic_vector	3	
App Ext Source Input				
AppExtSource_DatIn	in	Tsn_AppExtSource_Type	1	External Application Source Input Data
AppExtSource_ValIn	in	std_logic	1	External Application Source Input Valid
App Ext Sink Output				
AppExtSink_DatOut	out	Tsn_AppExtSink_Type	1	External Application Sink Output Data
AppExtSink_ValOut	out	std_logic	1	External Application Sink Output Valid
AXI4 Lite Slave				
AxiWriteAddrValid_ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady_RdyOut	out	std_logic	1	Write Address Ready
AxiWriteAddrAddress_AdrIn	in	std_logic_vector	32	Write Address

AxiWriteAddrProt_DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid_ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady_RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData_DatIn	in	std_logic_vector	32	Write Data
AxiWriteDataStrobe_DatIn	in	std_logic_vector	4	Write Data Strobe
AxiWriteRespValid_ValOut	out	std_logic	1	Write Response Valid
AxiWriteRespReady_RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid_ValIn	in	std_logic	1	Read Address Valid
AxiReadAddrReady_RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress_AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt_DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid_ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady_RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData_DatOut	out	std_logic_vector	32	Read Data Response

Table 27: TSN Plugfest App

4.3 Design Parts

The TSN Plugfest App 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.

4.3.1 Talker

4.3.1.1 Entity Block Diagram

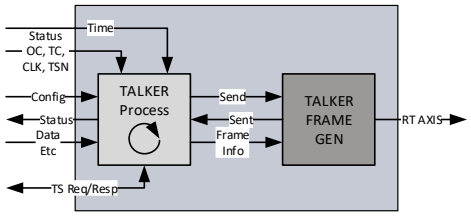

Figure 7: Talker

4.3.1.2 Entity Description

Talker Process

This process will calculate when it will send an OPC/UA Real-Time frame, based on the TX Offset, TX Delay and TX Interval. It will wait until an initial alignment has happened at a Seconds boundary and then periodically send a frame.
It will update some Frame Info (Timestamps, Frame counters, Vendor and Device Name, Flags, etc.), copy the Data either from the Application Input or from the Registerset and provide this to the Talker Frame Generator which will then Encapsulate the Frame Info into an OPC/UA Real-Time frame. The process controls the timely sending of the frame. The Talker will always send 32bytes of Data.
Once the frame is sent it will request a Timestamp from the Timestamper, this can take some time until it gets a positive Response since the frame has to go through the TSN core. If after some timeout value it didn't receive a Timestamp it will start all over again and mark that the TX Timestamp is not valid in the frame.

Talker Frame Generator

This module encapsulates the Frame Info and Data from the Talker Process into an OPC/UA Real-Time frame which is sent via AXI Stream. It also calculates a CRC.
The Talker Process controls when this module shall send the frame and will signal when it was sent.

4.3.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
AppRegSupport_Gen	-	boolean	1	If the Talker Data can also come from Register Values
LocalTimestamp_Gen	-	boolean	1	DO NOT USE If timestamps shall be taken internally in the core, this shall only be used when the external Timestampers are not used or e.g. no TSN core is in place.
Timeout Nanosecond_Gen	-	natural	1	How long the Talker shall wait in Nano-seconds to get a timestamp for the frame (this can take a while until the frame is processed by the TSN core, shall be less than the interval time)
VendorName_Gen	-	string	32	Vendor Name as ASCII String (must be filled with spac-es)
DeviceName_Gen	-	string	10	Device Name as ASCII String (must be filled with spac-

				es)
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 PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Config				
Config_DatIn	in	Tsn_TalkerConfigIIC_Type	1	Talker Configuration
Status				
Status_DatOut	out	Tsn_TalkerStatusIIC_Type	1	Talker Status
Status Inputs				
StaticStatusClk_DatIn	in	Clk_ClockStatic Status_Type	1	Static Status Data of the Clock Core
StaticStatusClk_ValIn	in	Clk_ClockStatic StatusVal_Type	1	Static Status Valid of the Clock Core
StaticStatusTsn_DatIn	in	Red_TsnStatic Status_Type;	1	Static Status Data of the TSN Core
StaticStatusTsn_ValIn	in	Red_TsnStatic StatusVal_Type;	1	Static Status Valid of the TSN Core
StaticStatus PtpOc_DatIn	in	Ptp_OrdinaryClock StaticStatus_Type;	1	Static Status Data of the PTP OC Core
StaticStatus PtpOc_ValIn	in	Ptp_OrdinaryClock StaticStatusVal _Type;	1	Static Status Valid of the PTP OC Core
StaticStatus PtpTc_DatIn	in	Ptp_Transparent ClockStatic Status_Type;	1	Static Status Data of the PTP TC Core
StaticStatus PtpTc_ValIn	in	Ptp_Transparent ClockStatic	1	Static Status Valid of the PTP TC Core

