NtpClientClock

Reference Manual

Product Info	
Product Manager	Sven Meier
Author(s)	Ioannis Sotiropoulos, Sven Meier
Reviewer(s)	Sven Meier
Version	1.2
Date	16.02.2024

Copyright Notice

Disclaimer

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

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

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

Overview

NetTimeLogic's NTP Client is a full hardware (FPGA) only implementation of a Simple Network Time Protocol v4 (SNTP) Client according to RFC 4330/5905. The whole protocol handling, algorithms and calculations are implemented in the core, no CPU is required. This allows running a NTP Client completely independent and standalone from the user application. The NTP Client can send Unicast requests to the NTP Server at configurable time intervals. After receiving the server's responses (Unicast or Multicast), the client calculates the offset and drift adjustments to be applied to the clock. The client can be configured either by signals or by an AXI4Lite-Slave Register interface.

Key Features:

- SNTP Client according to RFC 4330/5905
- Intercepts path between MAC and PHY
- Single Port
- Unicast requests at configurable intervals
- Process Unicast/Multicast responses from the NTP Server
- Hardware timestamping on (R)(G)MII level
- Support for UTC leap second handling (jump or smearing)
- Optional leap second smearing (configurable rate)
- AXI4Lite register set or static configuration
- MII/GMII/RGMII Interface support (optional AXI4 stream for interconnection to 3rd party cores)
- Timestamp resolution with 50 MHz system clock: 10ns or with a 250MHz aligned clock 4ns
- Hardware PI Servo

Revision History

This table shows the revision history of this document.

Version	Date	Revision
0.1	21.10.2022	First draft
1.0	04.11.2022	First release
1.1	03.01.2023	Added Vivado upgrade version description
1.2	16.02.2024	IPv6 added

Table 1: Revision History

Content

Definitions

Definitions	
Client	The NTP Device which requests the time from the Server
Server	The NTP Device answering requests

Table 2: Definitions

Abbreviations

Abbreviations	
AXI	AMBA4 Specification (Stream and Memory Mapped)
CRC	Cyclic Redundancy Check
ETH	Ethernet
FF	Flip Flop
FPGA	Field Programmable Gate Array
IP	Internet Protocol
IRQ	Interrupt, Signaling to e.g. a CPU
LUT	Look Up Table
MAC	Media Access Controller
NTP	Network Time Protocol
PHY	Physical Media Access Controller
RAM	Random Access Memory
ROM	Read Only Memory
SNTP	Simple Network Time Protocol
TB	Testbench
TS	Timestamp
UDP	User Datagram Protocol
VHDL	Hardware description Language for FPGA's

Table 3: Abbreviations

1 Introduction

1.1 Context Overview

The NTP Client (SNTP Client) is meant as a co-processor that calculates clock adjustments based on the time of an NTP Server. It intercepts the Media Independent Interface (MII) on the Ethernet path between the MAC and PHY where it handles all NTP traffic. This means it generates and processes NTP frames directly in hardware, using the same data path as the normal traffic coming from or going to the Ethernet MAC. The NTP Client is designed to work in cooperation with the Counter Clock core from NetTimeLogic (not a requirement). It contains an AXI4Lite slave for configuration from a CPU, this is however not required since the NTP Client can also be configured statically via signals/constants directly from within the FPGA.

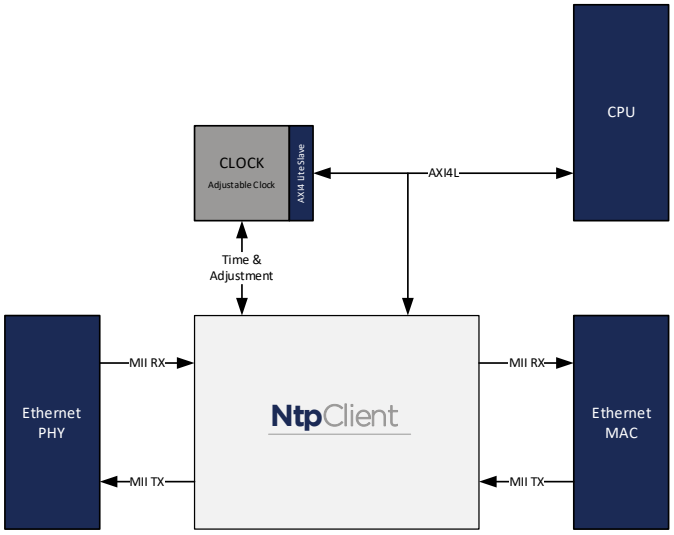

Figure 1: Context Block Diagram

1.2 Function

The NTP Client is a SNTP v4 Client according to RFC 4330/5905. As a client it receives NTP Responses from the NTP Server and calculates the adjustments to be applied to the Client's Counter Clock.
The client supports different modes, by accepting Unicast and Multicast responses from the server. The IP Core can act as an endpoint which fully handles NTP or it can also work in a throughput mode where it does handle NTP frames but all other traffic goes through the Core.

1.3 Architecture

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

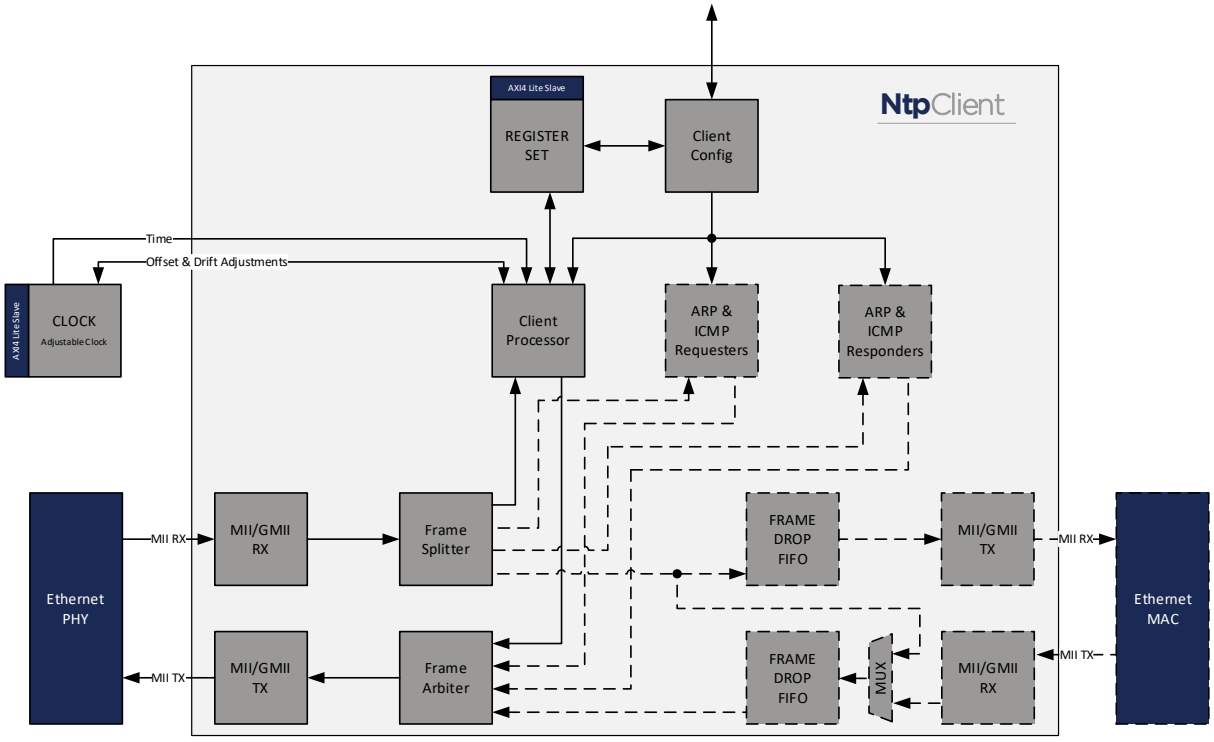

Figure 2: Architecture Block Diagram

Register Set

This block allows reading status values and writing configuration.

Client Config

This block stores the configuration of the client. There are multiple sources possible, like the configuration via the register set, static configuration from externally or also UTC information from externally.

Client Processor

This block is the heart the NTP Client. The Client Processor generates in periodic intervals the NTP request and timestamps the transmission time. When the NTP Server replies with the corresponding NTP response, the client timestamps the incoming frame and extracts the information from the NTP response. Based on the timestamps and the received information, the client calculates the average delay

to the server and the offset and drift adjustments to be applied to the client's clock. Additionally, UTC handling is done by the Client Processor block.

ARP & ICMP Requesters

These modules sends ARP and ICMP requests.

ARP & ICMP Responders

These modules answers ARP or ICMP requests with ARP or ICMP responses.

Frame Splitter

This module splits one AXIS interface to multiple, allowing the ARP Responder to receive ARP requests, forwarding frames through the core and forward frames to the client processor

Frame Arbiter

This module arbitrates multiple AXIS interfaces to one, allowing sending ARP requests, responding to incoming ARP requests, forwarding frames through the core and sending NTP messages from the client processor.

Frame Drop Fifo

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

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

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

2 NTP Basics

2.1 Protocol

NTP means Network Time Protocol and it is a protocol to synchronize device clocks over a network to a reference time base. The protocol exists since 1985 and it is specified in several Request for Comments (RFC). The most recent and relevant ones are RFC 5905 (NTPv4) and RFC 4430 (SNTPv4). NTP is built on the Internet Protocol (IP) and the User Datagram Protocol (UDP).
The principal of the protocol is based on periodically request from a client to the server for the precise UTC time reference. In the client-server mode the NTP server is listening for NTP packets on UDP port 123 and replies to these packets. NTP does also have a broadcast variant where the server sends periodically packets that can be received by multiple clients (but without the possibility to compensate for path delays).

2.2 Principles

The basic operation of NTP is timestamping of data packets transferred between the server and the client. The NTP packets sent from a client to the server and the responses from the server to the client have the same format. The server fills fields such as timestamps to the received packet and sends it back to the clients.

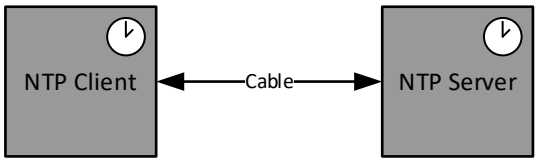

Figure 3: Simple setup

The NTP packet consists of four timestamps. The timestamps are 64 bits with 32 bits as second and 32 bits as fraction:

- Reference timestamp: Time when the system clock was last set
- Origin timestamp: Time at the client when the request departed
- Receive timestamp: Time at the server when the request arrived
- Transmit timestamp: Time at the server when the response left

The client sends the request to the server with the origin timestamp (T1). The server takes a timestamp of the time when the request packet is received (receive timestamp T2). The server sends the response packet back to the client with the

transmit timestamp (T3). The client stamps the time when the response packet is received (destination timestamp T4).

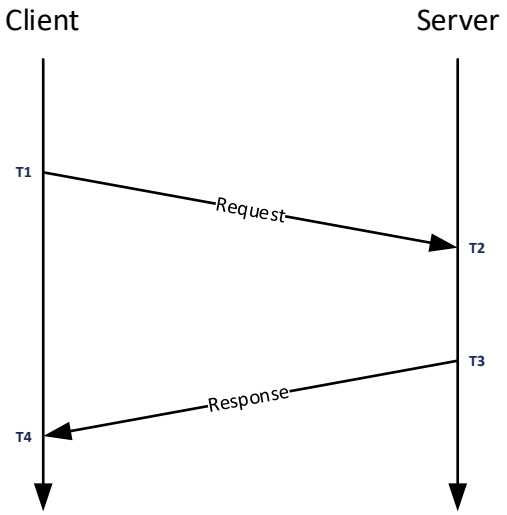

Figure 4: Message exchange simple setup

With these timestamps the offset and the roundtrip delay can be calculated:

$$Offset = \frac{(T2 - T1) + (T3 - T4)}{2}$$

$$Delay = (T4 - T1) - (T3 - T2)$$

2.3 Accuracy

The accuracy of the synchronization depends mostly on the jitter of the network between the Client and Server but also highly on the precision of the timestamps. They should reflect the send and receive time as precise as possible. The client's Offset and Delay calculations are based on the difference of timestamps taken at two different places.

