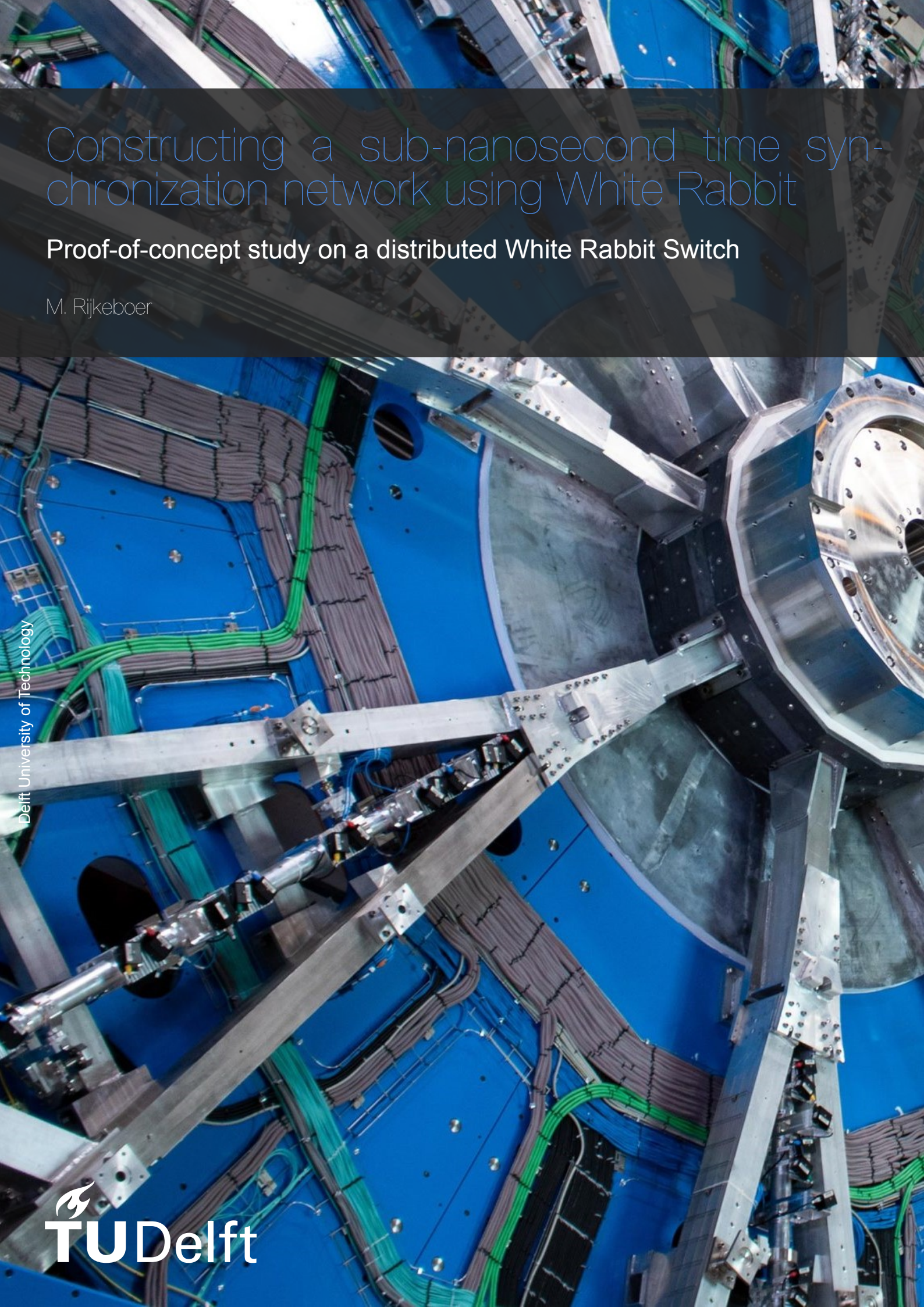
Constructing a sub-nanosecond time synchronization network using White Rabbit

Proof-of-concept study on a distributed White Rabbit Switch

M. Rijkeboer



Delft University of Technology

TU Delft

Constructing a sub-nanosecond time synchronization network using White Rabbit

Proof-of-concept study on a distributed White Rabbit Switch

by

M. Rijkeboer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday January 12, 2023 at 09:45 AM.

Student number: 4465237
Project duration: February 14, 2022 – November 30, 2022
Thesis committee:

Dr. ir. J.S.S.M. Wong,	TU Delft, supervisor
Dr. ir. N.P. van der Meijs,	TU Delft
Ir. D.O. van den Heuvel,	TOPIC Embedded Systems, supervisor

TU Delft

Contents

1

Introduction

State-of-the-art industrial and research systems consist of many sensors and sub-components. Time, or timing rather, is a key principle in control and data acquisition routines. An illustration of a large distant control- and sensor-network is the Large Hadron Collider (LHC) at CERN, which can be seen in Figure 1.1. This particle accelerator and collider resides in a 27-kilometre ring, packed with magnets and sensors [11]. High-precision timing is required to change polarity of the magnets and induce push and pull forces to accelerate particles. Each particle detector, temperature sensor or speed measurement device, requires ultra high timing precision in order to provide accurate reconstruction of the experimental data. To quote CERN: ”Collision of particles is similar to firing two tiny needles from 10 kilometres apart, expecting them to collide” [11]. One can imagine that slight inaccuracies in timing or measurement may quickly result in a miss.

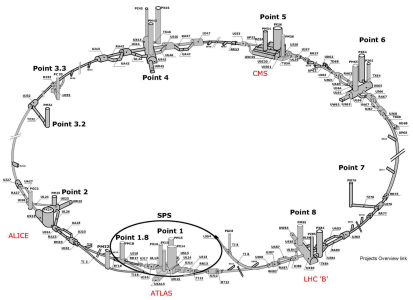


Figure 1.1: The chain of LHC collider and experiment points with sensors in a 27-kilometre ring at CERN. [45]

It is imperative to achieve a common shared notion of time throughout the LHC to provide accurate control signals. Hence, all clocks should tick at the same speed with identical offset at every instance. That is especially problematic when nodes are connected through miles and miles of cables, hubs and other delay-adding elements.

Custom-made solutions, such as in the GMT (General Machine Timing) network at CERN, are able to meet these sub-nanosecond requirements. Yet the custom-made solutions lack compatibility with other protocols, require extensive calibration procedures and function solely on dedicated time synchronization networks [15][49].

To overcome the aforementioned limitations, they developed the White Rabbit (WR) protocol: a new open-source project with enhanced accuracy, scalability and compatibility. Above all, White Rabbit runs on packet-based networks such as Ethernet, requiring no dedicated network infrastructure.

1.1. Function of a White Rabbit switch

To construct a packet-based network that supports White Rabbit, the present hardware has to be replaced with specialized White Rabbit devices, which use synchronization techniques to obtain an accurate and precise time reference at every point in the network. Each of these devices can be categorized as nodes (i.e., single-port) and switches (i.e., multi-port).

The function of the node devices can be simplified to dictating current time to adjacent nodes (master-mode) or receiving current time from adjacent nodes (slave-mode). White Rabbit switches are multi-port devices and are crucially important in the construction of network hierarchies. They maintain their input port in slave mode and accept current time from an adjacent node. The remaining ports are in master mode and distribute the acquired reference time to other downstream nodes and switches. Figure 1.2 illustrates the function of the switch in a White Rabbit network. The switch receives the current time, which is compensated for the transmission delay by White Rabbit. Subsequently, it outputs White Rabbit messages to the adjacent nodes, such that they also receive the current reference time. The switch has also compensated the current time for the transmission delay via White Rabbit.

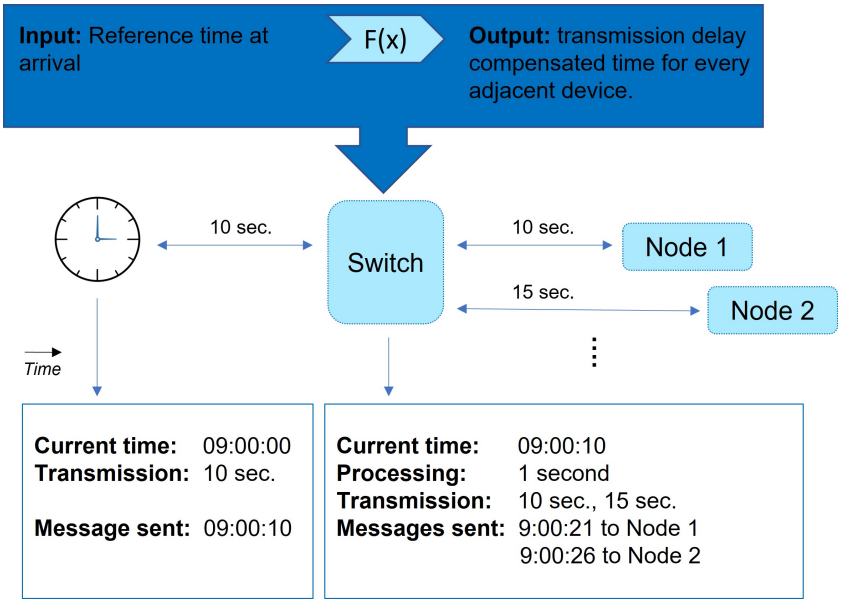


Figure 1.2: Schematic illustration of the function of a White Rabbit switch in a network. The switch receives a message containing the current reference time at arrival. Note that the master (represented by the clock image), compensated the time message for a transmission delay (10 sec.). The switch performs tasks represented by the F(x) notation. Lastly, the switch outputs messages to all adjacent nodes with similar transmission delay compensation, such that time information is valid upon arrival. The compensation combines internal processing time (1 sec.) and transmission time (10 and 15 sec.)

The following list epitomizes main tasks that are part of the function F(x):

- Acquisition of and adjustment to the reference time via the White Rabbit protocol.
- Compensation for internal transport delays and processing time.
- Distribution of the reference time to multiple downstream instances.
- Measurement of and compensation for the downstream transport delays (transceivers, optic cables etc.).

1.2. Motivation

Over the years, White Rabbit has been considered as a potential successor for other legacy timing networks: not only for particle accelerator systems, but also for distributed oscilloscopes, telescope arrays or metrology instruments (VSL, Delft) [32]. Even at smaller scales, timing networks are imperative for ultrasound or Light Detection and Ranging (Li-DAR) sensors on vehicles.

To encourage development, the White Rabbit project is open-source. Therefore, the designs for the nodes and switches are publicly available with different Field-Programmable Gate Array (FPGA) technologies [43]. In Figure 1.3, one can see an implementation of the open-source switch from Seven Solutions. Looking at the switch designs, a common characteristic among the switches stands out: each contains a single expensive FPGA (red) and a large set of Small Form-factor Pluggables (SFP) (yellow). The SFPs are controlled by transceivers in the FPGA and only expensive and advanced production lines contain enough transceivers (e.g., a Xilinx Virtex-6 FPGA in Figure 1.3). The size of the FPGA is also related to the fact that they replace regular Ethernet switches in a network and have to implement their features as well. However, the implementation lacks capabilities (e.g., user-friendly APIs, DDoS mitigation solutions) when compared to Ethernet switches from well-known brands (e.g., Cisco, Linksys, NETGEAR).

Figure 1.3: Implementation of the open-source switch from Seven Solutions. Accentuated in red, one can find the Virtex-6 FPGA. The yellow markings denote the 18 SFP-modules for the 18-port configuration of the switch [7].

To conclude, one observes that current White Rabbit switches are expensive, implement White Rabbit on FPGA platforms from ten years ago (Virtex-6), lack features and require entire replacement of present network infrastructure. Therefore, the embedded hardware company, Topic (Best, The Netherlands), wants to explore alternative concepts of a White Rabbit Switch to improve the state-of-the-art, to allow Ethernet switching from established brands, to mitigate the entire replacement of the present network infrastructure and to define a more cost-effective switch concept.

1.3. Research question

To explore alternative concepts of the White Rabbit switch, the process is guided by a research question that will be formulated as follows:

Can we define a distributed low-cost White Rabbit switch,
which requires minimal calibration and maintains sub-nanosecond synchronization accuracy?

To address each aspect of the research question, this work will formulate sub-questions to guide the research process.

1. **What is time synchronization?**

   This question will give a precise answer to the definition of time synchronization.

2. **What is White Rabbit and how does it work?**

   This question will provide an answer to the emergence, motivation and operation of White Rabbit.

3. **What are the limitations of White Rabbit in terms of design and implementation?**

   This question will initiate a study of the current open-source designs in terms of their specifications, components, implementation platforms and limitations

4. **What are the requirements for a newly defined White Rabbit Switch?**

   This question will initiate a translation of the limitations of current designs into improvements and requirements for the newly defined switch.

5. **Which functions have to be fulfilled and how to implement these?**

   The operation of the switch will be divided into smaller functions, after which there will be a search for possible implementations thereof.

6. **Does the definition fit the requirements?**

   This question will verify whether the newly defined switch fits the earlier proposed requirements

7. **How can the newly defined switch be improved?**

   This will trigger the search for improvements of the newly defined switch, such that subsequent works can continue research in those directions.

1.4. Outline

- **Chapter 2:** In the first chapter (’Background’), an overview on time references, time synchronization and the White Rabbit protocol is provided.
- **Chapter 3:** The design and implementation is thoroughly described in this specific chapter. Firstly, it discusses the current state of White Rabbit and its concurrent limitations, after which new design requirements and a distributed switch model is introduced. Lastly, it compares the design options and concludes with a proposed switch design.
- **Chapter 4;** This chapter verifies the newly introduced three-line asymmetry calculations from Chapter 3 and provides a cost comparison of the current and newly defined switch.
- **Chapter 5:** The concluding chapter summarizes the work, revisits the research question and concludes an answer to those questions. Lastly, recommendations are made regarding the continuation of the work.

2

Background

In the previous chapter, the White Rabbit switch and its function was introduced. Subsequently, the motivation was provided to define a new White Rabbit switch, followed by the research question and sub-questions.

This chapter investigates sources that provide the notion of time to digital systems (Section 2.1) and the process of time synchronization in a packet-based multi node network (Section 2.2). Subsequently, it explains recurrent terminology of Field Programmable Logic Arrays, regarding the relevant components and standards for communication of high performance signals and the variability of propagation delays (Section 2.3). Moreover, it motivates the complementary addition of two existing protocols into White Rabbit, whilst analyzing the operation (Section 2.4).

2.1. Time references

In data acquisition systems, such as the LHC, it is often necessary to register current timing information with a data sample to identify causal relationships between events. Therefore, a network requires a reference time to facilitate operation and protocols throughout.

2.1.1. Sources

The time references are often provided by atomic clocks. Atomic clocks count time based on different energy states of electrons in atoms by applying microwave radiation. Firstly, the electrons of a set of atoms is brought into a certain energy state. When this set is subjected to microwave radiation close to the atom oscillation frequency, some electrons will transition to other energy states. The closer the frequency to the oscillation frequency, the more electrons will transition. The oscillator for the microwave radiation is therefore tuned to the oscillation frequency of atoms by maximizing the amount of transitions. For a good reason, as the atom oscillation frequency is very stable and deterministic. A second, which is the SI unit of time, is defined by the multiplication of the oscillation period of Cesium and a numeric value. Thus, 9192631770 periods of the oscillation frequency of Cesium-133 is defined as 1 second [5][37]. It forms the basis for International Atomic Time (TAI), which composes current time in date, hours, minutes and seconds. The Coordinated Universal Time (UTC) is more widely used, as it is an advanced version of TAI that compensates for the variations in Earth's rotation period.

Albeit precise, Cesium clocks require expensive hardware to provide correct time. Therefore other types of atomic clocks exist, such as rubidium and hydrogen clocks. These are smaller and cheaper

than Cesium clocks, but require extensive re-calibration against Cesium clocks. Namely, their frequencies change over time due to internal and external influences, such as buffer gas and electromagnetic inferences [52].

Another standard to provide time is by using clocks based on the Global Positioning System (GPS). These clocks use the frequency of the carrier signals and code sequences received from the satellites to determine location and time. The satellites themselves will carry two Cesium clocks and two Rubidium clocks onboard to correctly keep track of time and to lock the transmitted carrier frequency [33]. In essence, GPS clocks derive their time and frequency from external atomic clocks. In high accuracy applications, a combination of Cesium and GPS clocks is found for added redundancy [32].

All previously-mentioned clock types have their output interface in common: Namely, a Pulse-Per-Second (PPS) signal, a 10 MHz reference clock and UTC time. The receiver of the information, will first receive UTC time after which it uses the PPS signal to update its local version of UTC time if necessary.

2.1.2. Offset and oscillation

It becomes clear from the previous paragraph, that correct notion of time is constructed from two variables: time offset and oscillation. To illustrate: a time signal (the offset, i.e. UTC) is received, and subsequently updated by a 1-PPS signal (the oscillation) to retrieve current time.

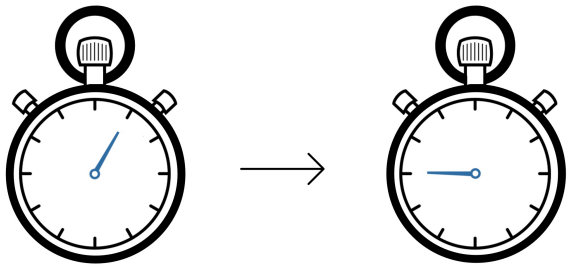


Figure 2.1: Schematic illustration of two concepts: time offset and oscillation. The same stopwatch at T_0 (left) and at T_{40} (right), whilst T_{offset} = 5 seconds and F_{osc} = 1 Hz

To illustrate, two stopwatches are introduced in Figure 2.1. The left stopwatch denotes seconds at T_0, whilst the right stopwatch denotes seconds at T_{40}, 40 seconds later. At T_0, the offset time T_{offset} is equal to 5 seconds. At T_{40}, the stopwatch shows correctly 45 seconds. The time of the stopwatch at time T_n can be mathematically described by the following equation:

$$T_n = n * F_{osc} + T_{offset} \tag{2.1}$$

In this case F_{osc} represents the update frequency of seconds on the stopwatch, whilst T_{offset} represent the initial offset time. The stopwatch counts ideally every increment of seconds. Therefore, in a well functioning stopwatch, F_{osc} should be equal to 1 Hz.

However, real-world oscillators have imperfections in phase and frequency. If F_{osc} changes only slightly to 0.99 Hz, the stopwatch will give inaccurate readings over time. Even Cesium clocks are not ideal and will oscillate with infinitely small imperfections. Even worse, most devices contain less stable oscillators than Cesium- or GPS-based clocks, due to size, material and cost constraints.

2.1.3. Crystal oscillators

Most digital devices contain internal crystal oscillators (abbreviated to XO). These oscillators put quartz crystals in resonation and are typically small, inexpensive and low in power usage [35]. However, they are also prone to errors induced by noise, ageing, temperature, voltage and other internal or external influences. Table 2.1 lists several types of crystal oscillators that contain features to reduce the influence from aforementioned external factors. Combinations of these features can also be found (i.e. Voltage Controlled Temperature Compensated Oscillator or VCTCXO).

Table 2.1: Several types of simple crystal oscillators with their respective abbreviation [35].

Oscillator	Acronym
Temperature compensated	TCXO
Voltage controlled	VCXO
Microprocessor controlled	MCXO
Oven controlled	OCX
Acceleration compensated	ACXO

As previously mentioned, crystal oscillators suffer from ageing and noise. Ageing is a long term variation due to induced stress or natural degradation of the materials used inside the oscillator. This may result in different oscillator frequencies (F_{osc}). Jitter on the other hand, is short term variation on the oscillation and may result in phase or frequency divergence.

2.1.4. Accuracy and precision

Drift and jitter lead to accuracy and precision loss. Both are used as quality indicators. If one refers to accuracy loss, one typically describes the average divergence from the desired value. To illustrate, if a crystal oscillator should resonate at 19 MHz, but on average resonates at 22 MHz, it is a 2 MHz accuracy loss. Figure 2.2 shows this example. If one refers to precision loss, one typically describes the deviation from its own average value. To illustrate, if one refers to a 1 MHz precision of the same 22 MHz oscillator, it indicates fluctuations of the frequency from 21 to 23 MHz. In conclusion, accurate and precise clock signals have a small average error compared to the desired value, whilst having small deviations throughout different measurements.

2.1.5. Summary

Considering previously mentioned imperfections in (crystal) oscillators, it appears non-trivial to let two digital devices present the exact same time. Our best approach would involve the attachment of Cesium clocks to all digital devices for the best accuracy and precision among the notions of time. However, attaching a clock that is at least the size of a briefcase, is rather impractical and too costly. Therefore, (crystal) oscillators are ubiquitously found among digital systems. To maintain a shared notion of time the offsets and drift/jitter from have to be compensated. Therefore, the following sections will introduce the concept of synchronization.

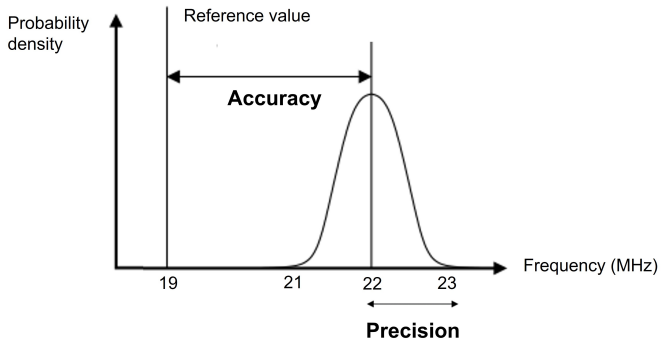


Figure 2.2: Graph visualizing the concept of accuracy and precision. The accuracy deviates 3 Mhz from a desired value, whilst the precision diverges maximally 1 MHz from the average value.

2.2. Synchronization

Synchronization is the process of tuning parameters of a secondary clock such that their presented time and update frequency are equal to a reference time and clock. Clock A represents a reference clock (i.e. Cesium clocks) presenting a correct reference time, with offset time $T_{A_{offset}}$ and frequency F_A. Clock B is a secondary clock that represents a device with incorrect offset time $T_{B_{offset}}$ and incorrect update frequency F_B. Both time references can be mathematically described by:

$$T_A = n * F_A + T_{A_{offset}} \tag{2.2}$$

$$T_B = n * F_B + T_{B_{offset}} \tag{2.3}$$

Herein, n represents the amount of sample time away from the initial measurement T_0. Equations 2.2 & 2.3 show that clock A will be synchronized to B when the time offsets are equal ($T_{B_{offset}}$ = $T_{A_{offset}}$) and the clocks are syntonized (F_B = F_A). Therefore, synchronization consists of two processes: time offset equalization and syntonization.

2.2.1. Time offset equalization

Time offset equalization is the process of assimilation of the offset time. Suppose Clock A could send its current offset time T_A to Clock B over a transmission line with one-way transmission delay δ. Then, Clock B will receive the value of T_A at time T_A + δ. The absolute correction of offset time T_B, can be expressed as:

$$\Delta = T_A - T_B + \delta \tag{2.4}$$

However, Clock B remains unaware of transmission delay δ and cannot adequately correct its offset time T_B. Only when Clock A and Clock B are synchronized, accurate one-way delay (OWD) can be estimated. This presents the exact problem that time offset equalization protocols try to solve by approximation.

2.2.2. Network Time Protocol

Network Time Protocol (NTP) is an Ethernet-based time offset protocol that shares reference offsets through a packet-based network. It was designed to provide time references to large groups of computers with Ethernet access. Therefore, a software-based solution was chosen to provide a wide platform support, without needing specialized hardware. Additionally, it requires no extra physical interconnection to time references, as it fully remains in the Ethernet domain.

NTP structures computers in hierarchical fashion. Figure 2.3 shows a schematic representation of such a network. Each level in the hierarchy is known as a stratum. The highest level devices are connected to atomic clocks and referred to as servers. Thereafter, the time reference is distributed through nodes, or clients, with increasing stratum levels 1 to maximally 16. The stratum number represents the distance to a reference time offset.

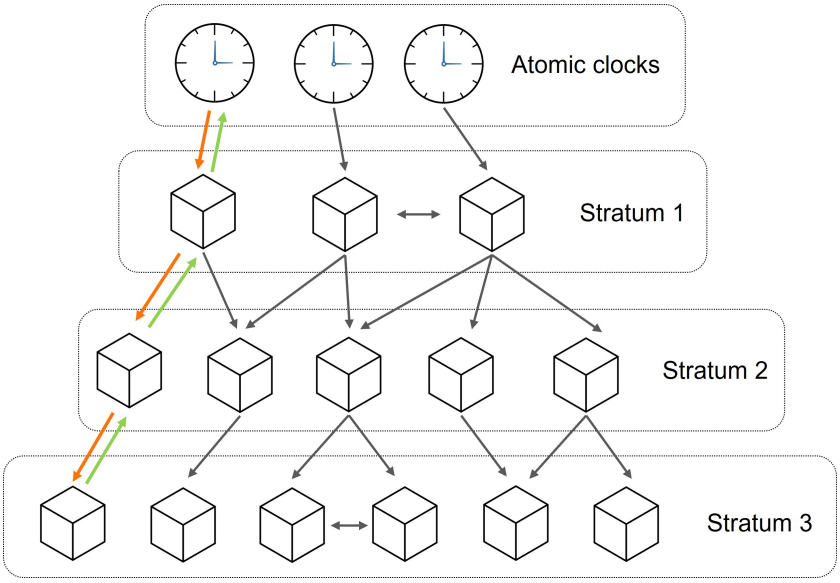


Figure 2.3: Hierarchical network of Ethernet nodes at different stratum levels. The stratum number represents the distance from the time reference source

Timestamp exchange

A client requires two sets of information to acquire a correct offset time: current time at a server and the transmission delay δ from the server to the client.

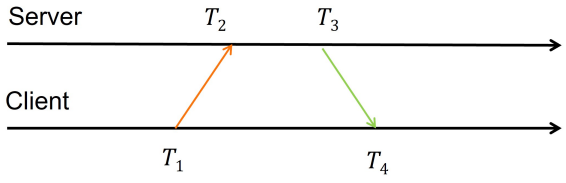


Figure 2.4: Symmetrical transmission delays from client to server and vice versa

The server-to-client delay is based upon the exchange of four timestamps. Figure 2.4 indicate them as T_1, T_2, T_3 and T_4. The registration and communication proceed in the following manner:

1. The client sends a request packet to the server at time T_1. The client registers time T_1 at which the request was sent via a software-based time-stamping unit.
2. The server receives the request at time T_2 and stores the timestamp.

3. The server modifies the request packet and sends it back to the client at time T_3
4. The client receives the modified request packet and stores the arrival time T_4.
5. Timestamps T_2 and T_3 are communicated to the client.

