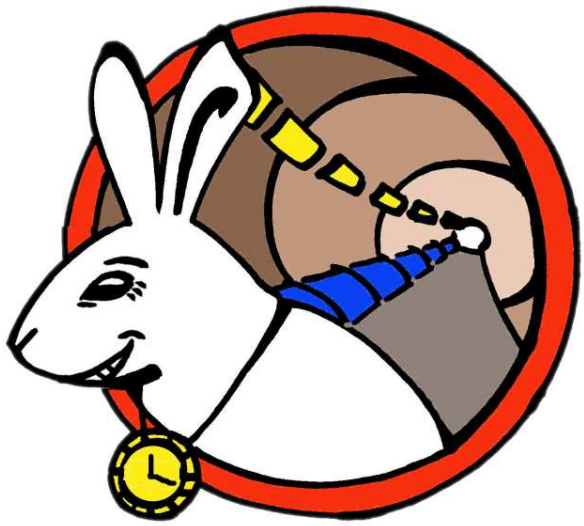
White Rabbit Specification: Draft for Comments

version 2.0
(06-07-2011)

Emilio G. Cota
Maciej Lipiński
Tomasz Włostowski
Erik van der Bij
Javier Serrano

July 2011



Revision History Table

Version	Date	Authors	Description
0.1	10/09/2010	E.G., M.L.	First draft for comments.
0.2	7/09/2010	T.W., M.L.	Added introduction about PTP.
0.3	8/09/2010	J.S., M.L.	Language&Style-related corrections, reference to Peter's paper.
0.4	14/09/2010	E.V.D.B., M.L.	1. Merged the FSM of the Slave and the FSM of the Master into a single and linear FSM, 2. Changed WR managementIDs, 3. Added authors, this rev. table, explanatory figures and descriptions.
0.5	17/09/2010	E.V.D.B., J.S.	1. Added Table of Contents. 2. Re-ordered appendixes. 3. Fixed some typos.
1.0	26/04/2011	M.L.	1. Included feedback from comments. 2. Created WR PTP Profile. 3. Added modification to BMC. 4. Enabled WR Switch to be a "real" Boundary Clock. 5. Added figure clarifying WRPTP vs. PTP FSMs operation from Power On. 6. Mentioned (missing) modification to PTP FSM.
1.1	2/05/2011	J.S., T.W., M.L.	Clarity and typos.
1.2	26/05/2011	M.L.	Modified according to John Eidson's feedback: . 1. Clear distinction between HW and WRPTP (interface defined). 2. Clarification of WR Message sending/receiving. 3. WR Link Setup in PTP Master state on the WR Master. 4. Added information about HW requirements and HW implementation. 5. WR Data Set field initialization defined and specified. Other cosmetic changes.
1.3	7/06/2011	M.L.	Added Appendixes: 1. Extract from Tomek's MSc. 2. WR computation of offset and delay with asymmetry correction.

Version	Date	Authors	Description
1.4	02/07/2011	M.L.	Changes based on ptpx daemon implementation: 1. Name changed: REQ_CALIBRATION to CALIBRATION. 2. Included Tx calibration into the CALIBRATION. 3. Added Link Down event description. 4. Specified additional wrModeOn's re-setting. 5. Re-modified BMC, completed DS Update tables. 6. Added fields to DS: calRetry and otherPortCalRetry. 7. Added calRetry field to WR TLV CALIBRATE Signaling Message.
2.0	04/07/2011	J.S.	Clarity and typos (version 2.0 release).

Contents

1 Introduction

White Rabbit (WR) is a protocol developed to synchronize nodes in a packet-based network with sub-ns accuracy. The protocol results from the combination of IEEE1588-2008 (PTP)[1] with two further requirements: precise knowledge of the link delay and clock syntonization[a] over the physical layer with Synchronous Ethernet (SyncE) [3].

A WR link is formed by a pair of nodes, master and slave . The master node uses a traceable clock to encode data over the physical layer, while the slave recovers this clock (synchronization) and bases its timekeeping on it. Absolute time synchronization between master and slave is achieved by adjusting the clock phase and offset of the slave to that of the master. The *offset* refers to the clock (e.g. time defined in Coordinated Universal Time or International Atomic Time standards), while the *phase* refers to the clock signal (e.g. 125 MHz clock signal). The phase and offset adjustment is done through the two-way exchange of PTP sync messages, which are time-stamped to achieve sub-ns accuracy thanks to hardware support such as described in section 5. Details of a sample hardware implementation are presented in Appendix B.

In White Rabbit, the precise knowledge of the link delay is obtained by accurate hardware timestamps (see section 5.3) and calculation of the delay asymmetry (see section 4) supported by knowledge of delays introduced by the hardware (see section 5.2).

The described single-link synchronization can be replicated. Multi-link WR networks are obtained by chaining WR links forming a hierarchical topology. This hierarchy is imposed by the fact that a frequency traceable to a common grandmaster must be distributed over the physical layer, resulting in a *cascade* of master and slave nodes. As a result of this topology, a WR network consists of two kinds of WR network devices: *WR boundary clocks* (WR Switches) and *WR ordinary clocks* (WR Nodes), see section 6.2. A WR boundary clock distributes the frequency and time retrieved from the upstream link to all the downstream links.

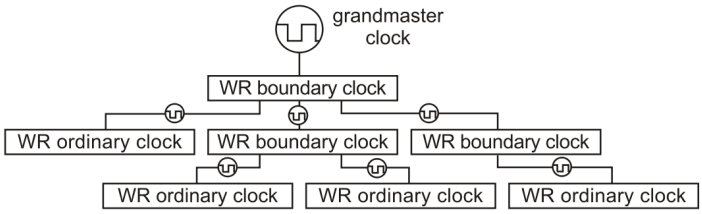


Figure 1: White Rabbit network; it forms a hierarchical topology.

Some applications need WR and IEEE1588-2008 nodes to coexist. Examples of this are networks where the need for highly accurate time synchronization is concentrated on a certain group of nodes. For this purpose the WR protocol enables WR Nodes or Switches to defer to IEEE1588 behavior when not connected to another WR Node or Switch.

[a]The adjustment of two electronic circuits or devices in terms of frequency.

2 Precision Time Protocol

The IEEE1588-2008 standard, known as Precision Time Protocol (PTP), is repeatedly referenced in this document. Knowledge of basic PTP concepts is required to read this specification. Therefore, they are explained in this section.

PTP is a packet-based protocol designed to synchronize devices in distributed systems. The standard defines two kinds of messages which are exchanged between *PTP nodes*: *event messages* and *general messages*. Both, the time of transmission and the time of reception of event messages are *recorded*. General messages are used by PTP nodes to identify other PTP nodes, establish clock hierarchy and exchange data, e.g. timestamps, settings or parameters. PTP defines several methods for node synchronization. Figure 2 presents the messages used when the *delay request-response mechanism* (with a *two-step clock*) is used, which is the case in White Rabbit.

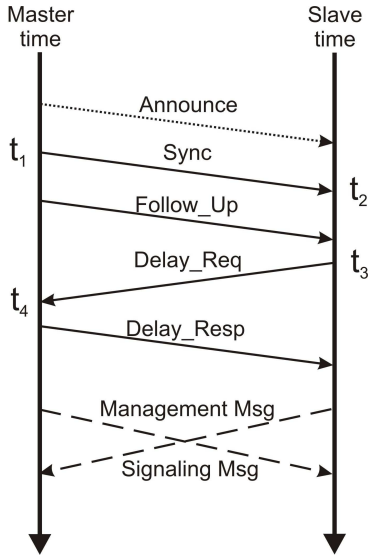


Figure 2: PTP messages used by WRPTP.

An *Announce Message* is periodically broadcast by the PTP node which is in the Master state. The message carries information about its originator and the originator's clock source quality. This enables other PTP nodes receiving the announce message to perform the Best Master Clock (BMC) Algorithm. This algorithm defines the role of each PTP node in the PTP network hierarchy; the outcome of the algorithm is the recommended next state of the PTP node and the node's synchronization source (grandmaster). In other words, a PTP node decides to which other PTP node it should synchronize based on the information provided in the announce messages and using the BMC algorithm. A PTP node which is in the SLAVE state synchronizes to the clock of another PTP node. A PTP node which is in the MASTER state is regarded as a source of synchronization for the other PTP nodes. The full PTP state machine with state descriptions is included in Appendix A.

Sync Messages and *Delay_Req Messages* are timestamped (t_1, t_2, t_3, t_4) and these timestamps are used to calculate the offset and the link delay between the nodes exchanging the messages. *Follow_Up Messages* and *Delay_Resp Messages* are used to send timestamps between Master and Slave (in the case of a two-step clock).

Management Messages are used only for configuration and administrative purposes. *Signaling Messages* are used for communication between clocks for optional or experimental features of the PTP standard as well as implementation-specific mechanisms. Both, Management and Signaling messages, are not essential for PTP synchronization.

The flow of events in the PTP delay request-response (two-step clock) mechanism is the following (simplified overview):

1. The master sends Announce messages periodically.
2. The slave receives the Announce message and uses the BMC algorithm to establish its place in the network hierarchy.
3. The master periodically sends a Sync message (timestamped on transmission, t_1) followed by a Follow_Up message which carries t_1.
4. The slave receives the Sync message sent by the master (timestamped on reception, t_2).
5. The slave receives the Follow_Up message (which carries the Sync transmission time, t_1) sent by the master .
6. The slave sends a Delay_Req message (timestamped on transmission, t_3).
7. The master receives the Delay_Req message sent by the slave (timestamped on reception, t_4).
8. The master sends the Delay_Resp message which carries t_4.
9. The slave receives the Delay_Resp.
10. The slave adjusts its clock using the clock offset and the link delay calculated with timestamps (t_1, t_2, t_3, t_4). This results in the Slave's synchronization with the Master clock.
11. Repeat 1-10.

3 Link Delay Model

The delay of a message traveling from master to slave (see Figure 3) can be expressed as the sum

$$\text{delay}_{ms} = \Delta_{tx_m} + \delta_{ms} + \Delta_{rx_s} \tag{1}$$

where Δ_{tx_m} is the fixed delay due to the master's transmission circuitry, δ_{ms} is the variable delay incurred in the transmission medium and Δ_{rx_s} is the fixed delay due to the slave's reception circuitry. In a similar fashion, the delay of a message traveling from slave to master can be decomposed as

$$\text{delay}_{sm} = \Delta_{tx_s} + \delta_{sm} + \Delta_{rx_m} \tag{2}$$

The characterization of the link is completed with an equation to relate the two variable delays δ_{ms} and δ_{sm}. Describing a procedure to obtain this equation is out of the scope of this document. However, section 3.1 provides such an equation obtained empirically for one scenario.

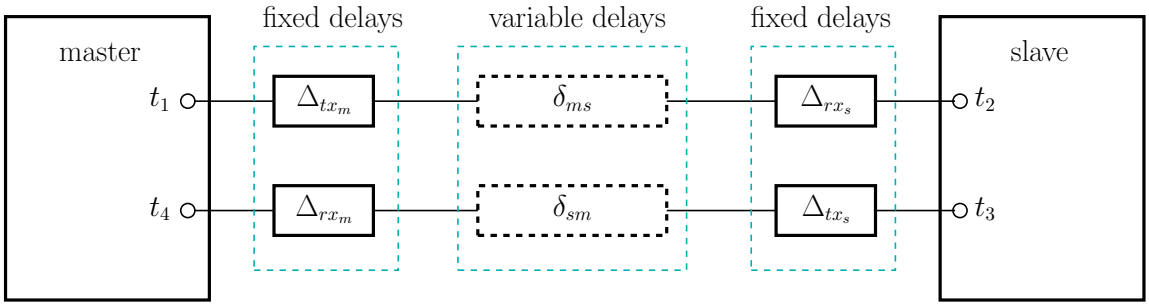


Figure 3: Delay model of a WR link. The timestamps are accurately corrected for link asymmetries by the usage of the four fixed delays $\Delta_{\{tx_m,rx_s,tx_s,rx_m\}}$ and the relationship between both variable delays $\delta_{\{ms,sm\}}$.

3.1 Relative Delay Coefficient

An accurate relation between both variable delays on the transmission line is essential for obtaining an acceptable estimate of the delay asymmetry on a WR link. The relation between δ_{ms} and δ_{sm} is represented in this document by the *relative delay coefficient* (α). Its origin is highly implementation-dependent. Thus this document just assumes that it exists and is known.

3.1.1 1000base-X over a Single-mode Optical Fiber

When a single-mode fiber is used as bi-directional communication medium, it can be shown that both variable delays are related by an equation of the form [5]:

$$\delta_{ms} = (1+\alpha)\,\delta_{sm} \tag{3}$$

4 Delay Asymmetry Calculation

Let us start from the PTP sync timestamps, represented by the familiar set t_1, t_2, t_3 and t_4. The mean path delay is then defined as

$$\mu = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{4}$$

Note that the transmission delays $t_2 - t_1$ and $t_4 - t_3$ can be expressed in terms of WR's Delay Model:

$$t_2 - t_1 = \Delta_{tx_m} + \delta_{ms} + \Delta_{rx_s} + \text{offset}_{ms} \tag{5}$$
$$t_4 - t_3 = \Delta_{tx_s} + \delta_{sm} + \Delta_{rx_m} - \text{offset}_{ms} \tag{6}$$

where offset_{ms} is the time offset between the slave's clock and the master's. Combining the three equations above we obtain

$$2\mu = \Delta + \delta_{sm} + \delta_{ms} \tag{7}$$

where Δ accounts for all fixed delays in the path, i.e.

$$\Delta = \Delta_{tx_m} + \Delta_{rx_s} + \Delta_{tx_s} + \Delta_{rx_m} \tag{8}$$

The delay asymmetry as specified in section 7.4.2 of the PTP standard is expressed in our own notation by using equations (1), (2) and (7) as follows:

$$\text{delay}_{ms} = \mu + \text{asymmetry} \tag{9}$$
$$\text{delay}_{sm} = \mu - \text{asymmetry} \tag{10}$$

4.1 Solution for Ethernet over a Single-mode Optical Fiber

Combining equations (3) and (7) we obtain:

$$\delta_{ms} = \frac{1+\alpha}{2+\alpha}(2\mu - \Delta) \tag{11}$$
$$\delta_{sm} = \frac{2\mu - \Delta}{2+\alpha} \tag{12}$$

The delay asymmetry can then be derived from equations (1), (9), (11) and (12):

$$\text{asymmetry} = \Delta_{tx_m} + \Delta_{rx_s} - \frac{\Delta - \alpha\mu + \alpha\Delta}{2+\alpha} \tag{13}$$

5 Hardware Support

This section gives a general overview of hardware requirements to support the White Rabbit protocol. Hardware supporting WR must fulfill the following requirements:

- It shall distribute frequency over physical layer with SyncE.
- It shall feature constant rx/tx latencies during operation and inform higher layers about these latencies.
- It shall provide timestamps with a sufficient precision.
- It shall be able to generate the WR calibration pattern (RD+K28.7 code group, Appendix 36A.2 of [2]).

Figure 4 (a) gives a general picture of the data flow in the WR protocol, WR Hardware Support and the interface between theme Figure 4 (b) explains the WR synchronization and synchronization scheme. An example hardware implementation is described in detail in Appendix B.

5.1 Synchronous Ethernet

SyncE is responsible for clock syntonization (frequency transfer) in the White Rabbit Network. In the SyncE scheme, the reference clock (125MHz) is used to encode the outgoing data stream. The same clock is retrieved on the other side of the physical link. The retrieved frequency can be further distributed and is always looped back to the sender. Having the same frequency allows the use of phase detector technologies as a means of evaluating delays. The measurement of phase can be used for increasing the precision of timestamps beyond the resolution allowed by the 125MHz clock (section 5.3) and to obtain the transmit/receive (Rx/Tx) fixed delays introduced by the PHY (section 5.2).

5.2 Fixed delays

The knowledge of all fixed delays $\Delta_{\{tx_m,rx_s,tx_s,rx_m\}}$ is necessary to calculate the delay asymmetry using the WR Link Model (section 3). They might include circuit, components (e.g. PHY, SFP) and FPGA internal latencies (Appendix B.6 describes fixed delays in the WR optical link model). Such delays may be constant for the lifetime of the hardware, its up-time or the duration of the link connection. Therefore, the method for obtaining fixed delays is medium-specific and implementation-dependent. The WR protocol requires the fixed delay values and enables PHY delay measurement, but does not specify how they shall be obtained.

The PHY's delays are measured (if necessary) and their values are distributed across the link during the process of establishing the WR link, which is called *WR Link Setup* in this document. A WR node participates in the measurement of another WR node's reception fixed delays ($\Delta_{\{rx_m,rx_s\}}$) upon request, e.g. by sending a calibration pattern in Gigabit Ethernet. The calibration pattern sent by WR nodes to measure fixed delays shall be generated by a repeated transmission of the RD+ K28.7 code-group. Measurement of fixed delays during WR Link Setup is optional and in principle needed only once, while the WR link is being set up. It is

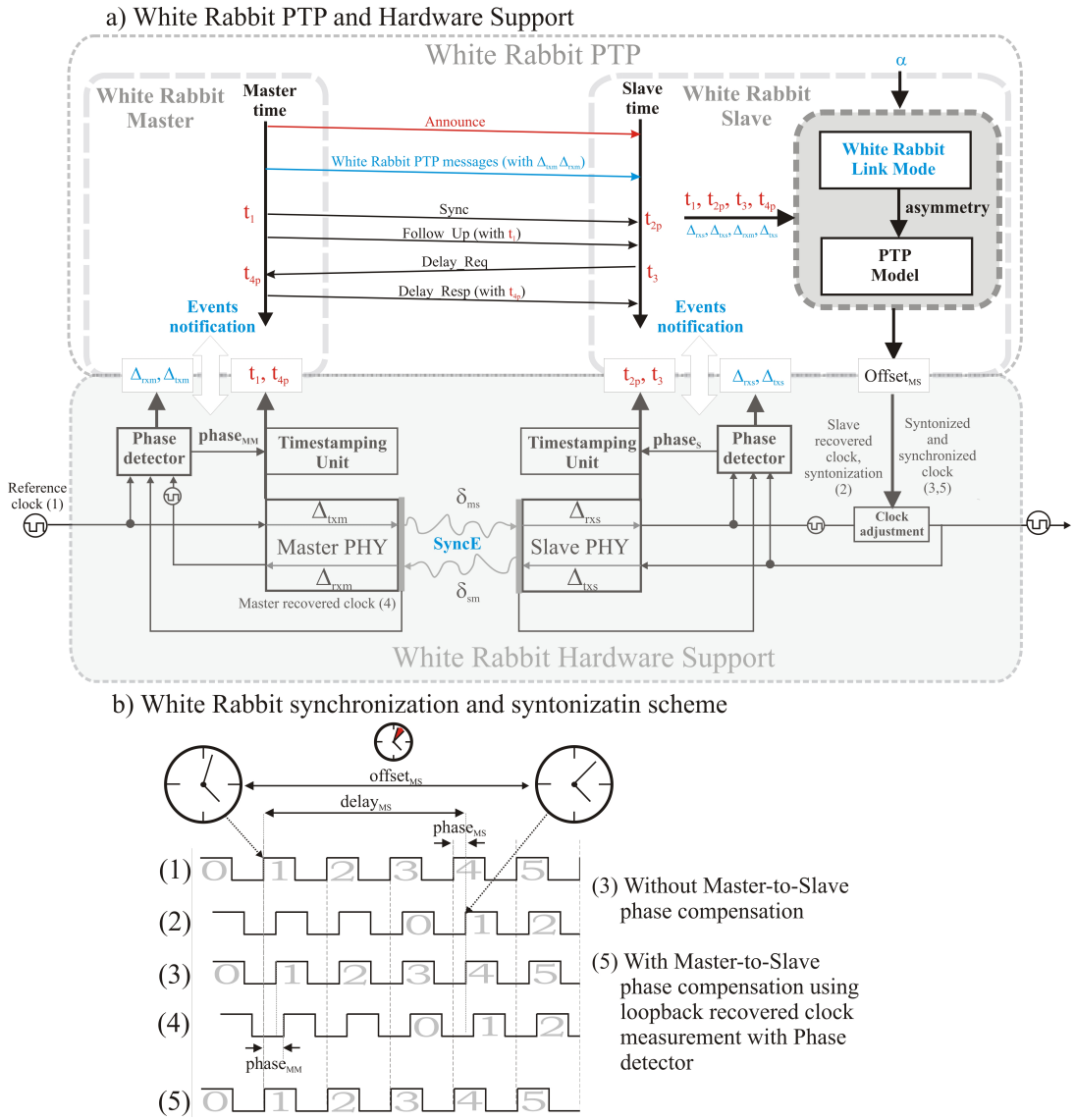


Figure 4: WR protocol and WR Hardware overview.

only required if non-deterministic reception/transmission elements (i.e. with latencies varying after each power cycle or lock) are used.