		StatusVal_Type;		
App Data Input				
AppId_DatIn	in	std_logic_vector	8	Application Identifier
AppData_DatIn	in	Common_Byte_Type	32	Application Data
AppDataValid_ValIn	in	std_logic	1	Application Data Valid
AppSequenceNr_DatIn	in	std_logic_vector	16	Application Sequence Number
AppTimestamp_DatIn	in	std_logic_vector	64	Application Timestamp
Timestamp				
Timestamp_DatIn	in	Clk_Time_Type	1	Timestamp Value
Timestamp Ok_ValIn	in	std_logic	1	Timestamp Ok (found)
Timestamp Resp_ValIn	in	std_logic	1	Timestamp searched
Timestamp Info_DatOut	out	std_logic_vector	32	Timestamp Info to find the correct one
Timestamp Req_ValOut	out	std_logic	1	Search Timestamp
Axi Real Time Output				
AxisRtValid_ValOut	out	std_logic	1	AXI Stream Real Time frame output
AxisRtReady_ValIn	in	std_logic	1	
AxisRtData_DatOut	out	std_logic_vector	32	
AxisRtStrobe_ValOut	out	std_logic_vector	4	
AxisRtKeep_ValOut	out	std_logic_vector	4	
AxisRtLast_ValOut	out	std_logic	1	
AxisRtUser_DatOut	out	std_logic_vector	3	

Table 28: Talker

4.3.2 Listener

4.3.2.1 Entity Block Diagram

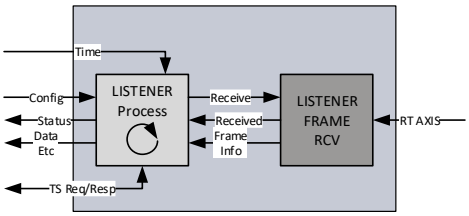

Figure 8: Listener

4.3.2.2 Entity Description

Listener Frame Receiver

This module receives the OPC/UA Real-Time frames via AXI Stream and extracts the Data and some Meta information which will be provided to the Listener Process. The Listener Process controls when this module shall receive frames and will signal when it has received a frame. It will signal if the frame was dropped or if a valid OPC/UA Real-Time frame was received or if an OPC/UA frame was received but with the Wrong Message identifier. It also extracts the Application Data Length even though it should be always 32 byte.

Listener Process

This process will allow receiving of OPC/UA Real-Time frames.
The process controls the receiving of the frames.
Once a frame is received it will request a Timestamp from the Timestamper, this should not take long until it gets a positive Response since the frame has already passed through the TSN core and should therefore be timestamped. If after some timeout value it didn't receive a Timestamp it will mark that the RX Timestamp is not valid and will start all over again. If it received a correct Response it will mark the Received Frame as valid and will provide the Data to the Applications.

4.3.2.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
LocalTimestamp_Gen	-	boolean	1	DO NOT USE If timestamps shall

				be taken internally in the core, this shall only be used when the external Timestampers are not used or e.g. no TSN core is in place.
Timeout Nanosecond_Gen	-	natural	1	How long the Listener shall wait in Nanoseconds to get a timestamp for the frame (can be short since timestamp shall be taken already)
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 PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Config				
Config_DatIn	in	Tsn_Listener ConfigIIC_Type	1	Talker Configuration
Status				
Status_DatOut	out	Tsn_Listener StatusIIC_Type	1	Talker Status
Frame Info Output				
FrameInfo_DatOut	out	Tsn_RtFrame Infolic_Type	1	Frame Information (Data, Meta Data, Timestamp Etc.)
FrameInfo_ValOut	out	std_logic	1	Frame Information