The achievable accuracy depends on:

- Network Jitter
- Network Symmetry
- Timestamp accuracy
- Clock stability
- Clock control loop characteristics
- Drift compensated clocks (i.e. adjusted time base of the Server)

2.3.1 Network jitter and symmetry

The accuracy of an NTP system is heavily dependent on a symmetrical jitter free network path between the client and the master. However, since this is never the case advanced filtering is required to get a good mean value and filter out outliners. It is aways important to know that it is always a tradeoff between fast synchronization and accuracy/stability. Also, a quite jittery network will lead to a long-term stable system but short term fluctuations.
Since for the offset calculation a symmetrical link is required (division by 2) any asymmetry will directly result in an offset which is hard to mitigate since paths can change at every moment.

2.3.2 Timestamp accuracy

As just stated, timestamp accuracy is the key to high accuracy. Timestamp support can be implemented at different layers with a decrease in accuracy in the higher layers. For this solution a timestamp point between MAC and PHY (on MII) was chosen to get the best possible accuracy without implementing PHY functionality This interface is perfect for the use of FPGAs since it is a strictly digital interface, standardized and has only a low frequency requirement. For this implementation the FPGA is intercepting the Path between MAC and PHY.

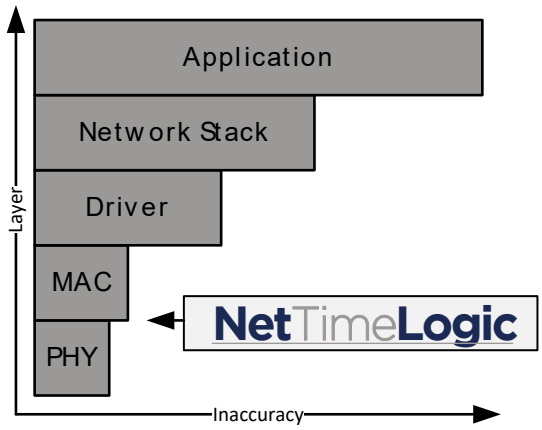

Figure 5: Timestamp Inaccuracy in the different Layers

Not only timestamping but also frame generation, handling and servo loops can be done in different stages. A wide spread approach is to have the timestamping at a very low layer and all other things in the application layer which implies that a MAC, CPU and Operating System with Drivers a Network Stack and a NTP application is in place. NetTimeLogic's approach is different: NetTimeLogic's NTP Server is completely implemented in an FPGA meaning that all the upper layers after the PHY

are not required. This decreases the complexity and dependencies between the different layers and allows running the system without CPU. It also allows to answer request at line speed

2.4 Difference between SNTP and NTP

Simple Network Time Protocol (SNTP) is a stripped-down version of NTP. SNTP and NTP share several similarities. The packets of data exchanged between the clients and the server are identical, which makes any time server compatible with both. The differences between NTP and SNTP is only affecting the synchronization on the client side.
The main difference is that no mitigation algorithms are implemented. In addition, SNTP is intended for primary server equipped with a single reference clock, so it is based on a single server-client relationship.

3 Register Set

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

3.1 Register Overview

Registerset Overview			
Name	Description	Offset	Access
Ntp ClientControl Reg	NTP Client Enable Control Register	0x00000000	RW
Ntp ClientStatus Reg	NTP Client Error Status Register	0x00000004	WC
Ntp ClientVersion Reg	NTP Client Version Register	0x0000000C	RO
Ntp ClientCountControl Reg	NTP Client Count Control Register	0x00000010	RW
Ntp ClientCountReq Reg	NTP Client Request Count Register	0x00000014	RO
Ntp ClientCountResp Reg	NTP Client Response Count Register	0x00000018	RO
Ntp ClientCountRespMissed Reg	NTP Client Response Missed Count Register	0x0000001C	RO
Ntp ClientConfigControl Reg	NTP Client Configuration Control Register	0x00000080	RW
Ntp ClientConfigMode Reg	NTP Client Configuration Mode Register	0x00000084	RW
Ntp ClientConfigVlan Reg	NTP Client Configuration VLAN Register	0x00000088	RW
Ntp ClientConfigMac1 Reg	NTP Client Configuration MAC 1 Register	0x0000008C	RW
Ntp ClientConfigMac2 Reg	NTP Client Configuration MAC 2 Register	0x00000090	RW
Ntp ClientConfigIp Reg	NTP Client Configuration IPv6 0 and IPv4 Register	0x00000094	RW
Ntp ClientConfigIpv61 Reg	NTP Client Configuration IPv6 1 Register	0x00000098	RW
Ntp ClientConfigIpv62 Reg	NTP Client Configuration IPv6 2 Register	0x0000009C	RW
Ntp ClientConfigIpv63 Reg	NTP Client Configuration IPv6 3 Register	0x000000A0	RW

Ntp ClientConfigServerMac1 Reg	NTP Client Configuration Server MAC1 Register	0x000000A4	RW
Ntp ClientConfigServerMac2 Reg	NTP Client Configuration Server MAC2 Register	0x000000A8	RW
Ntp ClientConfigServerIp Reg	NTP Client Configuration Server IPv6 0 and IPv4 Register	0x000000AC	RW
Ntp ClientConfigServerIpv61 Reg	NTP Client Configuration Server IPv6 1 Register	0x000000B0	RW
Ntp ClientConfigServerIpv62 Reg	NTP Client Configuration Server IPv6 2 Register	0x000000B4	RW
Ntp ClientConfigServerIpv63 Reg	NTP Client Configuration Server IPv6 3 Register	0x000000B8	RW
Ntp ClientConfigSubnetMask Reg	NTP Client Configuration Subnet Mask v6 0 and v4 Register	0x000000BC	RW
Ntp ClientConfigSubnetMaskv61 Reg	NTP Client Configuration Subnet Mask v6 1 Register	0x000000C0	RW
Ntp ClientConfigSubnetMaskv62 Reg	NTP Client Configuration Subnet Mask v6 2 Register	0x000000C4	RW
Ntp ClientConfigSubnetMaskv63 Reg	NTP Client Configuration Subnet Mask v6 3 Register	0x000000C8	RW
Ntp ClientConfigGatewayIp Reg	NTP Client Configuration Gateway IPv6 0 and IPv4 Register	0x000000CC	RW
Ntp ClientConfigGatewayIpv61 Reg	NTP Client Configuration Gateway IPv6 1 Register	0x000000D0	RW
Ntp ClientConfigGatewayIpv62 Reg	NTP Client Configuration Gateway IPv6 2 Register	0x000000D4	RW
Ntp ClientConfigGatewayIpv63 Reg	NTP Client Configuration Gateway IPv6 3 Register	0x000000D8	RW
Ntp ClientConfigFilterCoefB0 Reg	NTP Client Configuration Filter Coefficient B0	0x000000DC	RW
Ntp ClientConfigFilterCoefB1 Reg	NTP Client Configuration Filter Coefficient B1	0x000000E0	RW
Ntp ClientConfigFilterCoefB2 Reg	NTP Client Configuration Filter Coefficient B2	0x000000E4	RW
Ntp ClientConfigFilterCoefA1 Reg	NTP Client Configuration Filter Coefficient A1	0x000000E8	RW
Ntp ClientConfigFilterCoefA2 Reg	NTP Client Configuration Filter Coefficient A2	0x000000EC	RW
Ntp ClientConfigPiFactorP Reg	NTP Client Configuration PI Servo Factor P	0x000000F0	RW
Ntp ClientConfigPiFactorI Reg	NTP Client Configuration PI Servo Factor I	0x000000F4	RW
Ntp ClientUtcInfoControl Reg	NTP Client UTC Info Control Register	0x00000100	RW
Ntp ClientUtcInfo Reg	NTP Client UTC Info Register	0x00000104	RW
Ntp ClientOffsetFromServer Reg	NTP Client Offset From Server Register	0x00000200	RO
Ntp ClientMeanRoundtripDelay Reg	NTP Client Mean Roundtrip Delay Register	0x00000204	RO

Table 4: Register Set Overview

3.2 Register Descriptions

3.2.1 General

3.2.1.1 NTP Client Control Register

Used for general control over the NTP Client.

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

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

3.2.1.2 NTP Client Status Register

Shows the current status of the NTP Client.

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

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

3.2.1.3 NTP Version Register

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

Ntp ClientVersion 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

3.2.1.4 NTP Client Count Control Register

Used for clearing all statistic counters

Ntp ClientCountControl Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																															CLEAR
RO																															RW
Reset: 0x00000000																															
Offset: 0x0010																															

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

3.2.1.5 NTP Client Count Request Registers

Number of transmitted NTP Requests.

Ntp ClientCountReq 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
TX_REQ																													
RO																													
Reset: 0x00000000																													
Offset: 0x0014																													

Name	Description	Bits	Access
TX_REQ	Number of transmitted NTP requests	Bit: 31:0	RO

3.2.1.6 NTP Client Count Response Registers

Number of received NTP Responses.

Ntp ClientCountResp Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_RESP																													
RO																													
Reset: 0x00000000																													
Offset: 0x0018																													

Name	Description	Bits	Access
RX_RESP	Number of received NTP responses	Bit: 31:0	RO

3.2.1.7 NTP Client Count Response Missed Registers

Number of missed NTP Requests.

Ntp ClientCountRespMissed 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
RESP_MISSED																													
RO																													
Reset: 0x00000000																													
Offset: 0x001C																													

Name	Description	Bits	Access
RESP_MISSED	Number of missed responses	Bit: 31:0	RO

3.2.1.8 NTP Client Configuration Control Register

Valid flags of configuration data.

Ntp ClientConfigControl Reg																						
Reg Description																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RO																						PI_FACTORS_VAL	FILTER_COEF_VAL	GATEWAY_IP_VAL	SUBNET_MASK_VAL	SERVER_IP_VAL	SERVER_MAC_VAL	IP_VAL	MAC_VAL	VLAN_VAL	MODE_VAL
RO																						RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset: 0x00000000																															
Offset: 0x0080																															

Name	Description	Bits	Access
-	Reserved, read 0	Bit:31:10	RO
PI_FACTORS_VAL	PI Servo Factors valid (autocleared)	Bit: 9	RW
FILTER_COEFF_VAL	Filter Coefficients valid (autocleared)	Bit: 8	RW
GATEWAY_IP_VAL	Gateway IP valid (autocleared)	Bit: 7	RW
SUBNET_MASK_VAL	Subnet Mask valid (autocleared)	Bit: 6	RW
SERVER_IP_VAL	Server IP valid (autocleared)	Bit: 5	RW
SERVER_MAC_VAL	Server MAC valid (autocleared)	Bit: 4	RW

IP_VAL	IP valid (autocleared)	Bit: 3	RW
MAC_VAL	MAC valid (autocleared)	Bit: 2	RW
VLAN_VAL	VLAN valid (autocleared)	Bit: 1	RW
MODE_VAL	Mode valid (autocleared)	Bit: 0	RW

3.2.1.9 NTP Client Configuration Register

Configuration of the request-response mode support.

Ntp ClientConfigMode 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
~								POLL_INTERVAL								~										MULTICAST	UNICAST	-		IP_MODE	
RO								RW								RO										RW	RW	RO		RW	
Reset: 0x00000000																															
Offset: 0x0084																															

Name	Description	Bits	Access
~	Reserved, read 0	Bit:31:24	RO
POLL_INTERVAL	Poll Interval for Requests, MINPOLL=-4 and MAXPOLL=17	Bit:23:15	RW
-	Reserved, read 0	Bit:15:6	RO
MULTICAST	Support Multicast requests/responses	Bit:5	RW
UNICAST	Support Unicast requests/responses	Bit:4	RW
-	Reserved, read 0	Bit:3:2	RO
IP_MODE	IPv4 = 1, IPv6 = 2, 0&3 are illegal	Bit:1:0	RW

3.2.1.10 NTP Client VLAN Configuration Registers

VLAN configuration of the NTP packets.

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

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

3.2.1.11NTP Client MAC1 Registers

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

E.g. 0x01234567 => MAC: 67:45:32:01:XX:XX.

Ntp ClientConfigMac1 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: 0x008C																															

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

3.2.1.12NTP Client MAC2 Registers

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

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

Ntp ClientConfigMac2 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: 0x0090																															

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

3.2.1.13NTP Client Config IP Registers

IP address of the NTP Client. Used as source IP. LSB is transferred first on the network. This is the full IP address for IPv4 and the highest part of IPv6

Ntp ClientConfigIp 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: 0x0094																															

Name	Description	Bits	Access
IP(3)	IP Byte 3 (ipv4 and Ipv6)	Bit:31:24	RW
IP(2)	IP Byte 2 (ipv4 and Ipv6)	Bit:23:16	RW
IP(1)	IP Byte 1 (ipv4 and Ipv6)	Bit:15:8	RW
IP(0)	IP Byte 0 (ipv4 and Ipv6)	Bit:7:0	RW

3.2.1.14 NTP Client Config IP V6 1 Register

IPv6 address of the node. Used as source IP if in IPv6 mode, otherwise ignored. LSB is transferred first on the network.