Figure 4 (a) presents a possible method of fixed delay measurement for a non-deterministic Gigabit Ethernet PHY – detection of the clock signal's phase difference between the input and output of the PHY (detailed description in Appendix B.6.3).

5.3 Timestamps

WR requires precise timestamps. The precision of timestamps shall be sufficient to take advantage of the calculated asymmetry and is dictated by the required synchronization accuracy.

In standard implementations of timestamping units the precision is limited by the timestamping resolution (e.g. 8ns for 125MHz). In particular, the precision of the reception timestamps (t_2, t_4) needs to be enhanced. This can be done by using the common notion of frequency

distributed with SyncE to cast the problem of timestamping into a phase detection measurement.

An overview of a possible solution enabling highly precise timestamping is illustrated in Figure 4. The timestamping unit produces timestamps on both the rising and falling edges of the clock. It is provided with the phase measurement of the round-trip phase ($phase_{MM}$) in case of the Master, and the setpoint phase ($phase_S$) in case of the Slave. The phase measurement is used by the timestamping unit to choose the correct edge timestamp and enhance it to the precision allowed by the phase detector. Details of the implementation of the presented solution are included in Appendix B.5.

6 White Rabbit PTP

White Rabbit extends the IEEE1588-2008 (PTP) standard to achieve sub-ns accuracy while still benefiting from PTP's synchronization, management and messaging mechanisms. From now on in this document, the White Rabbit extension to the PTP standard will be referred to as *WRPTP*.

WRPTP takes advantage of PTP's customization facilities, i.e. *PTP profiles* and *Type-Length-Value* (TLV). It also defines *implementation specific* functionalities (e.g. WR-specific fields of Data Sets and hardware support requirements) which are compatible with the PTP standard. For the sake of simplicity and clarity, this section explains all the mechanisms of WRPTP without classifying them into PTP-profile and implementation-specific. The distinction is made in the last subsection, where the *White Rabbit PTP profile* is defined and the implementation-specific recommendations are listed.

6.1 Overview

WRPTP introduces *the White Rabbit Link Setup* (WR Link Setup), which is a process for establishing the WR link (Figure 6). It includes syntonization of the local clock over the physical layer, measurement of fixed delays (calibration) and distribution of the information about fixed delays over the link. WRPTP uses the Link Delay Model to obtain an accurate delay estimation, e.g. it uses the delay asymmetry equation (13) for Gigabit Ethernet over optical fiber. The additional communication between a master and a slave needed to exchange parameters and state information is done through extended PTP messaging facilities.

WRPTP is compliant with standard PTP. Figure 5 depicts the tree topology of a *hybrid* WR/IEEE1588 network with a grandmaster as a root.

WRPTP protocol cannot be performed over a non-WR network device (i.e. switches, repeaters, routers). For example, if a non-WR device is connected between two WR-compatible nodes, the standard PTP protocol will be used for synchronization.

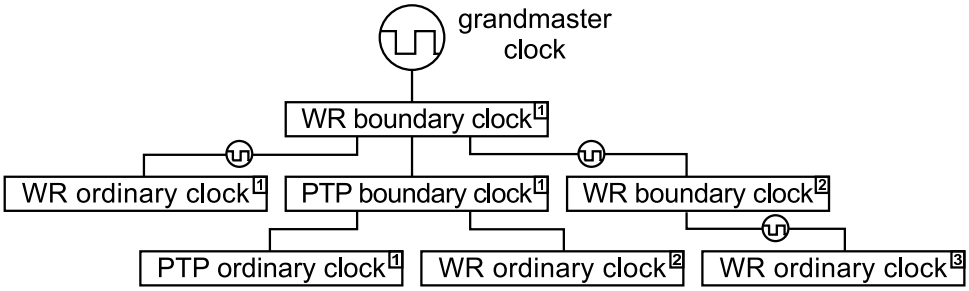


Figure 5: Hybrid WR/IEEE1588 network. White Rabbit nodes work transparently with PTP nodes. WR ordinary clock 3 is more accurately synchronized to the grandmaster than WR ordinary clock 2, which is below a standard PTP boundary clock.

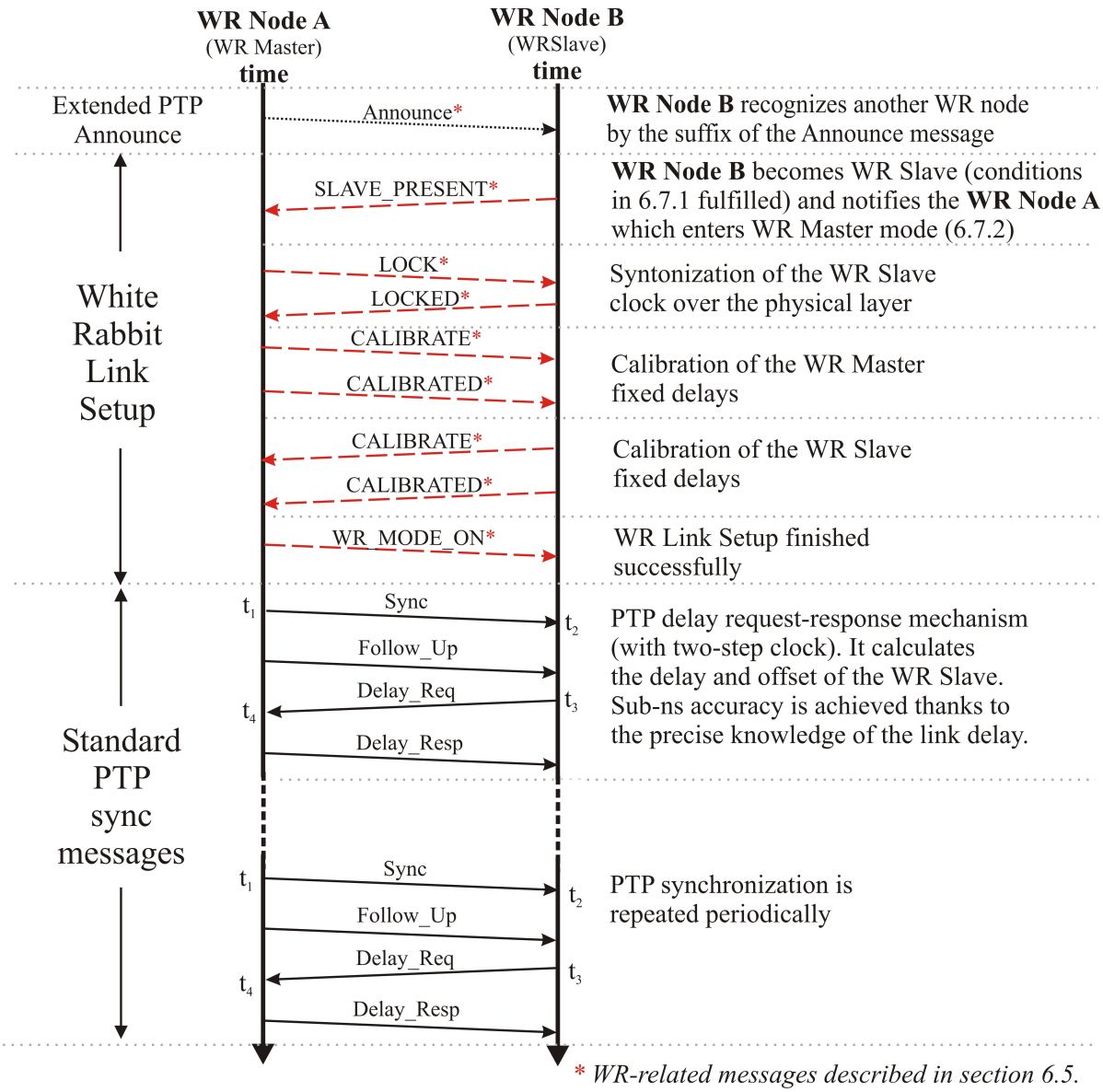


Figure 6: Simplified overview of the message flow in WRPTP.

The flow of events for standard PTP which is presented in section 2 is extended as depicted in Figure 6 and described below:

1. **WR Node A** which is in PTP_MASTER state periodically sends WR Announce messages with a custom suffix.
2. **WR Node B** receives Announce message(s), recognizes the WR Announce message and uses the modified BMC algorithm (section 6.4) to establish its place in the WR network hierarchy.
3. **WR Node B** enters WR Slave mode (based on the conditions in 6.7.1) and starts the WR Link Setup by sending the SLAVE_PRESENT WR Signaling message.
4. **WR Node A** enters WR Master mode (based on the conditions in 6.7.2) and sends the LOCK WR Signaling message to request the WR Slave to start syntonization.
5. The WR Slave sends the LOCKED WR Signaling message as soon as the syntonization process is finished (notification from the hardware).
6. The WR Master sends the CALIBRATE WR Signaling message to request the calibration pattern. It calibrates its transmission and reception fixed delay.
7. The WR Master sends the CALIBRATED WR Signaling message as soon as the calibration is finished (notification from hardware).
8. The WR Slave sends the CALIBRATE WR Signaling message to request the calibration pattern. It calibrates its transmission and reception fixed delay.
9. The WR Slave sends the CALIBRATED WR Signaling message as soon as the calibration is finished (notification from hardware).
10. The WR Master sends the WR_MODE_ON WR Signaling message to indicate completion of the WR Link Setup process.
11. The WR Master periodically sends a Sync message (timestamped on transmission, t_1) followed by a Follow_Up message which carries t_1.
12. The WR Slave receives the Sync message sent by the master (timestamped on reception, t_2).
13. The WR Slave receives the Follow_Up message sent by the master.
14. The WR Slave sends a Delay_Req message (timestamped on transmission, t_3).
15. The WR Master receives the Delay_Req message sent by the slave (timestamped on reception, t_4).
16. The WR Master sends a Delay_Resp message which carries t_4.
17. The WR Slave receives the Delay_Resp.
18. The WR Slave adjusts its clock using the clock offset and the link delay calculated with the timestamps (t_1, t_2, t_3, t_4). This results in the slave's synchronization with the master clock with sub-ns accuracy.
19. Repeat 1, 11-18.

6.2 Definitions

The following definitions used in this document are based on Clause 3.1 of PTP:
node: A device that can issue or receive Precision Time Protocol (PTP) communications on a network.
boundary clock: A clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain. It may serve as the source of time and frequency, i.e., be a master clock, and may synchronize and syntonize to another clock, i.e., be a slave clock. The frequency retrieved by the slave clock is re-distributed to syntonize other slave clocks connected to its ports.
ordinary clock: A clock that has a single Precision Time Protocol (PTP) port in a domain and maintains the timescale used in the domain. It may serve as a source of time and frequency, i.e., be a master clock, or may synchronize and syntonize to another clock, i.e., be a slave clock.

This document introduces the following definitions:
White Rabbit Port (WR Port): a WRPTP-enabled port of a boundary clock or an ordinary clock.
White Rabbit Master (WR Master): a WRPTP-enabled port of a boundary clock or an ordinary clock which acts as a source of time and frequency for another White Rabbit enabled port of a boundary or an ordinary clock.
White Rabbit Slave (WR Slave): a WRPTP-enabled port of a boundary clock or an ordinary clock which retrieves time and frequency sent over a link by the WR Master.
White Rabbit Switch (WR Switch): a boundary clock implementing WRPTP, compatible with standard PTP.
White Rabbit Node (WR Node): an ordinary clock implementing WRPTP, compatible with standard PTP.

6.3 WRPTP Data Set Fields

The PTP standard defines data sets (DS) to store the static and dynamic variables needed for the operation of the protocol (clause 8, PTP). WRPTP:

- adds fields to the DSs defined in the PTP standard to store the WR-specific parameters,
- defines a new DS: backupParentDS.

6.3.1 New fields of PTP Data Sets

Table 1 defines the additional DS fields required by WRPTP that are not part of the PTP standard (see Appendix E for a definition of primitive data types).

Table 1: WRPTP Data Sets fields

DS member	**DS name**	**Allowed values**	**Initialization values**	**D**ynamic or **S**tatic	**Unit**
wrConfig	portDS	NON_WR, WR_S_ONLY, WR_M_ONLY, WR_M_AND_S	-	S	-
wrMode	portDS	NON_WR,[b] WR_SLAVE, WR_MASTER	NON_WR	D	-
wrModeOn	portDS	TRUE, FALSE	FLASE	D	-
wrPortState	portDS	Table 14	IDLE	D	-
knownDeltaTx	portDS	UInteger64	-	S	$[2^{16}$ps$]$[c]
knownDeltaRx	portDS	UInteger64	-	S	$[2^{16}$ps$]$[c]
deltasKnown	portDS	TRUE, FALSE	-	S	-
calibrated	portDS	TRUE, FALSE	deltasKnown	D	-
deltaTx	portDS	UInteger64	knownDeltaTx	D	$[2^{16}$ps$]$[c]
deltaRx	portDS	UInteger64	knownDeltaRx	D	$[2^{16}$ps$]$[c]
wrStateTimeout	portDS	UInteger32	-	S	[ms]
wrStateRetry	portDS	UInteger8	-	S	-
calPeriod	portDS	UInteger32[d]	-	S	[us]
calRetry	portDS	UInteger8[e]	-	S	-
parentWrConfig	portDS	NON_WR, WR_S_ONLY, WR_M_ONLY, WR_M_AND_S	NON_WR	D	-
parentWrMode	portDS	NON_WR, WR_SLAVE, WR_MASTER	NON_WR	D	-
parentWrModeOn	portDS	TRUE, FALSE	FALSE	D	-
parentCalibrated	portDS	TRUE, FALSE	FALSE	D	-
otherPortDeltaTx	portDS	UInteger64	0x0	D	$[2^{16}$ps$]$[b]
otherPortDeltaRx	portDS	UInteger64	0x0	D	$[2^{16}$ps$]$[b]
otherPortCalPeriod	portDS	UInteger32	0x0	D	[us]
otherPortCalRetry	portDS	UInteger8	0x0	D	-
otherPortCalSendPattern	portDS	TRUE, FALSE	FALSE	D	-
primarySlavePortNumber	currentDS	UInteger16	0x0	D	-

[b] Not re-initialized on exiting IDLE state of WR State Machine.

[c] The value of $\Delta_{tx_m,rx_s,rx_m,tx_s}$ measured in picoseconds and multiplied by 2^{16}.

[d] Maximum shall be smaller than AnnounceInterval (see PTP).

[e] Maximum shall be 32.

6.3.1.1 (Re-)Initialization

Initialization rules listed in PTP (clause 8.1.3) apply to WR fields. In particular:

- Data set members shall be initialized before leaving the PTP INITIALIZATION state,
- Static members shall be initialized to the implementation-specific values meeting the specifications for the member.

Additionally, the values of dynamic WR fields shall be set to initializing values on the occurrence of the following events:

- exiting IDLE state of the WR State Machine (except for wrMode).
- EXCEED TIMEOUT RETRIES transition event.

The initialization values of dynamic WR fields are specified in Table 1, while the initialization values of static fields are implementation and configuration-specific (if not defined by implementation, the default values in Table 2 shall be used).

Table 2: Default values for static DS.

DS member	**DS name**	**Default value**
wrConfig	portDS	WR_M_AND_S
deltasKnown	portDS	FALSE
knownDeltaTx	portDS	0 $[2^{16}ps]^b$
knownDeltaRx	portDS	0 $[2^{16}ps]^b$
calPeriod	portDS	3000 [us]
calRetry	portDS	port number + 2
wrStateTimeout	portDS	1000 [ms]
wrStateRetry	portDS	3

6.3.1.2 New Fields Description

6.3.1.2.1 wrConfig

Determines the predefined function of a WR Port:

- WR_M_AND_S shall be the default on boundary and non-slave-only ordinary clocks.
- WR_SLAVE shall be the default on a WR Node which is a slave-only ordinary clock.
- WR_MASTER shall be default on a WR Node (ordinary clock) which is, by design, supposed to be always connected to a primary source of time, so-called master-only.
- NON_WR might be used on any kind of clock to disable the WRPTP protocol on a given port. Such configuration can result from administrative constraints or requirements.

6.3.1.2.2 wrMode

Contains the current WR-role of a WR Port which is determined using the modified BMC algorithm (section 6.4) by the conditions described in sections 6.7.1 and 6.7.2. The WR-role is recommended or active depending on the value of wrModeOn:

- wrMode is *recommended* when wrModeOn is FALSE.
- wrMode is *active* when wrModeOn is TRUE.

A WR Port in WR_SLAVE mode is called a WR Slave. Similarly, a WR Port in WR_MASTER mode is called WR Master. The wrMode field shall be reset to NON_WR on the reception of HW_LINK_DOWN hardware event notification (see 6.9.2) and on reception of an ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES event on the WR Slave.

6.3.1.2.3 wrModeOn

If TRUE, it indicates that the WR Link Setup has been performed successfully and the WR Port is synchronized using WRPTP, consequently the role defined in wrMode is active.

It is set to TRUE on successful completion of the WR State Machine (WR_LINK_ON state, section 6.7).

It is set to FALSE :

- when link-down is detected on the WR Port, section 6.9.2,
- on initialization and re-initialization as described in section 6.3.1.1,
- on occurrence of the WR_MODE_OFF event, section 6.11.
- on exiting the PTP_SLAVE state.

6.3.1.2.4 wrPortState

Stores the current state of the WRPTP state machine (see section 6.7).

6.3.1.2.5 knownDeltaTx

If the transmission fixed delay (Δ_{tx}) is not known a priori (i.e. a non-deterministic PHY is used, see 5.2), it shall be set to 0x0. Otherwise, it stores the value of Δ_{tx} measured in picoseconds and multiplied by 2^{16}.

6.3.1.2.6 knownDeltaRx

If the reception fixed delay (Δ_{rx}) is not known a priori (i.e. a non-deterministic PHY is used, see 5.2), it shall be set to 0x0. Otherwise, it stores the value of Δ_{rx} measured in picoseconds and multiplied by 2^{16}.

6.3.1.2.7 deltasKnown

TRUE if the fixed delays are known a priori (a deterministic PHY is used) and their values are provided in the knownDeltaTx and knownDeltaRx Data Set fields. Otherwise FALSE.

6.3.1.2.8 calibrated
If TRUE, it indicates that the fixed delays of the given port are currently known (measured and stored in deltaTx/deltaRx DS fields).

6.3.1.2.9 deltaTx
Port's Δ_{tx} measured in picoseconds and multiplied by 2^{16}.

6.3.1.2.10 deltaRx
Port's Δ_{rx} measured in picoseconds and multiplied by 2^{16}.

6.3.1.2.11 wrStateTimeout
Determines the timeout (in microseconds) for an execution of a state of the WR State Machine.

6.3.1.2.12 wrStateRetry
Determines the default number of times a state of WR State Machine is re-entered (as a consequence of wrStateTimeout expiration) before the WR Link Setup is abandoned. If the number of the given state execution retries equals wrStateRetry, the EXC_TIMEOUT_RETRY event is generated (see 6.7.4).

6.3.1.2.13 calPeriod
Calibration period in microseconds required to perform the measurement of the deltaTx and deltaRx fixed delays (see 5.2). Its value shall be less than the value of the AnnounceInterval of PTP determined by the portDS.logAnnounceInterval (see clause 15.5.3.7.2.1 of PTP). If its value is greater than 0x0, it overwrites wrStateTimeout for the *CALIBRATION* state of the WR State Machine. It is distributed to the partner-port to determine the timeout of its *RESP_CALIB_REQ* state.

6.3.1.2.14 calRetry
If greater then 0x0, it overwrites wrStateRetry (6.3.1.2.12) for the *CALIBRATION* state of the WR State Machine. It is distributed to the partner port and used to overwrite wrStateRetry for the *RESP_CALIB_REQ* state. The maximum allowed value is 32.

6.3.1.2.15 primarySlavePortNumber
Zero value indicates that no Primary Slave is selected. 1, 2 ,...N – values indicate the portNumber (clause 7.5.2.3 PTP) selected as the Primary Slave, see section 6.4.

6.3.1.2.16 parentWrConfig
Stores the value of wrConfig of the parent port which is sent in the Announce Message (section 6.5.2).

6.3.1.2.17 parentWrMode
Stores the value of wrMode of the parent port which is sent in the Announce Message (section 6.5.2).

6.3.1.2.18 parentWrModeOn

Stores the value of wrModeOn of the parent port which is sent in the Announce Message (section 6.5.2). It shall be set to TRUE in the WR_LINK_ON state on the WR Slave.

6.3.1.2.19 parentCalibrated

Stores the value of calibrated of the link-parent port which is sent in the Announce Message (section 6.5.2).