				is valid
FrameInfoApp_ValOut	out	std_logic	1	Frame Information is valid (Rx Timestamp maybe not) but this doesn't matter for the App
FrameDropped_ValOut	out	std_logic	1	Frame was dropped (not an OPC/UA frame or error)
FrameWrongApp_ValOut	out	std_logic	1	Frame was an OPC/UA frame but with the wrong App Version
Timestamp				
Timestamp_DatIn	in	Clk_Time_Type	1	Timestamp Value
Timestamp Ok_ValIn	in	std_logic	1	Timestamp Ok (found)
Timestamp Resp_ValIn	in	std_logic	1	Timestamp searched
Timestamp Info_DatOut	out	std_logic_vector	32	Timestamp Info to find the correct one
Timestamp Req_ValOut	out	std_logic	1	Search Timestamp
Axi Real Time Input				
AxisRtValid_ValIn	in	std_logic	1	AXI Stream Real Time frame input
AxisRtReady_ValOut	out	std_logic	1	
AxisRtData_DatIn	in	std_logic_vector	32	
AxisRtStrobe_ValIn	in	std_logic_vector	4	
AxisRtKeep_ValIn	in	std_logic_vector	4	
AxisRtLast_ValIn	in	std_logic	1	
AxisRtUser_DatIn	in	std_logic_vector	3	

Table 29: Listener

4.3.3 Analyzer

4.3.3.1 Entity Block Diagram

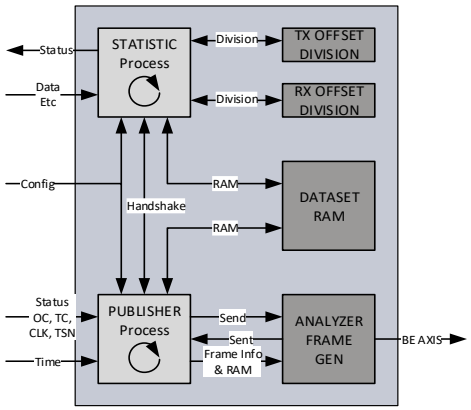

Figure 9: Analyzer

4.3.3.2 Entity Description

Statistic Process

This process waits for a new OPC/UA frame to arrive. It the calculates Statistics based on the previous frames and stores it per Talker Identifier in the Dataset RAM. The Talker Identifier is used as Index to the Dataset RAM and only Talker Identifiers between 0 and 255 are allowed. It calculates the TX Offset based on the TX Timestamp with a Division (Modulo) with the Period, the same is done with the RX Timestamp. For this a Division module is used. It also measures the Delay of the frame through the Network by subtracting the TX Timestamp from the RX Timestamp. For these 3 Values a Min, Max and the current value is stored in the Dataset RAM. It also checks if Frames were missed or if an Application update was missed and increments counters based on this. Additionally it calculates some Flags which are also stored in the RAM together with the State of the Talker.

RX & TX Offset Division

These modules are binary integer divisions, they take quite some time to do the division but are using less resources. Also after the division the reminder is provided which is what we need, since the reminder of an integer division is the modulo value.

Dataset RAM

This is a dual Port RAM which stores the Datasets of up to 256 Talkers. Each Dataset Entry uses 16 RAM Entries.

Publisher Process

This process periodically (1/s) goes through Dataset RAM and extracts the value of the available Talker Statics and provides this in the correct format to the Analyzer Frame Generator. Currently the Frame Generator can handle only up to 22 Datasets because of the Maximum size of an Ethernet frame (there is a mode where multiple Analyzer frames are sent until all entries in the table have been sent).
It only extracts valid Datasets of Talkers which are actually active, and also ages the Datasets to remove Talkers from the Dataset RAM which are no longer active. The Process can handle dynamic adding and removing of Talkers from the Network. Once all Datasets are ready it will update some Frame Info (counters, IP checksum etc.) and provide this to the Analyzer Frame Generator which will then Encapsulate the Frame Info into an OPC/UA Best-Effort frame. The process controls the timely sending of the frame.

Analyzer Frame Generator

This module encapsulates the Frame Info from the Publisher Process into an OPC/UA Best effort frame which is sent via AXI Stream. It also dynamically adds the Dataset entries to the frame and calculates a CRC.
The Publisher Process controls when this module shall send the frame and will signal when it was sent.

4.3.3.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriodNanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
MultiFrameSupport_Gen	-	boolean	1	If the Publisher shall split the results into multiple frames if more than 23 Nodes are seen
VendorName_Gen	-	string	32	Vendor Name as ASCII String (must be filled with spaces)