Ntp ClientConfigIpv61 Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP(7)								IP(6)								IP(5)								IP(4)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x0098																															

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

3.2.1.15 NTP Client Config IP V6 2 Register

IPv6 address of the node. Used as source IP if in IPv6 mode, otherwise ignored. LSB is transferred first on the network.

Ntp ClientConfigIpv62 Reg																															
Reg Description																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP(11)								IP(10)								IP(9)								IP(8)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x009C																															

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

3.2.1.16NTP Client Config IP V6 3 Register

IPv6 address of the node. Used as source IP if in IPv6 mode, otherwise ignored. LSB is transferred first on the network.

Ntp ClientConfigIpv63 Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP(15)								IP(14)								IP(13)								IP(12)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00A0																															

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

3.2.1.17NTP Client Config Server MAC1 Register

MAC address of the NTP Server . LSB is transferred first on the network.

E.g. 0x01234567 => MAC: 67:45:32:01:XX:XX.

Ntp ClientConfigServerMac1 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
SERVER_MAC(3)								SERVER_MAC(2)								SERVER_MAC(1)								SERVER_MAC(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00A4																															

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

3.2.1.18NTP Client Config Server MAC2 Register

MAC address of the NTP Server . LSB is transferred first on the network.
E.g. 0x0004567 => MAC: XX:XX:XX:67:45.

Ntp ClientConfigServerMac2 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
																SERVER_MAC(5)								SERVDER_MAC(4)							
RO																RW								RW							
Reset: 0x00000000																															
Offset: 0x00A8																															

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

3.2.1.19NTP Client Config Server IP Register

IP address of the NTP Server. LSB is transferred first on the network. This is the full IP address for IPv4 and the highest part of IPv6

Ntp ClientConfigServerMac1 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
SERVER_MAC(3)								SERVER_MAC(2)								SERVER_MAC(1)								SERVER_MAC(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00A4																															

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

3.2.1.20 NTP Client Config Server IPv6 1 Register

IP address of the NTP Server. Used if in IPv6 mode, otherwise ignored. LSB is transferred first on the network.

Ntp ClientConfigServerIpv61 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
SERVER_IP(7)								SERVER_IP(6)								SERVER_IP(5)								SERVER_IP(4)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00B0																															

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

3.2.1.21NTP Client Config Server IPv6 2 Register

IP address of the NTP Server. Used if in IPv6 mode, otherwise ignored. LSB is transferred first on the network.

Ntp ClientConfigServerIpv62 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
SERVER_IP(11)								SERVER_IP(10)								SERVER_IP(9)								SERVER_IP(8)							
RW								RW								RW								RW							
Reset: 0x00000000																													
Offset: 0x00B4																													

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

3.2.1.22 NTP Client Config Server IPv6 3 Register

IP address of the NTP Server. Used if in IPv6 mode, otherwise ignored. LSB is transferred first on the network.

Ntp ClientConfigServerIpv63 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
SERVER_IP(15)								SERVER_IP(14)								SERVER_IP(13)								SERVER_IP(12)							
RW								RW								RW								RW							
Reset: 0x00000000																													
Offset: 0x00B8																													

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

3.2.1.23 NTP Client Config Subnet Mask Register

Subnet mask of the Client's IP address. This is the full subnet mask for IPv4 and the highest part of IPv6

Ntp ClientConfigSubnetMask 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
SUBNET_MASK(3)								SUBNET_MASK(2)								SUBNET_MASK(1)								SUBNET_MASK(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00BC																															

Name	Description	Bits	Access
SUBNET_MASK(3)	Subnet mask Byte 3 (ipv4 and Ipv6)	Bit:31:24	RW
SUBNET_MASK(2)	Subnet mask Byte 2 (ipv4 and Ipv6)	Bit:23:16	RW
SUBNET_MASK(1)	Subnet mask Byte 1 (ipv4 and Ipv6)	Bit:15:8	RW
SUBNET_MASK(0)	Subnet mask Byte 0 (ipv4 and Ipv6)	Bit:7:0	RW

3.2.1.24 NTP Client Subnet Mask IPv6 1 Register

Subnet mask of the Client's IP address. Used if in IPv6 mode, otherwise ignored.

Ntp ClientConfigSubnetMaskv61 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
SUBNET_MASK(7)								SUBNET_MASK(6)								SUBNET_MASK(5)								SUBNET_MASK(4)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00C0																															

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

3.2.1.25 NTP Client Subnet Mask IPv6 2 Register

Subnet mask of the Client's IP address. Used if in IPv6 mode, otherwise ignored.

Ntp ClientConfigSubnetMaskv62 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
SUBNET_MASK(11)								SUBNET_MASK(10)								SUBNET_MASK(9)								SUBNET_MASK(8)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00C4																															

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

3.2.1.26 NTP Client Subnet Mask IPv6 3 Register

Subnet mask of the Client's IP address. Used if in IPv6 mode, otherwise ignored.

Ntp ClientConfigSubnetMaskv63 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
SUBNET_MASK(15)								SUBNET_MASK(14)								SUBNET_MASK(13)								SUBNET_MASK(12)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00C8																															

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

3.2.1.27 NTP Client Config Gateway IP Register

Gateway IP of the NTP Client's local network. This is the full IP address for IPv4 and the highest part of IPv6.

Ntp ClientConfigGatewayIp 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
GATEWAY_IP(3)								GATEWAY_IP(2)								GATEWAY_IP(1)								GATEWAY_IP(0)							
RW								RW								RW								RW							
Reset: 0x00000000																															
Offset: 0x00CC																															

Name	Description	Bits	Access
GATEWAY_IP(3)	Gateway IP Byte 3 (ipv4 and Ipv6)	Bit:31:24	RW
GATEWAY_IP(2)	Gateway IP Byte 2 (ipv4 and Ipv6)	Bit:23:16	RW
GATEWAY_IP(1)	Gateway IP Byte 1 (ipv4 and Ipv6)	Bit:15:8	RW
GATEWAY_IP(0)	Gateway IP Byte 0 (ipv4 and Ipv6)	Bit:7:0	RW

3.2.1.28 NTP Client Config Gateway IPv6 1 Register

Gateway IP of the NTP Client's local network. Used if in IPv6 mode, otherwise ignored.

Ntp ClientConfigGatewayIpv61 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
GATEWAY_IP(7)								GATEWAY_IP(6)								GATEWAY_IP(5)								GATEWAY_IP(4)							
RW								RW								RW								RW							
Reset: 0x00000000																													
Offset: 0x00D0																													

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

3.2.1.29 NTP Client Config Gateway IPv6 2 Register

Gateway IP of the NTP Client's local network. Used if in IPv6 mode, otherwise ignored.

Ntp ClientConfigGatewayIpv62 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
GATEWAY_IP(11)								GATEWAY_IP(10)								GATEWAY_IP(9)								GATEWAY_IP(8)							
RW								RW								RW								RW							
Reset: 0x00000000																													
Offset: 0x00D4																													

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

3.2.1.30 NTP Client Config Gateway IPv6 3 Register

Gateway IP of the NTP Client's local network. Used if in IPv6 mode, otherwise ignored.

Ntp ClientConfigGatewayIpv63 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
GATEWAY_IP(15)								GATEWAY_IP(14)								GATEWAY_IP(13)								GATEWAY_IP(12)							
RW								RW								RW								RW							
Reset: 0x00000000																													
Offset: 0x00D8																													

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

3.2.1.31NTP Client Config Filter Coef B0 Register

Coefficient B0 of the 2nd order IIR Filter.

Ntp ClientConfigFilterCoefB0 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
FILTER_COEF_B0																															
RW																															
Reset: 0x008A25E5																															
Offset: 0x00D8																															

Name	Description	Bits	Access
FILTER_COEF_B0	IIR Filter Coefficient B0 in signed fix point format Q4.27 format	Bit:31:0	RW

3.2.1.32 NTP Client Config Filter Coef B1 Register

Coefficient B1 of the 2nd order IIR Filter.

Ntp ClientConfigFilterCoefB1 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
FILTER_COEF_B1																															
RW																															
Reset: 0x01144BC9																															
Offset: 0x00DC																															

Name	Description	Bits	Access
FILTER_COEF_B1	IIR Filter Coefficient B1 in signed fix point format Q4.27 format	Bit:31:0	RW

3.2.1.33 NTP Client Config Filter Coef B2 Register

Coefficient B2 of the 2nd order IIR Filter.

Ntp ClientConfigFilterCoefB2 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
FILTER_COEF_B2																															
RW																															
Reset: 0x008A35E5																															
Offset: 0x00E0																															

Name	Description	Bits	Access
FILTER_COEF_B2	IIR Filter Coefficient B2 in signed fix point format Q4.27 format	Bit:31:0	RW

3.2.1.34 NTP Client Config Filter Coef A1 Register

Coefficient A1 of the 2nd order IIR Filter.

Ntp ClientConfigFilterCoefA1 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
FILTER_COEF_A1																															
RW																															
Reset: 0xF6DB2EBA																															
Offset: 0x00E4																															

Name	Description	Bits	Access
FILTER_COEF_A1	IIR Filter Coefficient A1 in signed fix point format Q4.27 format	Bit:31:0	RW

3.2.1.35 NTP Client Config Filter Coef A2 Register

Coefficient A2 of the 2nd order IIR filter.

Ntp ClientConfigFilterCoefA2 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
FILTER_COEF_A2																															
RW																															
Reset: 0x034D68D7																															
Offset: 0x00E8																															

Name	Description	Bits	Access
FILTER_COEF_A2	IIR Filter Coefficient A2 in signed fix point format Q4.27 format	Bit:31:0	RW

3.2.1.36 NTP Client Config PI Servo P Register

PI Servo Factor for Proportional Part

Ntp ClientConfigPiFactorP 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
FILTER_FACTOR_P																													
RW																													
Reset: 0x00002000																													
Offset: 0x00F0																													

Name	Description	Bits	Access
FILTER_FACTOR_P	PI Factor for Proportional Part in fraction of 2**16	Bit:31:0	RW

3.2.1.37 NTP Client Config PI Servo I Register

PI Servo Factor for Integral Part

Ntp ClientConfigPiFactorI 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
FILTER_FACTOR_I																													
RW																													
Reset: 0x00000800																													
Offset: 0x00F4																													

Name	Description	Bits	Access
FILTER_FACTOR_I	PI Factor for Integral Part in fraction of 2**16	Bit:31:0	RW

3.2.1.38 NTP Client UTC Information Control Register

Control Register for the UTC Information.

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

Name	Description	Bits	Access
READ_DONE	UTC Info was read	Bit: 31	RO
READ	Read UTC Info (autocleared)	Bit: 30	RW
-	Reserved, read 0	Bit 29:2	RO
UTC_SMEARING_VAL	UTC Smearing mode valid (autocleared)	Bit 1	RW
UTC_INFO_VAL	UTC Info valid (autocleared)	Bit 0	RW

3.2.1.39 NTP Client UTC Information Register

NTP UTC Information, depending on the ExtSync_Gen the UTC info can be set or is just read only. The only bit which is always settable (given that smearing is supported) is the UTC_LEAP_SMEARING bit

Ntp ClientUtcInfo Reg																													
Reg Description																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UTC_OFFSET																-		UTC_OFFSET_VALID	LEAP59	LEAP61	LEAP59_IN_PROGRESS	LEAP61_IN_PROGRESS	UTC_LEAP_SMEARING	-							
RW																RO		RW	RW	RW	RO	RO	RW	RO							
Reset: 0x00000000																															
Offset: 0x0104																															

Name	Description	Bits	Access
UTC_OFFSET	UTC Offset	Bit: 31:16	RW
-	Reserved, read 0	Bit 15:14	RO
UTC_OFFSET_VALID	If UTC Offset is valid	Bit: 13	RW
LEAP59	Whether a Leap 59 will happen	Bit: 12	RW
LEAP61	Whether a Leap 61 will happen	Bit: 11	RW
LEAP59_IN_PROGRESS	Whether a Leap 59 is in progress (smearing)	Bit: 10	RO
LEAP61_IN_PROGRESS	Whether a Leap 61 is in progress(smearing)	Bit: 9	RO
UTC_LEAP_SMEARING	If UTC Leap second smearing shall be done	Bit: 8	RW

-	Reserved, read 0	Bit 7:0	RO

3.2.1.40 NTP Client Offset from Server Register

Calculated offset of the NTP Client's clock from the NTP Server.