The client can calculate the round-trip time Δ of the request-packet as follows:

$$\Delta = (T_4 - T_1) - (T_3 - T_2) \tag{2.5}$$

NTP assumes that transmission delays are symmetric. Figure 2.4 shows equal transmission times from client to server and vice versa. Hence, the one-way server-to-client delay δ equals:

$$\delta = \frac{(T_4 - T_1) - (T_3 - T_2)}{2} \tag{2.6}$$

After determination of the one-way server-to-client delay, the server sends its local. The client can correct that offset at arrival by adding the one-way delay.

Performance and improvements

In most cases, transmission delays are asymmetric instead of symmetric [39]. To illustrate, Figure 2.5 shows the orange traversal path of packet from a client to a time reference server. Due to different routing decisions by the network nodes, the packet returns through the green path. Figure 2.6 shows transmission delays from the packet in time. Note that green path is longer than the orange one. Therefore, Equation 2.6 will not hold and induce substandard estimations. Moreover, network fluctuations in between NTP updates will contribute to accuracy loss. The following list contains elements that contribute to performance degradation combined with mitigation solutions:

- **Routing**

  Request packets are routed from client to server through a network of other nodes. Routing decisions from nodes may differ during a two-way packet exchange (Figure 2.5). Efforts have shown that back-tracing routing information can improve estimation of the server-to-client transmission delay [39].

- **Layers**

  Increased number of traversed stratum layers creates an accumulative increase of server-to-client delay and deteriorates the desired accuracy [17]. Therefore, cluster, combine and selection algorithms have been developed to allow NTP to select the closest time reference source from a bundle of atomic clocks [34]. Similarly, unstable reference sources can be excluded and replaced by the next best time reference source [28].

- **Congestion**

  Bandwidth and throughput limitations in the network influence traversal time for request packets. Therefore, efforts have been made to estimate congestion and queuing in nodes [20]. If a client sends two packets at a controlled difference interval, then the difference in receiving time can be presented as a metric for congestion.

- **Time-stamping**

  Software-based time-stamping introduces random and non-deterministic delays. Even if they are produced in the kernel space of the operating system, they still depend on current system workload. The overall delay of the time-stamping accuracy can be improved by size reduction of the NTP packet [17][27]. Thereby, the overall influence on the asymmetry is reduced, but the variability is not. One would assume that operating system load could be correlated to software-based delays. Based on this hypothesis, variability could be estimated. To the authors knowledge, no literature on this topic has been published. Other works have concluded that only hardware-based time-stamping will provide less total and more deterministic delay [28].

- **Syntonization**

  Oscillators in NTP nodes are not syntonized. In between NTP updates, local oscillators update time and consequentially depreciate accuracy when jitter or drift is present. Therefore, the original author of NTP showed a necessity for controlled and stable local oscillators in order to further improve accuracy [38].

- **Wide Area Network**

  NTP is designed to work in Wide Area Networks with many computers. Previously mentioned delay fluctuations of NTP packets over Ethernet, limit the accuracy of NTP to single-digit millisecond range. However, in Local Area Networks (LAN), delays are smaller and more deterministic leading to several tens of microseconds accuracy [28].

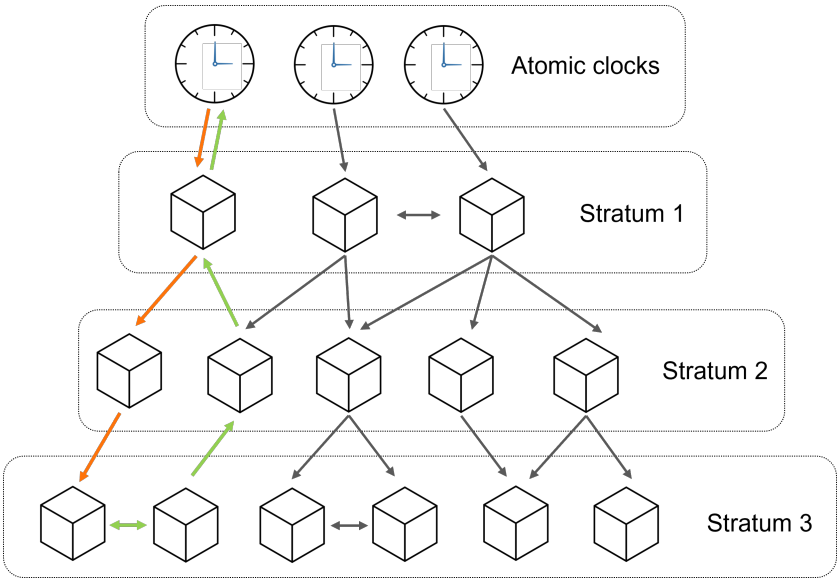


Figure 2.5: Hierarchical network of Ethernet nodes at different stratum levels. The stratum number represents the distance from the time reference source

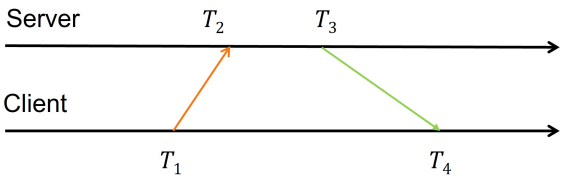


Figure 2.6: Asymmetrical transmission delays from client to server and vice versa

Summary

The accuracy of NTP is affected by the network type, asymmetric delays and local oscillator stability. When higher accuracy requirements are needed, specialized hardware implementations and setup calibration can lead to improvements. Nevertheless, the objective of NTP was to correct time offset in Ethernet networks with nodes from different platforms in a scalable and cost-effective manner. Further improving accuracy, defeats the objective of NTP.

2.2.3. Precision Time Protocol

The Precision Time Protocol (PTP) shifts the objective of NTP to Local Area Networks and improves the previously discussed limitations in NTP. Subsequently, single-digit microseconds to sub-microseconds accuracy can be maintained at the cost of needing more specialized hardware and configuration investments. On account of specialized hardware addition, PTP redefines names and functionalities in the network. These are defined as follows:

- **Master** (M)

  A master represents a node in the PTP network that can share its time offset with downstream nodes. The downstream nodes use the master as a time reference.

- **Slave** (S)

  A slave represents a node in the PTP network that has to adjust its time offset to an associated master clock.

- **Grand-master clock** (GM)

  Represent the highest level master node in a network. It provides reference time throughout the network via an atomic clock. It is therefore equivalent to a server in NTP.

- **Ordinary clock** (OC)

  An ordinary clock represents a lowest level node that has to adjust its time offset to an associated (grand-)master clock.

- **Boundary clock** (BC)

  A boundary clock, or switch, has an input and multiple output ports. The input port is configured as a slave, whilst the output ports are configured as masters. A boundary clock uses the slave port to adjust its time offset to an associated (grand-)master via PTP. Once adjusted, the output ports serve as masters for downstream nodes (e.g. slaves). Boundary clocks reduce the amount of layers necessary for request packets to traverse and improve distribution of the network. As a consequence, it improves routing and overcomes the limitation of an increased amount of traversed stratum layers in NTP, whilst reducing the amount of packets that grand-masters need to process.

- **Transparent clock** (TC)

  A transparent clock was later introduced in PTP version 2 (PTPv2) and greatly attributes to the accuracy in PTP [26]. These transparent clocks measure the variable residence time of request packets, whilst moving through the device. That time is registered in the packet and compensated by its slaves, as if they were transparent. This improves layer limitations compared to NTP, where cascading of multiple nodes throughout the path degraded accuracy exponentially. Similarly, variability in the time traversal of nodes can be measured, reducing the influence of congestion on one-way delay calculations.

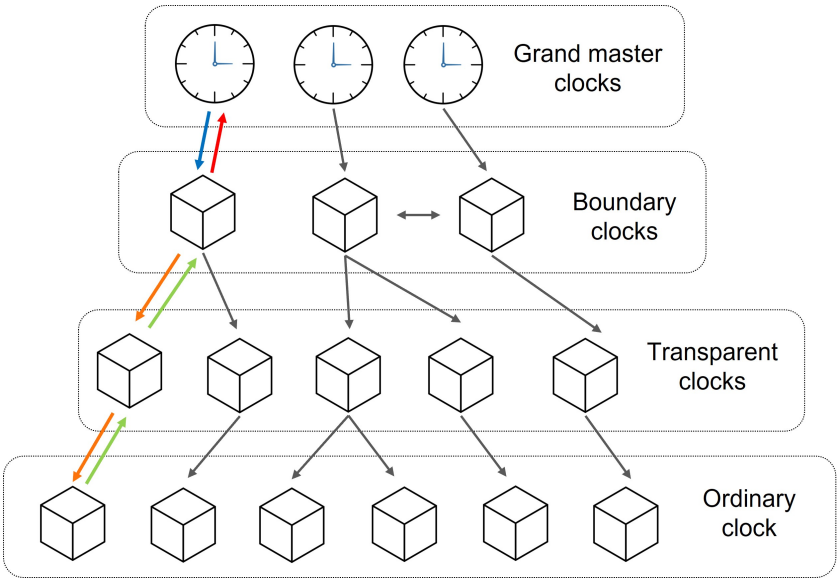


Figure 2.7: Hierarchical PTP network with specialized hardware to provide improved accuracy

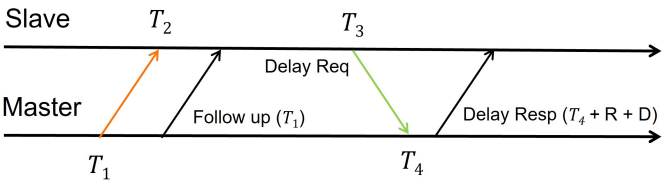


Figure 2.8: Timestamp exchange in PTP between master and slave for one-way transmission delay calculations (master-to-slave). The sum of variable residence times (R) in TCs is included in the communications, as is the asymmetric delay parameter (D)

Figure 2.7 shows the devices inside a PTP network. The time offset corrections proceed in two steps: from grand-master to boundary clock (slave) and from the boundary clock (master) to the ordinary clock (slave). Note the transparent clock in between the boundary clock (master) and slave.

Timestamp exchange

One significant difference between NTP and PTP is the source initiating the time offset equalization: In NTP, the client or slave initiates a request packet to correct its offset time, whereas in PTP the master alerts each of its slaves. Figure 2.8 shows the PTP timestamp exchange cycle, which is typically repeated every 1-2 seconds per slave [51]. Equations 2.8 and 2.7 are slightly modified versions of the equations from NTP. The correction of the sum of variable residence times (R) is now included. Similarly, if asymmetric delay properties of paths are known before timestamp exchange, an asymmetric delay parameter can be added [50].

$$\Delta = (T_4 - T_1) - (T_3 - T_2) - R - D \tag{2.7}$$

$$\delta = \frac{(T_4 - T_1) - (T_3 - T_2) - R - D}{2} \tag{2.8}$$

Performance and improvements

PTP improves the performance of NTP by adding specialized hardware. Namely, routing is improved by shortening the packets path to a time reference via boundary clocks. Transparent clocks correct the round-trip time for variability in the packet residence time (R) [41]. Measured delay parameters of certain paths (D) further improve the asymmetric limitations [50]. Local oscillators are sped up or tuned down by comparison of the time-offsets of the master and slave in a known interval [29]. Lastly, the time-stamping is now hardware-based instead of software based [29]. However, the following list contains elements that still contribute to performance degradation:

- **Routing**

  The addition of TC has reduced, but not annihilated, the influence of nodes in between master-slave links. Therefore, complete elimination of nodes in between master-slave connections would improve link asymmetry and routing problems.

- **Syntonization**

  PTP provides a correction of local oscillator frequency by comparison of the added time offsets from the master and the slave, as was suggested by the original author of NTP [38]. However clocks inside the network are still not shared and syntonized, which leaves room for improvement.

Summary

In short, PTP introduces specialized hardware in the form of transparent clocks and boundary clocks (switches) to reduce the amount of traversal layers and asymmetry in transmission paths. In combination with the addition of delay parameters, local oscillator control and hardware-based timestamping, accuracies up to sub-microseconds can be maintained. To further enhance accuracy, syntonization of local clocks and elimination of nodes in between master-slave connections, could be realized.

2.2.4. Syntonization

Syntonization is the process of frequency assimilation and refers to simultaneous updates of the notion of time. Devices can by syntonized by provision of equal oscillations to each device. Placing similar oscillators with each device may seem trivial. However, oscillators on digital devices drift and are therefore unequal. Devices could share their oscillators over transportation media, such as PCB traces, optic fibre cables or even wirelessly, at the cost of noise addition and degradation. If a clock A sends its oscillations, Clock B has to capture, measure and restore the received oscillation signal. Therefore, Phase-Locked loops (PLL) are often integrated to ordain its integrated oscillator to match the incoming frequency.

2.2.5. Phase-Locked Loop

A generic simplified Phase-Locked loop contains three main blocks: a phase-frequency-detector (PFD), a low-pass filter (LF), a Voltage Controlled Oscillator (VCO) and a frequency divider (FD). Figure 2.9 shows a block diagram of a generic PLL. The PFD generates a signal proportional to the phase difference between the incoming clock and feedback loop. A simple PFD can be realised by using the exclusive-or operation (XOR) that generates a difference signal between the input clock and feedback clock from the VCO. This can be seen in Figure 2.10. The low-pass filter generates a voltage signal from the square wave output of the PDF. If the bandwidth of the filter is too low, the ripple on the output voltage will be small and the voltage will change only slightly upon receiving a different square wave input from the PFD. The PLL will not be able to react to changes of the input signal. If the bandwidth is too high, the ripple will make the signal unstable. The low-pass change in output voltage will signal the

VCO to slow down or ramp up the VCO frequency. Subsequently, the VCO frequency goes through a frequency divider back to the PFD.

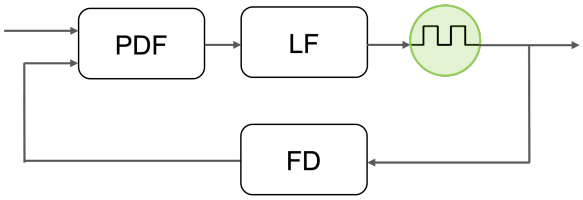


Figure 2.9: Block diagram of a generic PLL with four main building blocks: PFD, LF, VCO and FD

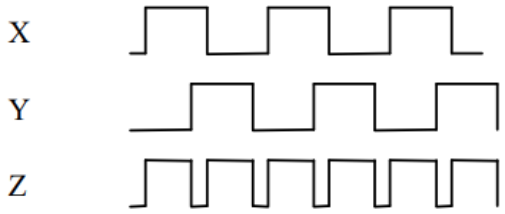


Figure 2.10: Phase-frequency-detector with exclusive-or operation on inputs X and Y, resulting in a pulse-width-modulated signal proportional to their phase difference. As the frequency of X and Y are equal, no change in the low-pass filter voltage will occur. Therefore, the VCO frequency remains unchanged.

2.2.6. Synchronous Ethernet

In regular Ethernet networks, each node contains a free-running oscillator for transmission of data packets. Synchronous Ethernet (SyncE) specifies additional capabilities to transfer and syntonize oscillation signals on the lowest physical layer of Ethernet (L1) via data carrier frequency transmission [24]. The source of the oscillation signals can be traced back to GPS or Cesium-based clocks.

Phased-Locked-Loops (PLL) are integrated to recover and lock the carrier frequency from received data signals. Nodes in a network receive this carrier frequency from GPS or Cesium-based reference clocks that inject the oscillations into the hierarchical network. Hence, the oscillations are equal in frequency, but not in phase as time offsets are not guaranteed. One can conclude that by the specified definitions, SyncE provides syntonization, rather than synchronization.

Figure 2.11 and 2.12 illustrate the difference between regular Ethernet and synchronous Ethernet. Both slave nodes N1 and N2 receive data from the master node with a carrier frequency injected from the reference clock. Slave nodes in regular Ethernet, send data back at their own free-running oscillator frequency. However, slave nodes in SyncE extract the reference oscillation frequency from the received data and use it for transmission back to the master node.

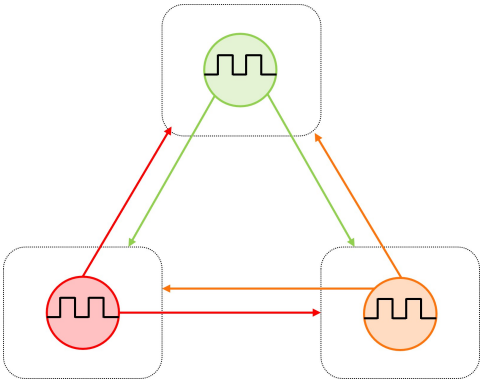


Figure 2.11: Different clock domains in regular Ethernet between nodes

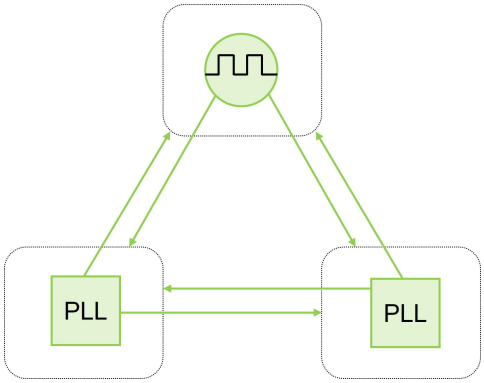


Figure 2.12: Recovered and restored clocks with PLL's between nodes

2.2.7. Summary

To provide equal notions of time, offsets and oscillations can be shared among the devices, better known as the process of synchronization. In packet-based networks, such as Ethernet, distribution protocols between devices, such as NTP and PTP, provide time offset equalization. It can be regarded that time offset equalization protocols share many features but present different objectives and trade-off decisions. NTP was designed for time offset distribution in large scale computer networks and values cost and flexibility constraints at millisecond to tens of microseconds time accuracy. In LAN networks, the accuracy improves to tens of microseconds. PTP aims for high-accuracy implementations in LAN networks by applying solutions to accuracy limitations in NTP, at the cost of hardware and configuration investments. It results in single-digit microsecond to sub-microsecond accuracy. Revisiting the requirements of this work, neither PTP nor NTP will adhere to sub-nanosecond accuracy on their own. However, it was also concluded that further improvement of accuracy could be found along the lines of clock syntonization and elimination of nodes in between the master-slave paths. Clock syntonization in packet-based networks could be realized through synchronous Ethernet (SyncE). PTP and NTP could therefore greatly benefit from the addition of SyncE to comply with the synchronization process. The section ”White Rabbit” will further discuss this option, to demonstrate that the accuracy can be stretched to the objective of this work by means of White Rabbit. The next section will first discuss Field Programmable Logic Arrays, which provide the platform for White Rabbit instances.

2.3. Field Programmable Logic Arrays

This section explains key characteristics of Field Programmable Logic Arrays (FPGAs). Among others, delay variability between devices is discussed, as it influences per device propagation delays. Secondly, it highlights signaling types and transmission options on the FPGA that could provide intercommunication with external devices such as sensors, micro-controllers and other FPGAs. Moreover, clock capable pins and transmission delay differences through optic links are touched upon.

2.3.1. Delay variability

Delay variability is generally described as the propagation delay deviation of a component compared to the expected delay value. This could be due to process variation during fabrication, degradation over time, temperature variations and difference of the supplied voltage [59]. Propagation delays are preferably minimized. Therefore, FPGA quality is graded based on the slowest hardware component at the initial test of the device after production. Throughout the course of its lifetime, the device is exposed to various other operating conditions and hardware loads. The speed grade of the FPGA is a poor indication of the actual physical delay of the components during initial measurements at the testing facilities. More importantly, it is a measure for the slowest element in the device, rather than an indication of the speed of the remaining pieces. There will be delay variability within a FPGA, which causes propagation delay differences between presumably identical components. Thus, they do not only suffer from inter-die delay differences, but also intra-die variations.

2.3.2. Intercommunication

FPGA’s contain several components that provide inter-communication to external devices, which can be selected based on speed, direction, voltage and throughput requirements. The terminology used in the upcoming subsections is frequently used in this work and are therefore briefly explained.

Buffers

Data signals move from internal paths to pins via Input/Output buffers. Their functions can be classified into either one-way receiving (input), one-way transmitting (output) and bidirectional (in-out) buffer [3]. The latter can be continuously reconfigured to either receive, transmit or power off via tri-state buffers.

Clock signals move through different regions in the FPGA via clock buffers (e.g., BUFG, BUFH, BUFR, BUFIO [3]). They reach separate lines in the FPGA to minimize jitter and skew behaviour on the clock signals.

Single-ended signaling

Traditional digital logic uses single-ended Input/Output buffers. Conventionally, these require a single link between two circuits to provide one-way communication whilst a common ground is assumed. A signal is transceived at a specified voltage, after which it is received and compared against a reference voltage at the other component. Thresholds determine if a digital high or digital low was received. Xilinx FPGA's handle several types of Single-ended standards (e.g., LVCMOS, LVTTL, HSTL etc. [3]) to comply with different connected devices. They can be broadly categorized into two groups: standards with fixed input thresholds and variable input thresholds. Variable threshold types can shift the threshold from ground (standard) to different values to remove dependence on the common ground, to lower the voltage swing and to allow control over the voltage levels. Primarily, the lower swing allows for higher frequency signals on the link [54].

Differential signaling

If high speeds or high applications are considered, differential signaling is employed. Namely, this standard trades off performance at the cost of two communication lines instead of one. Digital signal assertion is determined based on the voltage difference between the two lines. Similar to single-ended standards, the reference voltage differs per standard (e.g., LVDS, differential HSTL and SSTL [3]). If a communication line contains a higher voltage compared to the complement signal line, the signal is asserted as digital high. This way, differential signaling is less susceptible to noise, temperature and voltage biases.

Transceivers

Transceivers are dedicated devices embedded in FPGAs to facilitate high speed serial data communication with external components. High throughput can be realized at the cost of increased power consumption. Most FPGA's contain a limited amount of transceivers per chip.

Input/Output blocks

Input Output blocks (IOB) are elements on the FPGA through which communication with outside devices can be facilitated. Compared to their high speed counterparts, the transceivers, they provide more flexibility regarding protocols and standards, operate at lower speeds and are present in larger numbers.

Clocking Pins

Transceiver and IOBs connect through buffers to physical output pins. Some of these pins are Multiple Region Clock Capable (MRCC) and Single Region Clock Capable (SRCC). These posses minimal, deterministic and high quality routings to clock lines via buffers (e.g., BUFG [1]). Therefore, clock skew (to external entities) and jitter are minimized.

Small Form-factor Pluggable

Small Form Factor Pluggable (SFP) is a fiber optic module that typically connects to Ethernet-based devices via an SFP socket. It provides conversion from electrical signals to optical waves that travel through optical cables. This transmission type provides high speed bidirectional communication using light. The transmission wavelength differs from the receiving wavelength, such that data can be separated. In White Rabbit, 1310nm and 1490nm wavelengths are employed to provide bidirectional data transmission. Due to the difference in wavelenghts, a phenomenon called chromatic dispersion (i.e., interaction of light with electrons in the medium) causes velocity differences [44]. Whilst both wavelengths traverse the same optic cables, transmission delay between the two will be asymmetric. This means that transmission delays differ between sending data from a node A to a node B, compared to transmission in the opposite direction.

2.3.3. Summary

To summarize, FPGA's are susceptible to inter-die and intra-die propagation delay differences, which cause inequality between supposedly identical devices. Secondly, to transfer high performance signals, differential signaling standards should be adopted. Serialized signaling should be considered when high-throughput data is required, but is not beneficial for the quality of lower speed high accuracy signals. Intercommunication signals between FPGA's that are used as internal clocking signals, shall enter the chip via Multi-region Clock-capable pins and Single-region Clock-capable pins, as these provide minimal and deterministic input delay paths to clock lines inside the FPGA. To output clock signals, regular I/O pins can be adopted at the cost of increased nondeterministic I/O delay. Lastly, bidirectional optical transmission via SFP modules suffers from chromatic dispersion, causing delay differences between transmission in one direction compared to transmission in opposite direction.

2.4. White Rabbit

The White Rabbit project is a hardware development project of the open-source White Rabbit protocol. Initiated by CERN, the project is primarily maintained by academic laboratories and research institutions to find a combined solution for high-accuracy synchronization of time and high-speed transfer of data. Therefore, the following objectives can be formulated:

1. Synchronization of nodes at sub-nanosecond accuracy compared to reference time [41].
2. Establishment of a deterministic, synchronous and scalable network without the need for sophisticated configuration, maintenance or calibration procedures [15].
3. Provision of deterministic routing and robust delivery of packets [15].
4. In accordance with an open-source paradigm for hardware and software development [48].

To fulfill the requirements of synchronization, the resulting protocol comprises techniques mentioned in the previous chapter:

- **Synchronous Ethernet**

  Provides syntonization at the physical layer by transferring the frequency over data links.
- **Precision Time Protocol**

  Provides time offset equalization and improved link asymmetry compensation.

However, White Rabbit does not solely add two existing protocols: It adds phase compensation through a clock feedback loop to improve time-stamping and improves link asymmetry by complete removal of nodes in the path (such as TCs). Moreover, it adds a parameter based on fiber propagation asymmetries [39]. Figure 2.13 shows the new network situation in a White Rabbit network. Note the removal of transparent clocks compared to PTP, leaving the ordinary clocks (slave), the grand-master(master) and boundary clocks (slave-master).

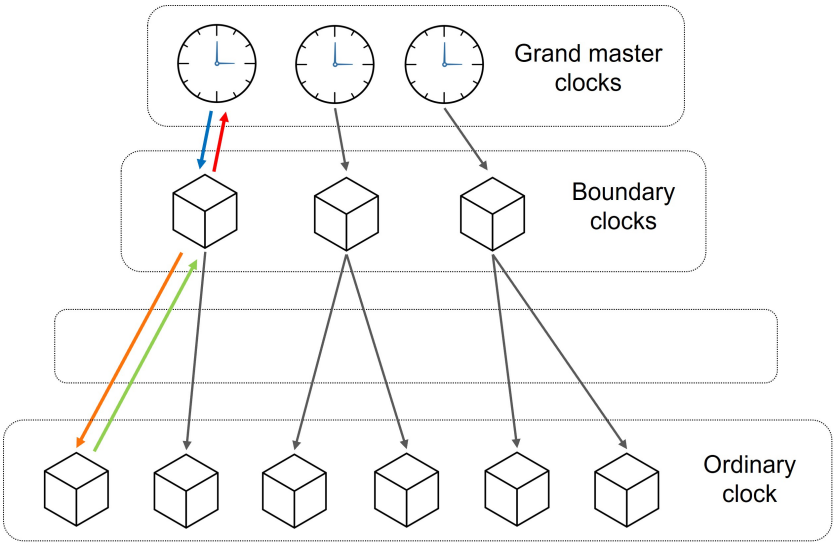


Figure 2.13: Hierarchical network of White Rabbit nodes, such as grand-master clocks, boundary clocks and ordinary clocks. Transparent clocks are no longer present in WR networks

To follow the steps in the protocol, a single master-slave connection (e.g. boundary to ordinary clock) is considered and depicted in Figure 2.14. It shows two entities, a master and slave, connected via an optical link without nodes in between. The bidirectional transmission connection via the optical link is represented by two one-way arrows. After all, transmission delays may differ according to Section 2.3. The two components inside the master and slave represent a phase detector and phase shifter respectively. Together, these add the phase compensation capability of White Rabbit. The figure also shows two transceivers and two receivers to facilitate the communication.