DeviceName_Gen	-	string	10	Device Name as ASCII String (must be filled with spac-es)
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 PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
Config				
Config_DatIn	in	Tsn_ConfigIIC_Type	1	Configuration of all parts (including Analyzer)
Status				
Status_DatOut	out	Tsn_Analyzer StatusIIC_Type	1	Status
Status Inputs				
StaticStatusClk_DatIn	in	Clk_ClockStatic Status_Type	1	Static Status Data of the Clock Core
StaticStatusClk_ValIn	in	Clk_ClockStatic StatusVal_Type	1	Static Status Valid of the Clock Core
StaticStatusTsn_DatIn	in	Red_TsnStatic Status_Type;	1	Static Status Data of the TSN Core
StaticStatusTsn_ValIn	in	Red_TsnStatic StatusVal_Type;	1	Static Status Valid of the TSN Core
StaticStatus PtpOc_DatIn	in	Ptp_OrdinaryClock StaticStatus_Type;	1	Static Status Data of the PTP OC Core
StaticStatus PtpOc_ValIn	in	Ptp_OrdinaryClock StaticStatusVal _Type;	1	Static Status Valid of the PTP OC Core
StaticStatus PtpTc_DatIn	in	Ptp_Transparent	1	Static Status Data

		ClockStatic Status_Type;		of the PTP TC Core
StaticStatus PtpTc_ValIn	in	Ptp_Transparent ClockStatic StatusVal_Type;	1	Static Status Valid of the PTP TC Core
Frame Info Input				
FrameInfo_DatIn	in	Tsn_RtFrame Infolic_Type	1	Frame Information (Data, Meta Data, Timestamp Etc.)
FrameInfo_ValIn	in	std_logic	1	Frame Information is valid
FrameDropped _ValIn	in	std_logic	1	Frame was dropped (not an OPC/UA frame or error)
FrameWrongApp _ValIn	in	std_logic	1	Frame was an OPC/UA frame but with the wrong App Version
Axi Best EffortOutput				
AxisBeValid_ValOut	out	std_logic	1	AXI Stream Best Effort frame output
AxisBeReady_ValIn	in	std_logic	1	
AxisBeData_DatOut	out	std_logic_vector	32	
AxisBeStrobe_ValOut	out	std_logic_vector	4	
AxisBeKeep_ValOut	out	std_logic_vector	4	
AxisBeLast_ValOut	out	std_logic	1	
AxisBeUser_DatOut	out	std_logic_vector	3	

Table 30: Analyzer

4.3.4 App LED

4.3.4.1 Entity Block Diagram

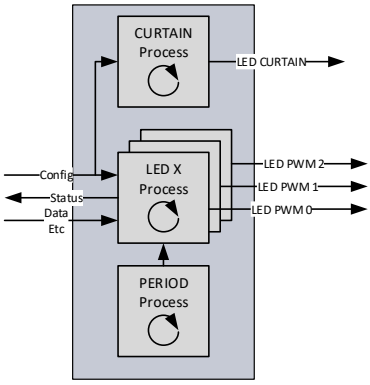

Figure 10: App LED

4.3.4.2 Entity Description

Curtain Process

This process just checks if the Application Data received was the correct Application Identifier and optionally also the correct Talker Identifier and VLAN. If this matches it will extract the Light Curtain value from the Data Bytes and provides this to the output.

Period Process

This process just creates a period counter which defines the PWM rate. When the counter wraps around it just starts at 0 again.

LED Processes

This process checks if the Application Data received was the correct Application Identifier and optionally also the correct Talker Identifier and VLAN. If this matches it will extract the brightness value and convert it into a pulse width (if larger than 100 it will truncate it to 100%) depending on the PWM period. This pulse width is then used to create the PWM signal: whenever the period counter is smaller than the pulse width it sets the LED output to 1 else to 0.
New pulse widths are only calculated at the end of a period to avoid inconsistent pulses.

4.3.4.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriod	-	natural	1	Clock Period in

Nanosecond_Gen				Nanosecond: Default for 50 MHz = 20 ns
PeriodNanosec-ond_Gen	-	natural	1	PWM Period in Nanosecond
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
Config_DatIn	in	Tsn_AppLed Configlic_Type	1	Static Configuration
Status				
Status_DatOut	out	Tsn_AppLed Statuslic_Type	1	Static Status
Frame Info Input				
FrameInfo_DatIn	in	Tsn_RtFrame Infolic_Type	1	Frame Information (Data, Meta Data, Timestamp Etc.)
FrameInfo_ValIn	in	std_logic	1	Frame Information is valid
LED App Output				
Curtain_DatOut	out	std_logic_vector	15	Adjusted PTP Clock Time
Led_DatOut	out	std_logic_vector	3	Adjusted PTP Clock Time valid

Table 31: App LED

4.3.5 App LED Gen

4.3.5.1 Entity Block Diagram

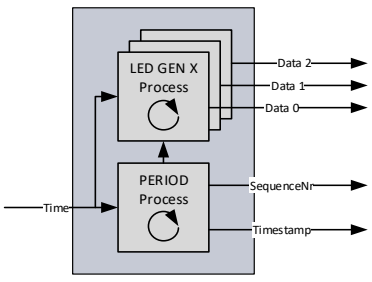

Figure 11: App LED Gen

4.3.5.2 Entity Description

Period Process

This process just creates a period counter which defines the Application update rate. When the counter wraps around the Application Sequence Number is incremented by 1 and a Snapshot of the Time taken which represents the Application Timestamp.