6.3.1.2.20 otherPortDeltaTx

Stores the value of deltaTx of the link-partner port exchanged during the WR Link Setup. It is sent in the WRPTP Signaling Message (section 6.5.3).

6.3.1.2.21 otherPortDeltaRx

Stores the value of deltaRx of the partner port exchanged during the WR Link Setup. It is sent in the WRPTP Signaling Message (section 6.5.3).

6.3.1.2.22 otherPortCalPeriod

Stores the value of the calPeriod parameter of the link-partner port which is sent in the CALIBRATE message (6.5.3.4). If greater than 0, the value defines the time (in microseconds) for which the calibration pattern should be sent by the port which received the message – it overwrites wrStateTimeout (6.3.1.2.11) for the *RESP_CALIB_REQ* state of the WR State Machine on the receiving port. It is exchanged during the WR Link Setup by the WRPTP Signaling Message (section 6.5.3).

6.3.1.2.23 otherPortCalRetry

Stores the value of the calRetry parameter of the link-partner port which is sent in the CALIBRATE message (6.5.3.4). If greater than 0x0, it overwrites the wrStateRetry (6.3.1.2.12) for the *RESP_CALIB_REQ* state of the WR State Machine on the receiving port.

6.3.1.2.24 otherPortCalSendPattern

Stores the value of calSendPattern sent by the WRPTP Signaling Message (section 6.5.3) which indicates whether sending of the calibration pattern is required.

6.3.2 backupParentDS data set specifications

A boundary clock shall maintain an implementation-specific backupParentDS Data Set for the purpose of qualifying redundant sources of synchronization, i.e.:

- backup paths to the current grandmaster clock, or
- paths to alternative grandmaster clock(s).

Each entry of the data set contains:

- backupParentDS.secondarySlavePortNumber - the number of the port on which the Announce message from a backup parent has been received (Secondary Slave port),

- backupParentDS.backupParentSourcePortIdentity - the portIdentity of the port on the backup master.

The implementation-specific backupParentDS set shall have a minimum capacity of three backup parent records. The order of the records is established using the Data Set Comparison Algorithm as described in section 6.4.2.

The initialization rules stated in PTP (clause 8.1.3) apply to the backupParentDS Data Set.

6.4 Modified Best Master Clock Algorithm

6.4.1 Overview

The Best Master Clock (BMC) algorithm is used in PTP to compare clocks (determine which clock is the *best*) and to recommend the next state of the PTP state machine (Clause 9.3 of PTP). The BMC is used to prune the physical network topology to obtain logical tree(s) with the *best* clock, the grandmaster (preferably the one synchronized to a primary reference time source), as a root. In the case when more than one clock in a network is synchronized to a primary reference time source, the BMC produces disjoint logic trees. Each tree has a single grandmaster.

As a result of BMC, no more than one port of a Boundary Clock can be in the PTP_SLAVE state. Such a solution allows for redundancy of the time source and topology but is not optimal for the continuity of the synchronization. For example, in case of a failure of one of the *best* clocks (reference-synchronized grandmasters), all the slave nodes in its logic tree need to switch synchronization to the alternate grandmaster. This requires the restart of the estimation of the clock drift, mean path delay and offset which might cause fluctuations in the notion of time.

The modified BMC allows for more than one *best* clock in a single domain, enabling the creation of a logic topology with multiple roots. A Boundary Clock running the modified BMC can have more than one port in the PTP_SLAVE state. This means that timing information is exchanged between the Boundary Clock and more than one source of time (Ordinary Clock or Boundary Clock). At any time any of these links can be used to perform synchronization, including a weighted average from all SLAVE ports as mentioned in [6].

The BMC algorithm includes the State Decision Algorithm (SDA) and the Data Set Comparison Algorithm (DCA) and defines which fields of Data Sets should be updated depending on the outcome of the SDA. The modifications to the BMC are detailed below. The DCA is not modified, therefore it is not discussed.

6.4.2 State Decision Algorithm (SDA)

The original SDA is depicted in Figure 26 of the PTP standard. The modified SDA, depicted in Figure 7, enforces the *BMC_SLAVE* state instead of the *BMC_PASSIVE* state on the clocks with Class field value greater than 127 (block-13 of Figure 7). A port being in a *PTP_SLAVE* state as a result of the SDA modification (block-13) is considered a Secondary SLAVE port. The port which enters the *PTP_SLAVE* state based on the decision at block-11, is considered the Primary SLAVE port. This means that:

- the Primary SLAVE port and the Secondary SLAVE port(s) are synchronized to the same grandmaster clock but the connection of the Primary SLAVE port is possibly better by path, or

- the Primary SLAVE port and the Secondary SLAVE port(s) are synchronized to different grandmaster clocks of the same clockClass range (1-127 or 128-255,see Table 5 of PTP) but the grandmaster clock connected to the Primary SLAVE port is better or better by path.

The best qualified Announce messages (E_{rbest}, see Clause 9.3.2.3 of PTP) from all Secondary SLAVE ports shall be compared with the DCA to determine the "second best master" and the lower order masters. The sequence of events is as follows (the idea originated from [6]):

- The best master is computed with the BMC – the Primary SLAVE port established (block-11 of Figure 7) and the parentDS data set updated (S1). The port number is stored in currentDS.primarySlavePortNumber.
- If the recommended state is BMC_SLAVE at block-13, the master is considered a second best master and its data is stored in E_{srbest}.
- The BMC algorithm is executed on all the E_{srbest}.
- The best master selected is considered second best master and its reception port is considered the best Secondary SLAVE port.
- The BMC is repeated excluding E_{srbest} of the best master until the set of E_{srbest} is empty.

The order of second and lower level best masters determined in this way is used to update the backupParentDS data set.

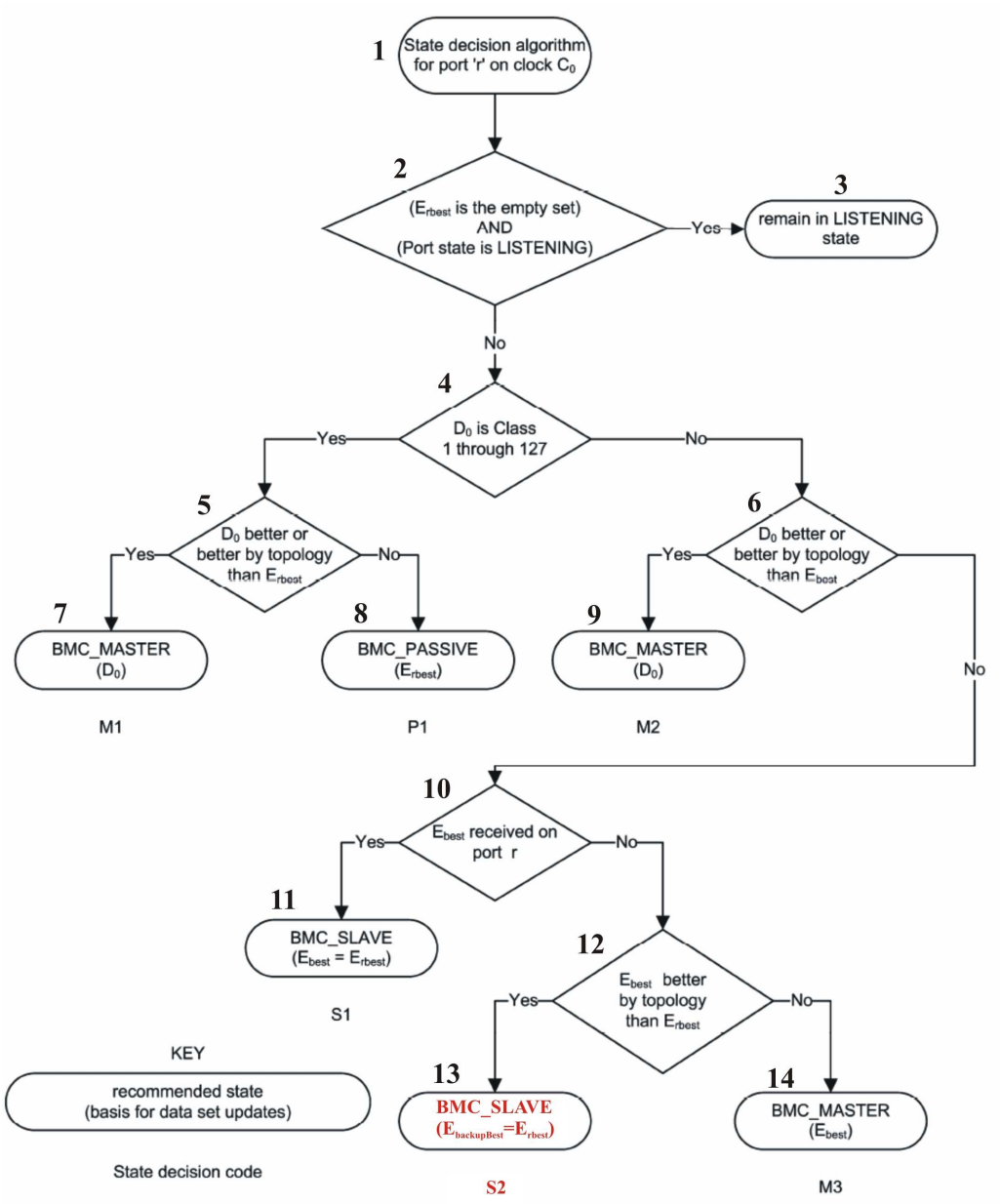


Figure 7: Modified State Decision Algorithm (modifications in red).

6.4.3 Update of Data Sets

The modifications to the rules governing the update of the data sets are twofold:

- Modifications to already existing ”update tables” (Table 13 and Table 16, clause 9.3 of PTP) to accommodate the new DS fields.
- Addition of a new ”update table” to accommodate the new backupParentDS data set and the modifications in the SDA.

Table 3: Modification to Table 13 of IEEE1588-2008: Update for state decision code M1 and M2.

Update this field	**From the indicated field of the defaultDS data set of the clock unless otherwise stated**
currentDS	
currentDS.primarySlavePortNumber	set to 0
portDS	
portDS.parentWrConfig	portDS.wrConfig
portDS.parentWrMode	portDS.wrMode
portDS.parentWrModeOn	portDS.wrModeOn
portDS.parentCalibrated	portDS.calibrated

Table 4: Modification to Table 16 of IEEE1588-2008: Update for state decision code S1.

Update this field	**From the indicated source**
currentDS	
currentDS.primarySlavePortNumber	portDS.portIdentity.portNumber

Table 5: Update for state decision code S2

Update this field	**From the indicated source**
backupParentDS	
backupParentDS.secondarySlavePortNumber	portDS.portIdentity.portNumber
backupParentDS.backupParentSourcePortIdentity	sourcePortIdentity of E_{rbest}

6.5 WRPTP Messages

White Rabbit benefits from PTP's messaging facilities. It defines a *WR Type-Length-Value* (WR TLV) extension to exchange WR-specific information and uses the two-step clock delay request-response mechanism for synchronization. In particular, it adds a suffix to the Announce message to enable recognition of WR nodes and uses Signaling Messages to exchange WR-specific information (Figure 6). During the exchange of timestamps, their fractional nanoseconds part is always included in the messages as defined in the PTP standard (cause 9.5.10) and explained in Appendix F.

A WR port in the *PTP_MASTER* state announces its presence by adding a suffix to the Announce message. The suffix is defined by the PTP standard as a set of TLV entities (section 13.4, PTP). Unrecognized by standard PTP nodes WR TLVs are ignored(section 14.1, PTP), but read and interpreted by White Rabbit nodes. The information provided in the WRPTP Announce message is sufficient for another WR port to decide whether the WR link can be established and maintained. A WR port receiving a WRPTP Announce message enters (if conditions in 6.7.1 are fulfilled) the WR Slave mode and starts the Link Setup process (performed in the *PTP_UNCALIBRATED* state, see Appendix D). It requests the sender of the WRPTP announce message to enter the WR Master mode (see section 6.7.2) and start the Link Setup process as well. During the WR Link Setup, communication between the WR Master and the WR Slave is performed using PTP Signaling Messages carrying WR TLVs (section 6.8). Once the WR link has been established, the WR nodes use a PTP delay request-response mechanism (section 11.3, PTP).

6.5.1 WR Type-Length-Value

All PTP messages can be extended by means of a standard *type, length, value* (TLV) extension mechanism. White Rabbit uses the *ORGANIZATION_EXTENSION* TLV type
(tlvType=0x0003) which is defined in clause 14.3 of PTP. TLVs represented by this type are designated for vendors and standard organizations to extend the protocol for specific needs. The organization-specific TLV fields for the WR extension are defined and described in Table 6.

The WR-specific TLVs are identified by wrMessageID which is defined in Table 7. The different types of WR messages are described in subsequent sections.

Table 6: Organization specific TLV fields for WR (see Table 35, PTP).

Bits								Octets	TLV Offset	WR TLV Content	Description
7	6	5	4	3	2	1	0				
tlvType								2	0	0x0003	Organization extension, see Table 34 PTP.
lengthField								2	2	8+N	N is an even number of wrDataField octets.
OrganizatinId								3	4	0x080030	OUI owned by CERN.
organizationSubType								2	7	magicNumber	0xABCD - identifies WRPTP within protocols identified by CERN's OUI.
								1	9	versionNumber	0x01 - WRPTP version.
DataField								2	10	wrMessageID	WR-specific messages identifier, see Table 7.
								N	12	wrDataField	Content of WR-specific message.

Table 7: White Rabbit Message ID values

WR Message Name	wrMessageId value (hex)	Sent in message type
SLAVE_PRESENT	0x1000	Signaling
LOCK	0x1001	Signaling
LOCKED	0x1002	Signaling
CALIBRATE	0x1003	Signaling
CALIBRATED	0x1004	Signaling
WR_MODE_ON	0x1005	Signaling
ANN_SUFIX	0x2000	Announce

6.5.2 WRPTP Announce Message

The standard PTP Announce Message is suffixed by one WR TLV entity. The WRPTP Announce message has the structure defined in Table 8. The *dataField* of the suffix WR TLV stores the *wrFlags* defined in Table 9.

Table 8: White Rabbit Announce Message.

Bits								Octets	TLV Offset	Content
7	6	5	4	3	2	1	0			
header								34	0	section 13.3, PTP.
body								30	34	section 13.5, PTP.
tlvType								2	64	0x0003, see 6.5.1.
lengthField								2	66	0xA.
OrganizationId								3	68	0x080030.
magicNumber								2	71	0xABCD.
versionNumber								1	73	0x01.
wrMessageId								2	74	0x0010.
wrFlags								2	76	see Table 9.

Table 9: White Rabbit flags.

Octet	Bit	Message type	Name	Description
0	0	Announce	wrConfig	The value of wrConfig parameter from portDS.
0	1			
0	2	Announce	calibrated	TRUE if the source port is calibrated.
0	3	Announce	wrModeOn	The wrModeOn value (6.3.1.2.3) of the sending port which shall be stored in parentWrModeOn (6.3.1.2.18) of the receiving port.

6.5.3 WRPTP Signaling Messages

White Rabbit uses Signaling Messages (defined in clause 13.12 of PTP) to exchange WR-specific information, except for the flags transported in the suffix of the Announce message.

The Signaling Messages conform to the format presented in Table 10. Each WR Signaling Message transports a single WR TLV structure as defined in Section 6.5.1. The *targetPortIdentity* shall be always set to the *clockIdentity* of the port on the other side of the link (conveyed in the Announce Message).

The Signaling Messages are used in WRPTP to trigger transitions in the WR State Machine and exchange WR-specific parameters. Each message is sent on entering particular states of the WR State Machine as described in section 6.8.

The distinction between WR Signaling Messages is made by the *wrMessageId* field of the *WR TLV*, defined in Table 7. Signaling Messages are exchanged only within a single link connection (no forwarding). This is ensured by setting the targetPortId appropriately. The rest of this subsection describes the WRPTP Signaling Messages in detail.

6.5.3.1 SLAVE_PRESENT

Message sent by the WR Port which became WR Slave (entered WR Slave mode) to the link-partner WR Port. It initiates the WR Master mode (the port becomes WR Master) on the link-

Table 10: PTP Signaling message fields (Table 33, PTP)

Bits								Octets	TLV Offset
7	6	5	4	3	2	1	0		
header								34	0
targetPortIdentity								10	34
WR TLV								M	44

partner WR Port which starts the WR Link Setup. The message shall have the form specified in Table 11.

6.5.3.2 LOCK

Message sent by the WR Master to the WR Slave to request the start of frequency locking. The message shall have the form specified in Table 11.

6.5.3.3 LOCKED

Message sent by the WR Slave to the WR Master. It indicates the successful completion of frequency locking. The message shall have the format specified in Table 11.

Table 11: WR TLV for SLAVE_PRESENT, LOCK, LOCKED AND WR_MODE_ON Signaling Messages.

Bits								Octets	TLV Offset	Content
7	6	5	4	3	2	1	0			
tlvType								2	0	0x0003, see 6.5.1.
lengthField								2	2	0x8.
OrganizationId								3	4	0x080030.
magicNumber								2	7	0xABCD.
versionNumber								1	9	0x01.
wrMessageId								2	10	Defined in Table 7.

6.5.3.4 CALIBRATE

Message sent by the WR port entering the *CALIBRATION* state (see section 6.7). It informs the other port whether sending a calibration pattern (see section 5.2) is required (defined by the value of *calSendPattern* flag). If calibration is required, it carries the calibration period. The message format and parameters are described in Table 12.

Table 12: WR TLV CALIBRATE Signaling Message

Bits								Octets	TLV Offset	Content
7	6	5	4	3	2	1	0			
tlvType								2	0	0x0003, see 6.5.1.
lengthField								2	2	0x0E.
OrganizationId								3	4	0x080030.
magicNumber								2	7	0xABCD.
versionNumber								1	9	0x01.
wrMessageId								2	10	CALIBRATE.
calSendPattern								1	12	The value determines whether the calibration pattern should be sent. If the value is 0x1, the calibration pattern is sent. If the value is 0x0, the calibration pattern is not sent. The value shall be based on the information provided by the *calibrated* Data Set field (6.3.1.2.8) It shall be stored in otherPortCalSendPattern (6.3.1.2.24).
calRetry (UInteger8)								1	13	The calRetry value (6.3.1.2.14) of the sending port which shall be stored in otherPortCalRetry (6.3.1.2.23) of the receiving port.
calPeriod (UInteger32)								4	14	The calPeriod value (6.3.1.2.13) of the sending port which shall be stored in otherPortCalPeriod (6.3.1.2.22) of the receiving port.

6.5.3.5 CALIBRATED

Message sent by a WR port entering the *CALIBRATED* state. If preceded by the *CALIBRATE* message with *calSendPattern* set to TRUE, it indicates successful completion of the calibration. The message provides the other port with the values of its fixed delays (Δ_{tx} and Δ_{rx}). The message shall have the format specified in Table 13.

6.5.3.6 WR_MODE_ON

Message sent by the WR Master to the WR Slave. It indicates the successful completion of the WR Link Setup process and requests the WR Slave to validate its WR mode (by setting wrModeOn to TRUE). The message shall have the format specified in Table 11.

Table 13: WR TLV for CALIBRATED Signaling Message.

Bits								Octets	TLV Offset	Content
7	6	5	4	3	2	1	0			
tlvType								2	0	0x0003, see 6.5.1.
lengthField								2	2	0x18.
OrganizationId								3	4	0x080030.
magicNumber								2	7	0xABCD.
versionNumber								1	9	0x01.
wrMessageId								2	10	CALIBRATED.
deltaTx (UInteger64)								8	12	The deltaTx value (6.3.1.2.9) of the sending port which shall be stored in otherPortDeltaTx (6.3.1.2.20) of the receiving port.
deltaRx (UInteger64)								8	20	The deltaRx value (6.3.1.2.10) of the sending port which shall be stored in otherPortDeltaRx (6.3.1.2.21) of the receiving port.

6.6 PTP State Machine

The WR Link Setup (WR State Machine) is performed in the PTP_MASTER state of the PTP state machine on a WR Port in WR Master mode (WR Master). On a WR Port in WR Slave mode (WR Slave), the WR State Machine is executed in the PTP_UNCALIBRATED state of the PTP state machine. Therefore, it is essential for a WR port to implement the PTP_UNCALIBRATED state as specified in the PTP state machine (1 in Figure 8): a transition state between the LISTENING or PRE_MASTER or MASTER or PASSIVE state and the SLAVE state. Additionally, WRPTP defines the implementation-specific *SYNCHRONIZATION_FAULT* and
MASTER_CLOCK_SELECTED PTP state transition events, see Figure 8.

6.6.1 MASTER_CLOCK_SELECTED

On successful completion of WR State Machine execution (transition from WR_LINK_ON to the IDLE state with wrModeOn set to TRUE) on the port being in the WR Slave mode and PTP_UNCALIBRATED state, the MASTER_CLOCK_SELECTED event (2 in Figure 8, defined in Clause 9.2.6.13 of PTP) shall occur. Consequently, the PTP_SLAVE state shall be entered.