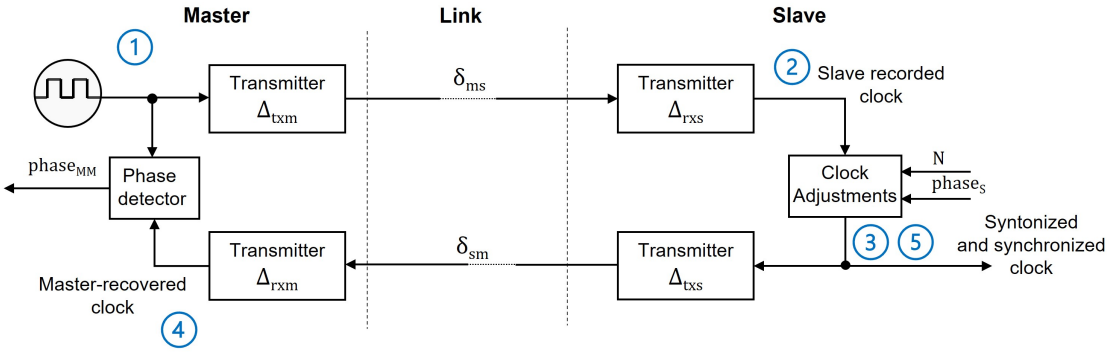


Figure 2.14: Schematic of the connections between master and slave over optic fibre (Adapted from Wlostowski et al. [57]). Variable delay sources are denoted by δ, whereas fixed delay values from calibration are denoted by Δ

The goal is to find an estimate of the adjusted offset time for the slave ($T_{ms_{offset}}$ based on the one-way transmission delays from slave to master ($delay_{sm}$) and master-to-slave delay ($delay_{ms}$). The latter two can be expressed as follows:

$$delay_{ms} = \Delta_{txm} + \delta_{ms} + \Delta_{rxs} \quad delay_{sm} = \Delta_{txs} + \delta_{sm} + \Delta_{rxm} \tag{2.9}$$

Herein, Δ_{txm} respresents the calibrated transmission circuit (tx) delay in the master (m), whereas the other subscripts present the receiver circuit (rx) and the slave (s).

2.4.1. Syntonization

According to Section 2.2, one can only relate current time with offset time, if the clock frequencies are equal. Hence, syntonization is firstly realised through synchronous Ethernet at 125 MHz clock oscillations. The reference clock (1 in Figure 2.14) is used to encode the data transmitter on the master side. By means of a PLL, the clock is then extracted from that data at the slave receiver as the slave recovered clock (2). Note that the clock signals are not in-phase and delayed by the one-way delay from master to slave. The phase of the retrieved clock can be adjusted by the phase shifter with the value $phase_s$ to represent the phase-shifted clock.

2.4.2. Coarse delay measurement

After syntonization of both nodes, the PTP time measurement is executed. Figure 2.15 shows the transmission delay from both connections. Through the exchange of PTP timestamps, the slave possesses the timestamps. Recall Equation 2.7 for the round-trip time from PTP measurements. Both factors R and D are not relevant, since there is no presence of nodes in between the link path. Therefore the round-trip time simplifies to:

$$delay_{mm} = (T_4 - T_1) - (T_3 - T_2) \tag{2.10}$$

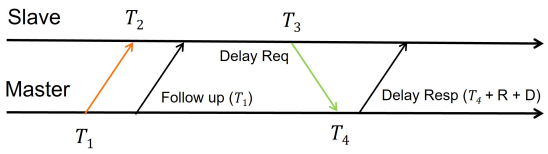


Figure 2.15: Symmetrical transmission delays from client to server and vice versa

2.4.3. Fine delay measurement

To perform measurements on the clock signal, the phase-shifted clock on the slave side is again embedded in the transceiver signal of the slave. The master then retrieves the master recovered clock(4). At arrival, the phase between the reference clock (1) and master recovered clock (4) is measured ($phase_{mm}$) and can be expressed as:

$$phase_{mm} = (\Delta + \delta_{ms} + \delta_{sm} + phase_s) \; mod \; T_{ref} \tag{2.11}$$

Herein, Δ represents the combined circuit delay from both receivers and transmitters, whereas $T_{ref} = 1/F_{ref}$. $phase_{mm}$ is typically referred to as the fine delay measurement, as it only represents the phase difference between the clocks. It cannot describe the amount of clock cycles that have passed during the round-trip time of the clock. This is expressed through the coarse delay measurements via PTP.

The increased accuracy of White Rabbit compared to PTP, lies in the use of the fine delay difference that was measured ($phase_{mm}$). Both PTP and White Rabbit timestamp incoming packets at 125 MHz. These timestamps are retrieved throughout different clock domains with fine delay differences, leading to inaccuracy in the timestamp calculations. White Rabbit compensates the fine delay differences by alteration of the timestamps.

In the authors opinion, White Rabbit documentation is rather overly-complex, erroneous and not comprehensible regarding the alteration of the timestamps. Moreover, related works seem to either restate or replicate the same exact explanation, or assume the protocol as common knowledge without proof of its functionality. A few of the main contributors to the project noted that improving the timestamps is not at all a trivial process [14]. One reckons that it is more illustrative to work out the understanding through equations and illustrations, rather than restating previous explanations.

Counters with clock domain synchronization

Firstly, it is important to understand the timestamp synchronizer-counters in both the slave and master. Figure 2.16 shows a block diagram of the counter. It contains two sets of three flip-flops, to align the incoming edges from the asynchronous trigger with the reference clock/compensated clock. Then two counters count from 0 to 124,999,999, each on a different clock transition of the reference clock. Upon receiving the asynchronous trigger, which is synchronized after the three flip-flops, the timestamps are saved into registers. Normally, the rising edge timestamp is used. However, if the phase between the reference clock and the phase-shifted clock is almost zero, the falling edge timestamp could be preferred due to the instability of the rising edge timestamps or vice-versa. The choice of a valid timestamp is dictated by a device specific parameter ϕ_{trans}, which has to be calibrated during boot or factory calibration [57].

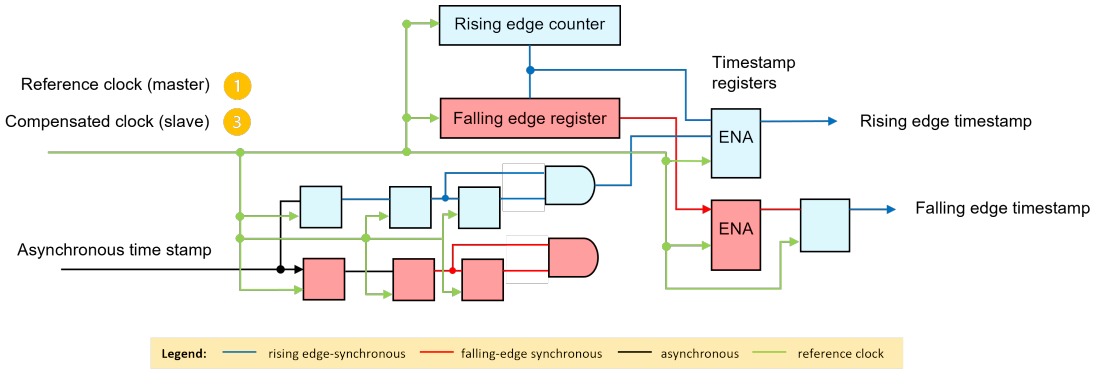


Figure 2.16: Time counter with synchronization of asynchronous trigger pulse (Adapted from Wlostowski et al. [57]).

Improved timestamps

To understand the correction that is applied to the timestamps, one should revisit Figure 2.14. In the master, only the reference clock (1) is used to produce timestamps, as it is synchronous to the system clock. Similarly, the phase-shifted clock (3) produces timestamps at the slave side. Both clocks also receive timestamp triggers from signals that are asynchronous to their own clock domain: Namely, the slave recovered clock (2) and the master recovered clock. These explain the use of synchronization flip-flops in the previously discussed timestamp counter. Figure 2.17 shows an example of a single time-stamping round between master and slave to clarify the inaccuracies in timestamps. The Figure 2.17 is accompanied by the textual explanation below.

Assume a PTP packet is transmitted from the master at 0 ns in the simulation waveform. Timestamp T1 is set to the value of $timestamp\_m\_rising$, namely $T1 = 0 * T_{ref}$. The start of the packet can be defined through a 8 ns pulse. Suppose that $delay_{ms}$ is equal to an arbitrary value, namely 17.4 ns. Therefore, the pulse will arrive at the slave at simulation time 17.4 ns. It is in-phase with the slave recovered clock, as the syntonization and transmission of the reference clock similarly experience $delay_{ms}$. Subsequently, the pulse moves into the timestamp counter. For simplicity and comprehensibility sake, the three cycles that are normally required for synchronization and thereafter compensated, are assumed to be zero. Therefore, the pulse is immediately synchronized at arrival to the clock domain of the counter.

The counter, counting at the rising edge of phase-shifted clock (3), notices the synchronized packet pulse and registers the timestamp $T2 = 18 * T_{ref}$ according to $timestamp\_s\_rising$. However, the packet arrived earlier at 17.4 ns, instead of 19.9 ns. This is due to the fact that the slave recovered clock and the phase-shifted clock are out of phase, due to an arbitrary value $phase_s = 2.5ns$.

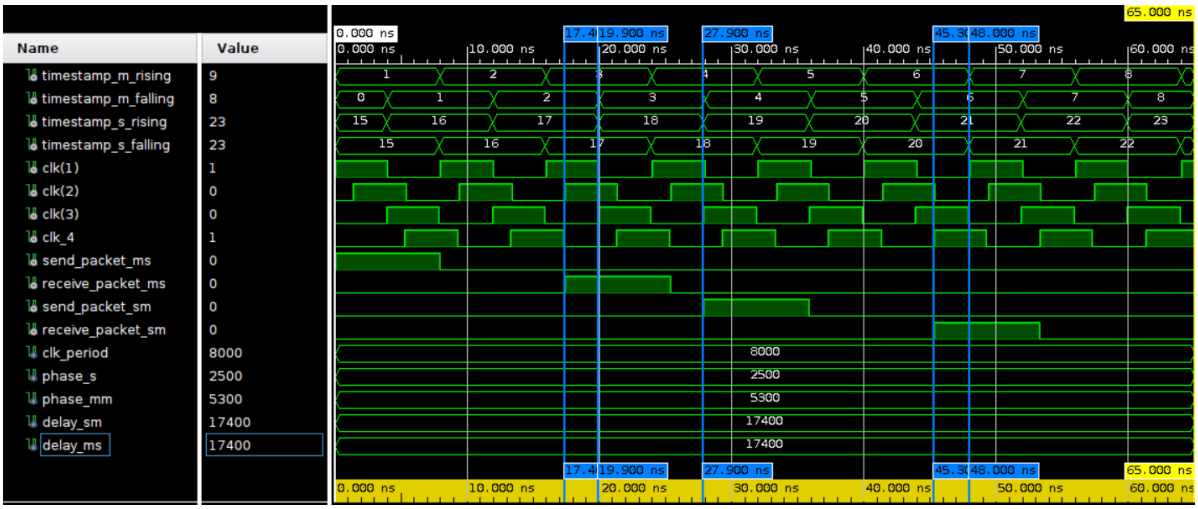


Figure 2.17: Illustration of the White Rabbit timestamp exchange in wave-form from Vivado

After the timestamp registration, the slave sends the packet back to the master, synchronously to the phase-shifted clock. It registers timestamp $T3 = 19 * T_{ref}$, at 27.9 ns. The packet experiences $delay_{sm}$ and arrives at 45.3 ns. However, the counter is synchronous to the reference clock (1). After immediate synchronization, the pulse is detected at the rising edge of the reference clock and stores $T4 = 7 * T_{ref}$ at 48 ns. The difference between the arrival time and the registration time is related to the phase difference between the reference clock and master recovered clock: $phase_{mm}$

One can see that the registered counter values and timestamps T_2 and T_4 are not registered synchronously to their respective arrival time due to phase differences. Hence, the timestamps are corrected and annotated by T_{4p} and T_{2p} in order to correctly calculate one-way transmission delays:

$$T_{4p} = T_4 - T_{ref} + phase_{mm} \tag{2.12}$$
$$T_{2p} = T_2 - phase_s$$

Equation 2.10 can be adjusted to the improved timestamps as follows:

$$delay_{mm} = (T_{4p} - T_1) - (T_3 - T_{2p}) \tag{2.13}$$

2.4.4. Asymmetric transmission delay

The improved timestamps provide a more accurate value of the round-trip delay $delay_{mm}$ due to the added fine phase measurement. However, it is not the round-trip delay, but rather the one-way delay from master to slave, that is desired. To calculate the one-way delay, asymmetry calculations are vital to provide sub-nanosecond accuracy. From Figure 2.14, it could be seen that circuit and Transceiver/Receiver asymmetry is calibrated through the parameters Δ_{txm}, Δ_{rxm}, Δ_{txs} and Δ_{rxs}. These parameters can be obtained via complex measurements or can be found in the White Rabbit database for specific Transceiver/Receiver devices. For example, the transceivers on the Xilinx 7 series devices can be found, as can parameters for the SFP modules. The combined set of circuit/transceiver/receiver delays is typically referred to as Δ. The difference in transmission line delays from master-to-slave (δ_{ms}) and from slave-to-master (δ_{sm}) are related through a calibrated factor α [14]:

$$\delta_{ms} = \delta_{sm}(\alpha + 1) \tag{2.14}$$

With these last these last parameters, the final calculation can be established:

$$delay_{mm} = delay_{ms} + delay_{sm} = \Delta + \delta_{ms} + \delta_{sm} \tag{2.15}$$

$$delay_{mm} = \Delta + \delta_{ms} + \frac{1}{1+\alpha}\delta_{ms} \tag{2.16}$$

$$delay_{ms} = \frac{1+\alpha}{2+\alpha}(delay_{mm} - \Delta) + \Delta_{txm} + \Delta_{rxs} \tag{2.17}$$

Based on the $delay_{ms}$, the counter and phase adjustments can be calculated to equalize the time offset to reach synchronization.

2.4.5. Fine delay compensation

The clocks of the White Rabbit instances can be aligned, such that they increment their time counters simultaneously. Since $delay_{ms}$ is a known value, the fine delay compensation can be calculated to align the edges of the reference clock (1 in Figure 2.14) and the phase-shifted clock (3 in Figure 2.14). Recall that the recovered clock (2 in Figure 2.14) can be phase shifted by a value $phase_s$, which can be calculated as follows:

$$phase_s = (delay_{ms})\ mod\ T_{ref} \tag{2.18}$$

Herein, $phase_s$ presents the clock phase correction and $delay_{ms}$ represent the one-way transmission delay.

2.4.6. Coarse delay compensation

Since the clocks at both instances are aligned, the counter offsets have to be equalized. Upon receiving a timestamp from the master via PTP, the slave should adjust its counter to that value added with:

$$coarse\_delay_s = \frac{delay_{ms} - phase_s}{T_{ref}} \tag{2.19}$$

2.4.7. Summary

In short, White Rabbit enhances the accuracy of PTP by complementary addition of SyncE, a clock feedback loop and removal of TCs. It can adhere to sub-nanosecond accuracy requirements, whilst still offering the scalability and robustness of an Ethernet network. Through (factory) calibration parameters Δ, ϕ_{trans} and α, it estimates the asymmetric one-way transmission delay based on the coarse and fine delay measurements. After the measurements and calculations, the phase of the slave clock is shifted such that the edges align with the master (reference) clock. Lastly, the coarse counters are updated with the result that the time offsets are equalized.

2.5. Conclusions

Highly accurate time sources, such as Cesium-based or GPS clocks, are the best candidates to provide a correct notion of time, but are too costly and impractical for implementation in every modern digital device. Ubiquitously, (crystal) oscillators are employed instead. To maintain a shared notion of time, the offsets and drift/jitter have to be compensated via the concept of synchronization, divided into time offset equalization and syntonization. Packet-based distribution protocols such as NTP and PTP can provide time offset equalization, whilst synchronous Ethernet can deliver syntonization. To reach high synchronization accuracy, PTP and synchronous Ethernet can be combined into an improved protocol, referred to as White Rabbit. Additionally White Rabbit removes Transparent Clocks to improve asymmetry estimations and introduces a clock feedback loop to improve accuracy. White Rabbit enhances the one-way delay calculation through fine and coarse delay measurements. These improve PTP timestamps to reach sub-nanosecond synchronization accuracy.

3

Design and implementation

The previous chapter defined time synchronization and provided the motivation and operation of White Rabbit.

This chapter introduces the current state of White Rabbit design and the concurrent limitations (Section 3.1). Subsequently, it motivates an alternative switch design and defines the design requirements (Section 3.2). Subsequently, the FPGA platform will be introduced (Section 3.3). Afterwards, a novel White Rabbit switch model is proposed to comply with the research question, whilst mitigating the previously discussed limitations (Section 3.4). Consecutively, it focuses on fine delay measurements and provides an implementation comparison (Section 3.5). In addition, it discusses mathematical approaches to mitigate the asymmetry problem through opting for a three-line interconnect between FPGAs, rather than a two-line approach with additional calibration parameters (Section 3.5). Also, it discusses and compares implementations of the fine delay compensation (Section 3.7). Lastly, it concludes the chosen combination of elements for a final model, which will be used for subsequent chapters (Section 3.8).

3.1. Current state and limitations of White Rabbit designs

As mentioned before, the White Rabbit protocol is implemented via two open-source designs, namely the design of the switch and the node [10]. The current White Rabbit protocol and its hardware design were explored in greater detail (Appendix B) to uncover whether improvements could be made. This lead to descriptive subsections on applicability, performance and costs in the current state of the White Rabbit design.

3.1.1. Applicability

White Rabbit requires (factory) calibration parameters Δ, ϕ_{trans} and α, to calculate link asymmetry in order to estimate a one-way transmission delay (see Equation 2.9). Firstly, the set of parameters is extensive and not intelligible. Whilst the parameters ϕ_{trans} and α consist of a single value, Δ represents a large set of values that configure and estimate the transmission through the transceivers and SFP modules. These sets are large and therefore difficult to understand and implement for the end-user. This is in contrast with the initial White Rabbit project objectives that state that no complex calibration sets should be needed. Future designs should not increase the amount of calibration parameters, in order to improve applicability for end-users. In addition, the calibration parameters are only available for hardware (e.g., SFP) of certain types and from certain manufactures. Namely, the hardware has to

be present in the CERN parameter database [9]. Therefore, White Rabbit cannot be implemented on existing or multi-purpose FPGA platforms that do not contain the correct hardware.

Lastly, the design cycle of the current switch design is slow, meaning that the last hardware update is based on the Virtex-6 (2009). The commercial interest of big manufacturers is low, due to the open-source nature of the protocol and the hardware. As a consequence, updates are sporadic when compared to the widely applied node designs, that are smaller and easier to work on for smaller parties. In addition to sporadic updates, built-in security features lack extensive feature sets from well-known brands (e.g., Cisco, Linksys, NETGEAR).

3.1.2. Performance

The aforementioned calibration parameters are static and do not account for hardware variability. FPGA platforms are susceptible to inter-die and intra-die propagation delay differences between logic that cause inequality between supposedly identical devices. This deteriorates the accuracy of the parameter based asymmetry estimation. Similar to die variations, interconnection lines, pins and buffers are also susceptible to delays that deviate from the expected value. To account for the variability, determination of the one-way delay clearly favours measurement rather than estimation through parameters.

3.1.3. Costs

Large hardware costs are mainly caused by the FPGA platform and the external components. Firstly, a more expensive FPGA (about 3000 US dollars) is required, since each port on the switch requires a transceiver. Moreover, as only one FPGA is present in this design, a lot of BRAM and logic cells are required (See Appendix B). The high accuracy external components necessary for the node and switch design, are expensive as well. This is mainly caused by the components for the softPLL, which costs around 80 dollars.

When implementing the White Rabbit protocol within your network, one has to replace all the devices present in the infrastructure, entailing high costs. Future designs should therefore focus on the preservation of present Ethernet switches, to overcome costs, as well as, applicability constraints.

3.2. Requirements

From the previous section, it can be concluded that White Rabbit has limitations: it relies on extensive calibration parameter sets, fiber-based one way transmission delay estimations, limited hardware options, costly components and expensive full-stack switches to build up the network.

The goal of this work is to define a White Rabbit switch, that mitigates the previously stated limitations at low-cost whilst maintaining sub-nanosecond synchronization with other devices. Therefore, the workload will be distributed over several FPGA's. The switch should also be modular, such that multiple configurations can be realised. This leads towards a design exploration that adheres to the following requirements:

- **Distributed and modular design**

  The synchronization task will be distributed over several smaller FPGA's. If a larger White Rabbit Switch is required, FPGA's can be added to sustain ports and workload. Moreover, distribution of the same workload over less expensive FPGA's may lead to cost reduction.

- **Minimal factory calibration**

  The calibration procedure has to be kept minimal and automatic to support modularity and keep user interference at a minimum. This includes one-way transmission delay calculations.

- **Low-cost implementation**

  To maintain a low-cost implementation, solutions for the synchronization tasks should reside in the FPGA as much as possible. In this manner, expensive external components can be reduced. This has an additional advantage, where it is possible for this design to be implemented on existing hardware boards.

- **Synchronization with sub-nanosecond accuracy and precision**

  White Rabbit specifications require sub-nanosecond time accuracy between nodes over large distances (5-10 km of fibre cables)[12]. The experimental setup presented in this study, requires distances of at most 50 centimeters of copper between White Rabbit ports. Therefore, inaccuracy and jitter will be introduced, similarly to the long optic fibre cables. The switch should provide methods to retain the signal quality and phase, such that the synchronization between nodes in a network can be maintained at sub-nanosecond accuracy and precision.

3.3. Platform

The FPGA platform that will be used for this research, is provided in the form of development boards specifically designed by Topic for this purpose. The set consists of two FPGA boards and a carrier board (See Figure 3.1). An FPGA board contains an Artix-7 FPGA, a SFP module, two oscillators, two Digital-to-Analog converters and other external components (e.g., flash and uart pins). Each of these boards contain all components to handle a White Rabbit node design. Besides power, the carrier board provides five differential and bidirectional copper pairs between the I/O pins of the FPGAs (See Figure 3.2).

Figure 3.1: Carrier board with two Topic Artix-7 boards attached. Both boards are denoted as WR_NODE_1 and WR_NODE_2 respectively. Logically, a carrier board for an 18-port White Rabbit switch will contain 18 slots and 18 attached boards.

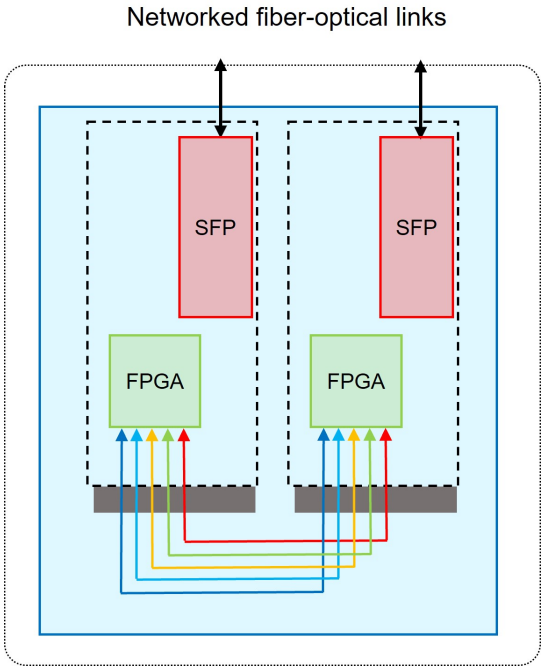


Figure 3.2: Block diagram of the Carrier board with two Topic Artix-7 boards. Accentuated in green, one can find the Artix-7 FPGAs. The red markings highlight the SFP-modules. The five colored lines represent five differential pairs of interconnection lines. The slotted FPGA boards, separated by dotted lines, can be programmed with the minimal White Rabbit node design.

3.4. Introduction to a new switch model

This section will introduce a new model of a White Rabbit Switch based on current designs of the White Rabbit nodes to distribute the operation of the White Rabbit switch over several smaller FPGAs. Recalling the definition of a White Rabbit switch (See Section 2.2.3), it is a multi-port device that contains a single input port that acts as a slave, whilst the several output ports dictate time to downstream devices in master mode. As discussed in Section 3.1, it is therefore costly and difficult to implement and modify.

The node design, on the other hand, is relatively small, well maintained and consists of a minimal set of components to provide the White Rabbit protocol. It is therefore opted to distribute the workload of the current switch design to several smaller FPGAs, implemented with a smaller node. To explore the idea further, the schematic illustration in Figure 3.3 is considered. The upper image (1) shows an abstraction of the current switch design containing: a slave port (orange), several master ports (red) and a large FPGA that controls both port types. The lower image (2) shows the proposed model, where each port (orange/red) contains a small FPGA (green) running the minimal implementation of the node in either slave or master configuration.

Since, each port contains a separate FPGA with separate time information, the nodes in the newly proposed model have to be synchronized. Moreover, the processing of Ethernet traffic can be off-loaded to separate Ethernet switches. This was previously performed by the large FPGA and a separate ARM processor (See Appendix B). The current node design contains the fabric interface, which guides White Rabbit traffic to the LM32 and passes other traffic through to the user code output ports (See Appendix A). Although this is a topic worth of further research, in this thesis the focus mainly lies on synchronization of time between the two White Rabbit nodes.