Led Generator Processes

Every time the wraparound of the Period counter happens, the LED pattern is also updated and the Data passed together with the Application Sequence Number and Application Timestamp to the Talker. The LED pattern is a simple up down counter which creates a sawtooth signal on each LED. The counters count from -100 to 100 and back to -100 (except initial start value which is smaller). The counter value is representing the brightness as a percentage of the LED, all negative values are truncated to 0. Since LEDs have not a linear brightness for the eye the brightness value is indexing a Gamma Table to get a close to linear impression of different brightness values. Each Led Generator Process starts initially at a different counter value, this means when the first is at top brightness the next will start fading in and the previous fading out.

4.3.5.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriodNanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
UpdatePeriodMillisecond_Gen	-	natural	1	How often the Generator calculates new brightness values in millisecond
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 PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Millisecond timer adjusted with the Clock
App Data Output				
AppId_DatOut	out	std_logic_vector	8	Application Identifier
AppData_DatOut	out	Common_Byte_Type	32	Application Data
AppDataValid_ValOut	out	std_logic	1	Application Data Valid
AppSequenceNr_DatOut	out	std_logic_vector	16	Application Sequence Number
AppTimestamp_DatOut	out	std_logic_vector	64	Application Timestamp

Table 32: App LED Gen

4.3.6 Timestamper

4.3.6.1 Entity Block Diagram

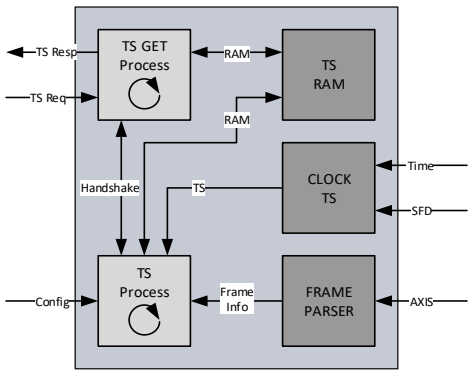

Figure 12: Timestamper

4.3.6.2 Entity Description

Timestamp RAM

This RAM stores the Timestamps, there is space for 256 Timestamps which corresponds to 256 Talkers, one timestamp per Talker. One Timestamp Entry in the RAM takes 4 RAM Entries of the 32bit wide RAM.

Clock Timestamper

This module takes a Timestamp when the SFD signal is asserted. It compensates the Timestamp for PHY delays, depending on the link speed.

Frame Parser

This module parses the frame, since we only timestamp OPC/UA Realtime frames. It tells the Timestamping Process if the frame is valid and passes the Talker ID and (Network) Sequence Number to it.

Timestamp Process

This process waits until the Timestamp from the Clock Timestamper and Frame Information from the Frame Parser are valid. If it is not an OPC/UA Real-Time Frame or if the interface has no link it ignores it and waits for the next frame.
If it is an OPC/UA Real-Time Frame, it uses the Talker Identifier (lower 8bits) as the index for the Timestamp RAM and writes the Timestamp with the Sequence Number and Talker Identifier as well as with a New Flag to the RAM. Then it waits for the next frame.

Timestamp Getting Process

This process waits until a Timestamp request comes. It then uses the Requester Information to check if for this specific Talker Id and Sequence Number a new Timestamp is available. As the Timestamper Process, it uses the Talker Identifier (lower 8bits) as the index for the Timestamp RAM and checks if the Meta Data (TalkerId & SequenceNr) matches the requested. If not, it signals a Response that the timestamp was not ok. If it matches, it will read the Timestamp from the RAM and will clear the New Flag and will then signal a Response that the timestamp is ok.