Ntp ClientOffsetFromServer 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_FROM_SERVER																															
RO																															
Reset: 0x00000000																															
Offset: 0x0200																															

Name	Description	Bits	Access
OFFSET_FROM_SERVER	Offset of the NTP client from the NTP server	Bit: 31:0	RO

3.2.1.41 NTP Client Mean Roundrip Delay Register

Moving average of the calculated roundtrip delay.

Ntp ClientMeanRoundtripDelay 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
MEAN_ROUNDTRIP_DELAY																															
RO																															
Reset: 0x00000000																															
Offset: 0x0204																															

Name	Description	Bits	Access
MEAN_ROUNDTRIP_DELAY	Mean value of roundtrip delay	Bit: 31:0	RO

4 Design Description

The following chapters describe the internals of the NTP Client starting with the Top Level, which is a collection of subcores, followed by the description of all subcores.

4.1 Top Level – NTP Client

4.1.1 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
ResetBuffer_Gen	boolean	1	If the reset shall be synchronized to the System Clock in the core
ArpSupport_Gen	boolean	1	If the core shall support sending ARP requests and responses
IcmpSupport_Gen	boolean	1	If the core shall support sending ICMP requests and responses
RoutingSupport_Gen	boolean	1	If the NTP Server is outside the client's LAN, the Gateway IP and the subnet mask need to be provided
UtcInitialUtcOffset_Gen	integer	1	Initial UTC offset value which is used
UtcLeapSmearing_Gen	boolean	1	If smearing for the UTC Leap Second handling is supported
UtcLeapSmearingRatePpb_Gen	natural	1	UTC Leap Second smearing rate in ns/60s
UnicastSupport_Gen	boolean	1	If the core shall accept Unicast NTP responses
MulticastSupport_Gen	boolean	1	If the core shall accept Multicast NTP responses
Layer3v4Support_Gen	boolean	1	If Ipv4 shall be supported

Layer3v6Support_Gen	boolean	1	If Ipv6 shall be supported
VlanSupport_Gen	boolean	1	Support for VLAN
PassThrough_Gen	boolean	1	If frames after the NTP shall be passed further
InternalLoopback_Gen	boolean	1	If loopback of the received packets to transmission shall be supported (possible only if PassThrough_Gen is disabled)
RedTagSupport_Gen	boolean	1	If the core shall support redundancy tag.
StaticConfig_Gen	boolean	1	If Static Configuration or AXI is used: true = Static, false = AXI
AdvancedFilter_Gen	boolean	1	If average over stored data shall be used for filtering the delay
AverageWindowNanosecond_Gen	natural	1	The allowed window of delay difference in nanoseconds
ExtSync_Gen	boolean	1	If external UTC information is used to sync: true = external, false = internal (register only)
ClockClkPeriodNanosecond_Gen	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
ClockClkPeriodFractNum_Gen	natural	1	Fractional Clock Period Numerator (0 if integer)
ClockClkPeriodFractDeNum_Gen	natural	1	Fractional Clock Period Denumerator (0 if integer)
RxDelayNanosecond10_Gen	integer	1	PHY receive delay (10Mbit)
RxDelayNanosecond100_Gen	integer	1	PHY receive delay (100Mbit)
RxDelayNanosecond1000_Gen	integer	1	PHY receive delay (1000Mbit)
TxDelayNanosecond10_Gen	integer	1	PHY transmit delay (10Mbit)
TxDelayNanosecond100_Gen	integer	1	PHY transmit delay (100Mbit)

TxDelayNanosec-ond1000_Gen	integer	1	PHY transmit delay (1000Mbit)
HighResSupport_Gen	boolean	1	If a high-resolution clock SysClkNx with alignment to SysClk is used
HighResFreqMulti-ply_Gen	natural	1	Multiplication factor of the high-resolution clock com-pared to SysClk
AxiAddressRange Low_Gen	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	std_logic_vector	32	AXI Base Address plus Regis-terset Size Default plus 0xFFFF
IoFf	boolean	1	If Input/Output filp-flops shall be added at the interface adapters
Sim_Gen	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis

Table 5: Parameters

4.1.2 Structured Types

4.1.2.1 Clk_Time_Type

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

Field Name	Type	Size	Description
Second	std_logic_vector	32	Seconds of time
Nanosecond	std_logic_vector	32	Nanoseconds of time
Fraction	std_logic_vector	2	Fraction numerator (mostly not used)
Sign	std_logic	1	Positive or negative time, 1 = negative, 0 = positive.

TimeJump	std_logic	1	Marks when the clock makes a time jump (mostly not used)

Table 6: Clk_Time_Type

4.1.2.2 Clk_TimeAdjustment_Type

Defined in Clk_Package.vhd of library ClkLib
Type represents the time adjustment that should be applied to the Clock over a specified interval.

Field Name	Type	Size	Description
TimeAdjustment	Clk_Time_Type	1	Seconds of time
Interval	std_logic_vector	32	Nanoseconds over which the adjustment should be spread.
Valid	std_logic	1	If the adjustment is valid

Table 7: Clk_Time_Adjustment_Type

4.1.2.3 Clk_UtcInfo_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
UtcOffset	std_logic_vector	15	UTC Offset in seconds
UtcOffsetValid	std_logic	1	UTC Offset is valid
Leap59	std_logic	1	Leap59 indication
Leap61	std_logic	1	Leap61 indication

Table 8: Clk_UtcInfo_Type

4.1.2.4 Ntp_ClientStaticConfig_Type

Defined in Ntp_ClientAddrPackage.vhd of library NtpLib
This is the type used for static configuration.

Field Name	Type	Size	Description

MulticastModeEnable	std_logic	1	If Multicast mode shall be used
UnicastModeEnable	std_logic	1	If Unicast mode shall be used
Ipv4Enable	std_logic	1	If Ipv4 shall be used
Ipv6Enable	std_logic	1	If Ipv6 shall be used
Vlan	Ntp_Vlan_Type	1	The Pcp,Dei and Vid of the VLAN
VlanEnable	std_logic	1	If VLAN shall be used
Mac	Com-mon_Byte_Type	6	The source MAC to be used index 0 = MSB
Ip	Com-mon_Byte_Type	16	The source IP to be used, 4 bytes Ipv4, 16 bytes Ipv6, index 0 = MSB
ServerMac	Com-mon_Byte_Type	6	The MAC address of the NTP Server
ServerIp	Com-mon_Byte_Type	16	The IPv4 (4 Bytes)/IPv6 (16 Bytes) address of the NTP Server, index 0 = MSB
SubnetMask	Com-mon_Byte_Type	16	The IPv4 (4 Bytes)/IPv6 (16 Bytes) subnet mask, index 0 = MSB
GatewayIp	Com-mon_Byte_Type	16	The IPv4 (4 Bytes)/IPv6 (16 Bytes) address of the gateway, index 0 = MSB
PollInterval	std_logic_vector	8	Poll Interval for Responses in Broadcast mode
UtcLeapSmearing	std_logic	1	If UTC leap second smearing shall be done
UtcInfo	Clk_UtcInfo_Typ e	1	The UTC information like offset, leap second etc.
FilterCoeff	Com-mon_Long_Type	5	The 5 IIR coefficients (0=B0, 1=B1, 2=B2, 3=A1, 4=A2) of the adjustment filter
PiFactor	Com-mon_Long_Type	2	The 2 PI Factors (0=P, 1=I) of the adjustment PI Servo

Table 9: Ntp_ClientStaticConfig_Type

4.1.2.5 Ntp_ClientStaticConfigVal_Type

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

Field Name	Type	Size	Description
Enable_Val	std_logic	1	Enables the NTP Client
Mode_Val	std_logic	1	If the Mode shall be set
Vlan_Val	std_logic	1	If the VLAN shall be set
Mac_Val	std_logic	1	If the MAC shall be set
Ip_Val	std_logic	1	If the IP shall be set
Server_Mac_Val	std_logic	1	If the Server MAC shall be set
Server_Ip_Val	std_logic	1	If the Server IP shall be set
SubnetMask_Val	std_logic	1	If the Subnet Mask shall be set
GatewayIp_Val	std_logic	1	If the Gateway IP shall be set
UtcInfo_Val	std_logic	1	If UtcInfo shall be set
FilterCoef_Val	std_logic	1	If FilterCoeff shall be set
PiFactor_Val	std_logic	1	If PiFactor shall be set

Table 10: Ntp_ClientStaticConfigVal_Type

4.1.2.6 Ntp_ClientStaticStatus_Type

Defined in Ntp_ClientAddrPackage.vhd of library NtpLib
This is the type used for static status supervision.

Field Name	Type	Size	Description
CoreInfo	Clk_CoreInfo_ Type	1	Infor about the Cores state
UtcInfo	Clk_UtcInfo_ Type	1	The UTC information like offset, leap second etc.
UtcLeap59InProgress	std_logic	1	Leap59 is in progress
UtcLeap61InProgress	std_logic	1	Leap61 is in progress

Table 11: Ntp_ClientStaticStatus_Type

4.1.2.7 Ntp_ClientStaticStatusVal_Type

Defined in Ntp_ClientAddrPackage.vhd of library NtpLib

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
UtcInfo_Val	std_logic	1	UTC Info valid

Table 12: Ntp_ClientStaticStatusVal_Type

4.1.3 Entity Block Diagram

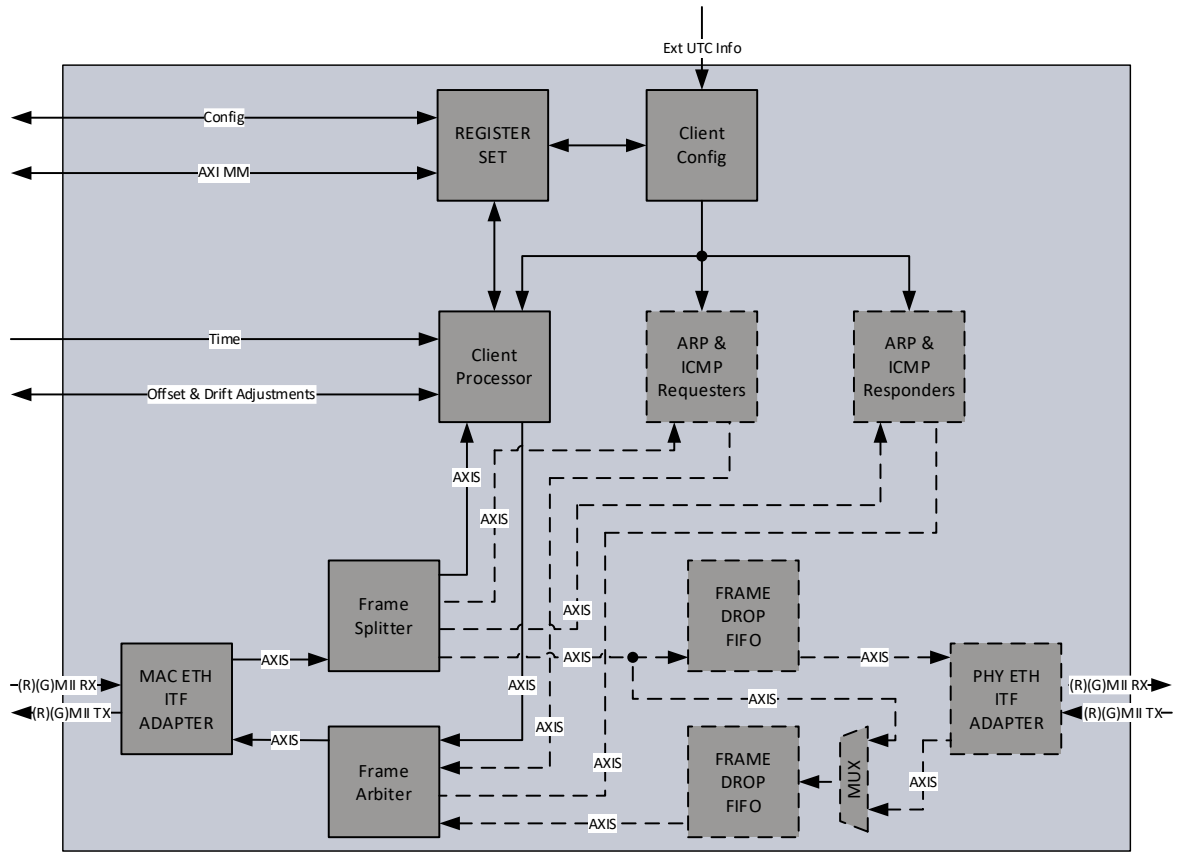

Figure 6: NTP Client

4.1.4 Entity Description

Frame Splitter

This module splits one AXI stream input to multiple AXI stream outputs, allowing the ARP Responder to receive ARP requests, the ARP Requester to receive ARP responds, to forward frames through the core and to forward frames to the client processor.