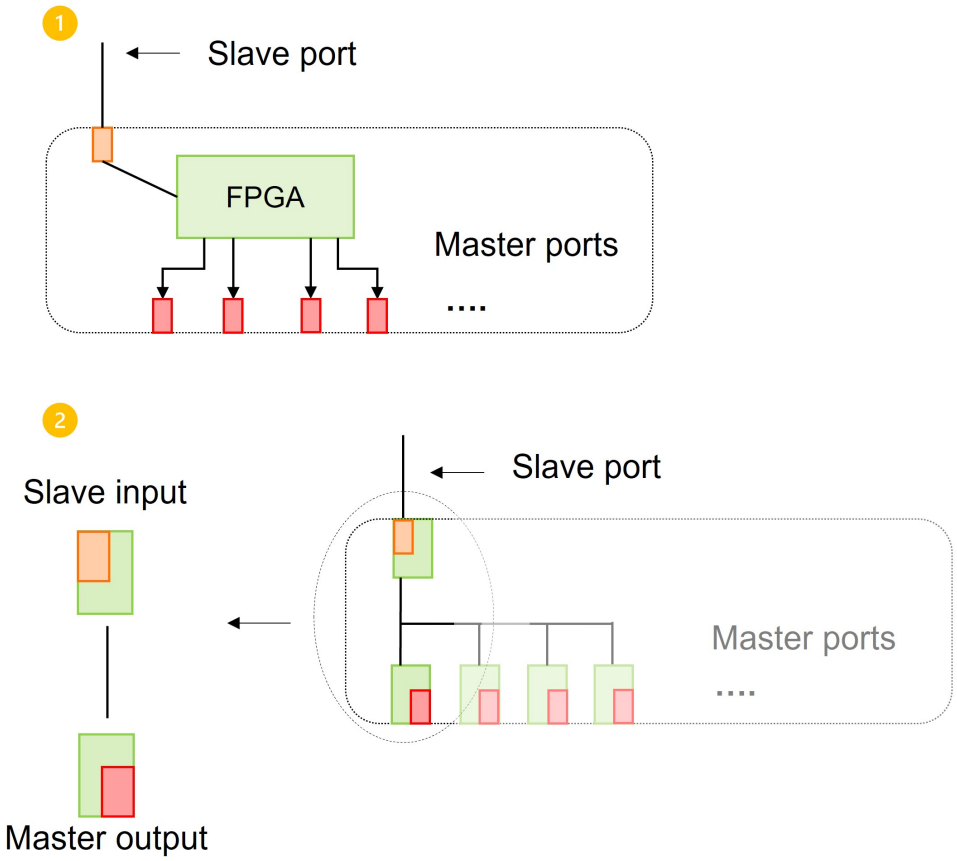


Figure 3.3: This schematic illustration shows the transition of the current White Rabbit switch into a multi-FPGA switch containing White Rabbit node design. The upper image (1) shows an abstraction of the current switch design containing: a slave port (orange), several master ports (red) and a large FPGA. The lower image (2) shows the proposed model, where each port (orange/red) contains a small FPGA (green) running the minimal implementation of the node. The third schematic (3) zooms in on the slave-master interconnection

To design a synchronization method that fits the proposed switch model in Figure 3.3, a 1-port switch is regarded, which contains a single input port (slave) and output port (master) interconnected through several copper lines on a PCB. As seen previously, transportation of signals between FPGA's is not infinitely fast, nor without added jitter and asymmetry. Similarly to the White Rabbit protocol, sub-nanosecond synchronization performance has to be retained through fine delay measurements, asymmetry estimations and fine delay compensation.

The following sections will discuss each of these tasks, compare selected solutions, in order to create an optimal switch design. Within each of these sections, the discussed design options will be scored on several criteria involving three categories: applicability, performance and costs. Applicability refers to the ability of the presented option to be used and implemented in the new switch design. Performance refers to the influence on the accuracy and precision. Costs provides a measure of the potential impact on resources and financial constraints.

3.5. Fine Delay Measurement

This section describes the options for fine delay measurements between the slave and master clocks. They were found by studying relevant literature regarding sub-clock-cycle time measurements in FPGAs. Three different techniques are considered, after which they are compared through simulation, specifications or characteristics.

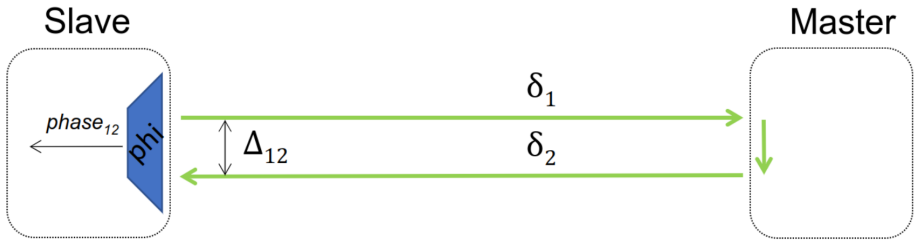


Figure 3.4: A schematic of the fine delay measurement performed by block phi. It measures $phase_{ij}$, which is the phase difference between the reference clock sent to the master with delay δ_1 and the received feedback clock delayed by δ_2. The sum of the delays is represented by Δ_{12}.

A fine delay measurement block represents an entity that measures the phase difference between the output clock and feedback clock over several paths. Figure 3.4 illustrates this principle in the form of measurement block phi, which represents the fine delay measurement. The following equation holds:

$$phase_{ij} = (\delta_i + \delta_j) \; mod \; T_{ref} \tag{3.1}$$

Herein, T_{ref} presents the transmitted clock period, $\delta_{i/j}$ represent the one-way transmission delays and Δ_{ij} represents the sum of both delays.

3.5.1. Option 1: Time-to-Digital converter

In application-specific integrated circuit (ASIC) designs, the phase-difference between clock signals is often determined via time-to-digital converters (TDC). Herein, the phase difference between syntonized clocks is regarded as a shift in time. The TDCs that are discussed in this section are typically referred to as flash TDCs [46]. The name hints at the relation with flash Analog-to-Digital converters, which compares an analog signal to a given set of voltages to construct a digital signal. Figure 3.5 shows an illustration of the flash TDC. It is seen that a delay line is constructed, through which the leading edge clock signal (green) propagates. The second lagging edge clock signal (orange) can toggle an array of flip-flops, which creates a temporary digital measurement of the passed delay elements. The amount of passed elements is related to the propagation delay of a single delay element. Combined, they provide a digital measure for phase difference.

To translate the ASIC based TDC to FPGA platforms, each delay element has to be replaced with an FPGA native element. The most important requirements for the implementation are high resolution and a measurement range of at least one clock cycle of the input clocks (8ns). Therefore, the carry chains are considered, as they are in fact the fastest native elements (i.e., with the smallest propagation delay). Each slice contains a CARRY4 element with four carry elements [2]. Each carry element can be configured by asserting digital high and low signals on the inputs of the carry component, which ensures that the carry-in propagates to the carry-out.

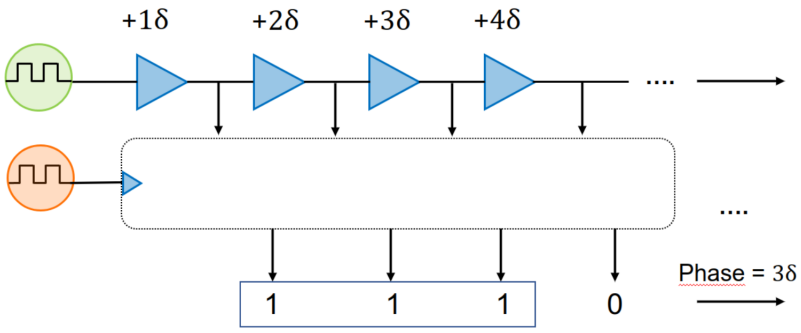


Figure 3.5: A schematic of the flash TDC can be seen, where a leading edge clock (green) propagates through a set of delay elements with delay δ. The lagging edge clock toggles the flipflop array upon reaching its rising edge. It therefore outputs the amount of delays δ that the leading edge clock precedes the lagging edge clock. The schematic shows an example with 3δ phase difference

Within each CARRY4 block, the interconnection between the carry elements is small, whilst the connection between the CARRY4 blocks is larger (approximately 1-2 ps). Therefore, an expected non-linearity is introduced per introduction of a new CARRY4 block. Moreover, this FPGA contains six clock domains, each containing 50 slices or 200 carry elements. Longer carry chains will have to cross a clock domain, which requires a larger physical gap to be crossed to the next CARRY4 element. Therefore, more non-linearity is introduced (approximately 10-100 ps). Additionally, when longer delay lines are considered, the variability of the propagation delay per element increases. Furthermore, one needs to be aware that carry elements are process/voltage/temperature (PVT) dependent, which introduces dynamic propagation delay differences [22].

Considerations

- **The resolution and measurement range are limited by the propagation delay of a CARRY4 element.**

  The carry-in to carry-out propagation is approximately 18 ps (Topic Artix 7, xc7a35tcsg325-1), which defines the maximum resolution of the type of TDC.

- **The measurement range can be traded off against area and power.**

  By using less delay elements, inherent area will go down. Power will also decrease, as less toggling delay elements and flip-flops are required.

- **The delay line is limited by length and cannot fit a full cycle of the input clock.**

  Variability of the propagation delay per element increases with longer delay lines. This causes uncertainty and linearity issues. Similarly, clock domain crossings induce extra delay after 200 carry elements. To measure a full cycle of the 125 MHz input, a delay line of approximately 450 elements is required.

3.5.2. Option 2: Digital Dual Mixer Time Difference

The Digital Dual Mixer Time Difference (DDMTD) is a digital counterpart of the Analog Digital Dual Mixer Time Difference (ADMTD) presented in 1975 [4]. It samples two clock signals to determine their phase difference. Normally, one may over-sample both signals greatly to count the amount of samples a clock signal proceeds the other. As a result, a value of the phase difference can be given with respect to the sampling frequency. However, oversampling a 125 MHz signal with an adequate resolution of 8 ps, would require at least a 125 GHz sampling frequency. Naturally, such speeds are impossible to reach inside FPGAs.

Under-sampling

The concept of under-sampling can be used to obtain similar sampling resolutions. For that, one needs a sampling clock (referred to as clk_{DDMTD}) close but not equal to the frequency of both input clocks. By using a flip-flop that is clocked by the sampling clock, both input clock signals are sampled each cycle at slightly different places compared to their previous cycle. Subsequently, the sampled signals will be deglitched, whilst the amount of samples between the edges of the sampled clocks is to be expressed in a phase difference value. A block diagram of the DDMTD can be found in Figure 3.6. Appendix C.1.2 contains a simulation of the DDMTD operation.

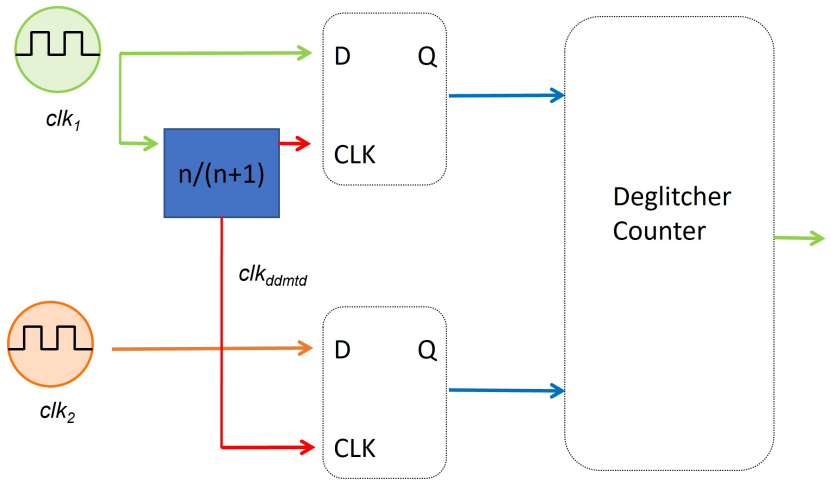


Figure 3.6: A schematic of the DDMTD under-sampler with syntonized out-of-phase input clocks and a close frequency DDMTD clock. The two syntonized clock signals (clk_1 and clk_2) are inserted at the inputs. The third clock clk_{DDMTD} is inserted, which deliberately differs slightly in frequency compared to the input clocks. A variable n is used to define the factor between the input frequency and the DDMTD frequency. A deglitcher counter, deglitches the sampled signals and counts the phase difference between them.

Generation of the sampling clock

White Rabbit utilizes the DDMTD technique and implements an external oscillator to create the close DDMTD sampling clock [47]. However, some authors experimented with clock creation from an internal PLL rather than generating the sampling clock from an external oscillator [40]. It was concluded that the internal PLL introduced approximately 34 ps more jitter on the DDMTD clock. Other works have worked with two cascaded PLLs inside the FPGA, that could provide reasonable resolutions, simpler designs and lower costs [30]. In Table 3.1, the resulted resolution is compared to two external oscillators that are typically implemented with DDMTDs.The White Rabbit repository states that the synchronization result is even slightly improved with cascaded PLLs compared to an external oscillator, but remains unclear about the improvement metric [8].

Table 3.1: Comparison of the DDMTD measurement resolution with two external oscillators (typically implemented with DDMTDs) and a set of cascaded internal PLL's in Xilinx Spartan 6 technology (XC6SLX45T-4CSG324C)

	VCXO (100 ppm)	**VCXO** (122 ppm)	**Hongming[30]**
Resolution (ps)	0.977	1.953	3.906
n	16384	8192	4096
Input clock (MHz)	62.5	62.5	62.5

Both external oscillators and internal PLL's contain phase jitter, which can cause inconclusive phase measurement results. An illustrative simulation of the phase jitter can be found in Appendix C.1.3). Similarly, the input clocks and metastability issues from the flip-flops will add unwanted noise. The metastability is mitigated by replacing each of the flipflops with three cascaded flipflops. Deglitching reduces the influence of jitter by means of a stability counter that counts the amount of consecutive samples at which the sampled clock is high, before addressing that a rising edge has been found. Thirdly, phase measurements are performed several times and subsequently averaged, to increase accuracy as a trade-off to execution time.

Considerations

From the above examples and further simulations, the following statements can be concluded:

- **The resolution depends on the difference frequency between the input and sampling clocks.**

  By correctly defining the PLL constants for the internal PLLs, a maximal n value of 4096 can be obtained (See Table 3.1). The external oscillator implementation uses an n value of 16384. If one compares both implementations, the external oscillator will provide a four times higher resolution whilst less jitter will be introduced. This will be at the cost of introducing a set of external components.

- **Accuracy is heavily influenced by jitter in the measured and reference clock signals**

  Jitter in the input signals causes inconclusive phase measurement results (See Appendix C.1.3). Therefore, a stability counter and averager are deployed to improve the performance, at the cost of increased execution time and area.

- **Higher resolutions increase measurement time and area.**

  By increasing the resolution, the measurement time increases approximately linearly. Similarly, larger counters, deglitchers and averagers are required, which increase the FPGA utilization.

- **The DDMTD can fully reside inside the FPGA.**

  The counters, flipflops, an averager, stability counters and a frequency divider can be realized inside the FPGA, provided that the frequency division is performed by an internal component (i.e. PLL). This is at the cost of measurement resolution and added phase jitter in the measurements.

- **Phase differences can only be measured if the input clocks are syntonized.**

  Without syntonized input signals, DDMTDs cannot be used and will provide inconsistant results.

3.5.3. Comparison

To compare both options, the following criteria are considered: Full cycle measurement, complexity, non-linearity, execution time, resolution, the necessity of external components and area on the FPGA.

Firstly, the TDC implementation cannot measure a full cycle of the 8 ns input clock, without crossing clock domains. The crossing of clock domains induces extra non-linearity to the measurement. The design of the DDMTD is invariant to the large period of the input clock and can measure the full clock cycle. However, the operation and construction of the TDC is less complex.

Moreover, due to the large stability window and the averager in the DDMTD, it can take several seconds-minutes before a result can be obtained. In comparison, a TDC can measure the phase difference in each cycle, with few additional cycles required for the FPGA to process the data. However, this is at the cost of increased area needed for the data processing.

Lastly, the DDMTD can maintain a resolution in the range of 1-4 ps, depending on the creation of the sampling clock. The TDC is limited to a resolution of 18 ps due to the CARRY4 elements.

Table 3.7 summarizes the discussed criteria and scores the design options. The favored option for that criteria is scored with a '1', whilst the other is scored with a '0'. In the case of equal benefit, both designs received a '1'. Based on this comparison, the DDMTD option is favored.

	Applicability		**Performance**			**Costs**		**Total**
	Full cycle	*Complexity*	*Non-linearity*	*Execution Time*	*Resolution*	*External components*	*Area*	
TDC	0	1	0	1	0	1	0	3
DDMTD	1	0	1	0	1	1	1	5

Figure 3.7: This table lists the discussed criteria, assigned to the following classes: applicability, performance and costs. The favored option for each criteria is scored with a '1', whilst the other is scored with a '0'. In the case of equal benefit, both designs received a '1'.

3.6. Asymmetric Transmission Delay

Synchronization protocols, such as White Rabbit, are often limited to a single bi-directional connection, simply because running two or three 100-km optic cables to another White Rabbit instance is rather expensive. Therefore, one-way transmission delays can only be estimated, unless the two nodes are both synchronized [18]. Some works try to estimate one-way delays by measurement of round-trip times with several other nodes in a network with bi-directional links [21]. Finally, a measurement determines the time of a packet, when it is sent through all nodes. The combination of these measurements results in improved one-way delay estimations.

However, when the new switch model is considered, one is not limited by a single bi-directional link connection to the other node. Instead, one PCB line can be used to send a clock signal, whilst another line can be used to receive the feedback clock. Even more links can be created as multiple links between nodes are available on PCBs. These are far less costly than running an extra 100m optic fiber cable to another node.

Therefore, the following situation is considered, whereby a slave node and master node are connected through two sets of sending and receiving links. Each link is noted by the link number and a direction arrow, whereas the transmission delay is denoted by δ and a number. Figure 3.8 shows a schematic of the situation. The left node (the slave) can measure round-trip-times between two lines (Δ). These measured round-trip times (Δ) can be expressed as follows:

$$\Delta_{ij} = \delta_i + \delta_j \tag{3.2}$$

The number i represents the forward path, whereas j represents the return path. To improve asymmetric link calculations, one wants to obtain the one-way transmission delays δ from the round-trip-times (Δ). Let A be an 4x4 matrix, relating the transmission delay vector $\vec{\delta}$ to the round-trip time measurements $\vec{\Delta}$. The system of equations can be expressed in the vector form:

$$A * \vec{\delta} = \vec{\Delta} \tag{3.3}$$

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \quad \vec{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix} \quad \vec{\Delta} = \begin{bmatrix} \Delta_{12} \\ \Delta_{32} \\ \Delta_{34} \\ \Delta_{14} \end{bmatrix}$$

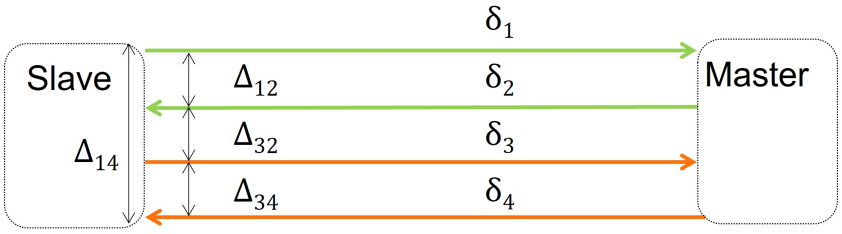


Figure 3.8: Interconnection via four transmission lines (2 forward, 2 backwards) between two nodes (slave, master)

Theorem 1 (Invertible Matrix [36]) *Let A be an n x n matrix, and let T:R^n be the matrix transformation T(x) = Ax. The following statements hold equivalent:*

1. *A is invertible.*
2. *A has n pivots*
3. *The columns of A are linearly independent*
4. *Det(A) $\neq$ 0*
5. *The columns of A span R^n.*
6. *Ax = b has a unique solution for each b in R^n*

According to Theorem 3.6, a system of equations can be solved with a unique solution for each δ, if the matrix A has an inverse. By using Laplace expansion [36] on the 4th row of A (A_4), one can deduce that the determinant is equal to zero. Therefore, A has no inverse and cannot be solved with a unique solution for $\vec{\delta}$. By using Gaussian elimination, one can find that indeed the rank of A is less than 4, whilst the diagonal elements confirm that the determinant value is zero:

$$Det(A) = 0 \qquad A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

To find a unique solution, assumptions have to be made to alter the transmission delay matrix A. Suppose one assumes that the one-way transmission delay δ_2 is equal to δ_3. Measurement Δ_{32} will have

no added value to the system of equations, as $\delta 3$ can now be obtained by solving for δ_2. Through this dimension reduction, the set of vectors and matrices change accordingly to:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad \vec{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_4 \end{bmatrix} \quad \vec{\Delta} = \begin{bmatrix} \Delta_{12} \\ \Delta_{34} \\ \Delta_{14} \end{bmatrix}$$

By using Laplace expansion [36] on the 1st row of A (A_1), it can be found that the determinant is equal to:

$$Det(A) = 1 * \begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix} + 1 * \begin{vmatrix} 0 & 1 \\ 1 & 1 \end{vmatrix} = 2 \tag{3.5}$$

The newly found system of equations has a unique solution for every δ_i according to Theorem 3.6. In that case the following equations hold, where I denotes the identity matrix and A denotes the inverse matrix of A:

$$A^{-1} * A * \vec{\delta} = A^{-1} * \vec{\Delta}$$

$$I * \vec{\delta} = A^{-1} * \vec{\Delta}$$

$$\vec{\delta} = A^{-1} * \vec{\Delta} \tag{3.6}$$

$$\vec{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_4 \end{bmatrix} \quad A^{-1} = \frac{1}{2} * \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix} \quad \vec{\Delta} = \begin{bmatrix} \Delta_{12} \\ \Delta_{34} \\ \Delta_{14} \end{bmatrix}$$

By considering multiple transmission lines and assuming that the one-way transmission delay of two of the lines is equal, improvements on the estimation can be made. To illustrate this, one may follow the numeric example with δ_1 = 2ns, δ_2 = 3ns and δ_4 = 4ns. The measurement data $\vec{\Delta}$ will consecutively be retrieved as follows:

$$\vec{\Delta} = \begin{bmatrix} 5 \\ 7 \\ 6 \end{bmatrix} ns$$

By substituting these numbers in Equation 3.6, one yields the following correct transmission delay for the first link:

$$\delta_1 = \frac{1}{2} * (5 - 7 + 6) = 2ns \tag{3.7}$$

Synchronization protocols (e.g., White Rabbit, PTP) would only posses the round-trip time and approximate the one-way delay to:

$$\delta_1 = \delta_2 = \frac{\Delta_{12}}{2} \approx 2.5ns \tag{3.8}$$

3.6.1. Discussion

The improved performance can only be acquired by using multiple interconnection lines and by assuming that the one-way transmission delay δ_2 is equal to δ_3. As of yet, this may seem to be an arbitrary hypothesis, but makes sense when Input/Output buffers are considered (See Section 2.3.2). Instead of using four separate lines, three lines can be used instead, where one of the lines is bi-directional, as it is connected with Input/Output buffers on the FPGAs. In essence, these buffers can be used to alternately send and receive signals through the same line. Naturally, such alink cannot be used to share a constant clock signal from the slave with the master, but would be perfect for periodic transmissions to measure the round-trip times. The opted idea is schematically illustrated in Figure 3.9. Again, this option can only improve the performance by assuming that the transmission delay over the newly added link, is approximately equal when a signal moves from slave to master and from master to slave. If that is the case, this method eliminates the error in one-way transmission delay calculations by calculating the asymmetry between the links.

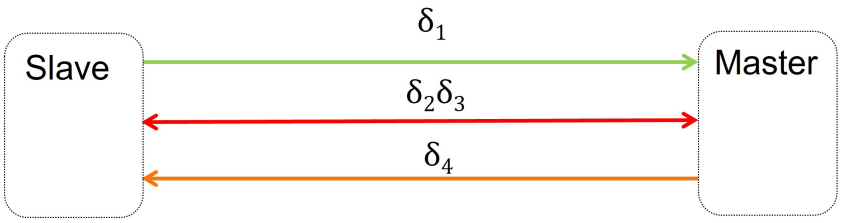


Figure 3.9: Schematic of the slave-master connection with three links instead of two. The direction of each path is denoted by the direction of the arrows.

3.7. Fine Delay Compensation

This section discusses three methods that can delay an input clock signal by a programmable amount. As presented in the requirements, the option should be low-cost and should therefore not require any external components. The function of the fine delay compensation block can be mathematically described as:

$$phase_c = (\delta_1) \; mod \; T_{ref} \tag{3.9}$$

Herein, $phase_c$ presents the clock phase correction and δ_1 represent the one-way transmission delay from slave to master, where the syntonized clock is sent through.

3.7.1. Option 1: Tapped Delay Line with CARRY4

A tapped delay line can typically be constructed by a cascaded set of small delays to induce a phase shift in the clock signals. It was previously seen that a CARRY4 chain can be used, as it provides a low delay per element in the FPGA [2]. Figure 3.10 shows a delay line constructed of CARRY4 elements. Each delay can be tapped to acquire a desired delayed version of the clock. However, non-linearity is introduced due to clock-crossings and continuation of the chain in other slices. The propagation delay of each delay unit is also affected by temperature, voltage and power fluctuations [31]. It should therefore be continuously calibrated against a known delay threshold.

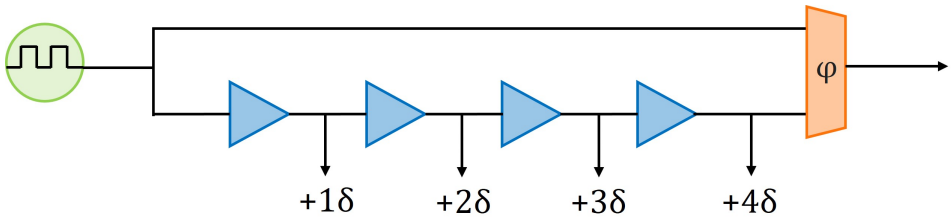


Figure 3.10: This schematic shows a chain of delay elements (blue) to delay a clock signal. The total chain delay is measured through phase measurement between the clock and the chain output clock

Considerations

The following characteristics should be regarded, when implementing a tapped delay line:

- **Resolution:** The resolution is limited to the propagation delay of the CARRY4 elements: Namely, 18 ps.
- **Chaining of elements:** Running an input clock with an 8 ns period through many cascaded delay elements cannot fill a full clock cycle in the Artix-7 FPGAs. The delay per carry element is small, leading to a large number of elements to fill a single clock cycle. Moreover, clock regions in the FPGA have to be crossed, which induces non-linearities. It is also possible to increase the clock frequency, which decreases the amount of elements that have to be passed, or use multiple phase-shifted versions of the clock as calibration points [23]. However, this makes the implementations more complex.
- **Delay variability:** Using voltage and temperature measurements to compensate delay differences is difficult, as it requires extensive measurements and calibration to rhyme temperature/-voltage measurements against delay change. Additionally, reduced voltage and increased temperature both increase the propagation delay, but these effects are complex and correlated to other non-linear CMOS phenomena, making it hard to find the cause and accurately compensate the effect.
- **Delay measurement:** To reduce the effect of total variability of the delay line, one could measure the added delay against a non-delayed reference, to approximate the delay per tap value. This option is depicted in Figure 3.10. However, the accuracy of the tapped delay is now dependent on the accuracy of the phase measurement component.