6.6.2 SYNCHRONIZATION_FAULT

The port being in the WR Slave mode and PTP_SLAVE state shall evaluate values of two WR-specific parameters: *wrModeOn* (6.3.1.2.3) and *parentWrModeOn* (6.3.1.2.18). If the value of at least one of the parameters is FALSE, the SYNCHRONIZATION_FAULT event (3 in Figure 8, defined in Clause 9.2.6.12 of PTP) shall occur and PTP_UNCALIBRATED shall be entered. Consequently the WR Link Setup is performed. Such implementation enables forced re-calibration of WR ports which can be triggered by both, WR Master and WR Slave.

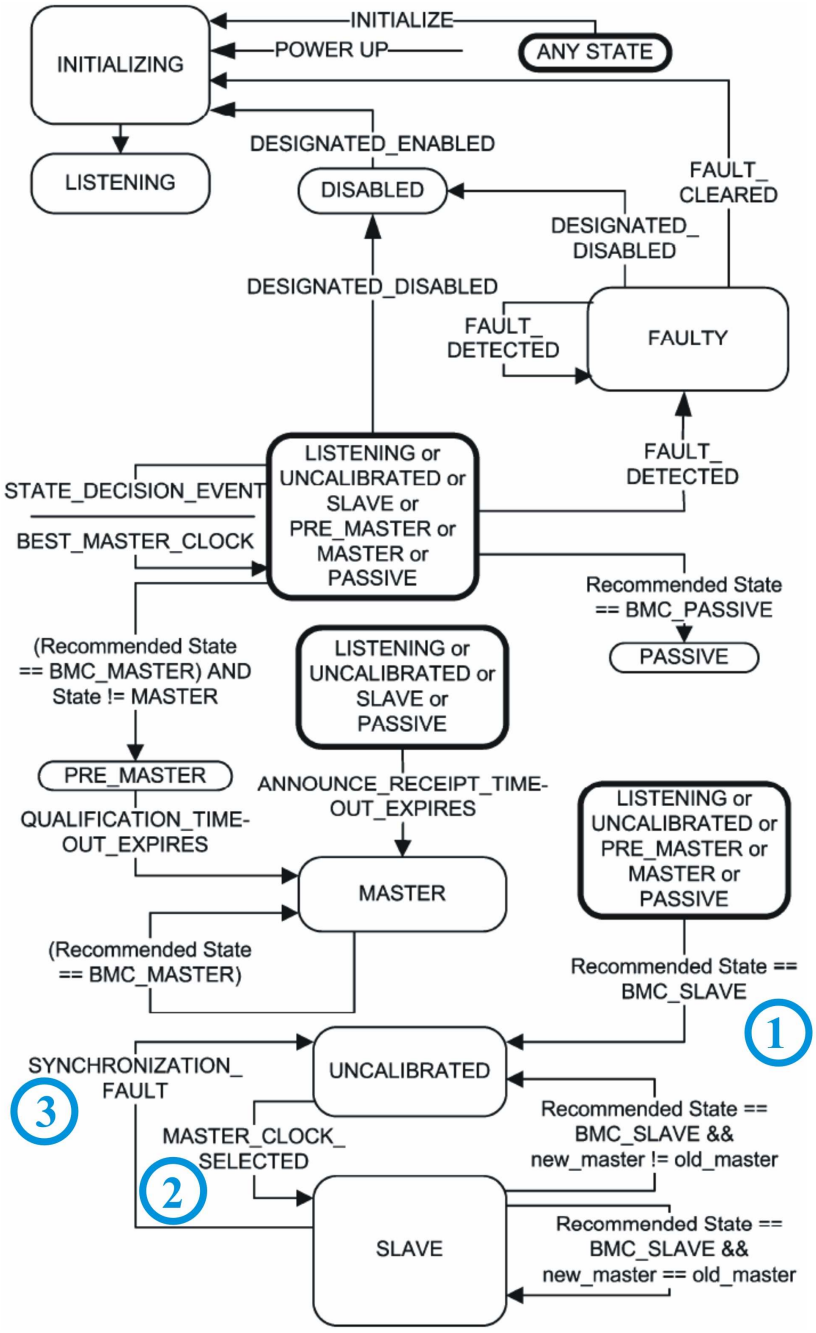


Figure 23—State machine for a full implementation

Figure 8: Modified PTP Finite State Machine.

6.7 White Rabbit State Machine

The White Rabbit Finite State Machine (WR FSM) controls the process of establishing a White Rabbit link between a WR Master and a WR Slave (WR Link Setup). It shall be non-preemptive with regards to the execution of the PTP State Machine. This means that, once the IDLE state of the WR State Machine is exited (WR State Machine execution started), no transitions on the PTP State Machine shall occur until the WR State Machine finishes execution and returns to the IDLE state.

WR Link Setup involves the recognition of two compatible WR ports, syntonization over the physical layer, optional measurement of fixed delays and exchange of their values across the link. The procedure differs between WR Master and WR Slave. Therefore three states of the WR FSM are Slave-only (entered only if the port is in WR Slave mode) and one state is Master-only (entered only if the port is in WR Master mode). The WR FSM shall be executed in the PTP UNCALIBRATED state on the WR Slave (WR Port in WR Slave mode) and in the PTP MASTER state on the WR Master (WR Port in WR Master mode). The WR FSM is depicted in Figure 9 and described in the rest of this section.

Successful completion of WR FSM results in the validation (*wrModeOn* set to TRUE) of the recommended WR mode (*wrMode*) and WRPTP synchronization. If EXC_TIMEOUT_RETRY (section 6.7.4) occurs, it means that WR Link Setup failed, WRPTP synchronization cannot be established and standard PTP is performed.

A typical flow of a complete WR Signaling Message exchange between a WR Slave and a WR Master during WR Link Setup is presented in Appendix C. For clarity, the cooperation of PTP and WR state machines from the power up is presented in Appendix D.

6.7.1 Conditions to enter WR Slave mode and start the WR FSM

A WR port becomes a WR Slave (portDS.wrMode = WR_SLAVE) and starts execution of the WR FSM by entering the *PRESENT* state, only when the following conditions are met:

- the port is WR Slave-enabled:
  portDS.wrConfig = (WR_S_ONLY ***OR*** *WR_M_AND_S)*
  AND

- the PTP state machine enters the PTP UNCALIBRATED state as a result of STATE_DECISION_EVENT: Recommended State == BMC_SLAVE (1 in Figure 8) **OR** SYNCHRONIZATION_FAULT event
  AND

- the parent port is WR Master-enabled:
  portDS.parentWrConfig = (WR_M_ONLY ***OR*** *WR_M_AND_S)*
  AND

- at least one of the ports on the link is not in WR Mode:
  portDS.wrMode = FALSE ***OR*** *portDS.ParentPortWrMode = FALSE*.

6.7.2 Conditions to enter WR Master mode and start the WR FSM

A WR port becomes a WR Master (portDS.wrMode = WR_Master) and starts execution of WR FSM by entering the *M_LOCK* state only when the following conditions are met:

- the port is WR Master-enabled:
  portDS.wrConfig = WR_M_ONLY ***OR*** *WR_M_AND_S*
  AND
- the port is in PTP_MASTER state
  AND
- the *SLAVE_PRESENT* Signaling Message has been received.

6.7.3 State Description

Table 14 specifies the WR states notation used in Figure 9.

Table 14: WR state definition

PTP portState	Description
IDLE	The WR FSM shall be in the IDLE state while WR Link Setup is not being performed.
PRESENT	Slave-only state. Upon entering this state, the WR Slave sends a SLAVE_PRESENT message to the WR Master and waits for the *LOCK* message.
M_LOCK	Master-only state. Upon entering this state, the WR Master sends the *LOCK* message. In this state, the WR Master waits for the WR Slave to finish successfully the locking process (reception of the *LOCKED* message).
S_LOCK	Slave-only state. The WR Slave locks its clock signal to the frequency distributed over the physical layer by the WR Master.
LOCKED	Slave-only state. Upon entering this state, the WR Slave sends the *LOCKED* message to inform that it is syntonized, and waits for the *CALIBRATE* message.
CALIBRATION	In this state, optional calibration of the port's reception and/or transmission fixed delays can be performed. Upon entering this state, the WR Port sends a *CALIBRATE* message to the other WR Port. If the calibration of reception fixed delay is needed, the *calSendPattern* flag in the *CALIBRATE* message is set to TRUE (0x1). If the calibration is not needed, the *calSendPattern* flag is set to FALSE (0x0). If calibration is not needed, the next state is entered, otherwise an indication from the hardware that the calibration has been finished successfully is awaited.
CALIBRATED	Upon entering this state, the WR Port sends a CALIBRATED message with the values of its fixed delays.
RESP_CALIB_REQ	The WR Port's action in this state depends on the value of the *calSendPattern* flag received in the *CALIBRATE* message. A TRUE value of the flag indicates that the calibration pattern shall be enabled while the WR State Machine is in this state. The pattern shall be disabled on exiting the state (after the timeout of *calPeriod* or on reception of the *CALIBRATED* message). If the value of the *calSendPattern* flag is FALSE, the *CALIBRATED* message is awaited for a timeout of calPeriod or wrStateTimeout (if calPeriod is 0x0). On reception of the *CALIBRATED* message, the next state is entered.
WR_LINK_ON	Upon entering this state, the WR Master sends the WR_LINK_ON message. In this state, the values of *wrModeOn (6.3.1.2.3)* and *parentWrModeOn (6.3.1.2.18)* are set to TRUE[f] and the *IDLE* state is entered unconditionally. The execution of the WR FSM is considered to be completed successfully.

[f]Depending on the implementation, it might be necessary to update the wrModeOn field of the best Foreign Master record (clause 9.3.2.4.1 of PTP).

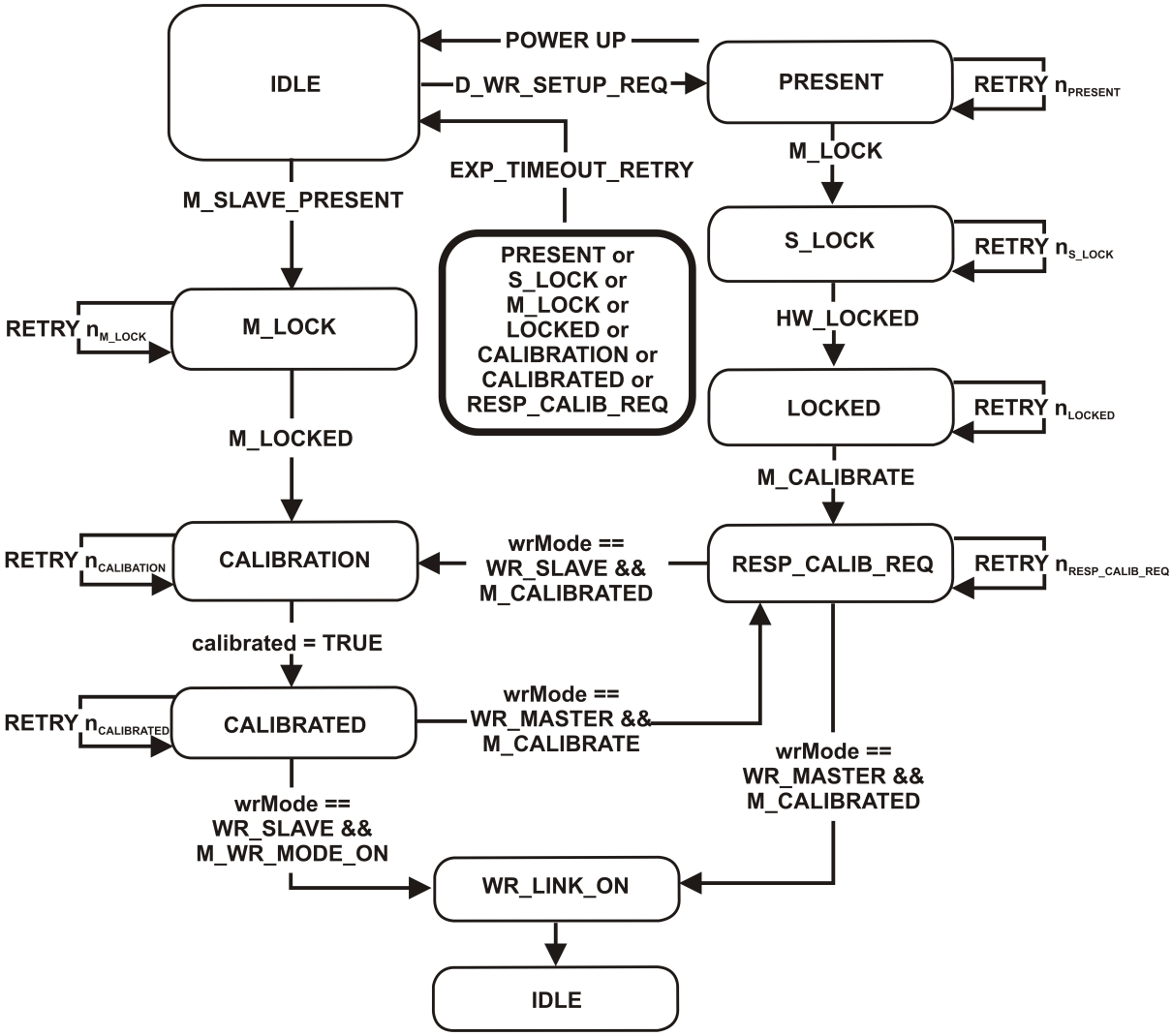


Figure 9: White Rabbit state machine.

6.7.4 WR FSM Transition Events and Conditions

POWERUP Turning on power to the device or resetting.

WR LINK SETUP REQUIRED DECISION (abrv. D_WR_SETUP_REQ) Event indicating that a Link Setup is required and that the WR FSM should be executed starting at the *PRESENT* state. It occurs when the condition presented in section 6.7.1 is fulfilled.

SLAVE PRESENT MESSAGE (abrv. M_SLAVE_PRESENT) Event triggered by the reception of the *SLAVE_PRESENT* Signaling message. It requests the recipient to enter the *M_LOCK* state.

LOCK MESSAGE (abrv. M_LOCK) Event triggered by the reception of the *LOCK* Signaling message.

LOCKED HARDWARE EVENT (abrv. HW_LOCKED) Event triggered by the reception of the hardware event notification (Table 17) indicating that frequency locking has been

completed successfully.

LOCKED MESSAGE (abrv. M_LOCKED) Event triggered by the reception of the *LOCKED* Signaling Message which notifies the WR Master that frequency locking has been completed successfully by the WR Slave.

CALIBRATE MESSAGE (abrv. M_CALIBRATE) Event triggered by the reception of the *CALIBRATE* Signaling message. It requests the recipient to enter the *RESP_CALIB_REQ* state.

CALIBRATED MESSAGE (abrv. M_CALIBRATED) Event triggered by the reception of the CALIBRATED Signaling message. It indicates that the port is calibrated. If the *CALIBRATED* message is received when the calibration pattern is being sent by the recipient, sending of the pattern shall be disabled.

WR MODE ON (abrv. M_WR_MODE_ON) Event triggered by the reception of the *WR_MODE_ON* Signaling message. It indicates that the WR Master finished successfully the WR Link Setup and set the wrModeOn flag to TRUE.

RETRY n_{name} The White Rabbit state machine waits in a given state for a transition event only for a limited time, called timeout. The default timeout for the following states is defined by *wrStateTimeout* (6.3.1.2.11): *M_LOCK*, *CALIBRATED*, *PRESENT*, *S_LOCK*, *LOCKED*, *CALIBRATION* and *RESP_CALIB_REQ*.
The timeout for the *CALIBRATION* state shall be overwritten by the calPeriod (6.3.1.2.13), if its value is greater than 0x0. Otherwise *wrStateTimeout* shall be used. The timeout for the *RESP_CALIB_REQ* shall be overwritten by the otherPortCalPeriod (6.3.1.2.22), if its value is greater than 0x0. Otherwise *wrStateTimeout* shall be used.
After the timeout expires, the state is re-entered. A state can be re-entered for a maximum number of $n_{\{M\_LOCK,CALIBRATION,CALIBRATED,PRESENT,S\_LOCK,LOCKED,RESP\_CALIB\_REQ\}}$ times. The default maximum number of state re-entering (retries) is defined by *wrStateRetry* (6.3.1.2.12). The number of n_{CALIB} shall be overwritten by *calRetry* (6.3.1.2.14) and $n_{RESP\_CALIB\_REQ}$ shall be overwritten by otherPortCalRetry (6.3.1.2.23) if their values are greater then 0x0. Otherwise, *wrStateRetry* shall be used.

EXCEED TIMEOUT RETRIES (abrv. EXC_TIMEOUT_RETRY) Indicates that a state has been re-entered for a maximum number of times ($n_{\{M\_LOCK,CALIBRATION,CALIBRATED,PRESENT,S\_LOCK,LOCKED,RESP\_CALIB\_REQ\}}$) and the n^{th} timeout has expired.

6.8 Communication between WR State Machines

This section clarifies the communication of the WR State Machines executed on different WR Ports on the same link during the WR Link Setup. The communication is performed using WR Signaling Messages (section 6.5.3). In particular, each WR Signaling Message is sent by a WR Port upon entering a particular state of WR State Machine. The same message received on the other WR Port triggers a transition in the WR State Machine (section 6.7.4).

Table 15 lists the WR Signaling Messages associating them with the moment of their transmission and the event they trigger on the reception WR Port. It also specifies the WR Mode of the sending port.

Table 15: Communication between WR State Machines on different Boundary Clocks.

WR Message	**Sent on entering WR state**	**Sent by port in wrMode**	**Triggers event**
SLAVE_PRESENT	PRESENT	WR Slave	M_SLAVE_PRESENT
LOCK	M_LOCK	WR Master	M_LOCK
LOCKED	LOCKED	WR Slave	M_LOCKED
CALIBRATE	REQ_CALIBRATION	WR Slave or WR Master	M_CALIBRATE
CALIBRATED	CALIBRATED	WR Slave or WR Master	M_CALIBRATED
WR_MODE_ON	WR_LINK_ON	WR Master	M_WR_MODE_ON

6.9 Communication between the WR State Machine and the hardware

White Rabbit extends the standard PTP communication with the hardware which includes transmission/reception of messages and retrieval of timestamps. The WR communication with the hardware is needed during the WR Link Setup process (by the WR State Machine) to control syntonization and fixed delay measurement and to retrieve their values. Additionally, outside of the WR Link Setup process (throughout the normal operation of the PTP State Machine) a quick detection of the link-down event is necessary.

6.9.1 Input to hardware

The information to the hardware by the WR State Machine is sent on the exiting or entering a state of this machine, as described in the Table 16. The inputs notify the hardware about the need to start/stop a process (i.e. syntonization, pattern transmission).

6.9.2 Output from hardware

The hardware shall communicate two types of information to the WR State Machine

- event notification, i.e. termination of fixed delays measurement, completion of phase locking or disconnection of the cable.
- parameter value, i.e. fixed delays.

An event notification shall be evaluated in a particular state of the WR State Machine (defined in Table 17) by means of polling or interrupt. Its occurrence in other state than defined in Table 17 shall be ignored.

The hardware output of a parameter value (i.e. deltaTx, deltaRx) shall be evaluated on the occurrence of an associated event notification (i.e. HW_CALIBRATED) and its value shall be stored in the local Data Set, as defined in Table 17.

Table 16: Input to hardware.

Hardware input	**Type**	**Sent on**	**Sent by port in wrMode**
HW_START_LOCKING	event notification	entering the S_LOCK State	WR Slave
HW_START_CALIBRATION[g]	event notification	entering the CALIBRATION State	WR Slave and WR Master
HW_ENABLE_PATTERN	event notification	entering the CALIBRATION and RESP_CALIB_REQ States	WR Slave and WR Master
HW_DISABLE_PATTERN	event notification	exiting the CALIBRATION and RESP_CALIB_REQ States	WR Slave and WR Master

Table 17: Output to hardware.

Hardware output	**Type**	**Evaluated**	**Evaluated by port in wrMode**	**Action**
HW_CALIBRATED[g]	event notification	in the CALIBRATION State	WR Slave and WR Master	*calibrated* Data Set field is set to TRUE, consequently WR State Machine transition is triggered
HW_LOCKED	even notification	in the S_LOCK State	WR Slave	Triggers transition in WR State Machine
HW_LINK_DOWN	even notification	always	WR Slave and WR Master	Defined in 6.10.
deltaTx[g]	UInteger64 [h]	on reception of HW_CALIBRATED	WR Slave and WR Master	Save the value in deltaTx DS field.
deltaRx[g]	UInteger64 [h]	on reception of HW_CALIBRATED	WR Slave and WR Master	Save the value in deltaRx DS field.