4.3.6.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				

General				
ClockClkPeriod Nanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
PreemptionSup-port_Gen	-	boolean	1	If we shall handle preempted frames
DelayNanosecond 10_Gen	-	integer	1	PHY Delay at 10Mbit/s
DelayNanosecond 100_Gen	-	integer	1	PHY Delay at 100Mbit/s
DelayNanosecond 1000_Gen	-	integer	1	PHY Delay at 1000Mbit/s
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 PTP Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted PTP Clock Time valid
Config				
Config_DatIn	in	Tsn_ConfigIIC_Type	1	Static Configuration
Link Inputs				
Link_DatIn	in	std_logic	1	If Link is Up
LinkSpeed_DatIn	in	Common_Link Speed_Type	1	10/100/1000Mbit
Timstamp Event Input				
SfdDetected_EvtIn	in	std_logic	1	Start of Frame Delimiter detect
Timstamp Event Output				
Timestamp Event_ValOut	out	std_logic	1	Timestamp Taken
Timestamp				
Timestamp_DatOut	out	Clk_Time_Type	1	Timestamp Value
Timestamp Ok_ValOut	out	std_logic	1	Timestamp Ok (found)
Timestamp Resp_ValOut	out	std_logic	1	Timestamp searched
Timestamp	in	std_logic_vector	32	Timestamp Info to

Info_DatIn				find the correct one
TimestampReq_ValIn	in	std_logic	1	Search Timestamp
Preemption Input				
PreemptionInfo_DatIn	in	std_logic_vector	16	AXI Stream frame input
PreemptionInfo_ValIn	in	std_logic	1	
Axi Input				
AxisValid_ValIn	in	std_logic	1	AXI Stream frame input
AxisReady_ValIn	in	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	

Table 33: Timestamper

4.3.7 Timestamp Merger

4.3.7.1 Entity Block Diagram

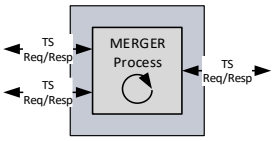

Figure 13: Timestamp Merger

4.3.7.2 Entity Description

Merger Process

This process merges the Timestamp Responses to one Timestamp Response. The Timestamp Request is duplicated to all Timestampers. The Merging is a selection, when a Timestamper marks the Response as ok, this is taken as the Response, if multiple are ok, the first is taken. It waits until all Timestampers have a valid Response.

4.3.7.3 Entity Declaration

Name	Dir	Type	Size	Description

Generics				
General				
NrOfInputs_Gen	-	natural	1	Number of Timestampers to merge
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Timestamp Multi				
Timestamp_DatIn	in	Clk_TimeArray_Type	NrOf Inputs _Gen	Timestamp Value
Timestamp Ok_ValIn	in	std_logic_vector	NrOf Inputs _Gen	Timestamp Ok (found)
Timestamp Resp_ValIn	in	std_logic_vector	NrOf Inputs _Gen	Timestamp searched
Timestamp Info_DatOut	out	Com-mon_Long_Type	NrOf Inputs _Gen	Timestamp Info to find the correct one
Timestamp Req_ValOut	out	std_logic_vector	NrOf Inputs _Gen	Search Timestamp
Timestamp Merged				
Timestamp_DatOut	out	Clk_Time_Type	1	Timestamp Value
Timestamp Ok_ValOut	out	std_logic	1	Timestamp Ok (found)
Timestamp Resp_ValOut	out	std_logic	1	Timestamp searched
Timestamp Info_DatIn	in	std_logic_vector	32	Timestamp Info to find the correct one
TimestampReq_ValIn	in	std_logic	1	Search Timestamp

Table 34: Timestamp Merger

4.3.8 Registerset

4.3.8.1 Entity Block Diagram

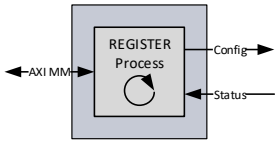

Figure 14: Registerset

4.3.8.2 Entity Description

Register Process

This module is an AXI4Lite Memory Mapped Slave. It provides access to all registers and allows configuring the TSN Plugfest App. AXI4Lite only supports 32 bit wide data access, no byte enables, no burst, no simultaneous read and writes and no unaligned access. 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.

4.3.8.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
TalkerSupport_Gen	-	boolean	1	If the Talker shall be instantiated
ListenerSupport_Gen	-	boolean	1	If the Listener shall be instantiated
AnalyzerSupport_Gen	-	boolean	1	If the Analyzer shall be instantiated
AppLedSupport_Gen	-	boolean	1	If the LED Application (PWM brightness generator) shall be instantiated
AppRegSupport_Gen	-	boolean	1	If the Talker Data can also come from Register Values