3.7.2. Option 2: Input/Output Delay

Input/Output Delay (IODelay) primitives are Xilinx specific Intellectual Property and are used to induce a tunable delay at input and output ports [3]. These can also be used for minor phase adjustments to increase the phase offset of a clock [53]. This work focuses on Artix-7 devices and therefore only the input delay (IDELAY2) elements are considered, as these solely contain High Range Banks (HR).

Inside the FPGA

IDELAY2 elements are delay elements that experience reduced influence from Process, Voltage and Temperature (PVT) through calibration inside the FPGA [60]. The amount of added delay can be adjusted in 31 taps. Although the IDELAY2 themselves are asynchronous, the resolution of the element is dictated by a reference clock to the IDELAYCTRL primitive, which configures the delay elements based on an input clock frequency. Table 3.2 shows delay per tap at specified frequencies.

Table 3.2: Delay specification for the IDELAY2 primitive in Xilinx Artix 7 series FPGAs with speed grade -1. [6]

Frequency (MHz)	**Per tap tunable delay** (ps)	**Maximum tunable delay** (ns)
200	78	2.496
300	52	1.664
400	39	1.248

Protected intellectual property

Both primitives are protected IP. It is therefore impossible to see what is inside the building blocks. Some effort has been made to obtain more information from Xilinx, but with limited results. Looking at the instantiation template, a generic attribute SIGNAL_PATTERN hints that different structures are used for either CLOCK or DATA signals. One could speculate that for phase shifts in clock signals, a structure close to a PLL could be probable with a temperature-compensated VCXO, combined with dynamic phase shift capabilities. However, this remains speculation.

Simulation

To test the functionality of the delay elements and gain more information about them, simulations are executed in Vivado 2021.2. For improved readability and intelligibility, both IDELAY2 and IDELAYCTRL are wrapped into one entity, as not all input ports are relevant in this situation. Table 3.3 contains each port signal in the wrapper entity.

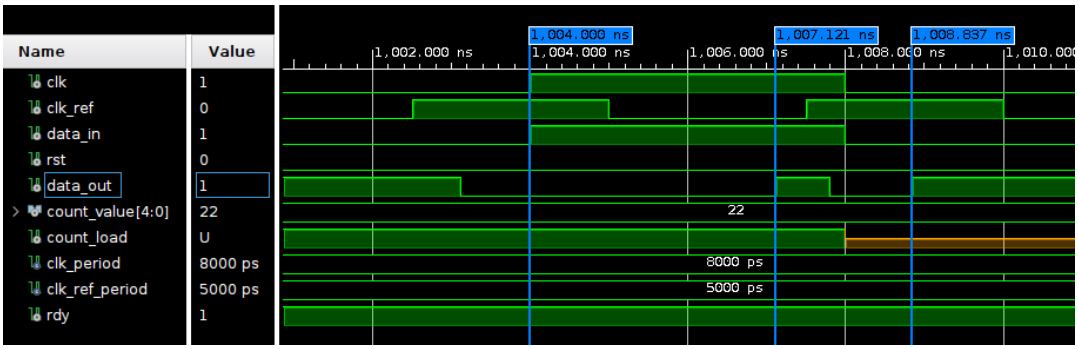
Table 3.3: Ports in the wrapper entity.

Signal	**Type**	**Direction**	**Function**
clk	std_logic	In	Clock to synchronize the input signals to (125 Mhz)
clk_ref	std_logic	In	Seperate clock signal to feed the IDELAYCTRL primitive
rst	std_logic	In	Active High Reset signal
data_in	std_logic	In	Input signal to be delayed, either clock or data signals
count_value	std_logic_vector	In	Set the delay value (0 to 31)
count_load	std_logic	In	Load count_value into the element
data_out	std_logic	Out	Delayed output signal

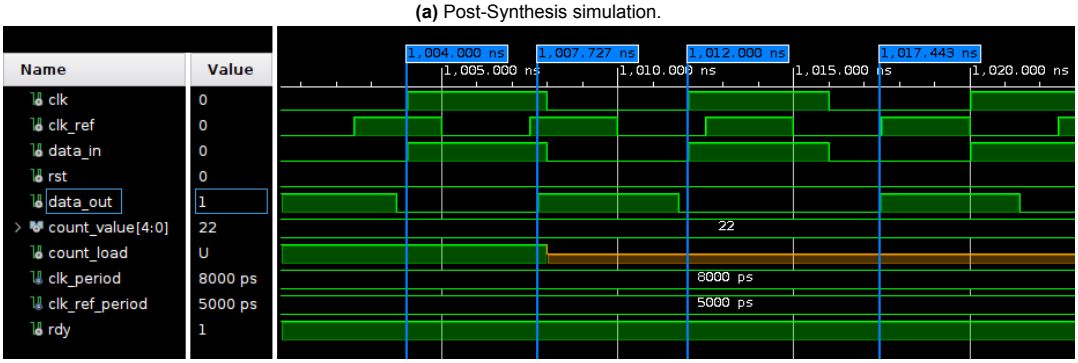
In simulation, a testbench sets the delay (count_value) to 22 taps and enables the phase shift. Figure 3.11 shows waveforms of the behaviour of the wrapper entity in a post-synthesis and post-implementation timing simulation. There is a difference in the static delays before phase shift between the post-synthesis and post-implementation simulations. The latter includes wire delay estimations, which explains a different initial or static delay compared to post-synthesis. The rest of the static delay is induced by I/O ports, port buffers and the delay element itself. Table 3.4, summarizes the delays per simulation for 22 taps.

Table 3.4: Delay specification for the IDELAY2 primitive in Xilinx Artix 7 series FPGAs with speed grade -1. [6]

Simulation type	**Frequency** (MHz)	**Static delay** (ns)	**Added delay (ns)**
Post-synthesis	200	3.121	1.716
Post-implementation	200	3.727	1.716



(a) Post-Synthesis simulation.



(b) Post-Implementation simulation.

Figure 3.11: Post-Synthesis and Post-Implementation simulations with clk_ref = 200 MHz. In the Post-Synthesis waveform, the first blue label highlights a rising edge of data_in and the second label shows the rising edge of data_out when no delay has been added. A glitch is introduced, as the signal is only high for a small bit, after which the added delay value kicks in. The third label shows the rising edge of data_out when delay has been added. In the Post-Implementation waveform, the first two blue labels show the initial delay difference between data_in and data_out. After count_load has finished, both show a correct addition of 1.716 ns delay, which aligns with 22 taps of 78 ps.

Considerations

The following characteristics were found by simulating and testing the IDELAY element:

- **Resolution:** Table 3.2 specifies per tap tunable delays ranging from 39 to 78 ps.
- **Input delay elements:** Artix-7 devices contain HR-banks that lack output delay elements.
- **Chaining of elements:** Several IDELAY elements can be cascaded to create larger delays [13].
- **Instant delay adjustment:** The adjustments cause glitches whenever delays happen near edges of the input clock signal. This behaviour can be seen in Figure 3.11a. Glitch-less behaviour can only be achieved if a sample point at the point of delay inset has the same value as the sample point from our desired delay value before. Xilinx states that in this case the state machines and Clock-Management-Tile elements (such as PLLs) running on the output clock signal will not be distorted [3]. However, one assumes that single increments of the tap delay are more desirable compared to large phase jumps.

- **Forwards phase adjustment:** With IDELAY elements, one can only delay a signal in time. If the edge of the clock signal has to be shifted to an earlier moment in time, the delay elements have to delay the input clock signal by almost an entire period of that clock signal to obtain our desired output clock. Instead it may have been easier to displace the clock forward in time. Nevertheless, the execution time of delay remains the same.
- **Static delay addition:** Timing reports in Vivado denote a 0.6 ns static delay for signals to pass through a single IDELAY element without added tap delays.
- **Ambience variations:** The IDELAYCTRL module provides a reference clock input that allows internal circuitry to derive precise tap delay values via a voltage bias, independent of process, voltage, and temperature [56][55][3].
- **Jitter addition:** High performance mode reduces period jitter in delay chains for clock signals to 0 ps per tap [6].
- **Setup/hold time:** The IDELAYCTRL primitive needs 3.67 ns to setup after reset, whilst setup and hold times for input signals in the IDELAY2 elements require at least 0.21 ns [6].
- **Stabilization clock:** IDELAYCTRL requires a stable clock signal up to 400 MHz to be delivered on an input pin or via an MMCM that drives a global clock buffer. It will enable synchronization and grouping of all delay elements in the same clock region.

3.7.3. Option 3: Mixed-Mode Clock Manager

As previously discussed, Phase Locked Loops are devices that generate periodic signals based on frequency characteristics of the input signal. They can also filter jitter, de-skew clocks and shift the phase of a clock [1]. The latter is explored in this section.

Inside the FPGA

Xilinx 7 series devices contain several Clock Management Tiles (CMT), which include two types of PLLs: a mixed-mode clock manager (MMCM) and a regular phase-locked loop (PLL) [1]. MMCMs are advanced PLLs that provide important features including dynamic phase-shifting, fine phase-shifting and fractional divide [1].

Phase Shifting

Every PLL and MMCM can output eight different output clocks of the same input clock. With static delays, one can create eight different phases of a clock with 45 degrees phase differences. The MMCMs can provide even more phase shifted versions through dynamically shifting a clock up to 360 degrees with linear steps depending on the Voltage-Controlled-Oscillator (VCO) frequency. The relation between the tap resolution and the VCO frequency can be found below in Equation 3.10 [1]. Figure 3.5 shows the maximum and minimum VCO frequencies and the calculated tap delays associated.

$$T_{tap} = \frac{1}{56 * F_{VCO}} \tag{3.10}$$

Table 3.5: Delay specification for a MMCM in Xilinx Artix 7 series FPGAs with speed grade -1. [6]

Name	**Description**	**Frequency** (MHz)	**Per tap rounded delay** (ps)
$F_{VCO_{min}}$	Minimum VCO frequency	600.00	29.76
$F_{VCO_{max}}$	Maximum VCO frequency	1200.00	14.88

Configuration of the MMCM

The frequency of the VCO influences the behaviour of the output clock. Namely, a higher VCO frequency induces less jitter and smaller phase errors [53]. The VCO frequency can be calculated by Equation 3.11. It can be seen that VCO frequency depends on the input frequency, the multiplication (M) factor and division (D) value for the frequency divider in the PLL/MMCM feedback loop. The goal is to make D and M values as small as possible, while maximizing VCO frequency for the aforesaid reasons [1]. Both values depend on the input frequency (F_{IN}), the maximal VCO frequency ($F_{VCO_{max}}$) and maximal PDF frequency ($F_{PDF_{max}}$). One finds the ideal values for M and D by using Equation 3.13 and 3.12.

$$F_{VCO} = F_{IN} * \frac{M}{D} \tag{3.11}$$

$$D_{ideal} = ceil(\frac{F_{IN}}{F_{PDF_{max}}}) \tag{3.12}$$

$$M_{ideal} = floor(\frac{D_{ideal} * F_{VCO_{max}}}{F_{IN}}) \tag{3.13}$$

MMCMs also support fractional frequency division. This means that the division factor of the frequency divider can take on fractional values with steps of 0.125. This way, one could create a wider variety of output frequencies at the cost of larger output jitter. However, integer division is preferred in the feedback path as it only needs internal paths inside the MMCM, which minimizes jitter [1].

Not only the VCO frequency and division values have influence on jitter behaviour from an MMCM: the update frequency of the feedback loop determines the influence of input jitter. If the feedback path bandwidth is minimized, the VCO will be adjusted less frequently through the feedback loop and is therefore less prone to short-time jitter from the input clock. This way, the MMCM can be used as a jitter filter. However, this is at the cost of increased output jitter and static delay between the input and output clock [53].

Simulation

To test the functionality of dynamic phase shifting for adjustment of a clock phase, simulations are executed in Vivado 2021.2. The design consist of two blocks: a configured MMCM in a clock wizard block and a MMCM controller.

Clock wizard block

Firstly, the MMCM primitive has to be initiated with appropriate configuration to match the desired output characteristics. The output clock should have high precision and therefore minimal output jitter. Additionally, the MMCM should preferably add minimal static delay to the system, that has to be compensated otherwise.

Based on aforementioned statements, the attributes and parameters for the MMCM can be seen in Table 3.6). These lead to estimated characteristics of the MMCM, that can be seen in Table 3.7. Finally, the MMCM primitive was instantiated via the Clock wizard block in block design to ease the process of interconnecting ports with different controller designs.

Table 3.6: Attributes and parameters chosen for a MMCM in Xilinx Artix 7 series FPGAs with speed grade -1. [6]

Attribute	Description	Value
BANDWIDTH	Feedback path bandwidth	High
CLKIN1_PERIOD	Input clock period	8 ns
CLKOUT_PERIOD	Output clock period	8 ns
M	Multiplication factor	9.000
D	Division factor	1.000

Table 3.7: Summary of estimated characteristics from Vivado based on a MMCM with attributes from 3.6

Attribute	Description	Value (ps)
Jitter	Pk-to-Pk jitter estimate with ideal input clock	111.194
	Pk-to-Pk jitter estimate with 10% Pk-to-Pk jitter on the input clock	190.304
Phase Offset	Phase difference estimate between the input and feedback clock	89.430
Per tap delay	Per tap delay based on Equation 3.10	15.873

Controller

To control the MMCM dynamic phase shift, a controller is designed to produce a sequence of pulses to reach a desired phase shift by means of single increments and decrements. From Table 3.8, it can be seen that the phase shift delay (count_value) denotes a positive number that represents the amount of phase shift to be added in order to align the positive edges of the output clock and the desired output clock. Subsequently, the controller has to decide whether incrementing or decrementing the phase is faster to reach the desired edge alignment. Therefore, it has to calculate the amount of steps in either direction. Figure 3.12 shows an example situation, where the the input phase shift delay is 6 ns, to align the clock edges. However, it is faster to use single decrements until -2 ns is reached, instead of increasing the phase to 6 ns. This means that the phase will be shifted at most 4 ns in either direction to align the clock edges. Projecting this onto the instantiated MMCM, it will take at most 252 taps (N_{tap}) according to Equation 3.14 with 4 ns (T).

$$N_{tap} = \frac{T}{T_{tap}} \tag{3.14}$$

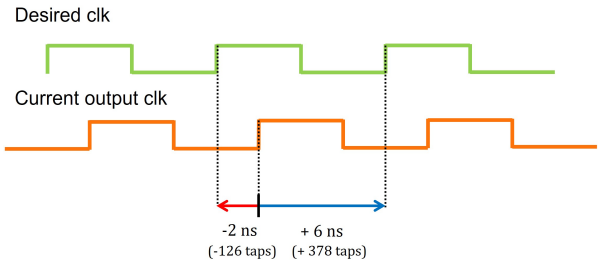
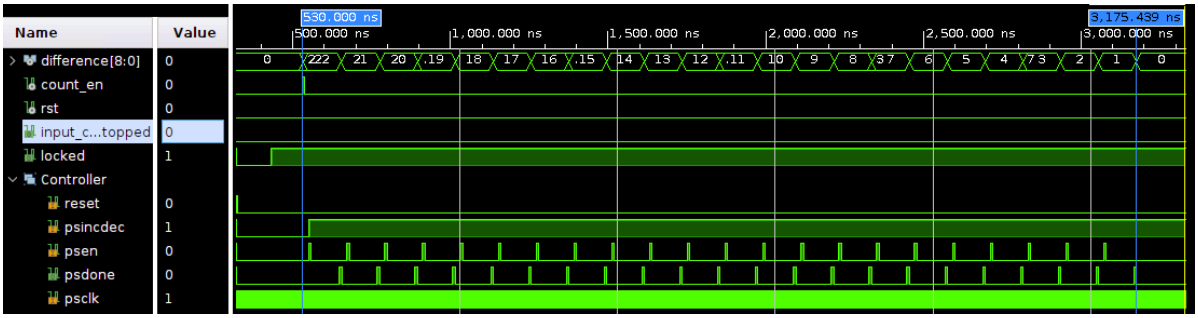


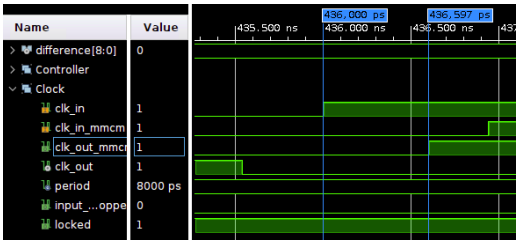
Figure 3.12: Example of the forward and backward phase shift capability of the MMCM

Table 3.8: Controller ports for dynamic phase shift in a MMCM[1]

Signal	Type	Direction	Function
clk	std_logic	In	Controller clock (125 Mhz)
count_value	std_logic_vector	In	Set the positive phase shift delay
count_en	std_logic	In	Enable phase shift
rst	std_logic	In	Active High Reset for the controller
psdone	std_logic	In	Indicates if the MMCM has finished a phase shift
locked_mmcm	std_logic	In	Indicates if the MMCM is locked
clk_missing	std_logic	In	Indicates if the MMCM is receiving an input clk
rst_mmcm	std_logic	Out	Active High Reset for MMCM
clkout_phase	std_logic	Out	Initial phase shift for MMCM
psclk	std_logic	Out	Phase shift control clock
psen	std_logic	Out	Enables a phase shift
psincdec	std_logic	Out	Indicates a decrement (0) or an increment (1) of phase shift



(a) Simulation of a phase shift of 22 taps.



(b) Phase difference before phase shift

(c) Phase difference after phase shift

Figure 3.13: Post-Implementation simulations with clk_in = 125 MHz. Figure 3.13a has two blue labels that denote the begin and end of the shift operation. Figure 3.13b and Figure 3.13c contain two blue labels that denote the phase difference between the input clock and output clock of the MMCM. The difference is 597 ps before phase shifting and 946 ps after. This means a correct addition of 349 ps, or 22 taps of 15.8 ps.

Simulation

Finally, a testbench was constructed that increased the phase difference to 22 taps (349 ps). The controller checked and correctly decided that 22 single increments is faster than 482 decrements, as can be seen in Figure 3.13a. One can see that the controller signals the MMCM 22 times to increment its phase via psen (See Table 3.8), after which the MMCM signals done via psdone. Figure 3.13b and Figure 3.13c denote the phase difference between the input clock and the output of the MMCM before and after phase shifting.

Considerations

The following characteristics were found by studying and simulating the fine phase shift capability of the MMCM element:

- **Resolution:** The resolution of the MMCM is 15.873 ps according to Table 3.7. It is bounded by the VCO frequency
- **Full cycle adjustments:** Can delay a clock for a full cycle of the input clock and is therefore suitable for fine phase delay compensation
- **Gradual phase adjustment:** The adjustments are glitch-less and shift the phase in a continuous manner over 12 cycles of the input clock [1].
- **Single increments or decrements** The phase can only be shifted in single increments or decrements.
- **Bidirectional phase adjustment:** The MMCM can increment or decrement the phase. Therefore, the execution time can be minimized by chosen the shortest path.
- **Static delay addition:** Timing reports in Vivado denote 89.430 ps static phase offset for the feedback clock with respect to the input clock according to Table 3.7
- **Jitter addition:** Minimized Output Jitter mode yields lowest Pk-to-Pk jitter estimates of approximately 100-200ps.

3.7.4. Comparison

To compare the options, the following criteria will be considered: Full cycle compensation, complexity, resolution, calibration, jitter and voltage-temperature invariance.

Firstly, it was found that CARRY4 based tapped delay line cannot compensate for a full cycle of 8 ns input clock, without crossing clock domains and introducing non-linearity in the delay line. The IDELAY has a maximal tunable delay of 1.248 ns at maximal resolution and cannot fill the cycle either. The elements can be chained, which also introduces delay through the connections. However, since the 32 tap delays in each element remain linear, the added delay through the connections can be regarded as a static delay addition. Thus, the delay taps remain 39 ps per increased tap delay. The MMCM can fully shift its delay over the entire 8ns cycle, whilst having the additional ability to shift the clock phase in either direction. Therefore, the implementation of the fine delay compensation with an MMCM, is regarded as the least complex, followed by the IDELAY and CARRY4 delay line.

Additionally, the IDELAY is automatically calibrated and process voltage temperature (PVT) invariant [54]. At least, it is configured through a separate external clocking entity that derives a PVT independent voltage bias. Similarly, it introduces the least jitter on the clock signals that pass-through (0 ps per tap [6]), whereas the MMCM is estimated to produce approximately 100-200 ps jitter.

Thirdly, the maximum resolution of the MMCM is 15.873 ps. The CARRY4 delay comes in a close second with 18 ps, whilst the IDELAY can reach resolutions of maximally 39 ps.

Lastly, the IDELAYs do not cost extra resources (e.g., LUTs, BRAM), as each I/O port already has one built-in. If the MMCM is considered, the Artix-7 FPGA on the Topic boards only contain five MMCM instances, of which three should be available for the White Rabbit node design. Therefore, it is more expensive in terms of area costs.

Table 3.14 summarizes the discussed criteria and scores the design options. The most favored option for that criteria is scored with a ’2’, whilst the least beneficial option is scored with a ’0’. In the case of equal benefit, both designs received the same score. Based on this comparison, the IDELAY and MMCM have an equal score. However, the IDELAY has the highest score in 4 categories, whilst the MMCM has the highest score in two categories. Also, the MMCM scores a lot higher on resolution, whereas the IDELAY scores better in terms of added jitter, PVT invariance and area. As the IDELAY scores better on three criteria, compared to the MMCM, the IDELAY is chosen for the final design.

	Applicability		**Performance**			**Costs**	**Total**
	Full cycle	*Complexity*	*PVT invariance*	*Added jitter*	*Resolution*	*Area*	
Tapped CARRY4	0	0	0	0	1	0	1
IDELAY	2	1	2	2	0	2	9
MMCM	2	2	1	1	2	1	9

Figure 3.14: This table lists the discussed criteria, assigned to the following classes: applicability, performance and costs. The most favored option for that criteria is scored with a ’2’, whilst the least beneficial option is scored with a ’0’. In the case of equal benefit, both designs received the same score.

3.8. Proposed 1-port switch design

Based on previously mentioned considerations, the proposed switch design includes a DDMTD for fine delay measurements, multiple line asymmetry approach to calculate δ_1 and IODELAYs for the fine delay compensation. The block-diagram has been graphically displayed in Figure 3.15. Four selectors (SEL) are added, such that the lines over which the clock travels, can be controlled. Similarly, each one-directional line contains an input-buffer (blue) and an output-buffer, whereas the bidirectional line contains two input/output buffers. The clock transmission delay is highlighted in gray and annotated with δ_1, whereas the other gray areas define the extra lines that are used by the three-line calculation (δ_2 and δ_3).

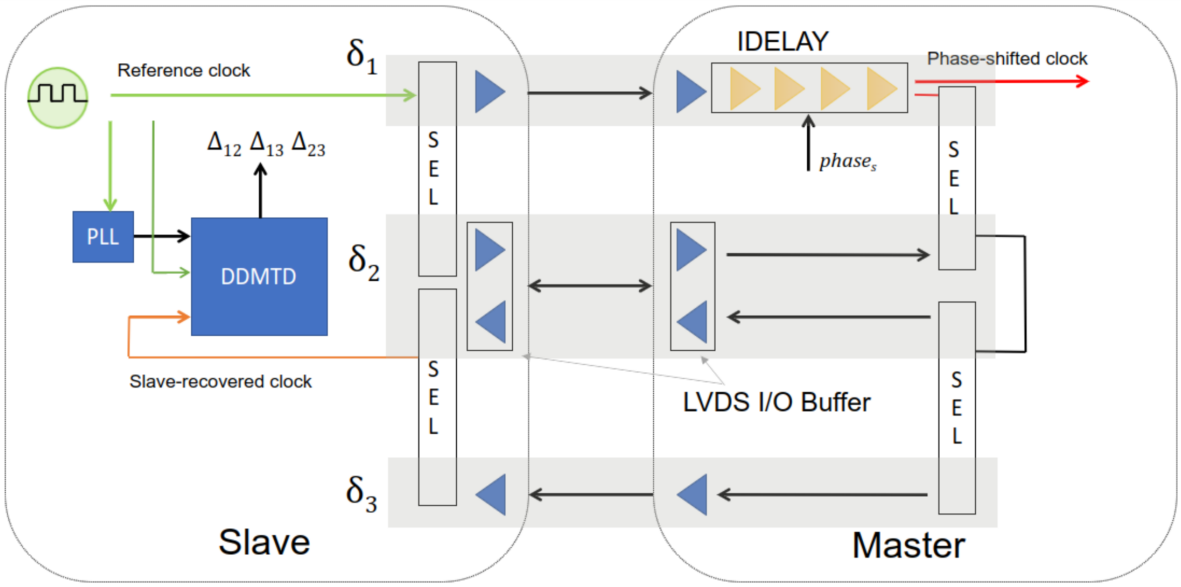


Figure 3.15: This block-diagram graphically displays the proposed 1-port switch design, which contains the DDMTD, IDELAY and multiple line one-way delay (δ_1) calculation. Four selectors (SEL) are added, such that the lines over which the clock travels, can be controlled. Similarly, each one-directional line contains an input-buffer (blue) and an output-buffer, whereas the bidirectional line contains two input/output buffers. The clock transmission delay is highlighted in gray and annotated with δ_1, whereas the other gray areas define the extra lines that are used by the three-line calculation (δ_2 and δ_3).

3.9. Conclusion

The White Rabbit project has limitations: it relies on extensive calibration parameter sets that are only available for limited devices and do not account for hardware variability. This is in contrast with the initial project objectives. Secondly, the update cycle of the hardware implementation is slow. Namely, the current White Rabbit switch is based on FPGA platforms from ten years ago. Thirdly, the switch contains costly components and requires full replacement of present network switches upon deployment.

This work opted to define a new switch design, which distributes the synchronization tasks over several smaller FPGAs to provide a low-cost implementation and improve applicability, whilst requiring minimal factory calibration and maintaining sub-nanosecond accuracy.