Frame Arbiter

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

Frame Drop Fifo

This module is AXI stream frame FIFO. It is always ready to receive data, so no throttling of the stream is necessary. It handles additionally a user drop-flag which allows explicitly dropping the current frame. In this design it is only used as store-and-forward FIFO, but with the drop-flag. The FIFO on the RX path allows to drop NTP and corrupted frames (they shall not be forwarded). The FIFO on the TX path buffers the incoming data stream for the case when an internal frame is processed (send a NTP or ARP). The FIFO in the loopback path also allows to drop NTP and corrupted frames.
The FIFOs are optional either for the through put mode or for an internal loopback.

Client Processor

This block is the heart the NTP Client. The Client Processor generates in periodic intervals the NTP request which includes the transmit timestamp. When the NTP Server replies with the corresponding NTP response, the client timestamps the incoming frame and extracts the information of the NTP response. Based on the timestamps and the received information, the client calculates the average delay to the server and the offset and drift adjustments to be applied to the client's clock. Additionally, UTC handling is done by the Client Processor block.
See 4.2.1 for more details.

Client Config

This block controls the configuration of the client. There are multiple sources possible, like the configuration via the register set, static configuration from externally or also UTC information from externally.
See 4.2.2 for more details.

ARP & ICMP Requesters

These modules generates ARP and ICMP requests to identify the MAC address of the NTP Server. This module is optional.
See 4.2.3 for more details.

ARP & ICMP Responders

These module answers ARP or ICMP requests with ARP or ICMP responses. This module is optional.
See 4.2.3.1 for more details.

MAC & PHY Ethernet Interface Adapter

This module converts the Media Independent Interface (MII) to AXI stream and vice versa. In parallel it has a SFD detector for each path which generates an event when a SFD is detected; this is used for timestamping in the RX/TX Processors. It is also in charge of generating correct Interframe Gaps and a Preamble with SFD. See 4.2.5 for more details.

Registerset

This module is an AXI4Lite Memory Mapped Slave. It allows to configure the NTP Client. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the Client is done via signals and can be easily done from within the FPGA without CPU. If in AXI mode, a AXI Master has to configure the Client with AXI writes to the registers, which is typically done by a CPU See 4.2.6 for more details.

4.1.5 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ResetBuffer_Gen	-	boolean	1	If the reset shall be synchronized to the System Clock in the core
ArpSupport_Gen	-	boolean	1	If the core shall support sending ARP requests and responses
IcmpSupport_Gen	-	boolean	1	If the core shall support sending ICMP requests and responses
RoutingSupport_Gen	-	boolean	1	If the NTP Server is outside the client's LAN, the Gateway IP and the subnet mask need to be

				provided
UtcInitialUtcOffset_Gen	-	integer	1	Initial UTC offset value which is used
UtcLeapSmearing_Gen	-	boolean	1	If smearing for the UTC Leap Second handling is supported
UtcLeapSmearingRatePpb_Gen	-	natural	1	UTC Leap Second smearing rate in ns/s (ppb)
UnicastSupport_Gen	-	boolean	1	If the core shall accept Unicast NTP responses
MulticastSupport_Gen	-	boolean	1	If the core shall accept Multicast NTP responses
Layer3v4Support_Gen	-	boolean	1	If Ipv4 shall be supported
Layer3v6Support_Gen	-	boolean	1	If Ipv6 shall be supported
VlanSupport_Gen	-	boolean	1	Support for VLAN
PassThrough_Gen	-	boolean	1	If frames shall be passed through the core
InternalLoopback_Gen	-	boolean	1	If loopback of the received packets to transmission shall be supported (possible only if PassThrough_Gen is disabled)
RedTagSupport_Gen	-	boolean	1	If the core shall support redundancy tag.
StaticConfig_Gen	-	boolean	1	If Static Configuration or AXI is used: true = Static, false = AXI

AdvancedFilter_Gen	-	Boolean	1	If average over stored data shall be used for filtering the delay
AverageWindowNanosecond_Gen	-	Natural	1	The allowed window of delay difference in nanoseconds
ExtSync_Gen	-	boolean	1	If external UTC information is used to sync: true = external, false = internal (register only)
ClockClkPeriodNanosecond_Gen	-	natural	1	Clock Period in Nanosecond: Default for 50 MHz = 20 ns
ClockClkPeriodFractNum_Gen	-	natural	1	Fractional Clock Period Numerator (0 if integer)
ClockClkPeriodFractDeNum_Gen	-	natural	1	Fractional Clock Period Denumerator (0 if integer)
RxDelayNanosecond10_Gen	-	integer	1	PHY receive delay (10Mbit)
RxDelayNanosecond100_Gen	-	integer	1	PHY receive delay (100Mbit)
RxDelayNanosecond1000_Gen	-	integer	1	PHY receive delay (1000Mbit)
TxDelayNanosecond10_Gen	-	integer	1	PHY transmit delay (10Mbit)
TxDelayNanosecond100_Gen	-	integer	1	PHY transmit delay (100Mbit)
TxDelayNanosecond1000_Gen	-	integer	1	PHY transmit delay (1000Mbit)
HighResSupport_Gen	-	boolean	1	If a high-resolution clock SysClkNx with alignment to SysClk

				is used
HighResFreqMulti-ply_Gen	-	natural	1	Multiplication factor of the high-resolution clock compared to SysClk
AxiAddressRang Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size Default plus 0xFFFF
IoFf	-	boolean	1	If Input/Output filp-flops shall be added at the interface adapters
Sim_Gen	-	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysClkNx_ClkIn	in	std_logic	1	High-resolution clock (multiple of Sys Clock)
SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Ntp_Client StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Ntp_Client StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Ntp_Client StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Ntp_Client StaticStatusVal _Type	1	Static Status valid
Timer				

Timer1ms_EvtIn	in	std_logic	1	1 millisecond Timer event, aligned to Adjusted Clock
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
(R)(G)Mii RX Clk/Rst Input				
(R)(G)MiiRxClk_ClkIn	in	std_logic	1	RX Clock
(R)(G)MiiRxRstN_RstIn	in	std_logic	1	Reset aligned with RX Clock
(R)(G)Mii TX Clk/Rst Input				
(R)(G)MiiTxClk_ClkIn	in	std_logic	1	TX Clock
(R)(G)MiiTxRstN_RstIn	in	std_logic	1	Reset aligned with TX Clock
(R)(G)Mii RX Data Input/Output				
(R)(G)MiiRxDv_Ena	in/ out	std_logic	1	RX Data valid
(R)(G)MiiRxErr_Ena	in/ out	std_logic	1	RX Error
(R)(G)MiiRxData_Dat	in/ out	std_logic_vector	2-8	RX Data MII:4, RMII:2, GMII:8, RGMII:4
(R)(G)MiiCol_Dat	in/ out	std_logic	1	Collision
(R)(G)MiiCrs_Dat	in/ out	std_logic	1	Carrier Sense
(R)(G)Mii TX Data Input/Output				
(R)(G)MiiTxEn_Ena	in/ out	std_logic	1	TX Data valid
(R)(G)MiiTxErr_Ena	in/ out	std_logic	1	TX Error
(R)(G)MiiTxData_Dat	in/ out	std_logic_vector	2-8	TX Data MII:4, RMII:2, GMII:8, RGMII:4
External UTC Info Input				
UtcInfoExtSync_DatIn	in	Clk_UtcInfo_Type	1	UTC information input from external

UtcInfoExtSync_ValIn	in	std_logic	1	UTC information input valid from external
UTC Time Output				
UtcTime_DatOut	out	Clk_Time_Type	1	UTC Time output
UtcTime_ValOut	out	std_logic	1	UTC Time 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 Re-sponse

Time Adjustment Output				
TimeAdjust-ment_DatOut	out	Clk_TimeAdjustment_Type	1	Time adjustment to be applied to Ad-justed Clock
TimeAdjust-ment_ValOut	out	Std_logic	1	TimeAdjust-ment_DatOut valid
Offset Adjustment Output				
OffsetAdjust-ment_DatOut	out	Clk_TimeAdjustment_Type	1	Offset adjustment to be applied to Adjusted Clock
OffsetAdjust-ment_ValOut	out	Std_logic	1	OffsetAdjust-ment_DatOut valid
Drift Adjustment Output				
DriftAdjust-ment_DatOut	out	Clk_TimeAdjustment_Type	1	Drift adjustment to be applied to Ad-justed Clock
DriftAdjust-ment_ValOut	out	Std_logic	1	DriftAdjust-ment_DatOut valid
Offset Adjustment Input				
OffsetAdjust-ment_DatIn	in	Clk_TimeAdjustment_Type	1	Offset adjustment feedback from the Adjusted Clock
OffsetAdjust-ment_ValIn	in	Std_logic	1	OffsetAdjust-ment_DatIn valid
Drift Adjustment Input				
DriftAdjust-ment_DatIn	in	Clk_TimeAdjustment_Type	1	Drift adjustment feedback from the Adjusted Clock
DriftAdjustment_ValIn	in	Std_logic	1	DriftAdjust-ment_DatIn valid

Table 13: NTP Client

4.2 Design Parts

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

4.2.1 Client Processor

4.2.1.1 Entity Block Diagram

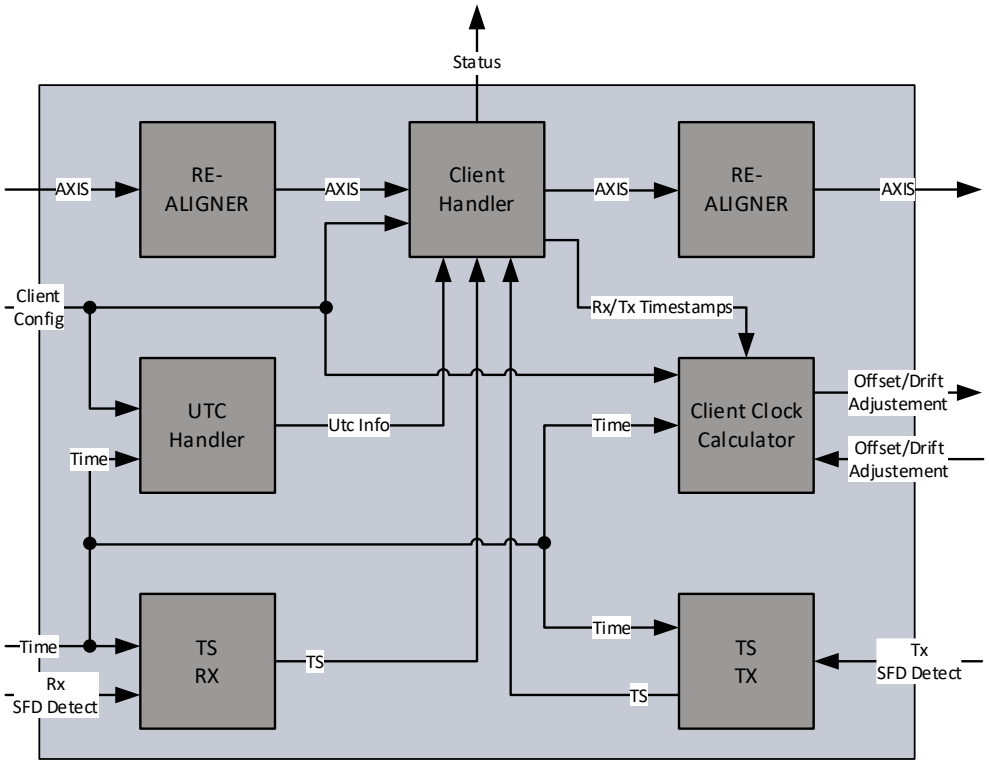

Figure 7: Client Processor

4.2.1.2 Entity Description

RX Timestamper

This module takes a snapshot of the UTC time when the SFD detected event was asserted by the Interface Adapter. The timestamp is buffered and aligned with the incoming frame.