[g]Calibration is optional and implementation specific.

[h]The value of $\Delta_{tx_m,rx_s,rx_m,tx_s}$ measured in picoseconds and multiplied by 2^{16}.

6.10 Link Down

Standard PTP can accommodate a temporary connection loss on a link performing the protocol. Therefore detection of link down does not mean instant break in the operation of the PTP protocol. Appropriate action is taken only after a timeout expires (i.e. PTP State Machine transition and Data Sets updates). In WR, any loss of connection on a link performing WRPTP results in syntonization failure and requires re-establishing the WR Link. Therefore, on reception of the HW_LINK_DOWN hardware event notification (6.9.2), the following actions shall be performed:

- wrModeOn (6.3.1.2.3) shall be set to the initialization value (Table 1).
- wrMode (6.3.1.2.2) shall be set to the initialization value (Table 1).
- calibrated (6.3.1.2.8) shall be set to the initialization value (Table 1).
- the record of Foreign Masters on the port on which the event occurred shall be cleared.

6.11 Re-establishing the WR Link

It shall be possible for an external event (user demand, management message) which occurs on a WR Port in an *active* WR Mode (wrModeOn=TRUE, section 6.3.1.2.2) to enforce re-establishing of the WR Link, i.e. by performing the WR Link Setup.

The WR_MODE_OFF event instantiation is WR-implementation-specific. This event should be instantiated whenever a WR Port is in the IDLE state of the WR State Machine (regardless of the PTP state) and the implementation circumstances are detected that require re-establishing the WR Link.

The WR_MODE_OFF event results in setting the wrModeOn DS field of a given WR Port to FALSE.

6.12 White Rabbit PTP Profile Summary

6.12.1 Identification

Table 18: Profile print form (clause 19.3.3 of PTP)

PTP Profile	
profleName	White Rabbit
profileVersion	1.0
profileIdentifier	08-00-03-00-01-00
organizationName	European Organization for Nuclear Research (CERN)
sourceIdentification	http://www.ohwr.org/projects/white-rabbit

6.12.2 PTP attribute values

All nodes shall support the ranges and shall have the default initialization values for the attributes as follows:

- portDS.logSyncInterval: The default initialization value shall be 0. The configuration range shall be -1 to 6.
- defaultDS.priority1: The default initialization value shall be 64.
- defaultDS.domainNumber: The default initialization value shall be 0. Only the default domain is allowed.

6.12.3 PTP Options

All options of 15.5.4.1.7 and clause 17 of PTP are permitted. By default, these options shall be inactive unless specifically activated by a management procedure.
The node management shall implement the management message mechanism of the IEEE1588-2008 standard.
The best master algorithm shall be the algorithm specified in section 6.4 of this document (Modified BMC).
The delay request-response mechanism shall be the only path delay measurement mechanism.
The TLV mechanism described in section 6.5 shall be supported.

6.13 Implementation-specific additions to PTP required by WRPTP

WRPTP shall implement the following implementation-specific features:

- issuing PTP messages with a WR TLV (i.e. suffixed Announce and Signaling messages), as described in section 6.5,
- proper handling of PTP messages with a WR TLV as described in 6.5,
- a non-preemptive WR state machine as defined in 6.7,

- WR data set and additional WR-specific fields to PTP data sets as defined in 6.3,
- SYNCHRONIZATION_FAULT state transition as defined in 6.6.2,
- MASTER_CLOCK_SELECTED state transition as defined in 6.6.1,
- communication with hardware as described in 6.9.

Appendix

A PTP State Machine

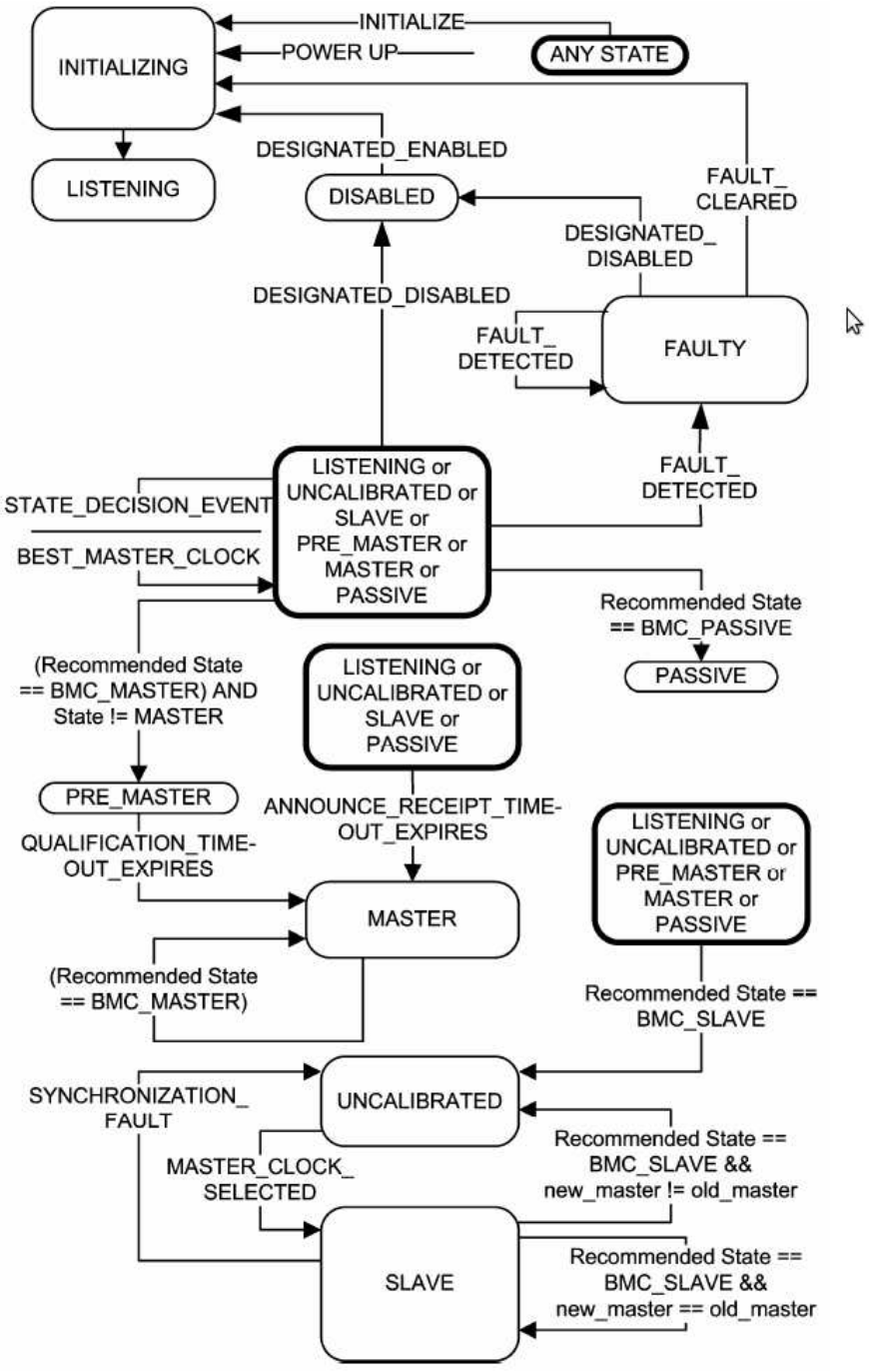


Figure 10: State machine for a full implementation of PTP (Figure 23, IEEE1588).

Table 19: PTP portState definition (Table 10, IEEE1588).

PTP portState	**Description**
INITIALIZING	While a port is in the INITIALIZING state, the port initializes its data sets, hardware, and communication facilities. No port of the clock shall place any PTP messages on its communication path. If one port of a boundary clock is in the INITIALIZING state, then all ports shall be in the INITIALIZING state.
FAULTY	The fault state of the protocol. A port in this state shall not place any PTP messages except for management messages that are a required response to another management message on its communication path. In a boundary clock, no activity on a faulty port shall affect the other ports of the device. If fault activity on a port in this state cannot be confined to the faulty port, then all ports shall be in the FAULTY state.
DISABLED	The port shall not place any messages on its communication path. In a boundary clock, no activity at the port shall be allowed to affect the activity at any other port of the boundary clock. A port in this state shall discard all PTP received messages except for management messages.
LISTENING	The port is waiting for the announceReceiptTimeout to expire or to receive an Announce message from a master. The purpose of this state is to allow orderly addition of clocks to a domain. A port in this state shall not place any PTP messages on its communication path except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, or signaling messages, or management messages that are a required response to another management message.
PRE_MASTER	The port shall behave in all respects as though it were in the MASTER state except that it shall not place any messages on its communication path except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, signaling, or management messages.
MASTER	The port is behaving as a master port.
PASSIVE	The port shall not place any messages on its communication path except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, or signaling messages, or management messages that are a required response to another management message.
UNCALIBRATED	One or more master ports have been detected in the domain. The appropriate master port has been selected, and the local port is preparing to synchronize to the selected master port. This is a transient state to allow initialization of synchronization servos, updating of data sets when a new master port has been selected, and other implementation-specific activity.
SLAVE	The port is synchronizing to the selected master port.

B Example White Rabbit Hardware Support Implementation

A White Rabbit switch, described in Tomasz Włostowski's Master Thesis [4], implements WRPTP and WR Hardware Support for Gigabit Ethernet over fiber. The most important fragments of the document are cited below (with necessary modifications). The citation is approved by the author.

B.1 WR link model

Knowledge of the physical model of the links connecting the clocks is a prerequisite for achieving the required synchronization accuracy. The model of a WR optical link is depicted in Figure 11. [...]

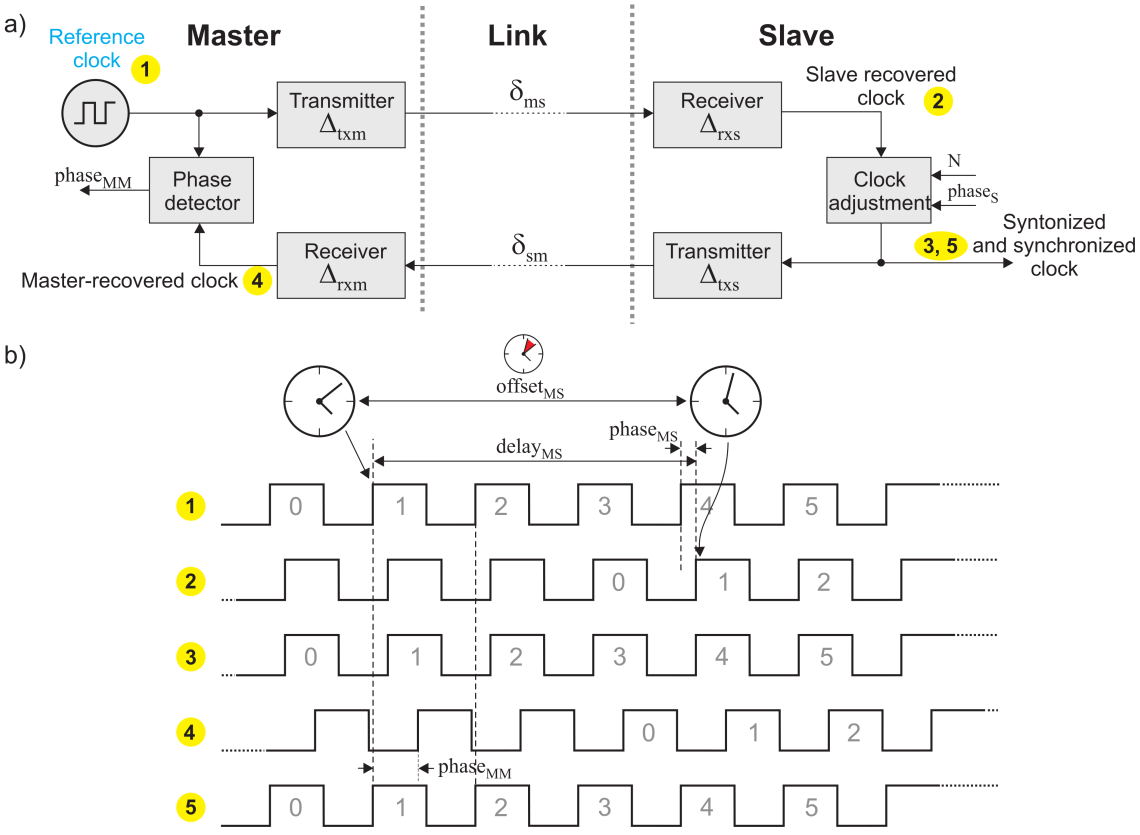


Figure 11: Model of a WR link (a) and relations between master and slave clocks (b)

WR employs the clock loopback technique to measure the round-trip phase shift $phase_{MM}$, which is a starting point for determining the precise two-way delay $delay_{MM}$ and thus the clock offset $offset_{MS}$. The clock signal is transferred between the master and the slave according to the following scheme:

1. The reference clock (1) is used to encode the master's transmitter output and then extracted from the data stream at the slave's receiver. The recovered clock (2) is a copy of

clock (1) delayed by $delay_{MS}$, where

$$delay_{MS} = \Delta_{txm} + \delta_{ms} + \Delta_{rxs} \tag{14}$$

expresses the master-to-slave latency introduced by the transceivers and the link.

2. The recovered clock (2) is fed to a clock adjustment unit which shifts its phase by a programmable value $phase_S$ to obtain the phase-compensated clock (3) – the final result of WR synchronization.

3. Clock (3) encodes the slave's outgoing data stream and is recovered at the master side as signal (4).

4. The master measures the phase shift $phase_{MM}$ between its outgoing (1) and incoming (4) clocks using a phase detector.

The value of $phase_{MM}$ can be expressed as:

$$phase_{MM} = \{\Delta + \delta_{ms} + \delta_{sm} + phase_S\} \bmod T_{ref} \tag{15}$$

where $\Delta = \Delta_{txm} + \Delta_{rxs} + \Delta_{txs} + \Delta_{rxm}$ and T_{ref} is the period of Gigabit Ethernet 125 MHz reference clock (8 ns).

The goal of the presented model is to calculate the precise value of master-to-slave offset $offset_{MS}$ by combining a coarse timestamp-based round-trip delay (4) with precise phase measurement $phase_{MM}$. Once the offset is computed, the WR slave can phase-shift its recovered clock (deriving $phase_S$ from $offset_{MS}$) to match the phase of the master clock, completing the synchronization. Determining the precise offset, however, is not a trivial task. Figure 12 shows the steps needed to achieve and maintain synchronization of a single WR link. The synchronization process can be split into two parts:

- **initial synchronization**, which determines the value of $offset_{MS}$ and fixes the slave's phase shift to compensate the offset.

- **phase tracking** which monitors the changes of $phase_{MM}$ over time and updates the phase shifter in the slave to follow these changes and sustain synchronization.

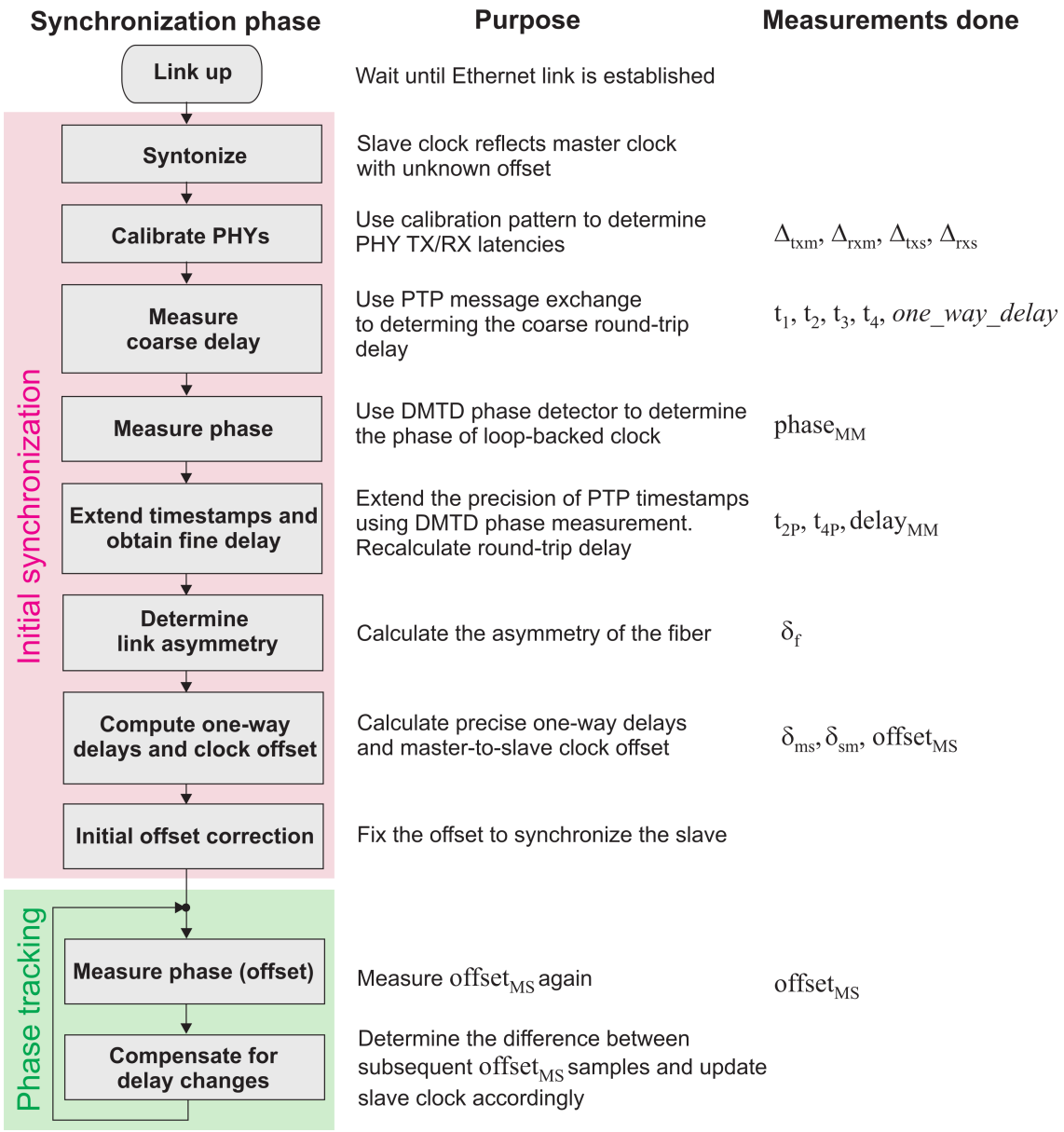


Figure 12: WR synchronization flow (DMTD is explained in Appendix B.4)

B.2 Link detection and syntonization

During the first two steps of the synchronization flow 12, a syntonized WR connection is established. Drawing 13 shows the order of hardware operations and message exchanges which result in a syntonized link. At the initial moment, the master and the slave are not connected to each other. Their PHYs are transmitting an Ethernet idle pattern (see Table 3.1 [4]), but are not receiving any meaningful bitstreams. As soon as the physical connection is present, the PHYs will start receiving valid idle patterns and the Ethernet link will become active (see [2] section 36.2.5.2.6). Presence of a valid physical link will trigger the following sequence:

1. The master starts broadcasting *ANNOUNCE* messages to look for a WR slave,
2. Eventually, the slave will respond with a *SLAVE_PRESENT* message, indicating that it

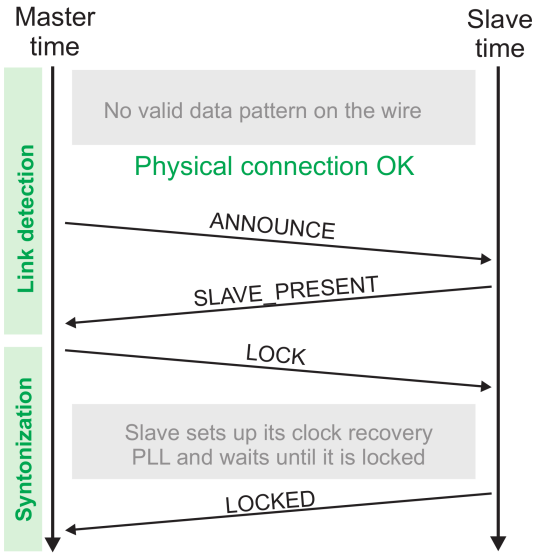


Figure 13: WR Link detection and syntonization

supports WR. If no response has been received within a predefined time, the master assumes that the slave is not WR compatible and aborts the synchronization process,

3. The master issues a *LOCK* message, commanding the slave to begin recovering the clock from its received data stream,

4. The slave sets up its PLL to use the RX clock as a reference and as soon as the PLL is locked, responds with a *LOCKED* message.