The new switch concept is based on the smaller and widely applied White Rabbit node design. The proof of concept is based on two interconnected FPGAs forming a 1-port White Rabbit switch. Fine delay measurements, asymmetry estimations and fine delay compensations are applied to reach synchronization between the FPGAs. The Digital Dual Mixer Time Difference (DDMTD) is preferred for the fine delay measurements due to its superior resolution, area and invariance to large clock periods. A three-line asymmetry calculation is introduced and motivated to provide an improved one-way

delay estimation. Lastly, the IDELAY was selected to provide the fine delay compensation due to its process-voltage-temperature (PVT) invariance, negligible jitter addition and area demands.

4

Verification

The previous chapter defined a new White Rabbit switch design, based on the smaller and widely applied White Rabbit node designs. To provide a proof of concept, a 1-port White Rabbit switch was displayed, in which the functions were filled by the favored implementations from the comparison sections.

This chapter addresses the verification procedure of the three-line asymmetry calculations (Section 4.1). Secondly, this chapter verifies the cost-effectiveness of the newly defined switch compared to the current switch design (Section 4.2). For both verifications, the goal and the procedure are explained after which the results are presented and discussed.

4.1. Three-line asymmetry

Whilst both the fine delay measurement and fine delay compensation were thoroughly discussed by means of specifications and simulations, the three-line asymmetry calculation was solely theoretically found and motivated, without verifying its physical ability to improve the regular two-line asymmetry calculation, as seen in White Rabbit. Therefore, the goal is to verify whether the opted three-line asymmetry calculation has benefits in the newly defined switch.

4.1.1. Verification procedure

The following two statements have to be verified in order to determine the benefits of the three-line asymmetry calculation in terms of accuracy:

- **The one-way transmission delays are dissimilar between each line.**

  Transmission lines have asymmetric delay when compared to one another. The transmission lines include: Input/Output buffers, intra-die connections and PCB traces.
- **The bi-directional transmission line has similar delay from slave to master and master to slave.**

  Transmission lines have symmetric delay when the delay is measured in either direction on the same line. The transmission lines include: Input/Output buffers, intra-die connections and PCB traces.

Platform

The platform from Section 3.3 will be deployed to verify the statements. The Topic carrier board is placed onto a Topic Florida Plus FPGA board, which will provide power and a LVDS offset voltage via a FPGA Mezzanine Card (FMC) connector. Namely, the voltage is required for the I/O banks, that connect to the PCB traces. The five length-matched PCB traces can be seen in Figure 4.1, represented by the five colored arrows. Similarly two single-ended traces (red/blue) are displayed, connected to the FMC connector. On these traces, two vias are displayed (yellow circles), which do not appear on the same distance from the respective source FPGA. Lastly, each FPGA contains an on-board 25 MHz crystal oscillator and PLL combination that generates 125 MHz clock signals, hereafter referred to as the 'oscillator'.

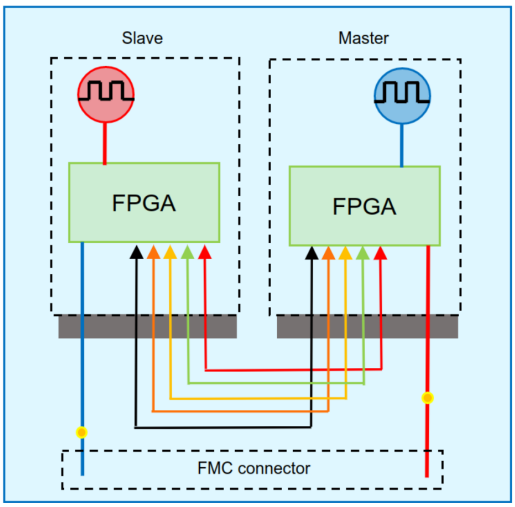


Figure 4.1: A schematic abstraction of the carrier board with two interconnected FPGAs. The five length-matched PCB traces are represented by the five colored arrows. Additionally, two single-ended traces (red/blue) are displayed, connected to the FMC connector. On these traces, two vias are displayed (yellow circles), which do not appear on the same distance from the respective source FPGA. Lastly, each FPGA contains an on-board oscillator and PLL combination that generates 125 MHz clock signals.

Acquiring samples

The fine delay difference between the lines can be measured by transmission of a clock signal over each line. Subsequently, the phase difference is acquired through comparison with the original oscillator signal. Ideally, a clock signal from a high-accuracy external source should be provided via coaxial connectors. Thereafter, the clock signal is transmitted to the other FPGA, which is subsequently outputted to the oscilloscope.

However, these FPGA boards are intentionally designed to be compact and therefore have no area to spare. Hence, the onboard 125 MHz oscillations are adopted as clock sources and their signals are measured through vias close to the oscillators. Additionally, the output clock signal at the other FPGA is measured through a via in the single-ended line.

The measurements can now be divided into two sets. The first set contains delay measurements from the slave oscillator to the master via (red set of arrows, see Figure 4.2). The other set contains measurements from the master oscillator to the slave via (blue set).

The resulting phase difference includes extra delay induced by the single-ended buffer, single-ended line and via. Hence, each measurement in the same set, includes the same extra induced delay. Thus, measurements in the same set can be compared.

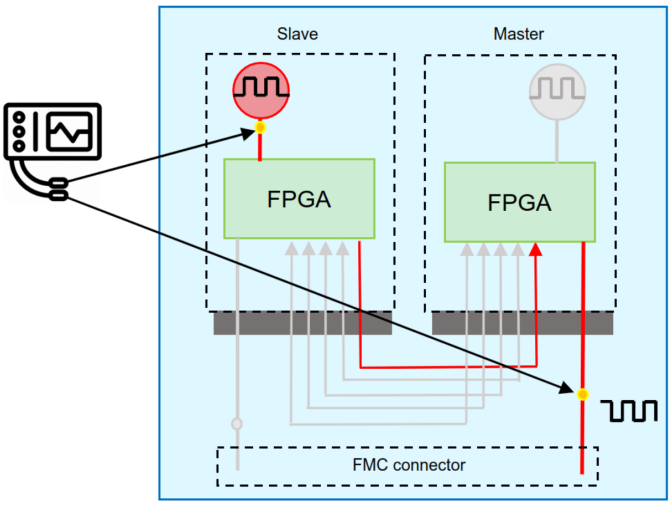


Figure 4.2: An illustration of the one way transmission delay measurement moving over one of the PCB traces. The direction of transmission is from the slave to the master, highlighted by the red lines and arrows. An oscilloscope measures the clock difference with two probes (black arrows).

Measurement devices

To measure the fine delay difference, a Teledyne LeCroy oscilloscope (WaveMaster 813Zi-B) is used in combination with two single-ended active probes (ZS2500) inserted with two via tips (PAAC-PT005). The active probes were chosen because they have lower capacitive loading compared to passive probes. Namely, the input capacitance of the probes causes the input impedance to reduce as the signal frequency increases. Since higher input impedance means that less loading is placed on the measurement point, less distortion is placed on the to-be measured signal. Thus, they provide more insight into fast signals, compared to their passive counterparts.

Figure 4.3: Stock image of the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. It has four inputs, which can be sampled with 40 GigaSamples/s

Figure 4.4: Stock image of the Teledyne LeCroy Active Probe of type ZS2500. It has a limited bandwidth of 2.5 GHz, an 8V range and an ideal input impedance of 1 $M\Omega$, which decreases over frequency.

4.1.2. Results

This section presents the results of the three-line asymmetry verification by comparing one-way transmission delays in terms of their deviation based on the transmission direction and the selected transmission line.

Clock measurement

Firstly, the clock signal from the oscillator at the slave was measured at a via close to the oscillator. The oscillator was set to its maximal bandwidth, which is band-limited to 2.5 GHz. The resulting measurement samples were divided into blocks of 320 samples (40Gs/s of a 125 MHz input signal). At the master via, the clock signal is also measured and graphically displayed in Figure 4.7. Figure 4.5 shows the measurement of the via near the slave oscillator. Some step can be observed in the clock signal, indicating that there is an impedance mismatch, which could be caused by the common mode voltage circuit that the respective via lies in.

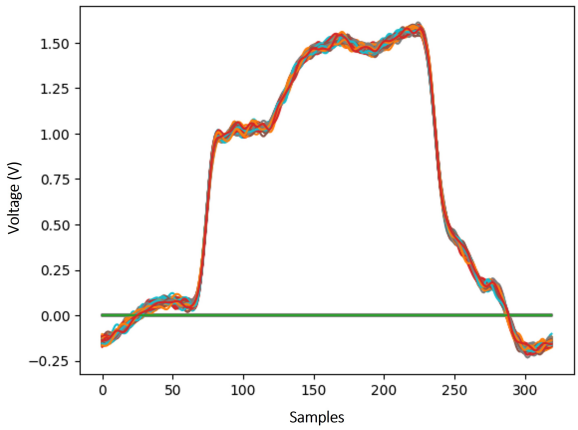


Figure 4.5: This graph plots the averaged clock pulse of the slave oscillator. The measured signal is divided into blocks of 320 samples (40 Gs/s on a 125 MHz signal). The measurement is taken from the closest via to the oscillator.

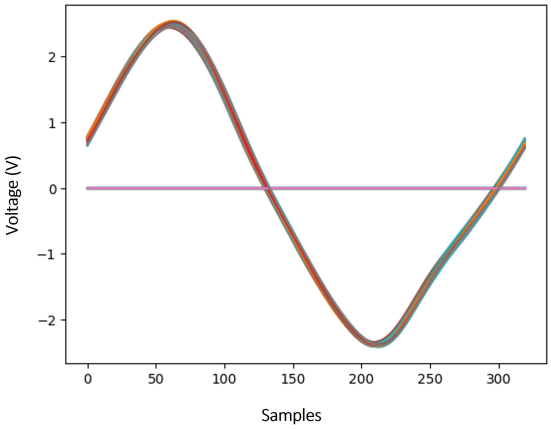


Figure 4.6: This graph plots the averaged clock pulse measured at the master via. The measured signal is divided into blocks of 320 samples (40 Gs/s on a 125 MHz signal). The measurement is taken from the closest via to the oscillator.

Transmission delays

Each transmission line was measured from slave to master (set of red arrows, see Figure 4.2) and master to slave (set of blue arrows). The oscilloscope measured the clock signals at 40 Gigasamples per second. Hence, it takes 320 samples per period of the 8 ns input clocks, at 25 ps accuracy. The measurement waveforms and tables from the oscilloscope can be found in Appendix D.0.2. The edge-to-edge delays and their standard-deviation are plotted in Figure 4.7. The x-axis shows each of the interconnection lines through the carrier PCB, whilst the y-axis shows the delay in ns. Based on the tables and graph, the following observations can be made:

- **Observation 1**: There is a maximal delay difference between two links in the red set of 300 ps. In the blue set, there is maximal difference of 280 ps.
- **Observation 2**: The minimal delay difference between two links in the red set is 25 ps. In the blue set, this is 10 ps.
- **Observation 3**: The maximal delay difference between different transmission directions in the same link are found in line 3 at 110 ps.

- **Observation 4**: The minimal delay difference between different transmission directions in the same link are found in line 4 at 75 ps.
- **Observation 5**: The transmission delay from slave to master shows the highest standard-deviation in line 3.

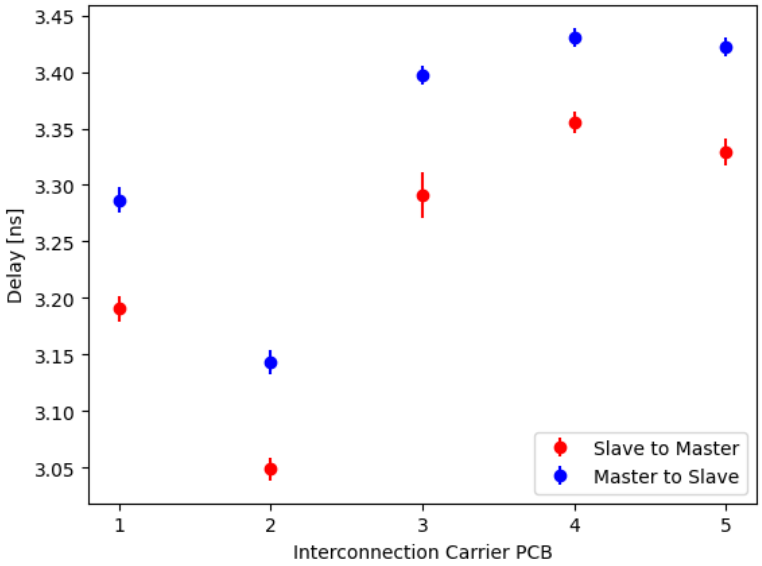


Figure 4.7: This graph plots the edge-to-edge delays and their standard-deviation acquired from the measurements in Appendix D.0.2. The x-axis shows each of the interconnection lines through the carrier PCB, whilst the y-axis shows the delay in ns.

4.1.3. Discussion

Based on observation 1 and 2, it can be concluded that there is large delay variability between the transmission lines. This could be caused by unequal PCB traces or routing differences in the FPGA. However, these are unlikely since the traces are length matched at 62.5 mm (See Table D.1 and internal routing differences are negligible due to adjacent I/O pins. The variability is most likely caused by input and output buffers, which are heavily influenced by hardware variability during fabrication [59]. Similarly, non-deterministic delays are added since the carrier board address regular I/O pins and not clock-capable pins(See Section 2.3). The regular I/O pins are not designed for deterministic delay from the IO pad to the internal clock lines [1].

Based on observation 3 and 4, there is less varying transmission delay difference in the same link between a slave-master transmission and a master-slave transmission (75-110 ps). Thus, a much more consistent delay for the direction measurements is observed compared to observations 1 and 2. This is graphically verified in the graph, since the red and blue points seem to be at a consistent delay difference from each other. Hence, it is concluded that a large contribution of the consistent delay difference is caused by the difference in the single-ended buffer, single-ended line and distance to the via. Naturally, these contributions will not be present in the implementation of the three-line delay calculation, since no measurement point is required.

Interestingly, the slave to master delay in line 3 has almost twice as much variance in the measured delay samples than the other transmission lines (observation 5). Therefore, the measurement has been repeated three times and the measurement time has been extended, to no avail. However, by switching the FPGAs in the slots, the increased standard deviation was seen on the measurement in the other direction. Therefore, the inaccuracy is caused by the output buffer of the slave device on line 3.

Based on the above discussed observations, it can be concluded that the assumption in Section 3.6.1 is valid: Namely, the transmission delay over the added bi-directional link is approximately equal when the signal moves from slave to master and master to slave. Similarly, the transmission delay over different links varies, which means that a two-line one-way calculation will suffer from inaccuracy induced by the link asymmetry. A three-line asymmetry calculation will therefore be beneficial in terms of one-way delay estimation.

4.2. Low-cost hardware implementation

The newly defined White Rabbit switch model has been thoroughly discussed and designed in the previous chapter. One of the requirements of the opted model is cost-efficiency compared to the current design. Hence, it was theoretically motivated to provide the switch functionality distributed over smaller and drastically less expensive FPGAs, without verifying that the new implementation is in fact less expensive. After all, extra costs such as multiple PCB assemblies, extra oscillators and carrier boards were not considered in the motivation. Therefore, the goal is to verify whether the newly defined switch has low-costs compared to the current switch implementation.

4.2.1. Verification procedure

To verify that the newly defined switch is cost-effective, the estimated costs of the current implementation are compared with the estimated hardware costs of the newly defined switch.

4.2.2. Results

This section presents the results of the cost-efficiency of the newly-defined White Rabbit switch. The hardware costs of the current White Rabbit switch design is estimated at $ 3935 according to Table B.4. Similarly the hardware costs of the currently designed node cost $ 205 according to Table B.2.

An extended version based of the distributed switch, would require one slave node and 17 master nodes. Under the assumption that additional costs are required for the carrier board, connectors and power distribution, the following cost estimation can be made (See Table 4.1).

Table 4.1: Main hardware components of a distributed White Rabbit switch

Component	**Function/type**	**Approximated cost** ($)
Node	1 Slave, 17 Masters	3690
Ethernet Switch	Generic 18 port	400
Assembly + PCB	Production costs	50
Connectors	Each node has one	90
Power adapter		15
Total costs		**4245**

However, this work also discussed the option to reduce the measurement accuracy from 4 to 1 ps, as presented by Hongming et al. [30] (See Table 3.1. Therefore, one oscillator (VCXO) and Digital-to-Analog converter (DAC) can be removed from the node devices. Additionally, master nodes contain an oscillator and Phase-Locked-Loop (PLL) combination, simply because the node designs have to be capable to become a slave or grand-master too. Since the masters in the switch, only have to be masters, it could be decided to remove the oscillator (VCTCXO + PLL). Hence, Table 4.2 shows the new costs per slave and master Table. Table 4.3 compares the costs of the current White Rabbit switch design, the distributed switch design and the distributed switch design with reduced node instances.

Table 4.2: Cost reduction in the Slave and Master nodes.

Component	**Elimination**	**Approximated cost** ($)
Slave	VCXO and DAC	183
Master	VCTCXO, VCXO, PLL and 2*DAC	130

Table 4.3: Summary of the current White Rabbit switch, the distributed switch and the reduced switch.

Component	**Approximated cost** ($)
Distributed switch	4245
Current switch	3935
Reduced switch	2948

4.2.3. Discussion

Table 4.3 showed that the distributed switch design can only be cost-efficient, if reduced master and slave nodes are applied. The removal of the phase measurement oscillator (VCXO), results in slightly decreased measurement performance. Since the fine delay elements (IDELAY) have a resolution of 39 ps, measurement accuracies below that resolution are not improving the synchronization accuracy. The elimination of the oscillator (VCXO) can therefore be motivated and replaced by a method with internal PLLs [30]. It was opted to remove the other oscillator as well in master devices, as they are not strictly required. However, the oscillator is used in a SoftPLL construction (See Section A.1), whereby it can be used as a jitter filter for the incoming clock signal from the slave nodes. Thus, the reduced distributed switch can be cost-effective as it reduces the hardware costs with approximately 25%. In contrast, it decreases the performance since input jitter filtering is reduced.

4.3. Conclusion

Whilst both the fine delay measurement and fine delay compensation were thoroughly discussed by means of specifications and simulations, the three-line asymmetry calculation had to rely on a theoretical discussion. As such, measurements have shown that the transmission delay over an added bi-directional link is approximately equal when the clock signal moves from slave to master and vice versa. A large delay variability between other transmission lines is subsequently exposed. The variability leads to inaccurate one-way estimations with a traditional two-line approach. A three-line asymmetry calculation will be conclusively beneficial for the synchronization accuracy. The low-cost requirement of the distributed switch can be maintained through elimination of components in the slave and master node. The reduced switch lowers the hardware costs with approximately 25% by trading off cost-effectiveness with jitter filtering and consecutive performance.

5

Conclusions

In this chapter, the thesis will be concluded. Firstly, a summary of this thesis is provided (Section 5.1). The contributions of this work are presented regarding the research question (Section 5.2). Finally, recommendations are made for research in future works (Section 5.3).

5.1. Summary

Highly accurate time sources, such as Cesium-based or GPS clocks, are the best candidates to provide a correct notion of time, but are too costly and impractical for implementation in every modern digital device. Ubiquitously, (crystal) oscillators are employed instead. To maintain a shared notion of time, the offsets and drift/jitter have to be compensated via the concept of synchronization, divided into time offset equalization and syntonization. Packet-based distribution protocols such as NTP and PTP can provide time offset equalization, whilst synchronous Ethernet can deliver syntonization. To reach high synchronization accuracy, PTP and synchronous Ethernet can be combined into an improved protocol, referred to as White Rabbit. Additionally White Rabbit removes Transparent Clocks to improve asymmetry estimations and introduces a clock feedback loop to improve accuracy. White Rabbit enhances the one-way delay calculation through fine and coarse delay measurements. These improve PTP timestamps to reach sub-nanosecond synchronization accuracy.

The White Rabbit project has limitations though: it relies on extensive calibration parameter sets that are only available for limited devices and do not account for hardware variability. This is in contrast with the initial project objectives. Secondly, the update cycle of the hardware implementation is slow. Namely, the current White Rabbit switch is based on FPGA platforms from ten years ago. Thirdly, the switch contains costly components and requires full replacement of present network switches upon deployment.

This work opted to define a new switch design, which distributes the synchronization tasks over several smaller FPGAs to provide a low-cost implementation and improve applicability, whilst requiring minimal factory calibration and maintaining sub-nanosecond accuracy between devices.

The new switch concept is based on the smaller and widely applied White Rabbit node design. The proof of concept is based on two interconnected FPGAs forming a 1-port White Rabbit switch. Fine delay measurements, asymmetry estimations and fine delay compensations are applied to reach synchronization between the FPGAs. The Digital Dual Mixer Time Difference (DDMTD) is preferred for the fine delay measurements due to its superior resolution, area and invariance to large clock periods. A three-line asymmetry calculation is introduced and motivated to provide an improved one-way delay estimation. Lastly, the IDELAY was selected to provide the fine delay

compensation due to its process-voltage-temperature (PVT) invariance, negligible jitter addition and area demands.

Whilst both the fine delay measurement and fine delay compensation were thoroughly discussed by means of specifications and simulations, the three-line asymmetry calculation had to rely on a theoretical discussion. As such, measurements have shown that the transmission delay over an added bi-directional link is approximately equal when the clock signal moves from slave to master and vice versa. A large delay variability between other transmission lines is subsequently exposed. The variability leads to inaccurate one-way estimations with a traditional two-line approach. A three-line asymmetry calculation will be conclusively beneficial for the synchronization accuracy. The low-cost requirement of the distributed switch can be maintained through elimination of components in the slave and master node. The reduced switch lowers the hardware costs with approximately 25% by trading off cost-effectiveness with jitter filtering and consecutive performance.

5.2. Contributions

This work set out to explore an alternative concept of the White Rabbit switch and find an answer to the following research question:

Can we define a distributed low-cost White Rabbit switch,
which requires minimal calibration and maintains sub-nanosecond accuracy?

The research question is translated into a design with four requirements. The following list discusses the requirements and defines whether the objective of the research question is attained:

- **Distributed and modular design**

  The new switch concept is based on the smaller and widely applied White Rabbit node design. The proof of concept is based on two interconnected FPGAs forming a 1-port White Rabbit switch. Fine delay measurements, asymmetry estimations and fine delay compensations are applied to reach synchronization between the nodes in the distributed switch. Additionally, these nodes lie on separate boards and the amount of output ports can be scaled. The design is therefore modular, easily upgradeable and replaceable. The requirement is therefore attained.

- **Minimal factory calibration**

  White Rabbit relies on extensive calibration parameter sets and the new switch does not add any. Instead of using calibration parameters and traditional two-line transmission delay estimations, a three-line asymmetry calculation is shown to improve performance. It provides a designer with flexibility regarding the physical implementation of the distributed switch, as the influence of asymmetry and trace length are reduced. This leads to minimal factory calibration and the attainment of the requirement.

- **Low-cost implementation** The low-cost requirement of the distributed switch can only be maintained through elimination of components in the slave and master node. The reduced switch lowers the hardware costs with approximately 25% by trading off cost-effectiveness with jitter filtering and consecutively, accuracy. Secondly, the reduction of the nodes decreases the modularity of the switch, since nodes are no longer interchangeable. Hence, the newly defined switch can be low-cost, but thereby reduces performance and modularity.

- **Synchronization with sub-nanosecond accuracy and precision**

  Fine delay measurements, asymmetry estimations and fine delay compensations are applied to reach synchronization between the FPGAs. The Digital Dual Mixer Time Difference (DDMTD) is preferred for the fine delay measurements due to its resolution of 3.9 ps, area and invariance to large clock periods. The IDELAY provides the fine delay compensation

with steps of 39 ps and due to its process-voltage-temperature (PVT) invariance, negligible jitter addition and area demands. In combination with the link asymmetry compensation, the accuracy can reach up to the resolution of the fine delay compensation (39 ps). Therefore, the switch can have a fine delay difference of 39 ps between the input and output port, which is considered low compared to the sub-nanosecond accuracy requirement between the White Rabbit devices. However, it cannot be concluded yet that the design fits the requirement, until the White Rabbit accuracy can be tested when connected to other White Rabbit devices.

The new switch design distributes the synchronization tasks over several smaller FPGAs to provide distribution of the load.

5.3. Future work

Multiple opportunities for potential improvements are identified during the elaboration of this work. These opportunities are either outside of the scope of this work or do not fit the time- frame of the project. Since these opportunities are not acted upon, recommendations are provided for future work. These recommendations can be found below:

- **Deployment of the newly defined switch with clock capable pins.**

  The current verification platform contains two FPGAs and a carrier board, which connects PCB traces regular I/O buffers (non-clock-capable). These buffers contain non-deterministic delays and induce more jitter on the passing clock signals. Non-determinism aims at the inability of the manufacturer to guarantee a constant propagation delay from the I/O pad to the clock lines. This also reduces accuracy and induces short-term changes that the asymmetry calculation cannot counter, since it only reduces the effects of long-term variation. A verification platform with interconnected clock-capable pins can be considered for improved performance in terms of short-term jitter and measurement accuracy.

- **Hardware control through the deployed microprocessor in the FPGA.**

  The components in the switch design are controlled through hardware controllers and manually scheduled through assertion of a control line. To improve applicability and performance, efforts could be made to use the residing microprocessor in the node designs to deploy software-based control (such as software-based Proportional-Integral-Derivative (PID) controllers). In that case, it is advisable to continue this exploration with a RISCV processor, since the upcoming version of White Rabbit will replace the LM32 with a RISCV core.

- **Increase of the SFP count per node to reduce the amount of nodes.**

  Since the node designs are single-ports, only one of the four transceivers of the Artix 7 FPGAs is occupied. Therefore, it is incredibly beneficial to increase the SFP count per module, to reduce the amount of needed master nodes. The load on the master nodes would increase, as four times more signals will have to be translated into packets and consequently separated by the Ethernet class devices (See Appendix A). The costs would drastically reduce since the node costs are the largest contributing factors to the total costs (See Table 4.1). Namely, five modified master nodes with four SFPs would be required, instead of 17 single-port master nodes.

- **Proof-of-concept and definition of the distribution network to multiple ports**

  This work focused on a proof of concept based on two interconnected FPGAs forming a 1-port White Rabbit switch. Fine delay measurements, asymmetry estimations and fine delay compensations were discussed and compared. A next step would be the definition and proof-of-concept of the network inside the distributed switch. Besides switching the transmission lines for asymmetry calculations, the slave node also has to divide its time over several attached master ports. Fortunately, White Rabbit is based on long-term delay variations and

does not require continuous measurement (e.g., a DDMTD takes minutes to calculate the phase difference).