AppExtIn Support_Gen	-	boolean	1	If an external Application Source shall be connected (to the Talker)
AppExtOut Support_Gen	-	boolean	1	If an external Application Sink shall be connected (to the Listener)
AxiAddress RangeLow_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddress RangeHigh_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size Default plus 0xFFFF
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
AXI4 Lite Slave				
AxiWriteAddrValid _ValIn	in	std_logic	1	Write Address Valid
AxiWriteAddrReady _RdyOut	out	std_logic	1	Write Address Ready
AxiWriteAddrAddress _AdrIn	in	std_logic_vector	32	Write Address
AxiWriteAddrProt _DatIn	in	std_logic_vector	3	Write Address Protocol
AxiWriteDataValid _ValIn	in	std_logic	1	Write Data Valid
AxiWriteDataReady _RdyOut	out	std_logic	1	Write Data Ready
AxiWriteDataData _DatIn	in	std_logic_vector	32	Write Data
AxiWriteDataStrobe _DatIn	in	std_logic_vector	4	Write Data Strobe
AxiWriteRespValid _ValOut	out	std_logic	1	Write Response Valid
AxiWriteRespReady _RdyIn	in	std_logic	1	Write Response Ready
AxiWriteResp Response_DatOut	out	std_logic_vector	2	Write Response
AxiReadAddrValid _ValIn	in	std_logic	1	Read Address Valid

AxiReadAddrReady_RdyOut	out	std_logic	1	Read Address Ready
AxiReadAddrAddress_AdrIn	in	std_logic_vector	32	Read Address
AxiReadAddrProt_DatIn	in	std_logic_vector	3	Read Address Protocol
AxiReadDataValid_ValOut	out	std_logic	1	Read Data Valid
AxiReadDataReady_RdyIn	in	std_logic	1	Read Data Ready
AxiReadData Response_DatOut	out	std_logic_vector	2	Read Data
AxiReadDataData_DatOut	out	std_logic_vector	32	Read Data Response
Config				
Config_DatOut	out	Tsn_ConfigIIC_Type	1	Static Configuration
Status				
Status_DatIn	In	Tsn_StatusIIC_Type	1	Static Status

Table 35: Registerset

4.4 Configuration example

4.4.1 AXI Configuration

The following code is a simplified pseudocode from the testbench: The base address of the TSN Plugfest App is 0x10000000.

```
-- Enable and Internal Application
AXI WRITE 20000000 00000001

-- TALKER MAC 00:11:22:33:44:55
AXI WRITE 20000104 33221100
AXI WRITE 20000108 00005544

-- TALKER ID 10
AXI WRITE 2000010C 0000000A

-- TX DELAY NS 500
AXI WRITE 20000110 000001F4
-- START TIME NS 100000
AXI WRITE 20000114 000186A0
-- INTERVAL TIME NS 100000
AXI WRITE 20000118 000186A0

-- RX DELAY NS 500
AXI WRITE 20000210 000001F4

-- ANALYZER IP 192.168.0.1
AXI WRITE 20000304 0100A8C0
-- ANALYZER UDP PORT 4840
AXI WRITE 20000308 000012E8

-- LED Talker ID all to listen
AXI WRITE 2000100C 0000FFFF

-- Start Talker and use Gen Data (LedGen or External)
AXI WRITE 20000100 00000001

-- Start Listener and ignore VLAN
AXI WRITE 20000200 00040001

-- Start Analyzer and ignore VLAN
AXI WRITE 20000300 00040001

-- Start App and ignore VLAN
AXI WRITE 20001000 00040001
```

Figure 15: AXI Configuration

4.5 Clocking and Reset Concept

4.5.1 Clocking

To keep the design as robust and simple as possible, the whole TSN Plugfest App, 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. All interfaces are run synchronous to this clock.. 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 PS runs on as well as the counter clock etc.
AXI Interface		
AXI Clock	50MHz (Default)	Internal AXI bus clock, same as the system clock

Table 36: Clocks

4.5.2 Reset

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

Reset	Polarity	Description
System		
System Reset	Active low	Asynchronous set, synchronous release with the system clock
AXI Interface		
AXI Reset	Active low	Asynchronous set, synchronous release with the AXI clock, which is the same as the system clock

Table 37: Resets

5 Resource Usage

Since the Resource heavily depends on the features instantiated a default configuration is used.