Upon successful completion of the above process, both nodes are syntonized and the delay measurements can begin.

B.3 Coarse delay measurement

The coarse delay measurement produces a set of timestamps $t_1 \ldots t_4$ which are obtained using the PTP End-to-End, two-way clock method as shown in fig. 2. The timestamps are generated in the hardware with a circuit depicted in fig. 14, guaranteeing single-cycle timestamping accuracy, which is necessary for reliably merging the coarse delay with the $phase_{MM}$ phase shift during fine delay calculation. Timestamps in WR are taken when the PCS detects a Start-of-Frame Delimiter (SFD) character in the incoming (RX timestamps) or outgoing (TX timestamps) data stream (see section 3.2.2 [4]).

Let's first focus on the blocks marked blue in Figure 14. Each time an SFD character is detected, the PCS produces a timestamp trigger pulse which causes the timestaming unit to take a snapshot of the free running counter CNTR_R with the D-type register DREG_R. The counter is counting from 0 to 124999999 which (given the reference clock frequency of 125 MHz) gives a period of one full second.

The counter CNTR_R works synchronously to the reference clock (master side, signal 1 in Figure 11) or the compensated clock (slave side, signal 5). Since RX trigger pulses come from different clock domains (2 or 4), they need to be synchronized to the reference clock with a chain of D flip-flops (SYNC). Single-cycle long trigger pulses are widened by the pulse

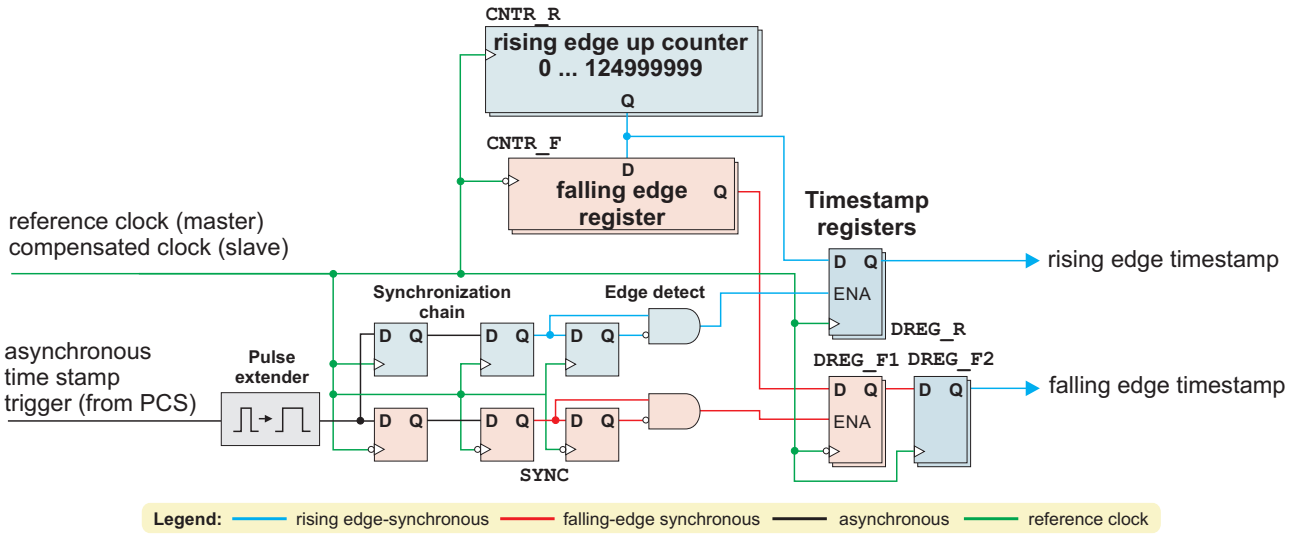


Figure 14: Structure of a WR timestamping unit (TSU)

extender before going to the synchronizer chain to ensure that no pulses are missed due to the metastability of synchronizer flip-flops. The counter value is latched in register DREG_R on the rising edge of the synchronizer output. TX timestamp triggers, which are generated within the reference clock domain (clocks 1 or 5) also pass through a synchronizer chain to obtain identical trigger reaction latency.

Unfortunately, due to the jitter of clock signals, crossing clock domains can make the gathered timestamps useless by causing a random ± 1 LSB error when the RX clock and the reference clocks are almost in phase. The problem is illustrated in Figure 15. Dashed lines show the transitions of ideal (jitter-free) clocks. If the jitter is neglected, the reference clock transition should be slightly ahead of the RX clock transition and the gathered timestamp should be equal to 2. However, if the clocks are jittery, the transitions may sometimes occur in reverse order, producing an erroneous timestamp of value 1. One possible way of addressing this issue is to

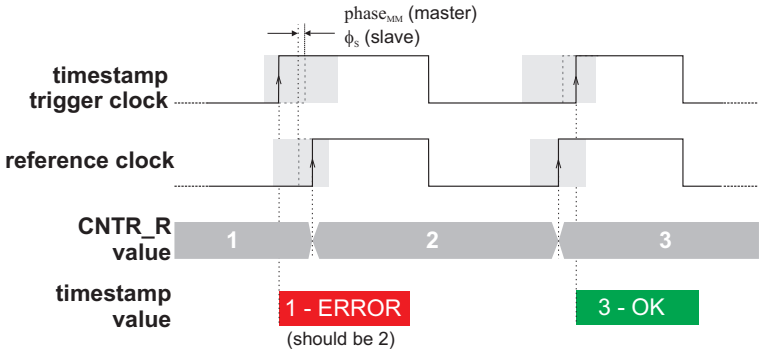


Figure 15: Timestamping errors caused by clock jitter

take RX timestamps on both reference clock edges. The falling edge part of the TSU is marked pink in 14. It does not have an independent counter – instead, the current value of the rising edge counter is latched in register CNTR_F on the falling edge of the reference clock, making the CNTR˙F a copy of CNTR_R delayed by a half of the clock period. This method ensures that at least one of the timestamps is valid at any moment (see Figure 16). The correct timestamp

is chosen depending on the current phase shift between clocks (see section B.5). In order to

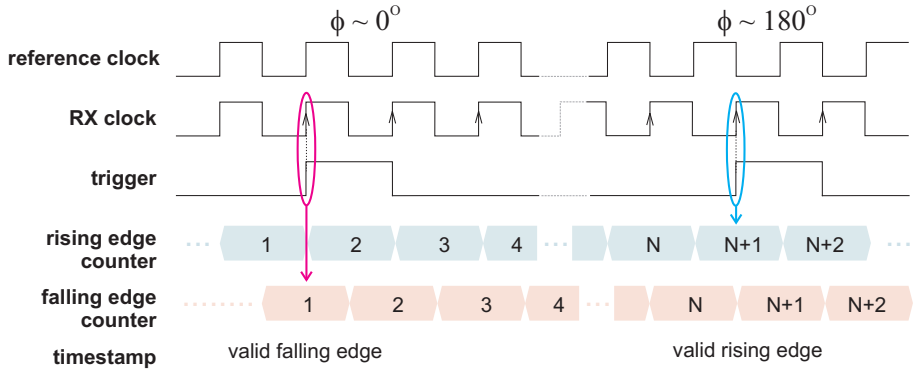


Figure 16: Dual-edge timestamping in WR

simplify the hardware design, the presented TSU can only measure the sub-second part of PTP timestamps. The UTC part is appended in the software using the algorithm shown in listing 1.

Listing 1: Producing UTC timestamps

```
  timestamp tstamp = get_hardware_timestamp();
  time t = get_utc_time();
// check if there was a transition of the UTC counter between the
// generation of the timestamp and its readout by the software
  if(t.milliseconds < tstamp.rising_edge)
   tstamp.seconds = t.seconds - 1;
  else
   tstamp.seconds = t.seconds;
```

B.4 Digital DMTD phase detector

The fine delay measurements in WR are based on a Dual Mixer Time Difference (DMTD) phase detector. Therefore, in this section a short introduction to DMTD technology will be presented. Figure 17 shows an analog DMTD system [7]. Let's assume that its input clocks $a(t)$ and $b(t)$ have identical amplitudes and frequencies equal to f_{clk}. The phases of both clocks are respectively ϕ_a and ϕ_b. The clocks are multiplied by the mixers with the signal $c(t)$ of frequency f_{offset} and phase ϕ_{offset}. The multiplication result can be expressed as:

$$\begin{aligned} a(t)\cdot c(t) &= cos(2\pi t\, f_{clk}+\phi_a)\cdot cos(2\pi t\, f_{offset}+\phi_{offset}) \\ &= \frac{1}{2}cos(2\pi t(f_{clk}+f_{offset})+\phi_a+\phi_{offset}) \qquad (16) \\ &+ \frac{1}{2}cos(2\pi t(f_{clk}-f_{offset})+\phi_a-\phi_{offset}) \qquad (17) \end{aligned}$$

Term 16 has higher frequency than the multiplied signals and it's removed by the low-pass filter, leaving only the low-frequency term 17. The downconversion process changes the frequency of the signals, but does not change their phase difference. Therefore, if the offset

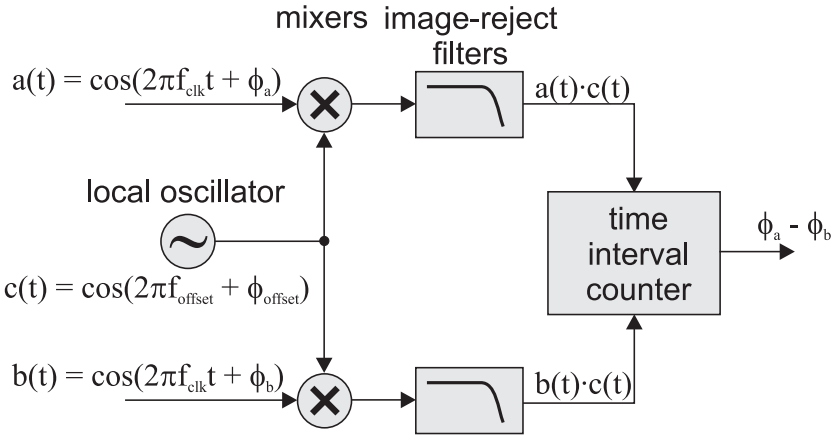


Figure 17: Structure of an analog DMTD phase detector

frequency f_{offset} is close to the input signal frequency f_{clk}, the phase shift can be measured by counting the time between the rising edges of the downconverted clocks. For example, a reference clock of 125 MHz and an offset clock of 124.99 MHz will produce an output signal of 10 kHz. At such frequencies, the phase shift can be very accurately measured using a simple counter.

Analog DMTDs provide excellent resolution and linearity, at the cost of several external discrete components (mixers and filters), which can be troublesome, especially in multi-port applications such as the WR switch. Fortunately, the analog mixing operation can be transformed into a digital sampling operation, resulting in a digital DMTD detector, shown on fig. 18. In a Digital DMTD (DDMTD) [8], the input signals are square waves, the mixers are re-

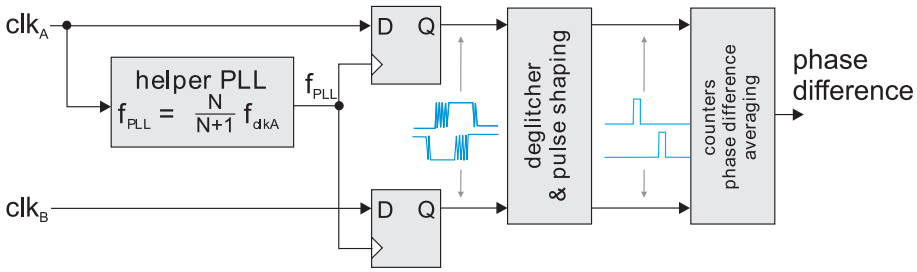


Figure 18: Structure of a digital DMTD phase detector

placed with simple D-type flip-flops and the offset clock is generated with a PLL from one of the input clocks. The sampling operation performed by the flip-flops can be mathematically proved to be equivalent to analog mixing (17), but the principle of a DDMTD can be explained in a more intuitive way. Figure 19a shows a vernier caliper. It has two scales - the big millimeter scale and a small vernier scale, with the units slightly smaller than the main scale. For a typical caliper, the vernier scale is split into 10 intervals with 4.9 mm spacing. If the length of the measured object has a fractional part, one of bars on the vernier scale will be in line with one of bars from the main scale. Translating to the language of electronics, the main scale is the input clock where each interval represents one cycle, the vernier scale is the offset clock and the transitions in the DDMTD output signals occur when bars on both scales are aligned. An example of signals produced by a DDMTD with $N = 8$ is shown in fig. 19b. The output

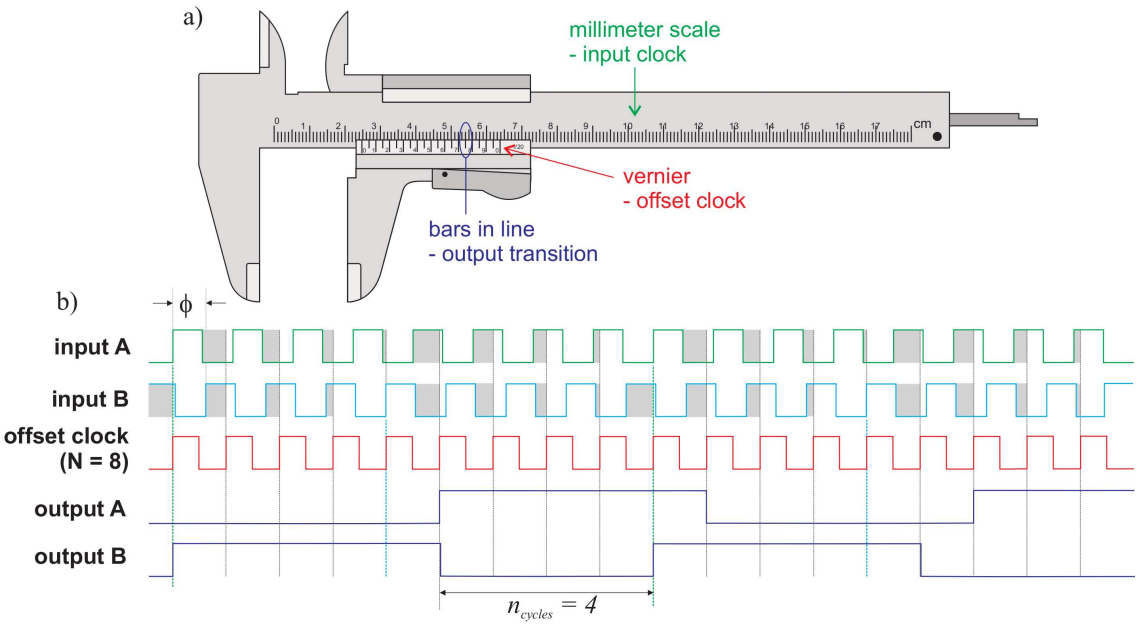


Figure 19: A vernier (a) and signals generated by DDMTD (b)

phase shift is proportional to the input phase shift ϕ by a factor of $(N+1)$. A general relation between the phase shift and the cycle interval between the DMTD outputs can be expressed as:

$$\phi[\text{ns}] = \frac{n_{cycles}}{N+1} \cdot \frac{1}{f_{in}} \tag{18}$$

Digital DMTDs have all the advantages of their analog counterparts while requiring only one external component (the oscillator producing the offset clock) which is shared among all measurement channels. This opens the way for low-cost FPGA implementations. If a picosecond-level accuracy is not required, even a PLL integrated inside an FPGA can be used, eliminating all external components. In practical DDMTD implementations, the output signals

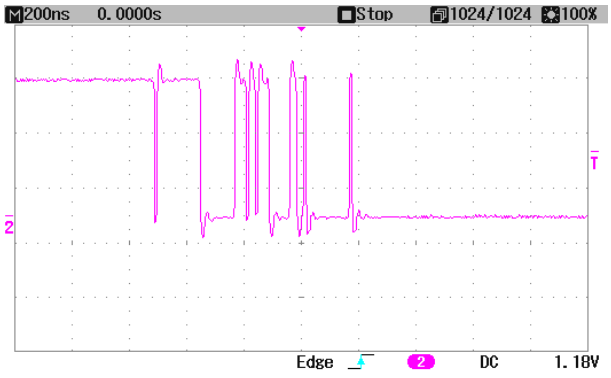


Figure 20: Glitches in the DMTD output caused by clock jitter

need to be additionally conditioned as the input clock jitter can introduce glitches, as shown on fig. 20. More details about the deglitching and postprocessing algorithm can be found in section 4.3.5 of [4].

B.5 Fine delay measurement

Having the knowledge of PTP timestamps and round-trip phase shift $phase_{MM}$, we can calculate the precise round-trip delay $delay_{MM}$. The calculation is performed in two steps:

- The accuracy of PTP timestamps t_2 and t_4 is enhanced beyond a single clock cycle using the knowledge of the round-trip phase $phase_{MM}$ and the slave PLL setpoint $phase_S$. As a result, precise timestamps t_{2p} and t_{4p} are obtained.
- The fine round-trip delay is calculated using the standard PTP formula:

$$delay_{MM} = (t_{4p} - t_1) - (t_3 - t_{2p}) \tag{19}$$

Only the reception timestamps need to be enhanced, as they are generated within clock domains asynchronous to the reference (or compensated) clock (see table 20). Transmission timestamps are always integer because packets are transmitted and timestamped using the same clock.

The flow graph of the algorithm used to merge PTP timestamps with phase measurements is shown on Figure 21. Figure 22 depicts sample measurements of inputs and results of the enhancement algorithm for t_{4p} timestamps, where the varying link delay was simulated using the slave's phase shifter.

Table 20: Timestamping clock domains

Timestamp	Origin	Trigger clock	Timestamping clock	Correction
t_1	Master TX	reference (1)	reference (1)	0
t_{2p}	Slave RX	slave recovered (2)	compensated (3)	$-phase_S$
t_3	Slave TX	compensated (3)	compensated (3)	0
t_{4p}	Master RX	master recovered (4)	reference (1)	$+phase_S$

The first step of the algorithm addresses the problem of glitches in RX timestamps, by choosing either the rising or falling edge timestamp depending on the phase between the RX clock and the reference clock. The key parameter of this step, ϕ_{trans} provides an approximate value of $phase_{MM}$ at which a transition should occur in the value of t_{4r}. In Figure 22, it is equal to 6.6 ns - it's the approximate intersection point of $phase_{MM}$ (blue, sawtooth-like trace) and the transition of the rising edge timestamp (red trace). The value of ϕ_{trans} is a device-specific constant. It can be determined once during the factory calibration of a WR device or measured upon every startup by sweeping a full clock period using the built-in phase shifter and searching for a transition in the RX timestamp value.

If the actual value of $phase_{MM}$ lies within a $\pm 25\% T_{ref}$ range from the transition point ϕ_{trans} (green zone in 22), the algorithm will use the t_{4f} timestamp, otherwise the t_{4r} timestamp will be taken (red zone). Note that because $phase_{MM}$ is bounded to $\left[0, T_{ref}\right)$, the range checks must be aware of the jump in the phase between T_{ref} and 0.

The second step checks if the $phase_{MM}$ value is ahead of the transition point ϕ_{trans}, and eventually increases the chosen timestamp by a full cycle (T_{ref}). That ensures the transition in t_4 will always occur when $phase_{MM} = \phi_{trans}$, eliminating the risk of transition glitches.