References

[1] *7 Series FPGAs Clocking Resources User Guide*. UG472. v1.14. Xilinx. 2018. URL: https://docs.xilinx.com/v/u/en-US/ug472_7Series_Clocking.

[2] *7 Series FPGAs Configurable Logic Block*. UG474. v1.8. Xilinx. 2016. URL: https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB.

[3] *7 Series FPGAs SelectIO Resources User Guide*. UG471. v1.10. Xilinx. 2018. URL: https://docs.xilinx.com/v/u/en-US/ug471_7Series_SelectIO.

[4] D.W. Allan and H. Daams. "Picosecond Time Difference Measurement System". In: (1975), pp. 404–411. DOI: 10.1109/FREQ.1975.200112.

[5] E. Felicitas Arias et al. "The 50th Anniversary of the Atomic Second". In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 65.6 (2018), pp. 898–903. DOI: 10.1109/TUFFC.2018.2823591.

[6] *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics*. DS181. v1.27. Xilinx. 2022. URL: https://docs.xilinx.com/v/u/en-US/ds181_Artix_7_Data_Sheet.

[7] E. van der Bij, G. Daniluk, and B. Rat. *White Rabbit Switch*. 2015. URL: https://ohwr.org/project/wr-switch-hw/wikis/home.

[8] Erik van der Bij. *White Rabbit Frequently Asked Questions*. 2022. URL: https://ohwr.org/project/wr-cores/wikis/wrpc-faq.

[9] Erik van der Bij. *White Rabbit SFP Database*. 2020. URL: https://ohwr.org/project/white-rabbit/wikis/SFP.

[10] Erik van der Bij. *WRPC FAQ*. 2022. URL: https://ohwr.org/project/wr-cores/wikis/Wrpc-faq?version_id=c4170f2b4beb22c08b8e836f91449ad7ef5d5482 (visited on 03/2022).

[11] CERN. *The Large Hadron Collider*. URL: https://home.cern/science/accelerators/large-hadron-collider.

[12] CERN. *The White Rabbit Project*. URL: http://white-rabbit.web.cern.ch/.

[13] N. Corna et al. "Programmable Delay-Line with High-Resolution Time Steps Implemented in a Digital-to-Time Converter IP-Core for FPGAs and SoCs". In: (2020), pp. 1–3. DOI: 10.1109/NSS/MIC42677.2020.9507750.

[14] Emilio G. Cota et al. "White Rabbit Specification: Draft for Comments". In: (2011).

[15] Grzegorz Daniluk. "White Rabbit PTP Core the sub-nanosecond time synchronization over Ethernet". thesis. Warsaw: Warsaw University of Technology, 2012. URL: https://ohwr.org/project/white-rabbit/uploads/479c4673c48eae8fe17816d144921217/GD_mgr.pdf.

[16] Grzegorz Daniluk. *WRPC v4.2 Spec Spartan 6*. 2018. URL: https://ohwr.org/project/wr-cores/tree/master/.

[17] Fardin Derogarian, João Canas Ferreira, and Vítor M. Grade Tavares. "A Precise and Hardware-Efficient Time Synchronization Method for Wearable Wired Networks". In: *IEEE Sensors Journal* 16.5 (2016), pp. 1460–1470. DOI: 10.1109/JSEN.2015.2501645.

[18] Nikolaos M. Freris, Scott R. Graham, and P. R. Kumar. "Fundamental Limits on Synchronizing Clocks Over Networks". In: *IEEE Transactions on Automatic Control* 56.6 (2011), pp. 1352–1364. DOI: 10.1109/TAC.2010.2089210.

[19] Guanghua Gong. *Cute-WR-DP, new version of SP*. 2020. URL: https://ohwr.org/project/cute-wr-dp/wikis/home.

[20] T. Gotoh, K. Imamura, and A. Kaneko. "Improvement of NTP time offset under the asymmetric network with double packets method". In: (2002), pp. 448–449. DOI: 10.1109/CPEM.2002.1034915.

[21] O. Gurewitz, I. Cidon, and M. Sidi. "One-way delay estimation using network-wide measurements". In: *IEEE Transactions on Information Theory* 52.6 (2006), pp. 2710–2724. DOI: 10.1109/TIT.2006.874414.

[22] Christian Hervé, J. Cerrai, and T. Caër. “High resolution time-to-digital converter (TDC) implemented in field programmable gate array (FPGA) with compensated process voltage and temperature (PVT) variations”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 682 (Aug. 2012), pp. 16–25. DOI: 10.1016/j.nima.2012.04.040.

[23] Santa C. Huerta et al. “FPGA based Digital Pulse Width Modulator with Time Resolution under 2 ns”. In: (2007), pp. 877–881. DOI: 10.1109/APEX.2007.357618.

[24] P.P.M. Jansweijer, H.Z. Peek, and E. de Wolf. “White Rabbit: Sub-Nanaosecond Timing over Ethernet”. In: (2012), pp. 1–10. URL: https://indico.cern.ch/event/143656/papers/1378076/files/1031-WHITE.pdf.

[25] Peter Jansweijer. *WRPC v4.2 CLBv3*. 2020. URL: https://ohwr.org/project/wr-cores/tree/peter_clbv2_3_4.

[26] J. Jasperneite, K. Shehab, and K. Weber. “Enhancements to the time synchronization standard IEEE-1588 for a system of cascaded bridges”. In: (2004), pp. 239–244. DOI: 10.1109/WFCS.2004.1377716.

[27] Yang. Kong et al. “A new design for precision clock synchronization based on FPGA”. In: (2009), pp. 411–414. DOI: 10.1109/RTC.2009.5321499.

[28] Martin Langer et al. “Analysis and Compensation of Latencies in NTS-secured NTP Time Synchronization”. In: (2020), pp. 1–10. DOI: 10.1109/IFCS-ISAF41089.2020.9234871.

[29] Martin Lévesque and David Tipper. “A Survey of Clock Synchronization Over Packet-Switched Networks”. In: *IEEE Communications Surveys Tutorials* 18.4 (2016), pp. 2926–2947. DOI: 10.1109/COMST.2016.2590438.

[30] Hongming Li. “CERN General Machine Timing System: status and evolution”. In: 2008. URL: https://indico.cern.ch/event/28233/contributions%20/1631188/attachments/519242/716376/Workshop%5C_CERN.pdf.

[31] Jiaqi Li et al. “Large Dynamic Range Accurate Digitally Programmable Delay Line with 250-ps Resolution”. In: 1 (2006). DOI: 10.1109/ICOSP.2006.345484.

[32] M. Lipinski et al. “White Rabbit Applications and Enhancements”. In: (Sept. 2018), pp. 1–7. DOI: 10.1109/ISPCS.2018.8543072.

[33] A.W. Love. “GPS, atomic clocks and relativity”. In: *IEEE Potentials* 13.2 (1994), pp. 11–15. DOI: 10.1109/45.283881.

[34] Akihiko Machizawa and Tsukasa Iwama. “Exponential degrading of NTP synchronization with number of network hops”. In: (2013), pp. 97–100. DOI: 10.1109/EFTF-IFC.2013.6702083.

[35] Leo A. Mallette, Joe White, and Pascal Rochat. “Space qualified frequency sources (clocks) for current and future GNSS applications”. In: (2010), pp. 903–908. DOI: 10.1109/PLANS.2010.5507225.

[36] D. Margalit and J. Rabinoff. *Interactive Linear Algebra*. 1st ed. Georgia: Georgia Tech, 2019. URL: https://textbooks.math.gatech.edu/ila/chap-algebra.html.

[37] W. Markowitz et al. “Frequency of Cesium in Terms of Ephemeris Time”. In: *Phys. Rev. Lett.* 1 (3 Aug. 1958), pp. 105–107. DOI: 10.1103/PhysRevLett.1.105. URL: https://link.aps.org/doi/10.1103/PhysRevLett.1.105.

[38] D.L. Mills. “Internet time synchronization: the network time protocol”. In: *IEEE Transactions on Communications* 39.10 (1991), pp. 1482–1493. DOI: 10.1109/26.103043.

[39] Faten Mkacher and Andrzej Duda. “Calibrating NTP”. In: (2019), pp. 1–6. DOI: 10.1109/ISPCS.2019.8886646.

[40] Pedro Moreira et al. “Digital dual mixer time difference for sub-nanosecond time synchronization in Ethernet”. In: (2010), pp. 449–453. DOI: 10.1109/FREQ.2010.5556289.

[41] Pedro Moreira et al. “White rabbit: Sub-nanosecond timing distribution over ethernet”. In: (2009), pp. 1–5. DOI: 10.1109/ISPCS.2009.5340196.

[42] *Mouser*. URL: https://www.mouser.com/.

[43] OHWR. *Repository for open-source WR IP cores*. URL: https://ohwr.org/project/wr-cores/wikis/wrpc-core.

[44] Rudiger Paschotta. *Chromatic Dispersion*. URL: https://www.rp-photonics.com/chromatic_dispersion.html.

[45] Sabrina Perrella. “Development of FPGA-based High-Speed serial links for High Energy Physics Experiments”. PhD thesis. Swiss: Università degli Studi di Napoli Federico II, 2013. URL: https://cds.cern.ch/record/2317008/files/getfile.pdf.

[46] T. Rahkonen and J. Kostamovaara. “The use of stabilized CMOS delay lines in the digitization of short time intervals”. In: (1991), 2252–2255 vol.4. DOI: 10.1109/ISCAS.1991.176828.

[47] Mattia Rizzi et al. “White rabbit clock characteristics”. In: (2016), pp. 1–6. DOI: 10.1109/ISPCS.2016.7579514.

[48] Mattia Rizzi et al. “White Rabbit Clock Synchronization: Ultimate Limits on Close-In Phase Noise and Short-Term Stability Due to FPGA Implementation”. In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 65.9 (2018), pp. 1726–1737. DOI: 10.1109/TUFFC.2018.2851842.

[49] Javier Serrano. “CERN General Machine Timing System: status and evolution”. In: 2008. URL: https://indico.cern.ch/event/28233/contributions%20/1631188/attachments/519242/716376/Workshop%5C_CERN.pdf.

[50] Natasa Simanic et al. “Compensation of asymmetrical latency for ethernet clock synchronization”. In: (2011), pp. 19–24. DOI: 10.1109/ISPCS.2011.6070151.

[51] Eugene Y. Song and Kang Lee. “An application framework for the IEEE 1588 standard”. In: (2008), pp. 23–28. DOI: 10.1109/ISPCS.2008.4659207.

[52] Leonardo Trigo and Daniel Slomovitz. “Rubidium atomic clock with drift compensation”. In: (2010), pp. 472–473. DOI: 10.1109/CPEM.2010.5544397.

[53] *UltraFast Design Methodology Guide for Xilinx FPGAs and SoCs*. UG949. v2022. Xilinx. 2022. URL: https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology/Using-Clock-Modifying-Blocks-MMCM-and-PLL.

[54] *UltraScale Architecture PCB Design User Guide*. UG583. v1.24. Xilinx. 2022. URL: https://docs.xilinx.com/r/en-US/ug583-ultrascale-pcb-design/UltraScale-Architecture-PCB-Design-User-Guide.

[55] *Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC Libraries Guide*. UG953. v2022.2. Xilinx. 2022. URL: https://docs.xilinx.com/r/en-US/ug953-vivado-7series-libraries.

[56] Hai Wang, Min Zhang, and Jie Liu. “High-resolution short time interval measurement system implemented in a single FPGA chip”. In: *Chinese Science Bulletin* 56.12 (Apr. 2011), pp. 1285–1290. ISSN: 1861-9541. DOI: 10.1007/s11434-011-4421-3. URL: https://doi.org/10.1007/s11434-011-4421-3.

[57] Tomasz Wlostowski. “Precise time and frequency transfer in a White Rabbit network”. thesis. Warsaw University of Technology, 2011. URL: https://white-rabbit.web.cern.ch/documents/Precise_time_and_frequency_transfer_in_a_White_Rabbit_network.pdf.

[58] Tomasz Wlostowski. *Why the SoftPLL is not a HardPLL....* 2019. URL: https://ohwr.org/project/wr-cores/wikis/Documents/Why-the-SoftPLL-is-not-a-HardPLL.

[59] J.S.J. Wong. “Delay Measurements and Self Characterisation on FPGAs”. PhD thesis. London: Imperial College London, 2011. URL: https://spiral.imperial.ac.uk/bitstream/10044/1/6434/1/Wong-JSJ-2011-PhD-Thesis.pdf.

[60] Min Zhang et al. “Phase Difference Measurement Method Based on Progressive Phase Shift”. In: *Electronics* 7 (June 2018), p. 86. DOI: 10.3390/electronics7060086.

A

Operation of White Rabbit components

Time offset calculations depend on the measurements and adjustments in the White Rabbit devices (e.g., phase measurements, counters and time-stamping). Most of these functionalities are provided or controlled by internal RTL components. This section discusses the common RTL components of the ordinary clock design (node) and the boundary clock design (switch).

The ordinary clock design represents the set of RTL components to form a single-port core that executes the White Rabbit protocol. Single-port cores cannot only represent slave devices, but can fulfil both the function of grand-master and master upon correct configuration. The boundary clock design represents the set of RTL components to form a multi-port core, which primarily contains similar components for the White Rabbit protocol as the single-port core design. The components can be grouped into two classes, based on their contribution. The operation of the phase measurement and correction class and Ethernet class, will be highlighted in subsequent sections, as these induce properties that are useful for the objective of this thesis.

A.1. Phase measurement and correction class

The phase measurement and correction class in White Rabbit, is responsible for fine delay measurements in master mode and provides fine delay compensation and syntonization in a slave configuration. In master configuration, the master recovered clock (4) from Figure 2.14 will be extracted from the data arriving at the master and compared to the reference clock via a phase measurement. When a slave is considered, the present oscillator will be syntonized to the received slave recovered clock (2) in Figure 2.14. After the execution of the White Rabbit protocol, the slave recovered clock (2) will be shifted to a phase-shifted clock (3), to align with the reference clock. When a boundary clock design is considered, the input port is in slave mode and distributes the phase-shifted clock to each output SFP module. For each of the master ports, the phase of the recovered clock has to be compared to the reference clock. Therefore, the phase measurement and correction class performs all three tasks at the same time: Namely, fine delay measurement, syntonization and fine delay compensation. These functionalities are aided and controlled by the software-based PLL and LM32 processor.

The software-based PLL (softPLL) is the core component of the White Rabbit synchronization process. It represents the exact difference between regular PTP and White Rabbit: Namely, fine delay measurement and fine delay compensation. The control of the component is handled by an LM32 microprocessor. Additionally, the phase measurement and correction class consists of two external oscillators (VCO), a PLL and digital-to-analog converters(DAC). Figure A.1 shows

an overview of the softPLL. The green set is used for syntonization and fine delay compensation. The orange set is used as a reference for the fine delay measurements (expressed in phase difference). The gray area with phase difference measurements and LM32 reside inside the FPGA fabric.

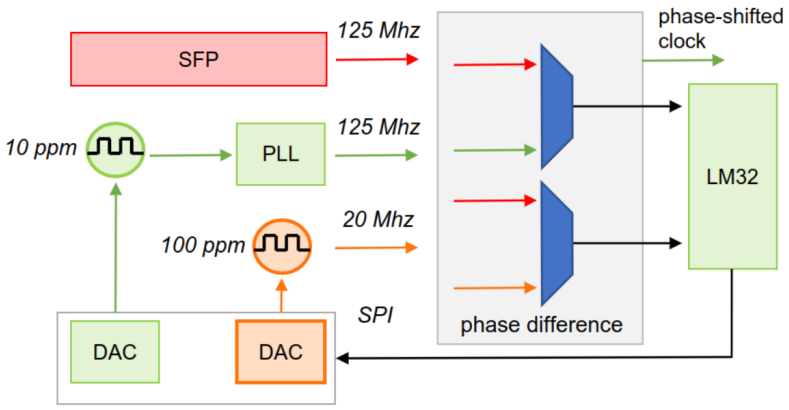


Figure A.1: Overview of the White Rabbit software-based PLL. The red clock set represents the extracted clock from the SFP. This can be the master recovered clock in master mode or the slave recovered clock in slave mode. The green set (Digital-to-analog converter, VCTCXO oscillator and PLL) present the clock that is used in the White Rabbit instance. This can either be the used reference clock in master mode, or the phase-shifted clock in slave mode. The oscillator has a 10 parts-per-million pullability range, which can be controlled through the DAC and serial peripheral interface (SPI). The oscillation frequency of the oscillator can therefore be tuned. The orange set represents a separate clock that is used as a reference for the phase measurements and can be tuned over a 100 ppm range. The gray set and LM32 measure and control each of the hardware components and reside fully inside the FPGA fabric

In contrast with a regular PLL, the phase difference between clock signals is not expressed in a voltage, but rather a digital value. A simple LM32 processor, processes the digital phase difference value and calculates an adjustment value for the DAC. This can be used to reach syntonization. After all, if one keeps the measured phase difference constant, the clocks are syntonized.

However, synchronization is desired and a one-way phase difference between the slave and master clock is extracted from timestamps and phase measurements. The LM32 can calculate what the value of the phase difference should be to compensate the acquired value from the fine delay measurement. A PI controller in the LM32 software, calculates the current digital value for the oscillator to gradually increase or decrease frequency, to alter the phase.

The motivation to construct a software-based PLL instead of a hardware-based PLL stems from the fact that hardware is harder to debug. In the current implementation, the controls are completely in hands of the LM32. Similarly, a software-based implementation appears to take less physical space and provides better controls over the complex algorithms that handle the phase shifts[58].

A.2. Ethernet class

White Rabbit uses gigabit optical Ethernet and the components that provide that service are part of the Ethernet class. The class can be composed of five main elements as can be seen in Figure A.2: the physical layer transceiver (PHY), the endpoint, the fabric interface, the mini-NIC and the user code.

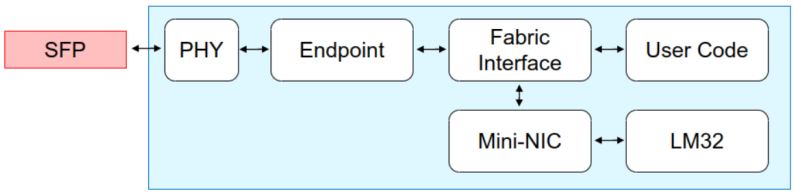


Figure A.2: Overview of the main components in the Ethernet class. Namely, the physical layer transceiver (PHY), the endpoint, the fabric interface, the mini-NIC and the user code.

The mini-NIC communicates with the LM32 microprocessor, as it constructs White Rabbit packets, or actually PTP packets. The LM32 places messages in a dual-port memory after which the mini-NIC constructs the network packet. However, not only White Rabbit packets can be sent through the network: Namely, user traffic may enter via the user code interface. Both instances, the mini-NIC and the user code, have to assert a high signal to the fabric interface. It analyzes the priority of the presented message, which normally means that user traffic is favoured over White Rabbit traffic. The fabric interface also receives packets, which are sent through to either the mini-NIC or the user code, depending on the packet type. After the fabric interface, the endpoint is reached. The endpoint chops the packets up into transmittable bits or constructs Ethernet frames upon receiving bits. Additionally, the endpoint provides the critical task of timestamping incoming packets. Subsequently, the PHY transceivers provide extraction of a clock signal and (de)serialization of data. Both the serialized data signal and clock from the endpoint, are offered to the SFP modules.

B

Preliminary hardware analysis of White Rabbit designs

The performance of the synchronization in terms of accuracy and precision, are directly related to the accuracy and precision of the physical implementations on the FPGA boards. Therefore, preliminary experiments were set up to implement and test the White Rabbit node on Artix-7 FPGA's to support the discussion on the White Rabbit protocol. Subsequently, a small summary is presented stating the required hardware and approximated costs for the White Rabbit node design. Lastly the boundary clock design (switch) is analyzed for which a similar hardware table is provided.

B.1. Ordinary clock design

To gain understanding about the ordinary clock design and perform tests on the newly-designed boards, a White Rabbit node design was built for the Topic boards based on the Artix 7 CLBv3 reference designs from Nikhef [25], the Spartan 6 reference design [16] and the compact CUTEWR reference design [19][30]. A custom Board Support Package (BSP) was build according to the White Rabbit abstraction hierarchy. It includes the Platform Support Package (PSP) constructed for Artix-7 devices specifically, in combination with the common VHDL modules for every White Rabbit instance. The composed set represents the core of the White Rabbit functionality, accompanied by a minimal standard software implementation for the LM32 micro-controller.

B.1.1. Utilization

Table B.1 summarizes the hardware utilization of a single-port White Rabbit implementation on a Topic Artix-7 board. Due to the SFP module, one of the four transceivers is occupied. Similarly, it is seen that utilization of BRAM is at 76 %. This may be due to the large presence of calibration parameters that are written once and read out during boot. Similarly, large dual-port ram is used in between the mini-NIC and LM32 processor, as is FIFO BRAM in between different clock domains.

Table B.1: Utilization report from Vivado 2022.2 of the White Rabbit node design for Topic. Note the large percentage of used BRAM elements. Similarly, the GTPE2 percentage corresponds with the use of one transceiver for the single SFP (4 transceivers are available). The utilization includes an Integrated Logic Analyzer, which does not exceed 0.80 % of each of the utilization percentages.

Type	**Usage** (%)
Slice LUTs	24.13
Slice Registers	13.95
Slice	27.09
Block RAM	76
GTPE2	25

B.1.2. Database

During testing, it could be verified that White Rabbit relies on calibration sets from CERN's database. In this configuration, an unsupported SFP module was plugged in the SFP socket. The initialization process could not find a matching ID in the calibration database, throwing an error and stopping the White Rabbit initialization process (See Figure B.1). Similarly, the process looks for t24p calibration parameters. These are read from the external EEPROM memory and required for the calibration of the ϕ_{trans} parameter. This means it falls back on pre-calibrated values.

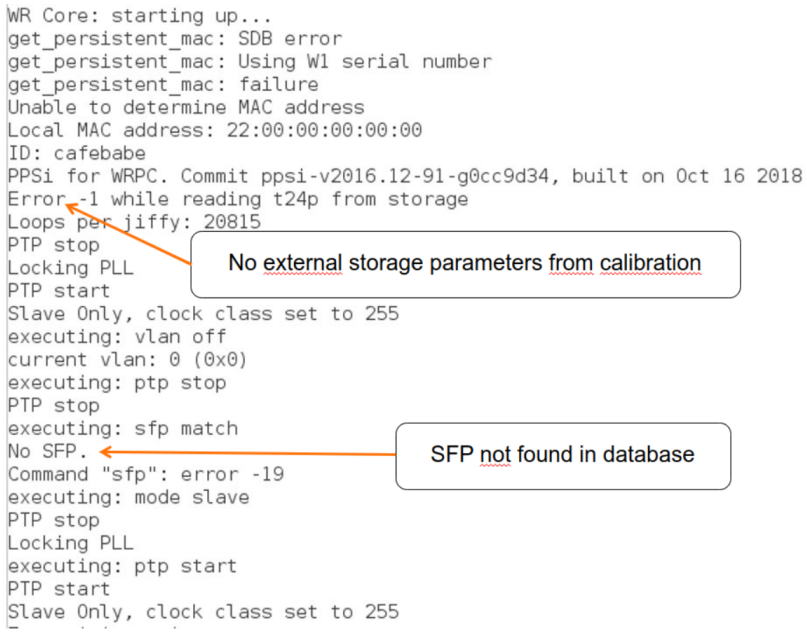


Figure B.1: Screenshot from the UART terminal connected with the White Rabbit node design, modified and running on the Topic Artix-7 board. Note that the system cannot find calibration parameters for the currently plugged in SFP module, from previous calibrations or in the database. Hence, it is not calibrated at all!

B.1.3. Hardware components

This section provides a list for the hardware components needed to construct a White Rabbit node. The Topic boards are considered to provide the hardware platform for the White Rabbit nodes. Hence, the estimated costs are approximated based on components from that board in combination with the prices from Mouser [42]

Table B.2: Main hardware components of an ordinary White Rabbit clock

Component	**Function/type**	**Approximated cost** ($)
VCTCXO	Voltage to 25 MHz	30
PLL	25 to 125 MHz	15
VCXO	20 MHz (phase reference)	15
DAC	2 * SPI to voltage	15
FPGA	Artix-7	50
EEPROM	Storage parameters	5
Passive components	Connectors, buttons, etc.	10
SFP		15
Assembly + PCB	Production	50
Total costs		**205**

B.2. Boundary clock design

B.2.1. Utilization

Table B.3 summarizes the hardware utilization of a multi-port White Rabbit switch implementation on a Virtex-6 FPGA (XC6VLX240T).

Table B.3: Utilization overview of WRS-3 on the Virtex-6 FPGA (XC6VLX240T). Adapted from Seven Solutions

Type	**Available**	**Usage** (%)
Slice LUTs	150K	24.13
FF	301K	13.95
Block RAM	14.9 Mb	76
GTPE2	24	75

B.2.2. Hardware components

This section provides a list for the hardware components needed to construct a White Rabbit switch. The component estimations are extracted from the current leading implementation from

Seven Solutions (Latest version v3.4) [7]. Hence, the estimated costs are approximated based on components from that board in combination with the prices from Mouser [42]

Table B.4: Main hardware components of the White Rabbit boundary clock (rev. 3.4)

Component	**Function/type**	**Approximated cost** ($)
VCTCXO	Voltage to 25 MHz	30
PLL	25 to 125 MHz	15
VCXO	20 MHz (phase reference)	15
DAC	2 * SPI to voltage	15
FPGA	Virtex 6 (XC6VLX240T)	3500
CPU	Offloads the FPGA (ARM)	15
EEPROM	Storage parameters	5
SFP	18 * SFP	270
Passive components	Connectors, buttons, etc.	20
Assembly + PCB	Production	50
Total costs		**3935**

C

Supporting schematics, tables and simulations

C.1. Digital Dual Mixer Time Difference

This section provides waveforms and tables to support concepts of the main text with more detail

C.1.1. Table: Comparison of the resolutions of external oscillators and internal PLLs

This table illustrates the difference in resolution between the external oscillator solution and the internal double PLLs [30]:

Table C.1: Comparison of the DDMTD measurement resolution with two external oscillators (typically implemented with White Rabbit devices) and a set of cascaded internal PLL's in Xilinx Spartan 6 technology (XC6SLX45T-4CSG324C)

	VCXO (100 ppm)	**VCXO** (122 ppm)	**Hongming[30]**
Resolution (ps)	0.977	1.953	3.906
n	16384	8192	4096
Input clock (MHz)	62.5	62.5	62.5

C.1.2. Simulation: The operation of a DDMTD

To illustrate the operation of a DDMTD through simulation, an example waveform can be seen in Figure C.1. It can be seen that the under-sampling frequency is equal to the difference frequency between clk_1 and clk_{DDMTD}. Namely, the clk_{DDMTD} edge shifts with the difference period time (t_{DDMTD} = 1ns) compared to the edges of the sampled clocks. The resolution of the phase measurement therefore depends on the frequency difference between the sample clock and input clock [40].