5.1 AMD/Xilinx (Kintex 7)

Configuration	FFs	LUTs	BRAMs	DSPs
Default (Talker, Listener, Analyzer, LED App, LED App Gen, REG App, No EXT App, 1 RX & TX Timestamper)	5800	6800	8	22

Table 38: Resource Usage AMD/Xilinx

6 Delivery Structure

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

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

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

TSN                                 -- TSN library folder
 |-Core                             -- TSN library cores
 |-Doc                              -- TSN library cores documentations
 |-Library                          -- TSN library component sources
 |-Package                          -- TSN library package sources
 |-Refdesign                        -- TSN library cores reference designs
 |-Testbench                        -- TSN 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
```

Be aware that this core is only a companion core to the TSN IP core

7 Testbench

The TSN Plugfest App testbench consist of 2 parse/port types: AXI, ETH.
The ETH ports AXI0 and AXI1 simulate two Real-Time ports, the ETH port AXI2 simulates the Best-Effort port. The two Real-Time Ports are connected to RX and TX Timestampers and for each direction an Timestamp Merger.
The Ext Application port of the DUT is looped back so whatever Data is sent into the DUT is looped back as Data. Frames sent by the DUT can be checked for their format and timely arrival as well as Frames generated and checked how the DUT reacts.
In addition for configuration and result checks an AXI read and write port is used in the testbench.

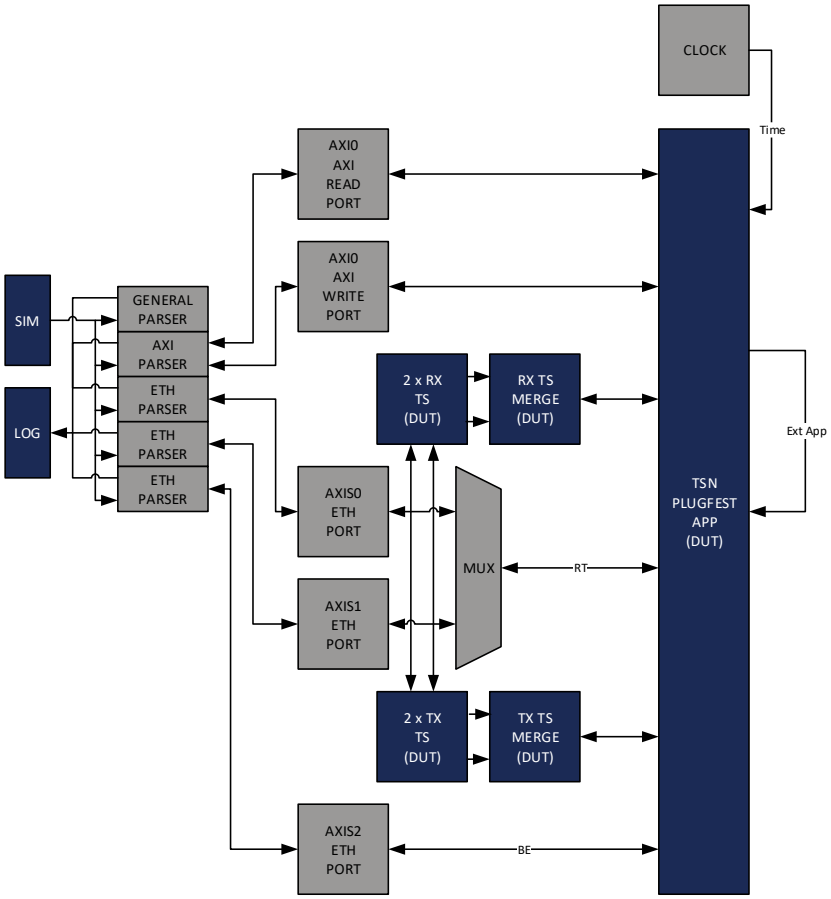

Figure 16: Testbench Framework

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/TSN/Testbench/Core/PlugfestApp/Script/run_Tsn_PlugfestAppIicAxi_Tb.tcl
```

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

8 Reference Designs

The Reference Design is only there to show how the core shall be interconnected and used. No further information is provided here since it is meant as companion core to the NetTimeLogic TSN Core.

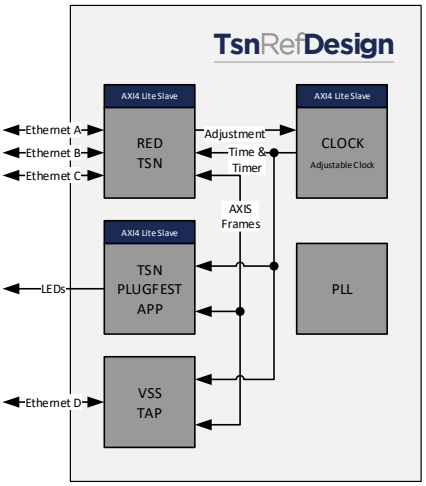

Figure 17: Reference Design

If you need further information on how to use it in combination with the TSN Core contact us: support@nettimelogic.com

A List of tables

B List of figures