The last part of the algorithm simply adds the picosecond part (which is the DMTD phase corrected with the transition offset ϕ_{trans}) to the coarse deglitched timestamp t_4. If the resulting

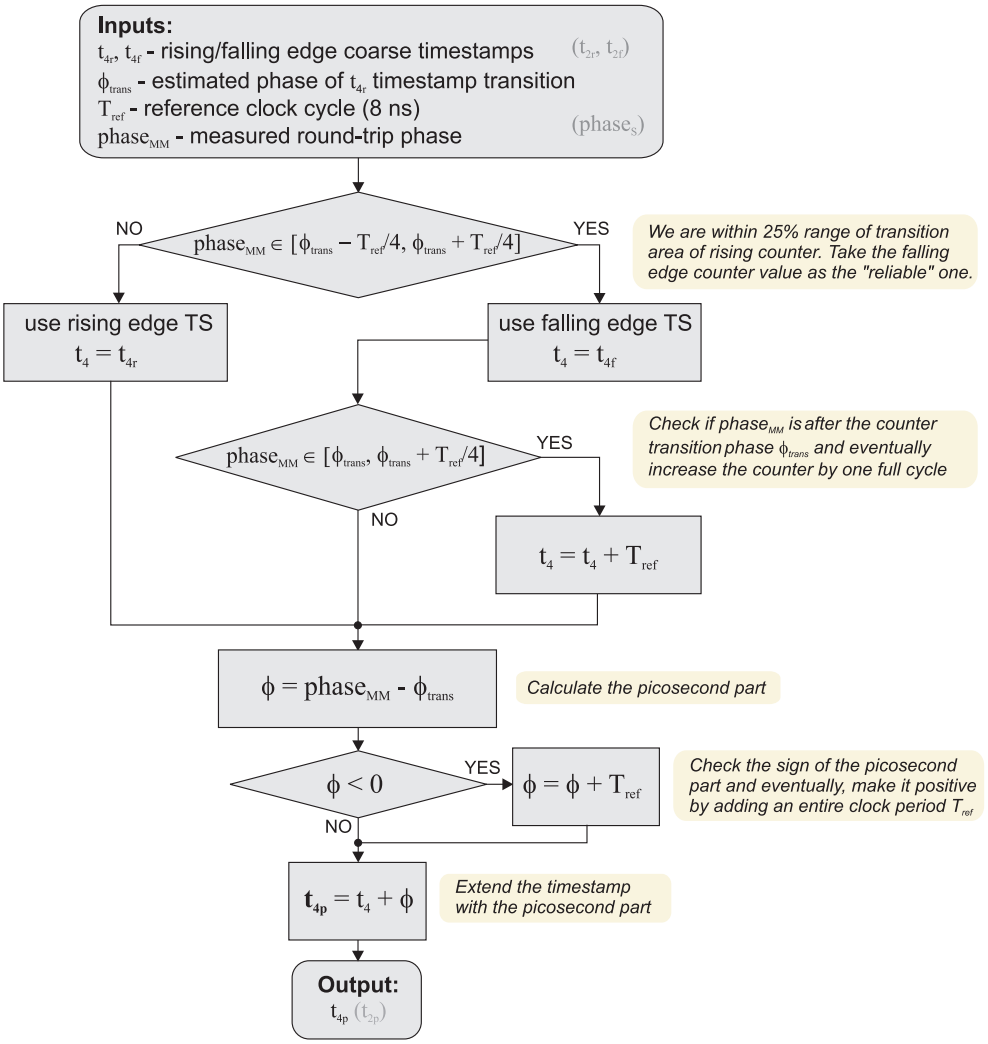


Figure 21: Algorithm for enhancing coarse timestamps with DMTD phase.

picosecond part (ϕ) is negative, an additional full cycle is added to the result. The final output of the merging algorithm is shown in 22 as the thick navy trace. The enhancement operation for the t_2 timestamp is done in a very similar way by replacing t_4 with t_2 and $phase_{MM}$ with $phase_S$. Note that changing the slave's phase shift $phase_S$ will result in a change of the values of both t_{2p} and t_{4p} (see table 20). Increasing $phase_S$ by a certain δ will cause t_{4p} to also increase by δ (assuming that the link delay stays constant). Simultaneously, the value of t_{2p} will be decreased by the same amount, as for the slave's RX timestamps, the timestamping clock is $phase_S$ ahead of the trigger clock (so if the phase shift is increased, the timestamp value goes down). Therefore, the calculated fine delay is not affected by the changes of $phase_S$:

$$delay_{MM} = (t_{4p} - t_1) - (t_3 - t_{2p}) = t_{4p} + \cancel{phase_S} - t_1 - t_3 + t_{2p} - \cancel{phase_S} \tag{20}$$

B.6 Link asymmetry estimation

In order to compute an accurate master-slave offset, one must determine the asymmetry of the link delay. This asymmetry cannot be measured directly (as a difference between the overall M-

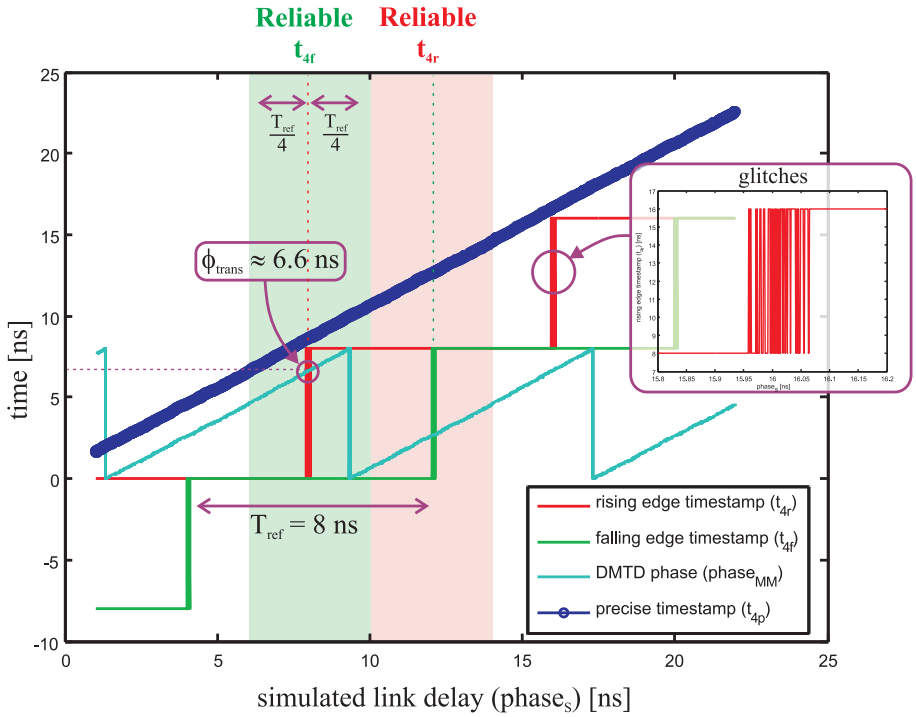


Figure 22: Example of t_{4p} timestamp enhancing.

S and S-M delays), as it would require the nodes to be synchronized prior to the measurement. Therefore it is only possible to estimate the asymmetry from round-trip delay $delay_{MM}$, using the knowledge of the properties of the components constituting the link.

In the WR optical link model, the following sources of asymmetry were taken into consideration:

- Propagation delays of electronic components and PCB traces (circuit asymmetry),
- Asymmetry of optical transceivers (SFPs),
- Difference between TX and RX wavelengths in the fiber,
- Internal structure and clocking of the PHY (SerDes) chips.

Figure 23 shows the reference asymmetry model used in WR devices. The device's asymmetric delays (as shown in Figure 11) are expressed as sums of circuit, SFP and PHY delays between the phase detector inputs (the PD measuring $delta_{MM}$ on the master side and the PD in the phase shifting PLL on the slave side) and the SFP optical input/output. The fiber asymmetry is compensated separately in the slave's PTP servo.

$$\begin{aligned} \Delta_{tx(m/s)} &= \delta_{TX\_PHY(M/S)} + \delta_{TX\_CIR(M/S)} + \delta_{TX\_SFP(M/S)} \\ \Delta_{rx(m/s)} &= \delta_{RX\_PHY(M/S)} + \delta_{RX\_CIR(M/S)} + \delta_{RX\_SFP(M/S)} \end{aligned} \tag{21}$$

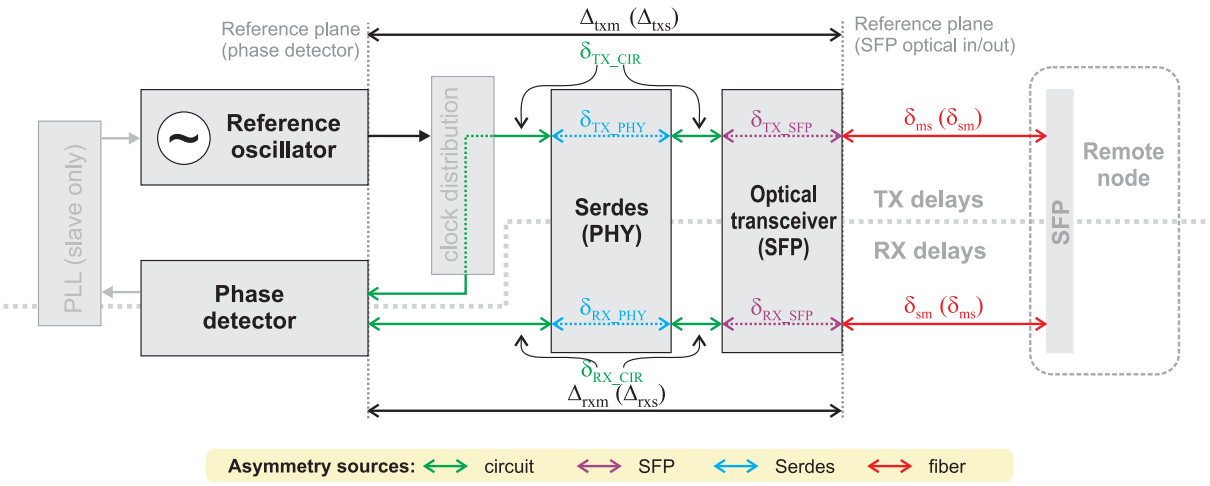


Figure 23: Delay asymmetries in WR optical link.

B.6.1 Circuit and SFP asymmetry

Circuit asymmetry stems from differing PCB trace lengths, propagation delays of the electronic components of the clock distribution system and FPGA logic cell placement and routing delays. SFP asymmetry is a result of different reception and transmit delays between the electrical and optical ports of the SFP transceiver.

These asymmetries can vary with changes of the device's operating conditions (i.e. temperature, supply voltage). Depending on the synchronization accuracy requirements, they can be:

- Treated as time-invariant and measured once during factory setup.
- Actively compensated, following the changes in temperature and/or supply voltage read from built-in sensors (requires a model of delay vs. temperature and voltage),
- Eliminated by building a system where both master and slave have identical (or zero) asymmetry and operate in similar conditions.

The last method is particularly useful for compensating the delays inside the FPGA on the way between the PHY (oscillator) and the phase detector, as they cannot be externally measured during the factory calibration process. They are also significantly affected by temperature and voltage changes. Therefore, the only practical way of dealing with the FPGA part of the circuit asymmetry is to equalize the delays on all phase detector inputs. This can be done by constraining the routing delays or manually placing and routing the phase detector block. This approach also allows for reducing the impact of the temperature-voltage changes on the asymmetry to a negligible value, as two equal paths placed next to each other would have very similar temperature/voltage delay coefficients.

SFP asymmetries can be compensated in a similar manner by choosing pairs of SFP transceivers whose TX and RX delays differ by similar values (i.e. $\delta_{TX\_SFPM} - \delta_{RX\_SFPM} = \delta_{TX\_SFPS} - \delta_{RX\_SFPS}$). This can be done in a laboratory system where all other asymmetric delays are already compensated by selecting SFPs that together give minimum clock offset.

B.6.2 WDM fiber asymmetry

As mentioned in section 3.2 of [4], WR uses different wavelengths for transmitting and receiving the data (for example 1550/1310 nm). Due to chromatic dispersion of the fiber, the refractive indexes for these wavelengths are slightly different, causing different propagation speeds (and thus, different delays) between the master and the slave.

The refractive index at a given wavelength can be derived using Sellmeier's equation [9] (22):

$$n^2(\lambda) = 1 + \frac{B_1\lambda^2}{\lambda^2 - C_1} + \frac{B_2\lambda^2}{\lambda^2 - C_2} + \frac{B_3\lambda^2}{\lambda^2 - C_3} \tag{22}$$

where B_i and C_i are material-specific coefficients.

For a standard G.652 telecom fiber, the refractive indexes are respectively: $n_{1550} = 1.467$ and $n_{1310} = 1.466$. In order to simplify the asymmetry calculations in the hardware, the WR specification defines a custom fiber asymmetry coefficient (23) expressing the ratio between the M-S and S-M fiber propagation delays:

$$\alpha = \frac{\delta_{ms}}{\delta_{sm}} - 1 = \frac{n_{1550}}{n_{1310}} - 1 \tag{23}$$

Unfortunately, refractive indexes may vary slightly between different fiber manufacturers, making the direct calculation of α unreliable. WR can solve this problem by characterizing the fiber asymmetry using laboratory measurements of $delay_{MM}$ (done by PTP) and clock offset $offset_{MS}$ (done using an oscilloscope). The measurements are performed when all the other asymmetries are already compensated. The value of α 25 is calculated by solving equation system 24:

$$\begin{cases} delay_{MM} = \delta_{ms} + \delta_{sm} + \Delta \\ offset_{MS} = \frac{\delta_{ms} - \delta_{sm}}{2} \end{cases} \tag{24}$$

where Δ accounts for all fixed delays in the path (i.e. $\Delta_{txm} + \Delta_{rxs} + \Delta_{txs} + \Delta_{rxm}$). Substitution of δ_{ms} and δ_{sm} with the results of 24 gives the final value of α:

$$\alpha = \frac{delay_{MM} - \Delta + 2 \cdot offset_{MS}}{delay_{MM} - \Delta - 2 \cdot offset_{MS}} \tag{25}$$

B.6.3 Transceiver asymmetry

[This chapter has been modified to add clarity]

Transceiver asymmetry is a result of the internal structure of the SerDes's circuitry. Most of the SerDes chips nowadays are optimized for low power consumption and fast locking to the incoming data stream. Unfortunately, because of these optimizations, PHYs may not keep constant transmit/receive latencies. The problem is illustrated in Figure 24. PHY asymmetry manifests itself as a random latency between the rising edge of the TX/RX clock and the inter-symbol boundary in the transmitted (received) serial data stream. In most PHYs, this latency can be different for each PLL/CDR lock cycle, but once the PHY is locked, the delays shall remain constant. PHYs whose TX/RX delays can change when the link is active are unsuitable for WR devices.

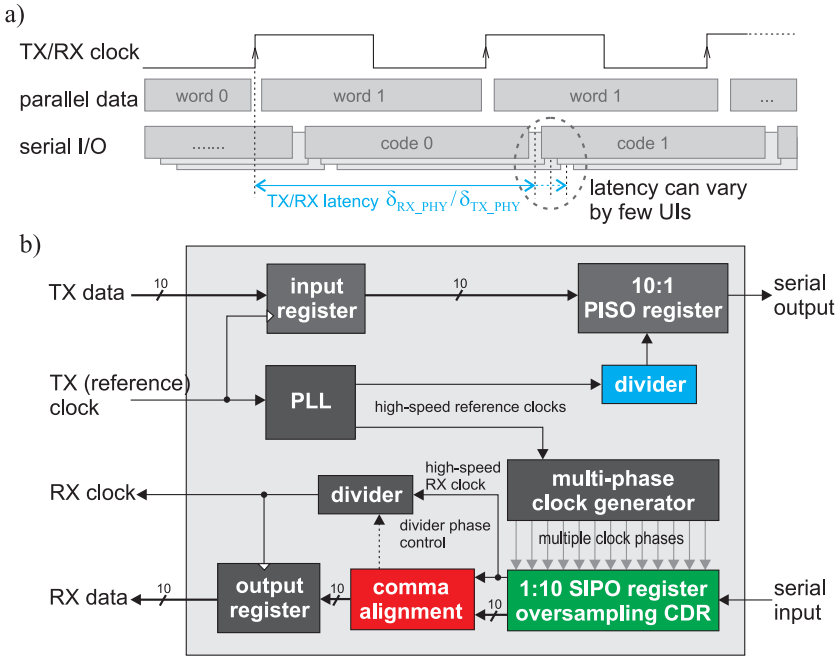


Figure 24: Random delays in gigabit SerDes devices (a) and blocks causing them (b).

In the PHYs which have been evaluated for usability in WR hardware (TLK1221 from Texas Instruments and Virtex-6/Spartan-6 GTP transceivers), the following random delays were identified:

- RX alignment latency, observed in PHYs which do not correct the RX clock phase when aligning to the inter-symbol boundary (red, observed for Xilinx GTP),
- RX latency resulting from the internal structure of the digital oversampling CDR (green, observed for TLK1221),
- TX latency caused by the clock divider between the internal PLL and the parallel-to-serial register (blue, observed for TLK1221).

Alignment latency can be measured every time the link goes up by disabling automatic comma alignment and bit-shifting the unaligned output data until a valid 8B10B code sequence is detected (*bitslip trick*, see [5]). Compensation of the latter two latencies, however, requires additional calibration logic. An example method (used in the WR switch) is shown in Figure 25. Such methods can by applied to the PHYs whose maximum ($\delta_{PHY(M/S)_{max}}$) and minimum ($\delta_{PHY(M/S)_{min}}$) delays are known (provided in the datasheet) and the delay's variation is

$$\delta_{\{TX,RX\}\_PHY(M/S)_{variable}} \in \langle 0 : \delta_{PHY(M/S)_{max}} - \delta_{PHY(M/S)_{min}} \rangle \tag{26}$$

e.g. below 10 bit times in the case of Gigabit Ethernet. Therefore, a fixed delay can be expressed as a sum of a constant value ($\delta_{\{TX,RX\}\_PHY(M/S)_{min}}$) and a variable part ($\delta_{\{TX,RX\}\_PHY(M/S)_{variable}}$) which needs to be measured:

$$\delta_{\{TX,RX\}\_PHY(M/S)} = \delta_{\{TX,RX\}\_PHY(M/S)_{min}} + \delta_{\{TX,RX\}\_PHY(M/S)_{variable}} \tag{27}$$

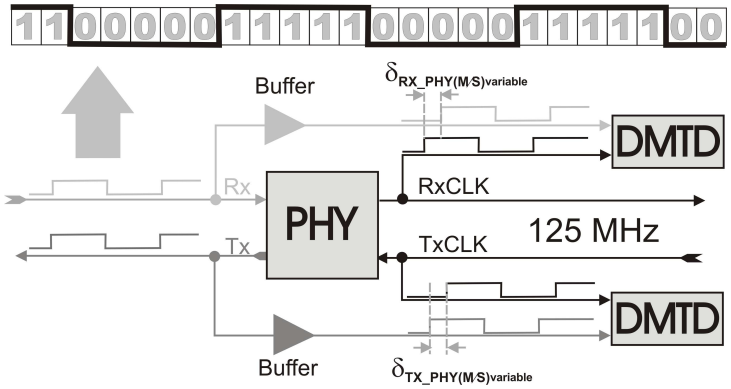


Figure 25: PHY latency measurement using calibration patterns.

The PHY transmit path is fed with a sequence of RD+ K28.5 characters (1111100000), effectively producing a 125 MHz square wave on the serial outputs. The repeated pattern of five ”0” and five ”1” is defined by the IEEE 802.3 standard [2] as *Low-frequency test pattern* (Appendix 36A.2). The phase shift between the TX clock and the output bitstream can be measured using a DDMTD phase detector, giving the value of the TX variable latency ($\delta_{TX\_PHY(M/S)_{variable}}$). The same method can be used to measure the RX variable latency ($\delta_{RX\_PHY(M/S)_{variable}}$) by commanding the link partner to send the calibration pattern. Note that since the K28.5 character contains a comma (5 consecutive ones), a burst of subsequent K28.5 symbols will cause improper operation of the PHY's comma alignment unit. Therefore, comma alignment must be disabled during the calibration process.