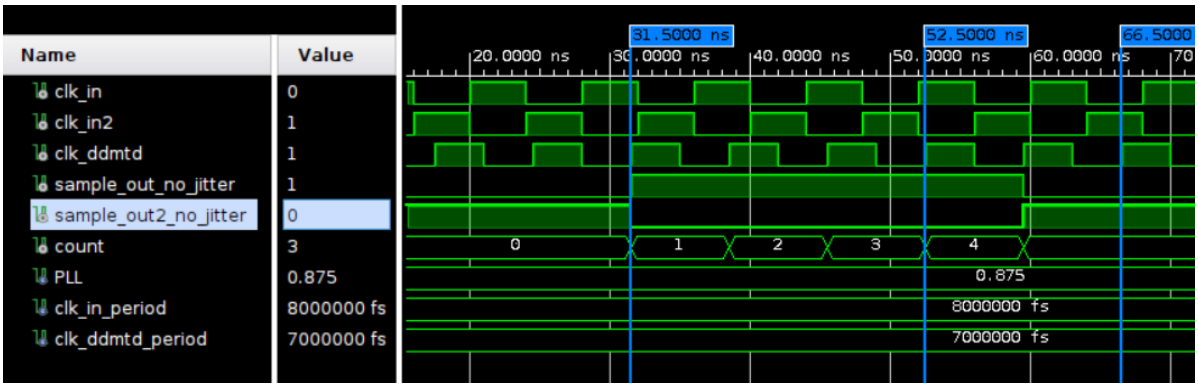


Figure C.1: A simulation waveform with clk_1 = 8ns, clk_{DDMTD} = 7ns and a phase difference of 4ns. The blue markers denote several sampling points. By counting the high registered samples of sample_out ($clk1$ sampled) before the sample_out2 ($clk2$) samples are high, one denotes that the result is 4 samples, as can be seen in the value count. Therefore, the phase difference can be calculated as 4ns.

To obtain the maximum resolution (as defined by Hongming et al. [30]), a n-value of 4096 is applied. The difference in period between the incoming clock clk_a (8 ns) and the DDMTD clock (8.0018 ns). Hence, the under-sampling happens at steps of 1.8 ps, which defines the resolution of the implementation. A phase difference of 3888 * 1.8 indicates 6.9984 ns where in fact 7.0000 ns difference was set. The error is 1.6 ps in simulation, which is not realistic considering that jitter-less clock signals are simulated.

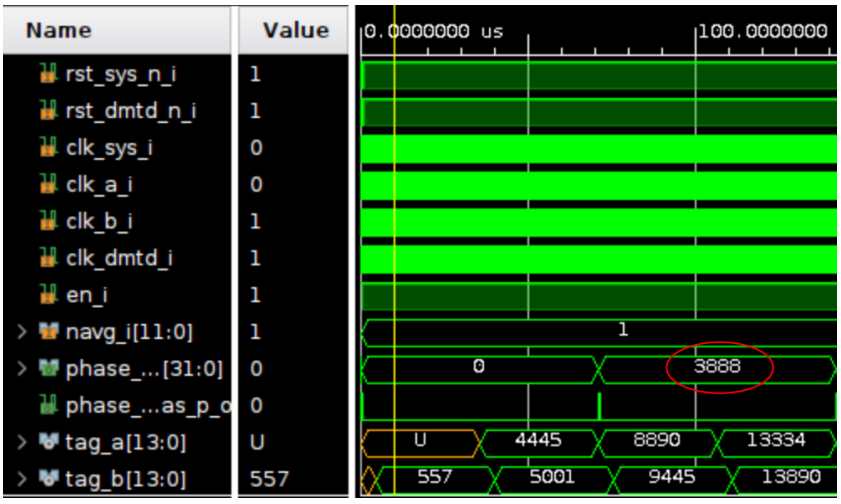


Figure C.2: A simulation waveform with clk_1 = 8ns, clk_{DDMTD} = 8.0018ns and a phase difference of 7.0000ns. The red marker shows the amount off counted samples of the ddmtd clock. It corresponds to 1.8 * 3888 = 6.9984 ns phase difference, which means that there is a 1.6 ps error. However this example is not realistic, as jitter-less clock signals are considered.

C.1.3. Simulation: Phase jitter on the input and sampling clocks

An example can be seen in Figure C.3, where phase jitter was placed on the DDMTD clock (5% of the period) and on the input clock signals (1% of the period). Without deglitching and averaging the phase measurements, the resulting phase can be inconclusive.

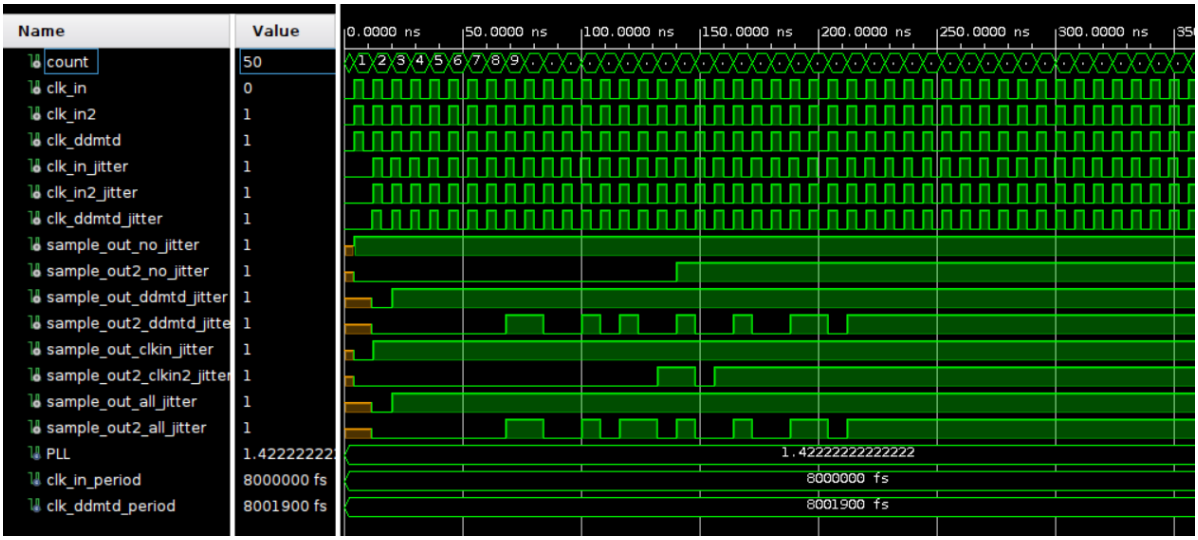
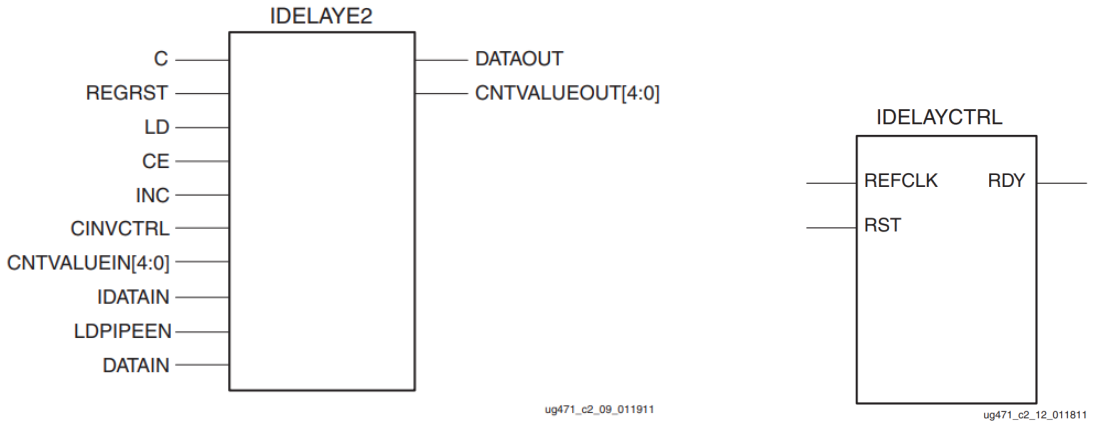


Figure C.3: A simulation waveform to illustrate the instability in the DDMTD with jitter addition on clock signals. The DDMTD is formed with single sampling flipflops and without presence of deglitching logic. The clock DDMTD is set with $n = 4096$ and is simulated with 5% phase jitter from its period. The input clocks are simulated with 1% jitter noise from the input clock. The phase difference between the clocks is 30 ps. Different versions of sample_out compare no jitter, jitter on the DDMTD clock, on the input clocks or on both.

C.2. IO Delay

C.2.1. Schematic: The block diagrams of an IDELAY2

A block diagram of the inputs and outputs of the IDELAY2 and IDELAYCTRL primitives can be found in Figure C.4a and C.4b.



(a) IDELAY2 primitive

(b) IDELAYCTRL primitive

Figure C.4: Block diagrams of IDELAY2 and IDELAYCTRL primitives [3]

C.2.2. Graph: One-way transmission delay measurements

D

Measurements and specifications for verification

D.0.1. Table: Trace delay specifications

Table D.1: Trace delay specifications from Altium and specifications sheets.

Connection	**Type**	**Length** (mm)	**Delay** (ps)
Between FPGA's	LVDS	62.535	437.444
FPGA to Mezzanine	LVCMOS	32.185	216.291

D.0.2. Graph: One-way transmission delay measurements

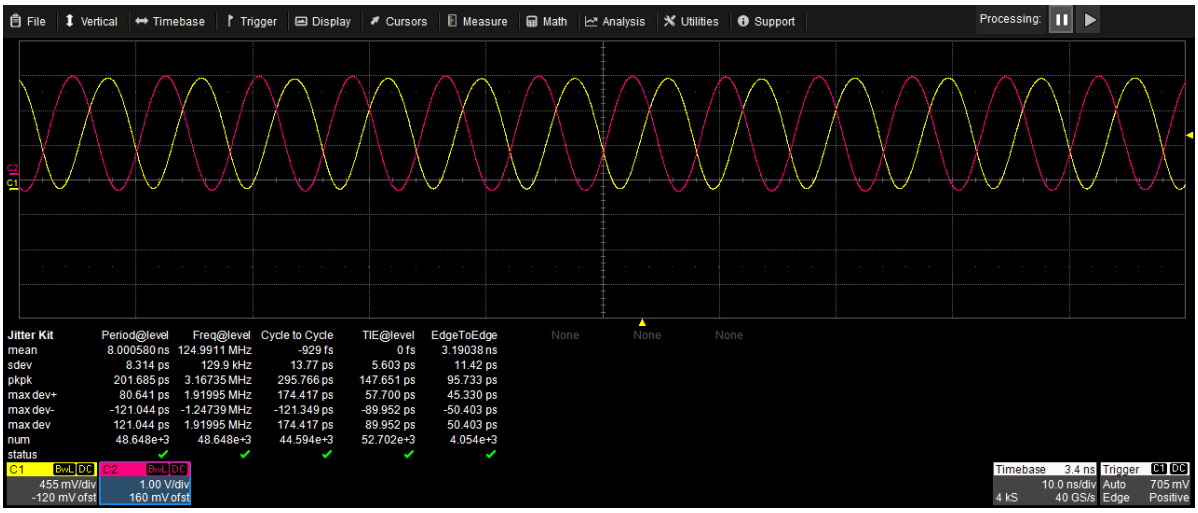


Figure D.1: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 1. The direction: the slave is measured at its oscillator, and the master is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

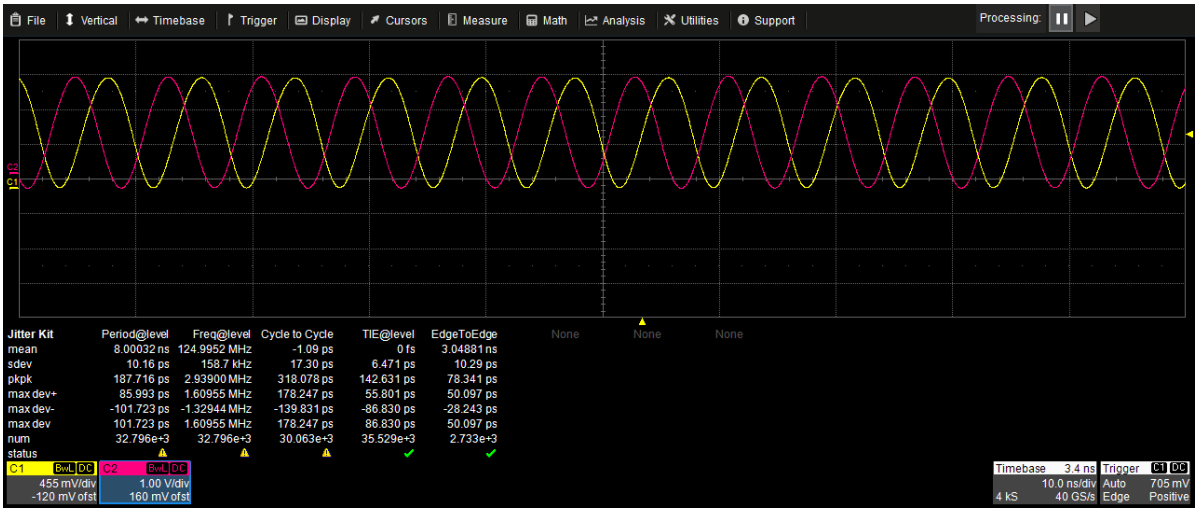


Figure D.2: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 2. The direction: the slave is measured at its oscillator, and the master is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

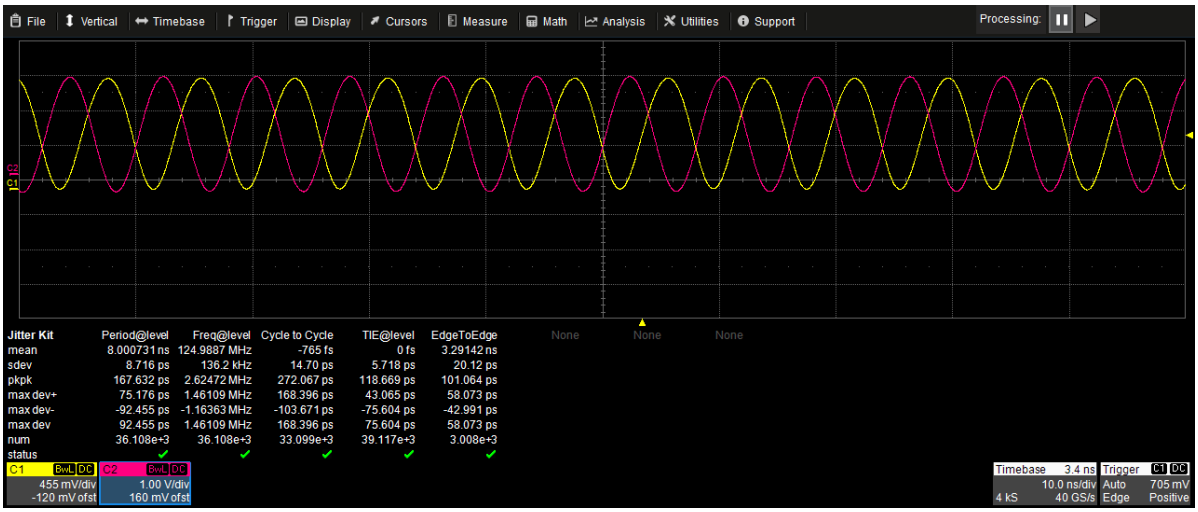


Figure D.3: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 3. The direction: the slave is measured at its oscillator, and the master is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

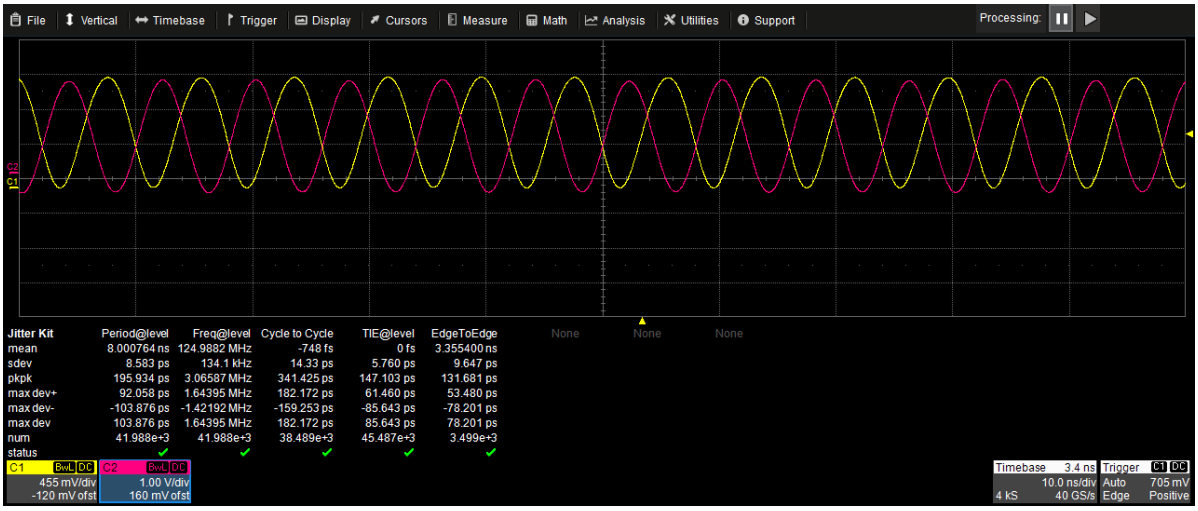


Figure D.4: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 4. The direction: the slave is measured at its oscillator, and the master is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

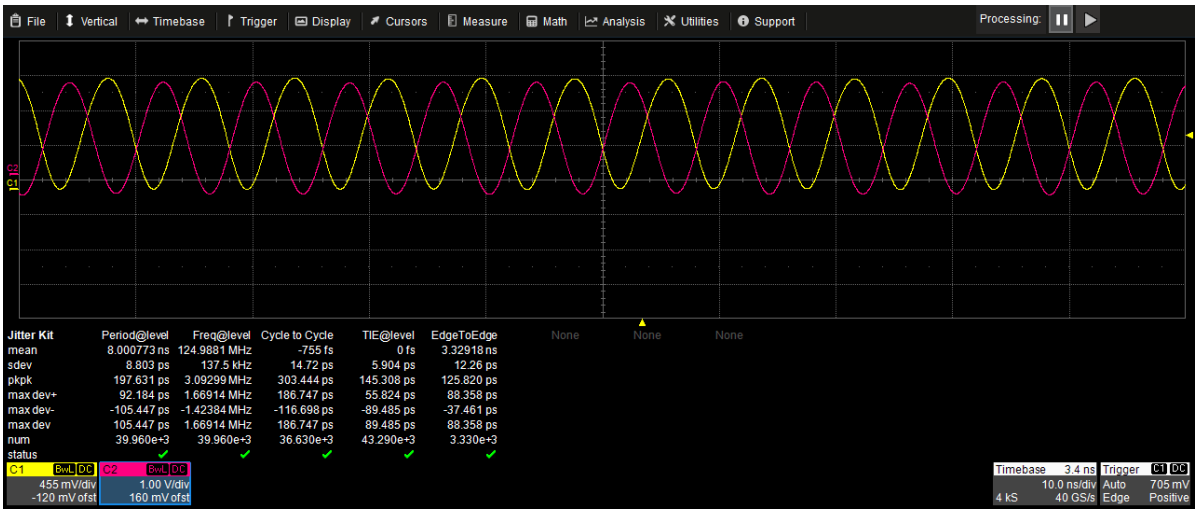


Figure D.5: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 5. The direction: the slave is measured at its oscillator, and the master is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

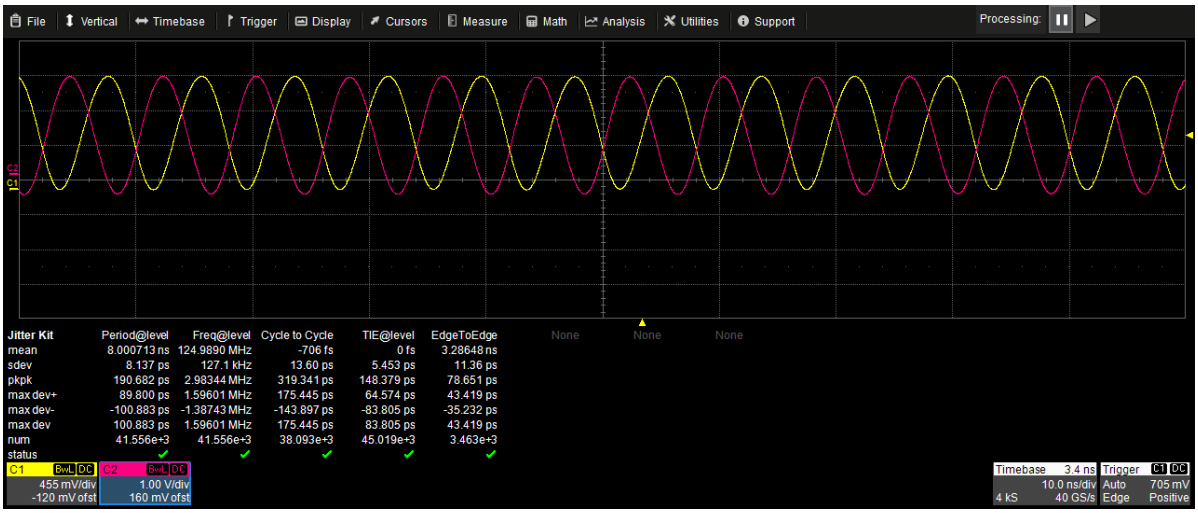


Figure D.6: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 1. The direction: the master is measured at its oscillator, and the slave is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

Figure D.7: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 2. The direction: the master is measured at its oscillator, and the slave is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

Figure D.8: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 3. The direction: the master is measured at its oscillator, and the slave is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

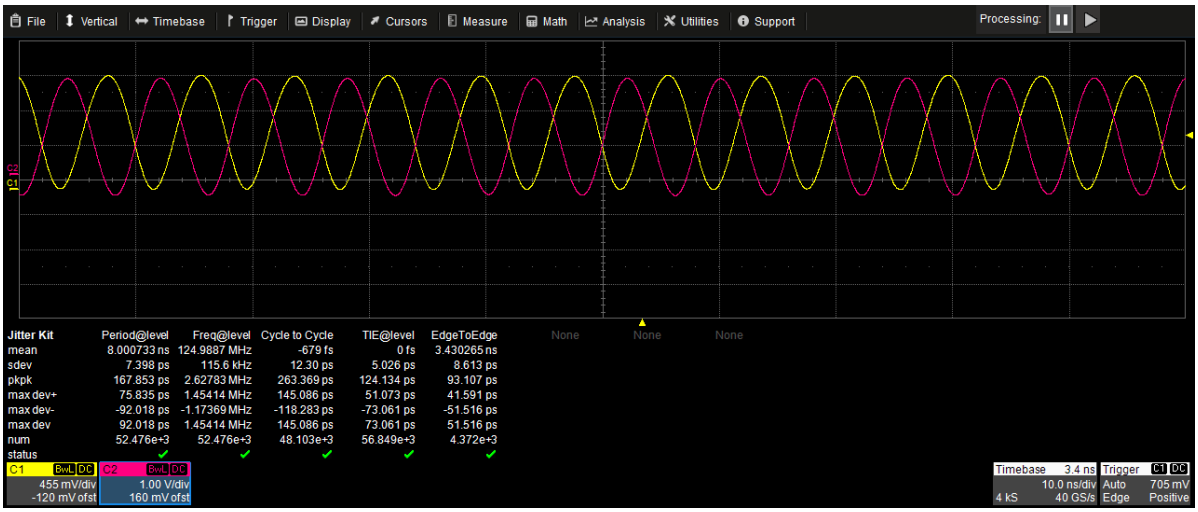


Figure D.9: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 4. The direction: the master is measured at its oscillator, and the slave is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge

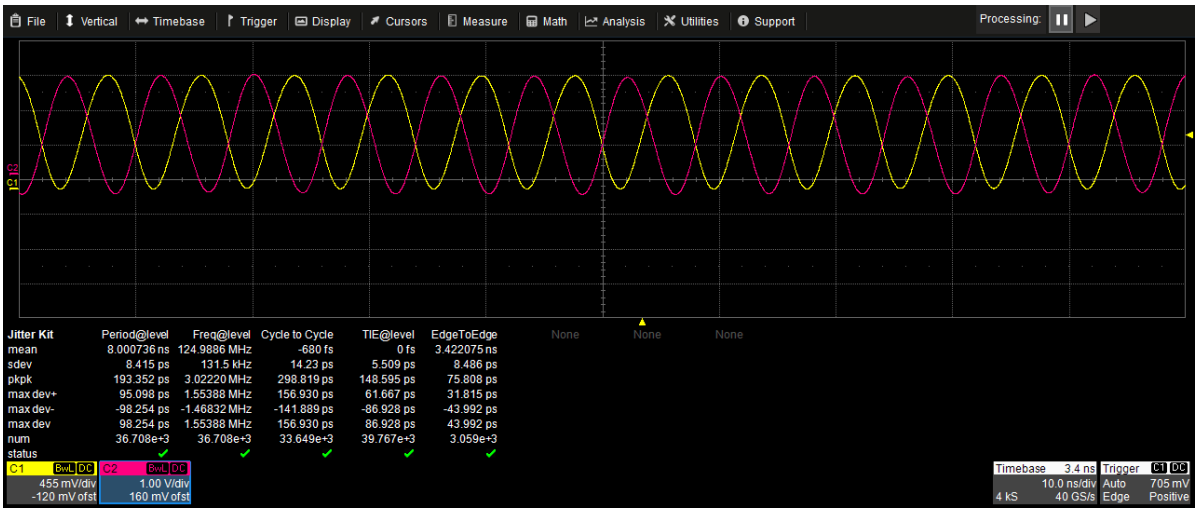


Figure D.10: A screenshot from the Teledyne LeCroy Oscilloscope of type waveMaster 813Zi-B. The transmission delay is measured on line 5. The direction: the master is measured at its oscillator, and the slave is measured at its via on the single-ended line. The bandwidth was limited to 200 MHz on both input channels. The transmission delay is expressed by the variable EdgeToEdge