TX Timestamper

This module takes a snapshot of the UTC time when the SFD detected event was asserted by the Interface Adapter. The timestamp is buffered and aligned with the outgoing frame.

Client Handler

This module generates in periodic intervals the NTP request which includes the transmit timestamp. When the NTP Server replies with the corresponding NTP response, the client timestamps the incoming frame and extracts the information of the NTP response.

Client Clock Calculator

This module receives the NTP request/response timestamps and calculates the delay. An average delay is afterwards updated to filter out outliers, and an IIR Filter filters the adjustments to be applied to the Adjusted Clock.

UTC Handler

This module handles based on the time (TAI) from the Counter Clock the UTC Offset and the leap seconds. Additionally, smearing of the leap second is done by the UTC handler once enabled.

Re-Aligner

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

4.2.1.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriodNano-second_Gen	-	natural	1	Clock Period in Nanosecond: De-fault for 50 MHz = 20 ns
AdvancedFilter_Gen	-	Boolean	1	If average over

				stored data shall be used for filtering the delay
AverageWindowNanosecond_Gen	-	Natural	1	The allowed window of delay difference in nanoseconds
RxDelayNanosecond10_Gen	-	integer	1	PHY receive delay (10Mbit)
RxDelayNanosecond100_Gen	-	integer	1	PHY receive delay (100Mbit)
RxDelayNanosecond1000_Gen	-	integer	1	PHY receive delay (1000Mbit)
TxDelayNanosecond10_Gen	-	integer	1	PHY transmit delay (10Mbit)
TxDelayNanosecond100_Gen	-	integer	1	PHY transmit delay (100Mbit)
TxDelayNanosecond1000_Gen	-	integer	1	PHY transmit delay (1000Mbit)
HighResSupport_Gen	-	boolean	1	If a high-resolution clock SysClkNx with alignment to SysClk is used
HighResFreqMultiply_Gen	-	natural	1	Multiplication factor of the high-resolution clock compared to SysClk
RxNoCrcCheck_Gen	-	boolean	1	No CRC check at the received NTP messages
TxNoCrcGen_Gen	-	boolean	1	No CRC generation at the transmitted NTP messages (send “0” instead)
UtcInitialUtcOffset_Gen	-	integer	1	Initial UTC offset value which is used
UtcLeapSmearing_Gen	-	boolean	1	If smearing for the UTC Leap Second handling is support-

				ed
UtcLeapSmear-ingRatePpb_Gen	-	natural	1	UTC Leap Second smearing rate in ns/60s
UnicastSupport_Gen	-	boolean	1	If the core shall accept Unicast Response NTP
MulticastSupport_Gen	-	boolean	1	If the core shall accept Multicast Response NTP
Lay-er3v4Support_Gen	-	boolean	1	If Ipv4 shall be supported
Lay-er3v6Support_Gen	-	boolean	1	If Ipv6 shall be supported
VlanSupport_Gen	-	boolean	1	Support for VLAN
RedTagSupport_Gen	-	boolean	1	If the core shall support redundancy tag.
Sim_Gen	-	boolean	1	If in Testbench simulation mode: true = Simulation, false = Synthesis
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysClkNx_ClkIn	in	std _logic	1	High resolution Timestamping Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Enable Input				
Enable_EnaIn	in	std_logic	1	Core Enabled
Time Input				
ClockTime_DatIn	in	Clk_Time_Type	1	Adjusted Clock Time
ClockTime_ValIn	in	std_logic	1	Adjusted Clock Time valid
Timer				
Timer1ms_EvtIn	in	std_logic	1	Adjusted Clock aligned 1 millisecond Timer event

UTC Ignore				
UtcInfoIgnore_VatOut	out	std_logic	1	Ignore UTC info
UTC Time				
UtcTime_DatOut	out	Clk_Time_Type	1	UTC Time
UtcTime_ValOut	out	std_logic	1	UTC Time valid
Stratus Output				
RequestSent_ValOut	out	std_logic	1	NTP request has been sent
ResponseRe-ceived_ValOut	out	std_logic	1	NTP response has been received
Re-sponseMissed_Valout	out	std_logic	1	NTP Response has been missed
Ut-cLeap59InProgres_ValOut	out	std_logic	1	UTC Leap59 smear-ing in progress
Ut-cLeap61InProgres_ValOut	out	std_logic	1	UTC Leap61 smear-ing in progress
Link Speed Input				
LinkSpeed_DatIn	in	Common_ LinkSpeed_Type	1	Link Speed of the interface
Frame Input				
SfdDetectedRx_EvtIn	in	std_logic	1	Start of RX Frame Delimiter detected
Axis 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	
Drop Output				
Drop_ValOut	out	std_logic	1	If frame shall be dropped
Frame Output				
SfdDetectedTx_EvtIn	in	std_logic	1	Start of TX Frame Delimiter detected
Axis Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	

AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	
Selcted Input				
Selected_DatIn	in	std_logic	1	Select input from the frame Arbiter
Client Config Output				
ClientConfig_DatOut	out	Ntp_ClientConfig_Type	1	NTP Client configuration
ClientConfig_ValOut	out	std_logic	1	NTP Client configuration valid
Client Config Input				
ClientConfig_DatIn	in	Ntp_ClientConfig_Type	1	NTP Client configuration
Offset Adjustment Input				
OffsetAdjustment_DatIn	in	Clk_TimeAdjustment_Type	1	Offset adjustment feedback from the Adjusted Clock
OffsetAdjustment_ValIn	in	Std_logic	1	OffsetAdjustment_DatIn valid
Drift Adjustment Input				
DriftAdjustment_DatIn	in	Clk_TimeAdjustment_Type	1	Drift adjustment feedback from the Adjusted Clock
DriftAdjustment_ValIn	in	Std_logic	1	DriftAdjustment_DatIn valid

Table 14: Client Processor

4.2.2 Client Config

4.2.2.1 Entity Block Diagram

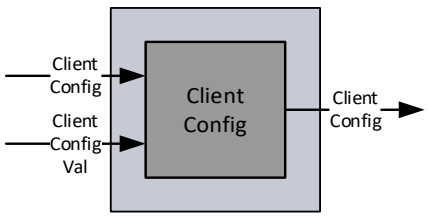

Figure 8: Client Config

4.2.2.2 Entity Description

Client Config

This module controls the configuration of the client. It checks if a configuration is supported and it stores the configured parameter once it is valid.

4.2.2.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
RoutingSupport_Gen	-	boolean	1	If the NTP Server is outside the client's LAN, the Gateway IP and the subnet mask need to be provided
UtcInitialUtcOff-set_Gen	-	integer	1	Initial UTC offset value which is used
UtcLeapSmear-ing_Gen	-	boolean	1	If smearing for the UTC Leap Second handling is support-ed
UnicastSupport_Gen	-	boolean	1	If the core shall support Unicast NTP
MulticastSupport_Gen	-	boolean	1	If the core shall support Multicast NTP
Lay-er3v4Support_Gen	-	boolean	1	If Ipv4 shall be supported
Lay-er3v6Support_Gen	-	boolean	1	If Ipv6 shall be supported
VlanSupport_Gen	-	boolean	1	Support for VLAN
ExtSync_Gen	-	boolean	1	If external UTC information is used to sync:

				true = external, false = internal (register only)
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Client Configuration Input				
ClientConfig_DatIn	in	Ntp_ClientConfig_Type	1	NTP Client Configuration
ClientConfig_ValIn	in	Ntp_ClientConfigVal_Type	1	NTP Client Configuration valid
Client Configuration Output				
ClientConfig_DatOut	in	Ntp_ClientConfig_Type	1	Stored NTP Client Configuration

Table 15: Client Configuration

4.2.3 ARP & ICMP Requesters

4.2.3.1 Entity Block Diagram

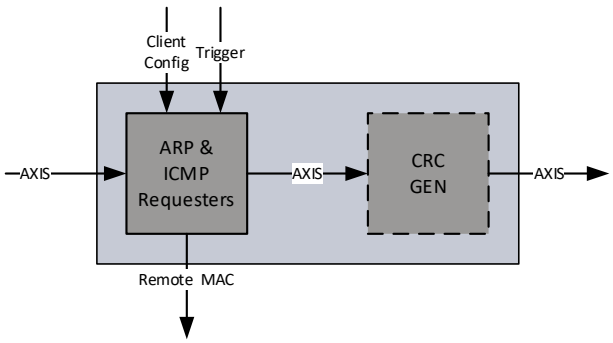

Figure 9: ARP Requester

4.2.3.2 Entity Description

ARP & ICMP Requesters

This module receives triggers to generate the ARP or ICMP requests and when it receives an ARP or ICMP response, it extracts the MAC address of the responder .

The ARP requester gets its configuration like MAC address and IP from the Client Config.

CRC GEN

This module generates the CRC of the ARP request.

4.2.3.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ClockClkPeriodNano-second_Gen	-	Natural	1	Clock Period in Nanosecond: De-fault for 50 MHz = 20 ns
NoCrc_Gen	-	boolean	1	No CRC generation
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Enable Input				
Enable_EnaIn	in	std_logic	1	Module enable
Trigger Input				
Trigger_ValIn	in	std_logic	1	Trigger for an ARP Request generation
Config Input				
RemoteIpv4_DatIn	in	Common_Byte_Type	4	The IPv4 address of the NTP Server, index 0 = MSB
RemoteIpv6_DatIn	in	Common_Byte_Type	16	The IPv6 address of the NTP Server, index 0 = MSB
LocalMAC_DatIn	in	Common_Byte_Type	6	The MAC address of the NTP client
LocalIpv4_DatIn	in	Common_Byte_Type	4	The IPv4 address of the NTP Client, index 0 = MSB
LocalIpv6_DatIn	in	Common_Byte_Type	16	The IPv6 address of the NTP Client,

				index 0 = MSB
Config Output				
RemoteMAC_DatOut	out	Common_Byte_Type	6	The MAC address of the NTP server
RemoteMAC_ValOut	out	std_logic	1	The MAC address of the NTP server is valid
RemoteMAC Timeout_ValOut	out	std_logic	1	The MAC address could not be re-solved
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	
Axi Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	
AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	

Table 16: ARP Requester

4.2.4 ARP Responder

4.2.4.1 Entity Block Diagram

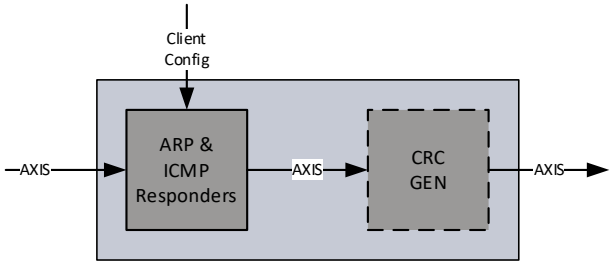

Figure 10: ARP Responder

4.2.4.2 Entity Description

ARP/ICMP Responder

These modules receives the ARP or ICMP requests and answer with a corresponding ARP or ICMP response to it. The ARP/ICMP responder gets its configuration like MAC address and IP from the Server Config.
These are two individual blocks

CRC GEN

This module generates the CRC of the ARP response.