B.7 Establishing and maintaining synchronization

Having obtained the values of round-trip delay $delay_{MM}$ and link asymmetry, we can calculate the one-way master to slave delay $delay_{MS}$ by solving equation (28):

$$delay_{MM} = \Delta + \delta_{ms} + \delta_{sm} \tag{28}$$

where $\Delta = \Delta_{txm} + \Delta_{rxs} + \Delta_{txs} + \Delta_{rxm}$. Substituting δ_{sm} with 23 and solving for δ_{ms}, we obtain the one-way fiber delay 29:

$$\delta_{ms} = \frac{1+\alpha}{2+\alpha}(delay_{MM} - \Delta) \tag{29}$$

which after adding the circuit, SFP and PHY asymmetric delays present on the master to slave path gives the final master-slave delay 30 and offset 31:

$$delay_{MS} = \frac{1+\alpha}{2+\alpha}(delay_{MM} - \Delta) + \Delta_{txm} + \Delta_{rxs} \tag{30}$$

$$offset_{MS} = t_1 - t_{2p} - delay_{MS} \tag{31}$$

The value of $offset_{MS}$ is the input for the slave's offset adjustment algorithm which controls the slave's clock servo. The flow diagram of the algorithm is shown in fig. 26a and an example servo design can be found in fig. 26b. The algorithm assumes that the frequency has been

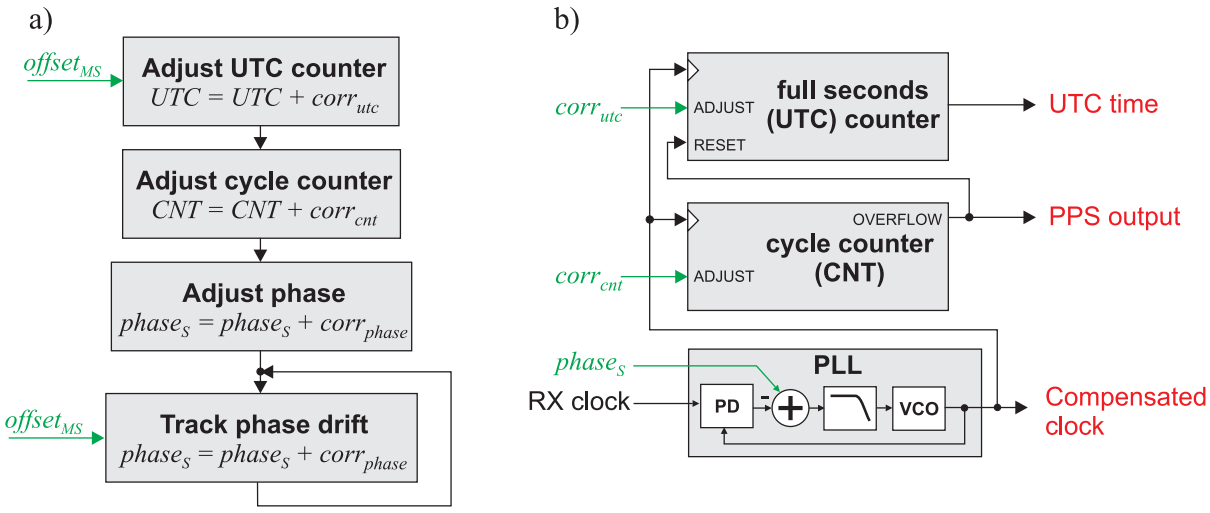


Figure 26: WR slave offset adjustment (a) and clock servo (b)

already syntonized by means of Sync-E and only the clock offset needs to be corrected. Offset correction is split into 3 steps:

1. **UTC time adjustment**: the UTC counter in the servo is increased (or decreased) by the number of full seconds 32 in $offset_{MS}$:

$$corr_{utc} = \lfloor \frac{offset_{MS}}{1\text{ s}} \rfloor \tag{32}$$

2. **Clock cycle counter adjustment**: The reference clock cycle counter, which produces the PPS signal is adjusted by the number of full T_{ref} (8 ns) cycles 33.

$$corr_{cnt} = \lfloor \frac{offset_{MS} - corr_{UTC}}{T_{ref}} \rfloor \tag{33}$$

3. **Phase adjustment**: The slave's phase shifter is adjusted with the remaining sub-cycle part of the offset 34

$$corr_{phase} = offset_{MS} - [offset_{MS}] \tag{34}$$

Voilà! Now the slave's clock and PPS signals are synchronized to the master with sub-nanosecond accuracy. Since the offset can vary with operating conditions, it is measured at regular intervals and the difference between subsequent measurements is added to the slave's phase shift to compensate for phase drift:

$$corr_{phase} = offset_{MS} - offset_{MS\_previous} \tag{35}$$

Because the phase drift is mainly caused by temperature variations, the rate of subsequent adjustments can be very low, even in the scale of a single adjustment per hour.

Note that the way $corr_{phase}$ is calculated requires the phase shifter to be able to change the phase relatively by any arbitrary value with no "jumps" in the signal when the value of $corr_{phase}$ crosses the inter-cycle boundary. For example, one can use a PLL with a phase detector capable of handling wrap-around phase transitions, but not a programmable delay line. An example design of such phase detector and PLL is described in section 4.3.7 of [4].

C Typical flow of WR Signaling message exchange

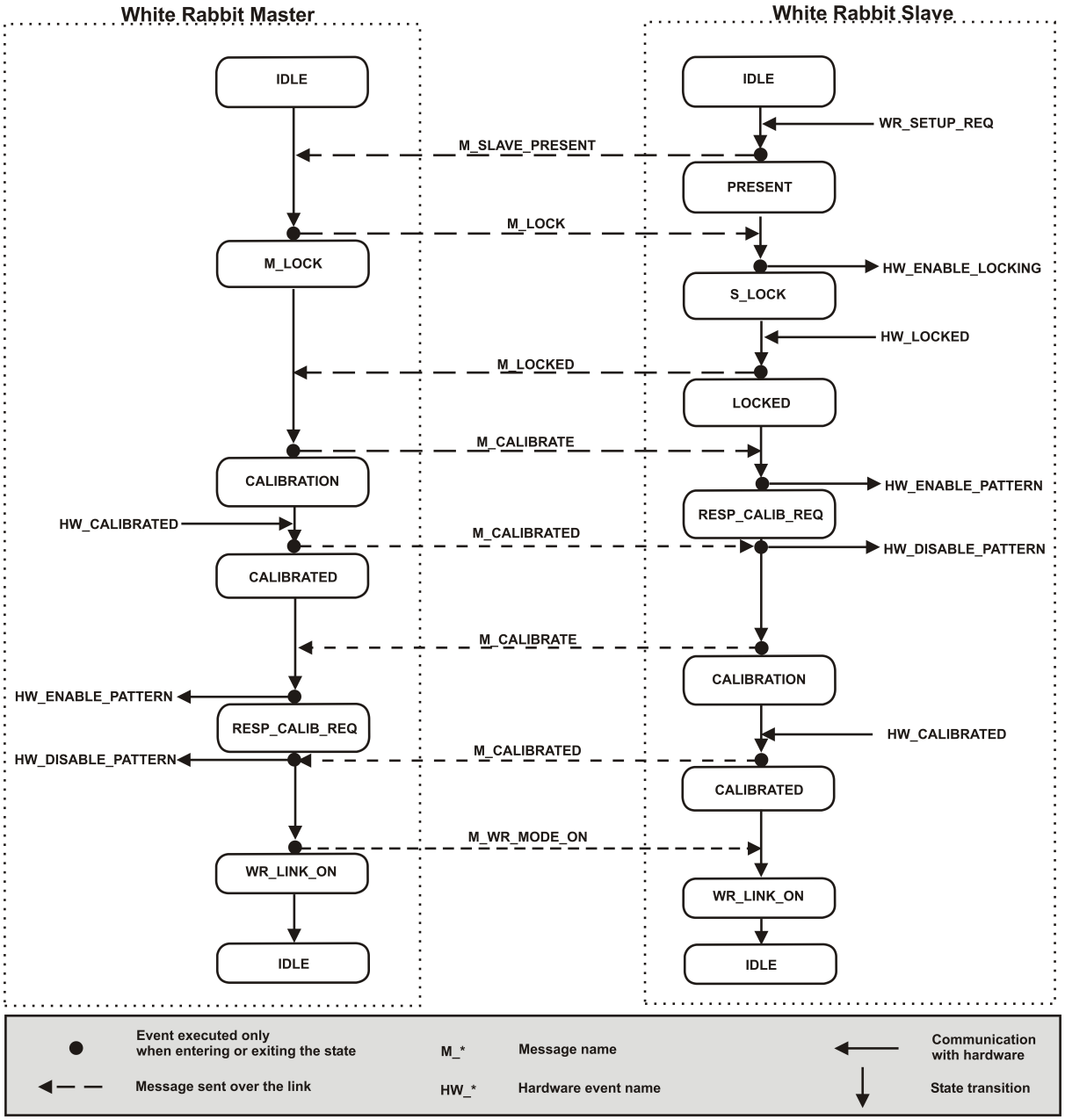


Figure 27: Typical flow of events (no exceptions) during WR Link Setup.

D PTP and WR FSMs from POWER ON use case

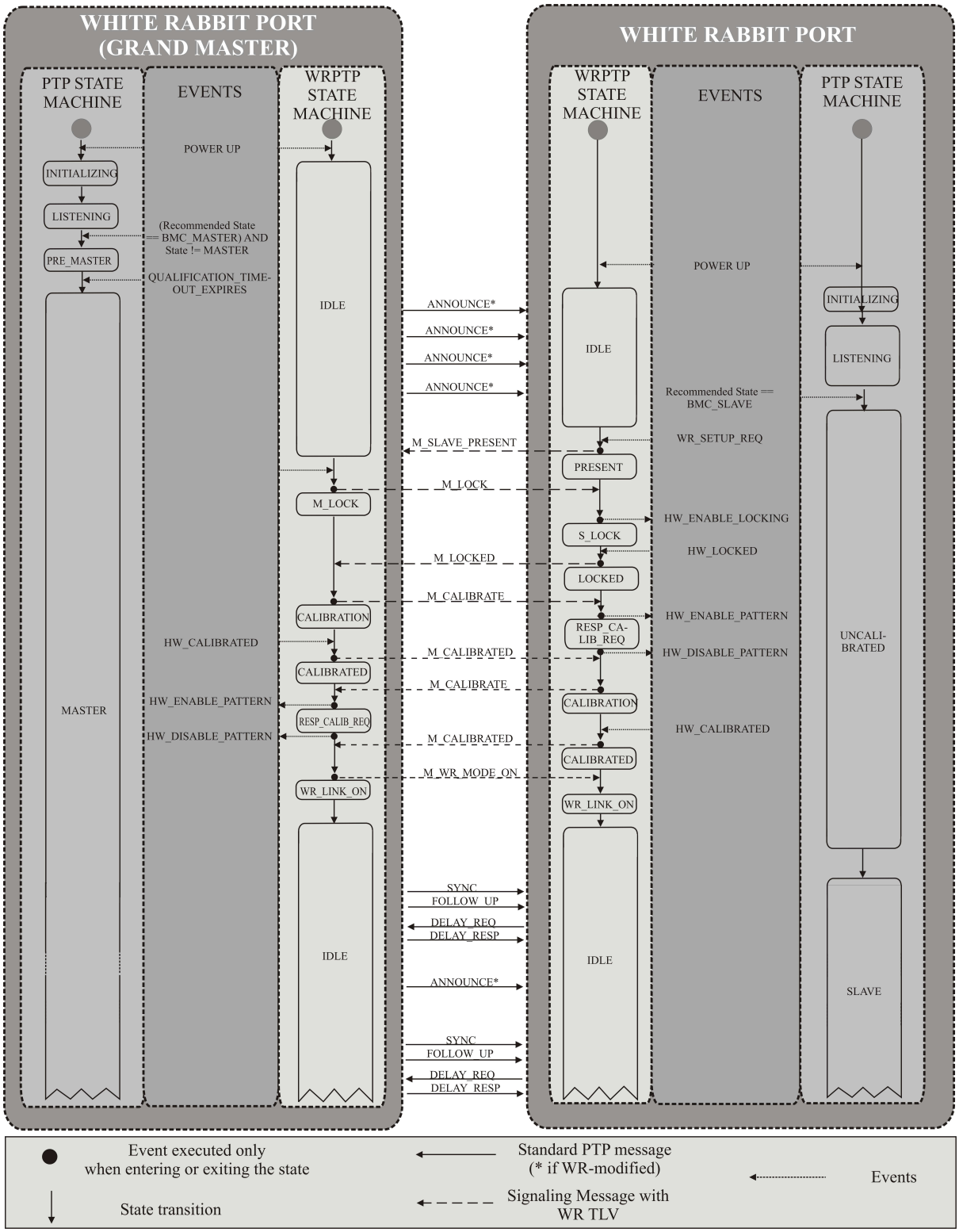


Figure 28: PTP and WR FSMs from POWER ON use case

E Primitive data types

Table 21: Primitive data types

Data type	Definition
UInteger8	8-bit unsigned integer
UInteger16	16-bit unsigned integer
UInteger32	32-bit unsigned integer
Integer64	64-bit integer
UInteger64	64-bit unsigned integer

F Computations using the two-step delay request-response mechanism with asymmetry correction

The method of incorporating asymmetry obtained using the WR Link Model by the WR Slave is WRPTP-implementation specific. Below, a summary of PTP offset ($< offsetFromMaster >$) and mean delay ($< meanDelayPath >$) calculations are presented. They are followed by two methods of including WR asymmetry, with the later being recommended. The method used might vary among WR Slaves.

F.1 Overview of PTP offset and mean path delay calculations

[Citing [1]]
The two-step delay request-response for a boundary or ordinary clock,:

1. Sync message transmission (section 9.5.9 of [1]):
   - The $< syncEventEgressTimestamp >$ (t_1) shall be generated upon transmission of the Sync message.
   - The originTimestamp field of the Sync message shall be 0.
   - The correctionField of the Sync message shall be set to 0.

$$Sync.originTimestamp = 0 \tag{36}$$
$$Sync.correctionField = 0 \tag{37}$$

2. Sync message reception (section 9.5.4 and 11.6.2 of [1]):
   - The t_2 shall be generated upon reception of the Sync message.
   - The sync message correctionField shall be adjusted for asymmetry by adding the value of the ingress path delayAsymmetry prior to its any use in a computation.

$$Sync.correctionField = delayAsymmetry \tag{38}$$

3. Follow_Up message transmission (section 9.5.10 of [1]):

   - The preciseOriginTimestamp field of the Follow_Up should be the $< syncEventEgressTimestamp >$ of the associated Sync message excluding any fractional nanoseconds of the associated Sync message.
   - The correctionField field of the Follow_Up shall be the fractional nanoseconds part of the $< syncEventEgressTimestamp >$ of the associated Sync message.

$$Follow\_Up.preciseOriginTimestamp = seconds\_and\_nanoseconds(t_1) \quad (39)$$
$$Follow\_Up.correctionField = fractional\_ns\_part(t_1) \quad (40)$$
$$< syncEventEgressTimestamp >= t_1 = Follow\_Up.preciseOriginTimestamp + Follow\_Up.correctionField \quad (41)$$

4. Delay_Req message transmission (section 11.6.3 and 11.3.2 of [1]):

   - Prior to transmission on an egress port the correctionField of the Delay_Req message shall be modified by subtracting the value of the egress path delayAsymmetry from its correctionField.
   - The t_3 timestamp shall be generated upon transmission of the Delay_Req message.

$$Delay\_Req.correctionField = -delayAsymmetry \quad (42)$$
$$Delay\_Req.originTimestamp = 0 \quad (43)$$

5. Delay_Req message reception (section 9.5.6 of [1]):

   - The $< delayReqEventIngressTimestamp >$ (t_4) shall be generated upon receipt of the Delay_Req message.

6. Delay_Resp message transmission (section 9.5.12 and 11.3.2 of [1]):

   - The receiveTimestamp field of the Delay_Resp message shall be set to the seconds and nanoseconds portion of the time t_4.
   - The correctionField of the Delay_Req message shall be set to 0 and then the correctionField of the Delay_Resp message shall be added to it and the fractional nanoseconds portion of t_4 shall be subtracted from the corrctionField of Delay_Resp message.

$$Delay\_Resp.correctionField = Delay\_Req.correctionField \quad (44)$$
$$- fractional\_ns\_part(t_4) \quad (45)$$
$$= -(delayAsymmetry + fractional\_ns\_part(t_4))$$
$$Delay\_Resp.receiveTimestamp = seconds\_and\_nanoseconds(t_4) \quad (46)$$

7. Delay_Resp message reception (section 11.3.2 of [1]):

- The $< meanPathDelay >$ shall be computed as follows

$$\begin{aligned} < meanPathDelay > = [&(t_2 - t_3) \\ &+ (Delay\_Resp.receiveTimestamp \\ &- Follow\_Up.preciseOeriginTrimestamp) \\ &- Sync.correctionField \\ &- Follow\_Up.correctionField \\ &- Delay\_Resp.correctionField]/2 \end{aligned} \tag{47}$$

8. Computation of the clock offset (section 11.2 of [1]):

$$\begin{aligned} < offsetFromMaster > = &\, t_2 \\ &- Follow\_Up.preciseOriginTimestamp \\ &- < meanPathDelay > \\ &- Sync.correctionField \\ &- Follow\_Up.correctionField \end{aligned} \tag{48}$$

F.2 WR asymmetry as PTP communication path asymmetry

The asymmetry obtained using the WR Link Model (*asymmetry*, section 4) can be seen as a communication path asymmetry *delayAsymmetry* as defined in 7.4.2 of [1]

$$delayAsymmetry = asymmetry \tag{49}$$

and incorporated into the PTP calculations as described in Appendix F.1, equations (38) and (42).

F.2.1 Solution for Ethernet over a Single-mode Optical Fiber

In the case of the solution for Ethernet over a single-mode Optic Fiber (section4.1), it can be calculated using directly equation (13), where μ is $< meanDelayPath >$ obtained using equation (47):

$$delayAsymmetry = \Delta_{tx_m} + \Delta_{rx_s} - \frac{\Delta - \alpha * < meanDelayPath > + \alpha * \Delta}{2 + \alpha} \tag{50}$$

It needs to be noted, that the *delayAsymmetry* used in calculation of $< meanDelayPath >$ and $< offsetFromMaster >$ is computed based on a previous measurement of the $< meanDelayPath >$. This is because the standard [1] seems to assume that, if available, the *delayAsymmetry* is constant and known in advance.

Therefore, a direct incorporation of the asymmetry calculated using the WR Link Model into $< meanDelayPath >$ and $< offsetFromMaster >$ is recommended.

F.3 Direct WR asymmetry incorporation into PTP computations

The asymmetry obtained using the WR Link Model (*asymmetry*, section 4) can be incorporated directly into the computation of $< meanDelayPath >$ and $< offsetFromMaster >$. In such case, the *delayAsymmetry* as defined in 7.4.2 of [1], is set to 0x0 (equations (38) and (42)). The $< meanDelayPath >$ value is calculated using (47) and corrected with *asymmetry* to obtain $< pathDelay_{MS} >$,

$$< pathDelay_{MS} >=< meanDelayPath > + asymmetry \tag{51}$$

which is used to compute $< offsetFromMaster >$ using (48):

$$\begin{aligned} < offsetFromMaster > &= t_2 \\ &\quad - Follow\_Up.preciseOriginTimestamp \\ &\quad - < pathDelay_{MS} > \\ &\quad - 0 \\ &\quad - Follow\_Up.correctionField \end{aligned} \tag{52}$$

$$= t_2 - t_1 - < pathDelay_{MS} > \tag{53}$$

F.3.1 Solution for Ethernet over a Single-mode Optical Fiber

A WR Slave, which assumed communication path asymmetry to be 0 ($delayAsymmetry = 0$) and measured timestamps t_2 and t_3, can assemble the timestamps received from the WR Master follows:

$$\begin{aligned} t_1 = &Follow\_Up.preciseOriginTimestamp \\ &+ Follow\_Up.correctionField \end{aligned} \tag{54}$$

and

$$\begin{aligned} t_4 = &Delay\_Resp.receiveTimestamp \\ &- Delay\_Resp.correctionField \end{aligned} \tag{55}$$

Having all timestamps (t_1, t_2, t_3, t_4), the fixed delays (Δ_{txm},Δ_{rxs}, Δ_{txs}, Δ_{rxm}) and relative delay coefficient (α), the WR Link Model can be combined with the PTP computation to obtain directly the correct values of $< delayPath_{MS} >$ and $< offsetFromMaster >$:

$$< delayPath_{MS} > = \frac{1+\alpha}{2+\alpha}[(t_4 - t_1) + (t_3 - t_2) - \Delta] + \Delta_{txm} + \Delta_{rxs} \tag{56}$$

$$< offsetFromMaster > = t_2 - t_1 - < delayPath_{MS} > \tag{57}$$

where

$$\Delta = \Delta_{txm} + \Delta_{rxs} + \Delta_{txs} + \Delta_{rxm} \tag{58}$$

References

[1] IEEE Std 1588-2008 *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. IEEE Instrumentation and Measurement Society, New York, 2008, http://ieee1588.nist.gov/.

[2] IEEE Std 802.3-2008 *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and Metropolitan area networks – Specific requirements*. IEEE Computer Society, New York, 2008.

[3] ITU-T G.8262/Y.1362 *Timing characteristics of a synchronous Ethernet equipment slave clock*. TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 07/2010.

[4] Tomasz Włostowski *Precise time and frequency transfer in a White Rabbit network*. Warsaw University of Technology, 05/2011.

[5] P.P.M. Jansweijer, H.Z. Peek, *Measuring propagation delay over a 1.25 Gbps bidirectional data link*. National Institute for Subatomic Physics, Amsterdam, 2010, http://www.nikhef.nl/pub/services/biblio/technicalreports/ETR2010-01.pdf.

[6] Takahide Murakami and Yukio Horiuchi, *A Master Redundancy Technique in IEEE 1588 Synchronization with a Link Congestion Estimation*. ISPCS Proccedings, 2010,

[7] D.B Sullivan, D.W. Allan, D.A. Howe, F.L. Walls, *Characterization of Clocks and Oscillators*. http://tf.nist.gov/general/pdf/868.pdf, NIST Technical Note, 1990.

[8] J. Serrano, P. Alvarez, M. Cattin, E. G. Cota, J. H. Lewis, P. Moreira, T. Włostowski and others, *The White Rabbit Project*. ICALEPCS TUC004, 2009.

[9] Bahaa E.A. Saleh and Malvin Carl Teich, *Fundamentals of Photonics*. Wiley-Interscience, 2007.