4.2.4.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
NoCrc_Gen	-	boolean	1	No CRC generation
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock
SysRstN_RstIn	in	std_logic	1	System Reset
Enable Input				
Enable_EnaIn	in	std_logic	1	Module enable
Config Input				
LocalMAC_DatIn	in	Common_Byte_Type	6	The MAC address of the NTP client
LocalIpv4_DatIn	in	Common_Byte_Type	4	The IPv4 address of the NTP Client, index 0 = MSB
LocalIpv6_DatIn	in	Common_Byte_Type	16	The IPv6 address of the NTP Client, index 0 = MSB
Axis 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	
Axis Output				
AxisValid_ValOut	out	std_logic	1	AXI Stream frame output
AxisReady_ValIn	in	std_logic	1	
AxisData_DatOut	out	std_logic_vector	32	
AxisStrobe_ValOut	out	std_logic_vector	4	
AxisKeep_ValOut	out	std_logic_vector	4	
AxisLast_ValOut	out	std_logic	1	
AxisUser_DatOut	out	std_logic_vector	3	

Table 17: ARP Responder

4.2.5 Ethernet Interface Adapter

4.2.5.1 Entity Block Diagram

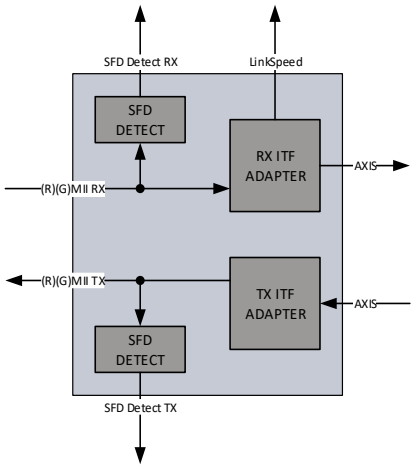

Figure 11: Ethernet Interface Adapter

4.2.5.2 Entity Description

SFD Detector

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

RX Interface Adapter

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

TX Interface Adapter

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

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

4.2.5.3 Entity Declaration

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

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

Table 18: Ethernet Interface Adapter

4.2.6 Registerset

4.2.6.1 Entity Block Diagram

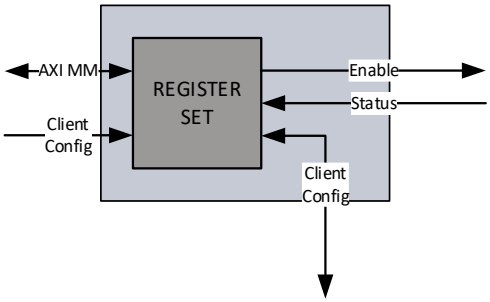

Figure 12: Registerset

4.2.6.2 Entity Description

Register Set

This module is an AXI4Lite Memory Mapped Slave. It provides access to the client status and allows configuring the NTP Client. AXI4Lite only supports 32 bit wide data access, no byte enables, no burst, no simultaneous read and writes and no unaligned access. It can be configured to either run in AXI or StaticConfig mode. If in StaticConfig mode, the configuration of the Client is done via signals and can be easily done from within the FPGA without CPU. For each parameter a valid signal is available, the enable signal shall be set last (or simultaneously). To change parameters the NTP Client has to be disabled and enabled again. If in AXI mode, a AXI Master has to set the configuration with AXI writes to the registers, which is typically done by a CPU. Parameters can in this case also be changed at runtime.

4.2.6.3 Entity Declaration

Name	Dir	Type	Size	Description
Generics				
General				
ArpSupport_Gen	-	boolean	1	If the core shall support sending ARP requests and responses
RoutingSupport_Gen	-	boolean	1	If the NTP Server is outside the client's

				LAN, the Gateway IP and the subnet mask need to be provided
UtcInitialUtcOff-set_Gen	-	integer	1	Initial UTC offset value which is used
UtcLeapSmear-ing_Gen	-	boolean	1	If smearing for the UTC Leap Second handling is support-ed
UnicastSupport_Gen	-	boolean	1	If the core shall support Unicast NTP
MulticastSupport_Gen	-	boolean	1	If the core shall support Multicast NTP
Lay-er3v4Support_Gen	-	boolean	1	If Ipv4 shall be supported
Lay-er3v6Support_Gen	-	boolean	1	If Ipv6 shall be supported
VlanSupport_Gen	-	boolean	1	Support for VLAN
ExtSync_Gen	-	boolean	1	If external UTC information is used to sync: true = external, false = internal (register only)
Register Set				
StaticConfig_Gen	-	boolean	1	If Static Configura-tion or AXI is used
AxiAddressRange Low_Gen	-	std_logic_vector	32	AXI Base Address
AxiAddressRange High_Gen	-	std_logic_vector	32	AXI Base Address plus Registerset Size
Ports				
System				
SysClk_ClkIn	in	std_logic	1	System Clock

SysRstN_RstIn	in	std_logic	1	System Reset
Config				
StaticConfig_DatIn	in	Ntp_Client StaticConfig_Type	1	Static Configuration
StaticConfig_ValIn	in	Ntp_Client StaticConfigVal _Type	1	Static Configuration valid
Status				
StaticStatus_DatOut	out	Ntp_Client StaticStatus_Type	1	Static Status
StaticStatus_ValOut	out	Ntp_Client StaticStatusVal _Type	1	Static Status 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
Error Input				
Ntp_ErrIn	in	std_logic	1	An error happened
UTC Ignore				
UtcInfoIgnore_VatIn	in	std_logic	1	Ignore UTC info
Status Input				
RequestSent_ValIn	in	std_logic	1	NTP request has been sent
ResponseReceived_ValIn	in	std_logic	1	NTP response has been received
ResponseMissed_Valin	in	std_logic	1	NTP Response has been missed
UtcLeap59InProgres_ValIn	in	std_logic	1	UTC Leap59 smearing in progress
UtcLeap61InProgres_ValIn	in	std_logic	1	UTC Leap61 smearing in progress
Config Output				
ClientConfig_DatOut	out	Ntp_ClientConfig_Type	1	NTP Client Configuration data output
ClientConfig_ValOut	out	Ntp_ClientConfigVal_Type	1	NTP Client Configuration valid output
Enable Output				
NtpClientEnable_DatOut	out	std_logic	1	Enable NTP Sever

Table 19: Registerset

4.3 Configuration example

4.3.1 Static Configuration

```
constant NtpStaticConfig_Con : Ntp_ClientStaticConfig_Type := (
    MulticastModeEnable                             => '0',
    UnicastModeEnable                               => '1',
    Ipv4Enable                                      => '1',
    Ipv6Enable                                      => '0',
    Vlan                                            => Ntp_Vlan_Type_Rst_Con,
    VlanEnable                                      => '1',
    MAC                                             => (
        0                                           => x"4E",
        1                                           => x"54",
        2                                           => x"4C",
        3                                           => x"10",
        4                                           => x"00",
        5                                           => x"00"),
    Ip                                              => (
        0                                           => x"C0",
        1                                           => x"A8",
        2                                           => x"00",
        3                                           => x"11",
        others                                      => x"00"),
    Server MAC                                      => (
        0                                           => x"4E",
        1                                           => x"54",
        2                                           => x"4C",
        3                                           => x"10",
        4                                           => x"00",
        5                                           => x"01"),
    Server Ip                                       => (
        0                                           => x"C3",
        1                                           => x"8D",
        2                                           => x"BE",
        3                                           => x"BE",
        others                                      => x"00"),
    Subnet Mask                                     => (
        0                                           => x"C0",
        1                                           => x"A8",
        2                                           => x"FF",
        3                                           => x"FF",
        others                                      => x"00"),
    GatewayIp                                       => (
        0                                           => x"C0",
        1                                           => x"A8",
        2                                           => x"00",
```

```
            3                                           => x"80",
            others                                      => x"00"),
        PollInterval                                    => x"04"
        UtcLeapSmearing                                 => '0',
        UtcInfo                                         => Clk_UtcInfo_Type_Rst_Con,
        FilterCoeff                                     => (0 => Ntp_DefaultFilterCoefB0_Con,
                                                            1 => Ntp_DefaultFilterCoefB1_Con,
                                                            2 => Ntp_DefaultFilterCoefB2_Con,
                                                            3 => Ntp_DefaultFilterCoefA1_Con,
                                                            4 => Ntp_DefaultFilterCoefA2_Con),
        PiFactor                                        => (0 => Ntp_DefaultPiFactorP_Con,
                                                            1 => Ntp_DefaultPiFactorI_Con)
    );

    constant Ntp_ClientStaticConfigVal_Con : Ntp_ClientStaticConfigVal_Type := (
        Enable_Val                                      => '0',
        Mode_Val                                        => '0',
        Vlan_Val                                        => '0',
        Mac_Val                                         => '0',
        Ip_Val                                          => '0',
        ServerMac_Val                                   => '0',
        ServerIp_Val                                    => '0',
        SubnetMask_Val                                  => '0',
        GatewayIp_Val                                   => '0',
        UtcInfo_Val                                     => '0',
        FilterCoef_Val                                  => '0',
        PiFactor_Val                                    => '0'
    );
```

4.3.2 AXI Configuration

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

```
-- NTP
-- Config Client
-- Poll Interval 1, Unicast mode, IPv4
AXI WRITE 10000084,0x00010011

-- VLAN 0x4000 (unused)
AXI WRITE 10000088,0x00004000

-- Mac: 00:01:02:03:04:05
AXI WRITE 1000008C,0x03020100
AXI WRITE 10000090,0x00000504

-- IP: 192.168.1.1
AXI WRITE 10000094,0x0101A8C0
```

```
-- Server IP: 192.168.0.2
AXI WRITE 100000AC,0x0200A8C0

-- Gateway IP: 192.168.0.1
AXI WRITE 100000CC,0x0100A8C0

-- Subnetmask 255.255.255.0
AXI WRITE 100000CC,0x00FFFFFF

-- Set Config valid bits
AXI WRITE 10000080,0x0000002F

-- set UTC Offset 37, valid, no leap, no smearing
AXI WRITE 10000104,0x00252000
-- set UTC Info valid
AXI WRITE 10000100,0x00000001

-- enable NTP Client
AXI WRITE 10000000,0x00000001
```

4.4 Clocking and Reset Concept

4.4.1 Clocking

To keep the design as robust and simple as possible, the whole NTP Client including the Counter Clock and all other cores from NetTimeLogic are run in one clock domain. This is considered to be the system clock. Per Default this clock is 50MHz. Where possible also the interfaces are run synchronous to this clock. For clock domain crossing asynchronous Fifos with gray counters or message patterns with meta-stability flip-flops are used. Clock domain crossings for the AXI interface is moved from the AXI slave to the AXI interconnect.
To increase the timestamping resolution for some very small logic parts a high resolution clock is used. This clock is optional and it must be a multiple of the System Clock.

Clock	Frequency	Description
System		
System Clock	50MHz (Default)	System clock where the NTP Client runs on as well as the counter clock etc.
High Resolution Clock	250 MHz (Default)	Optional High-resolution clock (multiple of Sys Clock)
(R)(G)MII Interface		
PHY (R)(G)MII RX Clock	2.5/25/125MHz	Asynchronous, external receive clock from the PHY also used for the MAC. Depending on the interface not all frequencies apply.
PHY (R)(G)MII TX Clock	2.5/25/125MHz	Asynchronous, external transmit clock to/from the PHY also used for the MAC. Depending on the interface not all frequencies apply.
AXI Interface		
AXI Clock	50MHz (Default)	Internal AXI bus clock, same as the system clock

Table 20: Clocks

4.4.2Reset

In connection with the clocks, there is a reset signal for each clock domain. All resets are active low. All resets can be asynchronously set and shall be synchronously released with the corresponding clock domain. All resets shall be asserted

for the first couple (around 8) clock cycles. All resets shall be set simultaneously and released simultaneously to avoid overflow conditions in the core. See the reference designs top file for an example of how the reset shall be handled.

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

Table 21: Resets

5 Resource Usage

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

5.1 Intel/Altera (Cyclone 10)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (Only Unicast, ARP Support, No routing support, No leap second smearing, No VLAN and No Red-Tag support, No Passthrough, No internal loopback. No external sync, no advanced filter)	7700	18893	5	68
Maximal (Unicast, Multicast, ARP Support, leap second smearing, VLAN and RedTag support, passthrough, external sync, advanced filter)	7409	19562	27	68

Table 22: Resource Usage Intel/Altera

5.2 AMD/Xilinx (Artix 7)

Configuration	FFs	LUTs	BRAMs	DSPs
Minimal (Only Unicast, ARP Support, No routing support, No leap second smearing, No VLAN and No Red-Tag support, No Passthrough, No internal loopback. No external sync, no advanced filter)	6722	11695	0	40
Maximal (Unicast, Multicast, ARP Support, leap second smearing, VLAN and RedTag support, passthrough, external sync, advanced filter)	7371	12695	4	40

Table 23: 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

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

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

7 Testbench

The NTP Client testbench consist of 5 parse/port types: ARP, AXI, CLK, ETH and NTP. Multiple instances exist. NTP0 CLK, NTP1 CLK, NTP0 NTP, NTP1 NTP, NTP0 ARP, NTP1 ARP and MAC0 ETH ports are all multiplexed with the MAC0 ETH MUX to one Ethernet channel connected to the port going to the PHY from the DUT (which acts like a MAC). PHY0 is connected to the port going to the MAC from the DUT (which acts like a PHY)
The NTP Server ports take the CLK ports times as reference and set the timestamps aligned with the time from the CLK ports. The NTP Server ports are responding to ARP Requests of the DUT and also send ARP Requests and check that the DUT is responding with ARPs. When NTP requests are received from the DUT, the corresponding NTP Server port takes the time of the Clock instance as reference to generate NTP Responses.. In addition, for configuration and result checks an AXI read and write port is used in the testbench and for accessing more than one AXI slave also an AXI interconnect is required.
With this Setup multiple NTP servers can be simulated.

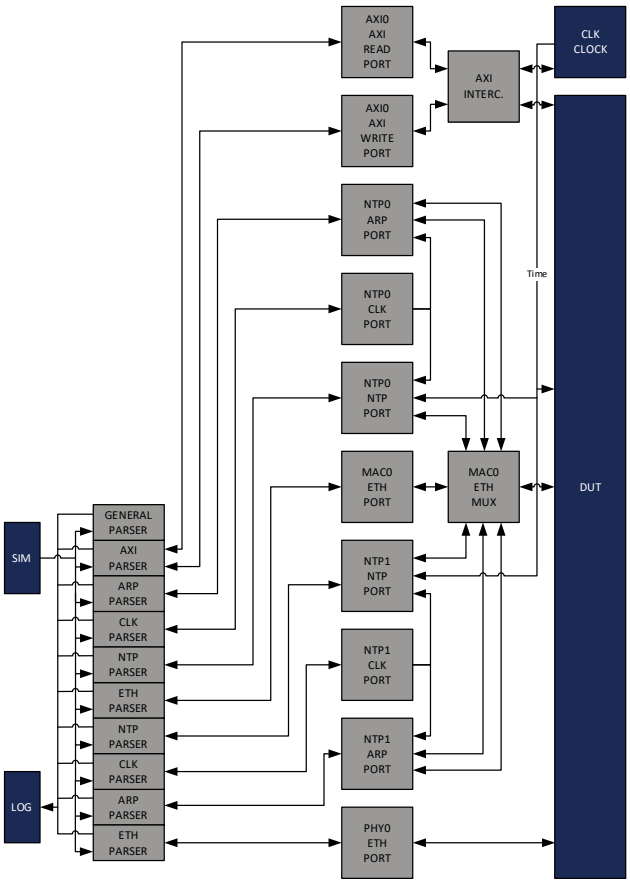

Figure 13: Testbench Framework

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/NTP/Testbench/Core/NtpClient/Script/run_Ntp_ClientMii_Tb.tcl
```

or

```
src XXX/NTP/Testbench/Core/NtpClient/Script/run_Ntp_ClientGMii_Tb.tcl
```

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

8 Reference Designs

The NTP Client design contains a PLL to generate all necessary clocks (cores are run at 50 MHz), an instance of the NTP Client IP core and an instance of the Adjustable Counter Clock IP core (needs to be purchased separately). The Reference Design is intended to be connected to any NTP Server. The Reference Design is using MII in 100Mbit full duplex as Ethernet link. If a second Ethernet Port is available on the FPGA board pass through can be enabled.
All generics can be adapted to the specific needs.

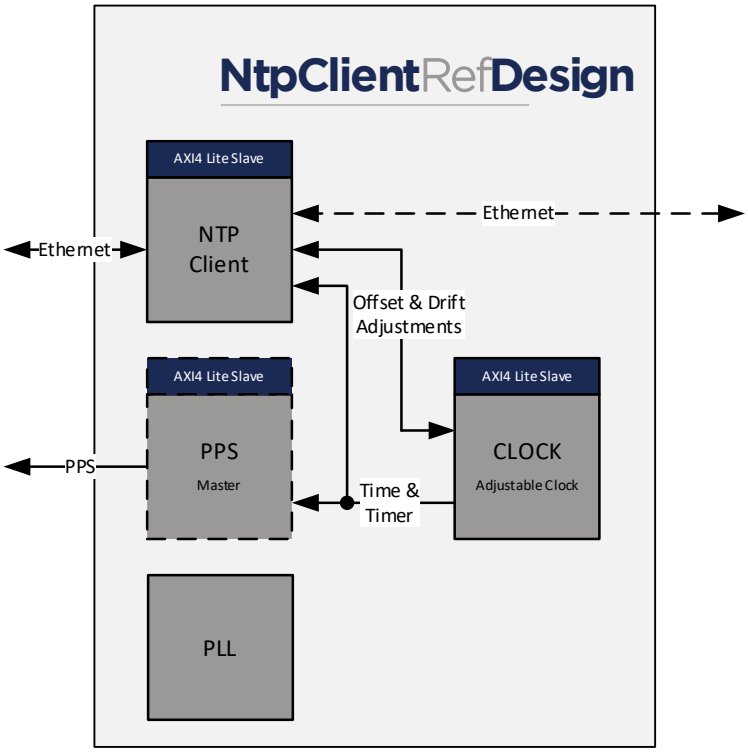

Figure 14: Reference Design

8.1 Intel/Altera: Cyclone 10 LP RefKit

The Cyclone 10 LP RefKit board is an FPGA board from Trenz Electronic. with a Cyclone 10 FPGA from Intel/Altera. (https://shop.trenz-electronic.de/de/TEI0009-02-055-8CA-Cyclone-10-LP-RefKit-10CL055-Development-Board-32-MByte-SDRAM-16-MByte-Flash).

1. Open Quartus 18.1
2. Open Project
   /NTP/Refdesign/Altera/C10LpRefKit/NtpClientMii/NtpClient.qpf

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

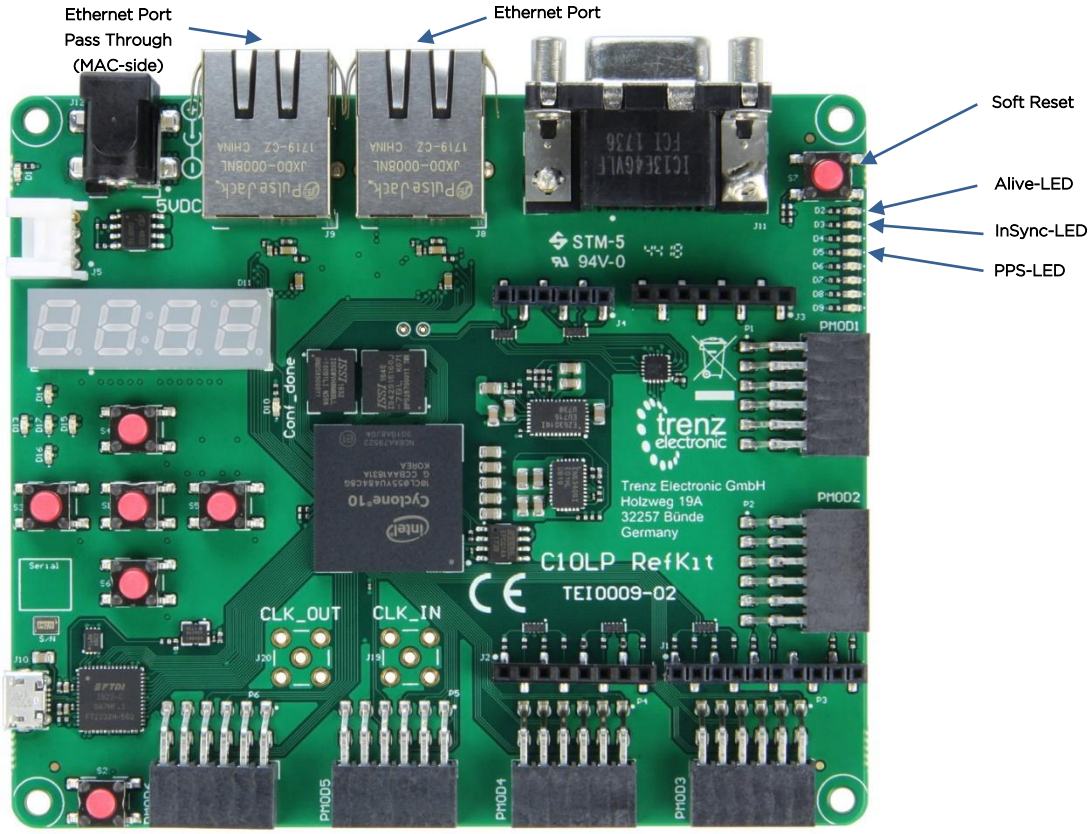

Figure 15: Cyclone 10 LP RefKit (source Trenz Electronic)

8.2 AMD/Xilinx: Digilent Arty

The Arty board is an FPGA board from Digilent Inc. with an Artix7 FPGA from AMD/Xilinx. (http://store.digilentinc.com/arty-board-artix-7-fpga-development-board-for-makers-and-hobbyists/)

1. Open Vivado 2019.1
   Note: If a different Vivado version is used, see chapter 8.3.
2. Run TCL script
   /NTP/Refdesign/Xilinx/ArtyA7/NtpClientMii/NtpClientClock.tcl
   a. This has to be run only the first time and will create a new Vivado Project

3. If the project has been created before open the project and do not rerun the project TCL
4. If the optional core PPS Master Clock is available add the files from the corresponding folders (PPS/Core, PPS/Library and PPS/Package) to the corresponding Library (PpsLib).
5. Change the generics (PpsMasterAvailable_Gen) in Vivado (in the settings menu, not in VHDL) to true for the optional cores that are available.
6. Rerun implementation
7. Download to FPGA via JTAG

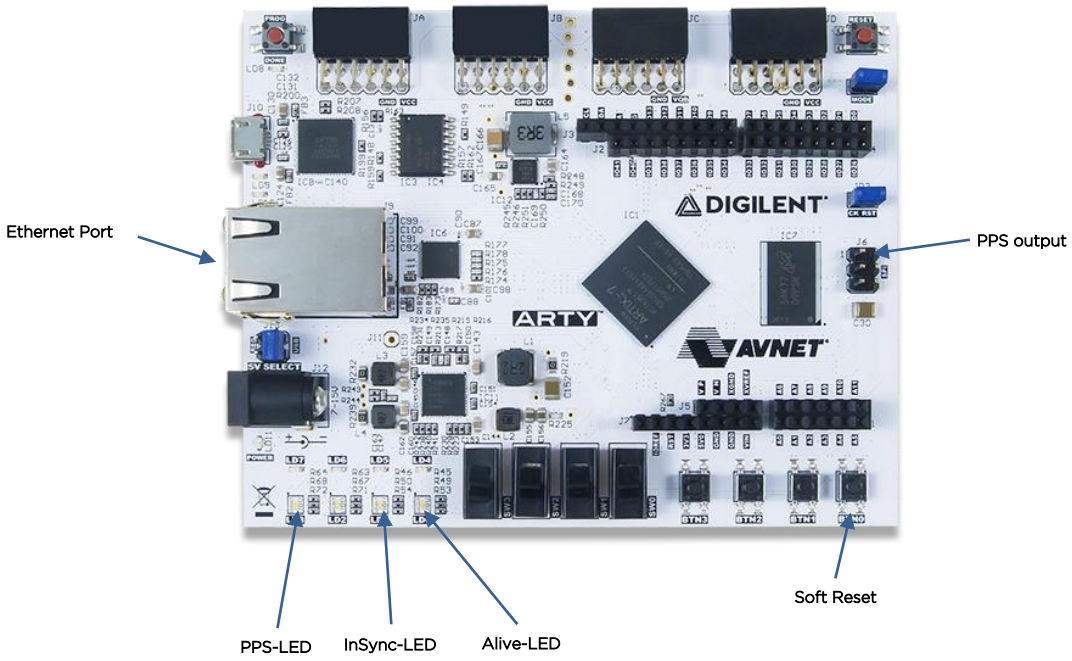

Figure 16: Arty (source Digilent Inc)

8.3 AMD/Xilinx: Vivado version

The provided TCL script for creation of the reference-design project is targeting AMD/Xilinx Vivado 2019.1.
If a lower Vivado version is used, it is recommended to upgrade to Vivado 2019.1 or higher.
If a higher Vivado version is used, the following steps are recommended:

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

- The statement occurrences:

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

  shall be replaced by:

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

- The statement occurrences:

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

  shall be replaced by:

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

- After executing the project creation TCL script, the AMD/Xilinx IP cores, such as the Clocking Wizard core, might be locked and a version upgrade might be required. To do so:
  1. At "Reports" menu, select "Report IP Status".
  2. At the opened "IP Status" window, select "Upgrade Selected". The tool will upgrade the version of the selected IP cores.

A List of tables

B List of figures