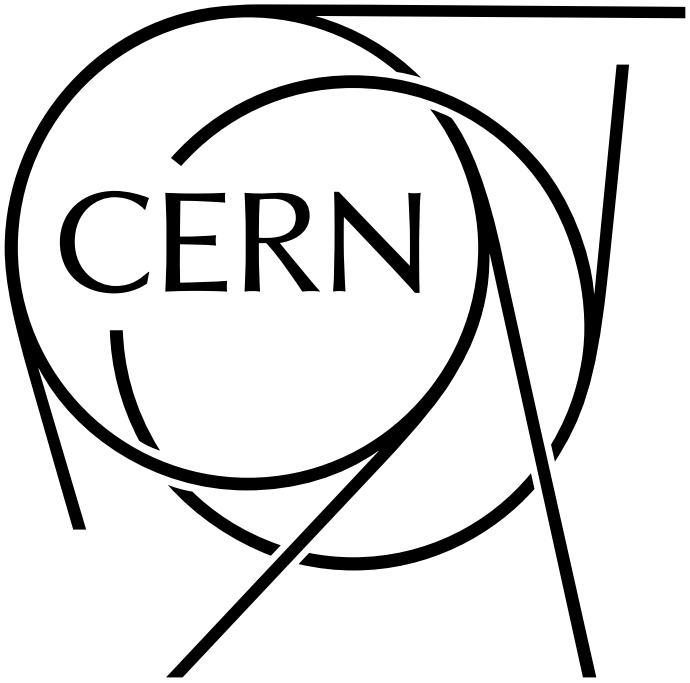
Packet Based Time Synchronisation



Project Report

Group 1073

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg Ø
http://es.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Packet Based Time Synchronisation

Theme:
Time Synchronisation

Project Period:
Spring Semester 2021

Project Group:
Group 1073

Participant:
Magnus Bøgh Borregaard Christensen

Supervisors:
Petar Popovski
Anders Ellersgaard Kalør
Tomasz Podzorny (CERN)

Page Numbers: 70

Date of Completion:
June 17, 2021

Abstract:

This thesis concludes a nine months research, design and evaluation of a packet based synchronisation solution for a new data acquisition system being developed by the TE-MPE-EP section at CERN. Prior to this thesis, a synchronisation test system was developed applying the IEEE 1588 standard to measure the time difference between two clocks in a network. In this thesis the causes of time drift between disjoint clocks is thoroughly analysed theoretically. Subsequently, the theory is applied to identify and quantify the noise sources present in our test system. Based on these measurements, it was found that quantisation in the IEEE 1588 timestamping logic was the main source of error. Additionally, when exposed to high network load, timestamping is distorted by irregularly occurring outliers. Finally, to implement synchronisation, a rudimentary control solution using the inferred time offsets between the system clocks was added. This control proved effective and demonstrated highly accurate synchronisation with a time error in the order of 100's of nanoseconds is attainable; meeting the design objectives. The results of the thesis will be implemented in the coming months.

Abbreviations

ADEV Allan Deviation

AVAR Allan Variance

CERN The European Organisation for Nuclear Research

CTR Central Time Receiver

DQAME Data Acquisition Board

FDPLL Fractional Digital Phased Locked Loop

FEC Front End Computer

FM Frequency Modulated

HL-LHC High Luminosity Large Hadron Collider

LHC Large Hadron Collider

MAD Median Absolute Deviation

MTIE Maximum Time Interval Error

NIC Network Interface Card

PDV Packet Delay Variation

PI Proportional-Integral

PM Phase Modulated

PPS Pulse Per Second

PTP Precision Time Protocol

TDEV Time Deviation

TE-MPE-EP Technology Department, Machine Protection and Electrical Group, Electronics for Protection Section

TIE Time Interval Error

TSU Time Stamping Unit

Contents

Chapter 1

Introduction

The European Organisation for Nuclear Research (CERN) is in the process of developing the technologies required to upgrade their state-of-the-art particle accelerator, the Large Hadron Collider (LHC). This upgraded machine, known as the High Luminosity Large Hadron Collider (HL-LHC) aims to increase the number of particle collision (formally referred to as luminosity) by a factor of 10 with the goal of increasing the number of observations of rare particle phenomena. While the actual installation is expected to commence in 2027, the various engineering sections at CERN are already diligently working towards this milestone.

One aspect of this upgrade involves the modernization of the data acquisition system used for monitoring the LHC's quench detection systems. Currently this system is based on an old Fieldbus network, and bandwidth limited to 1 kbit/s. To bring the data acquisition system into the 21st century, the TE-MPE-EP section at CERN is working on a replacement based on Ethernet and IP traffic. It is anticipated that this approach will improve data acquisition in several key areas, including bandwidth, synchronisation accuracy, and network flexibility.

While the ultimate goal of the new data acquisition system is the deployment in HL-LHC for the purpose of quench detection monitoring, there are other activities at CERN that may benefit from the system. One of these is CERN's state-of-the-art superconducting magnetic test facilities, known as SM18. To perform testing, SM18 uses the equipment developed for quench detection to probe and measure the voltage across magnets. SM18 wishes to take advantage of the new data acquisition system to improve test flexibility and allow for easy and high precision synchronisation of different test equipment. It is currently envisaged that deployment of this new system in SM18's test facilities will begin in the summer of 2021 with the time synchronisation component from the work covered in this thesis. The adoption of the acquisition system within SM18 presents a great opportunity to test, validate, and mature the system before installation in HL-LHC.

The development and deployment of the new data acquisition system is a sizeable undertaking that requires a broad and multifaceted development effort. Aspects include areas such as hardware development of the acquisition cards, protocol development, methods to store the acquired data, and many more. This thesis focuses on the development and implementation of synchronisation techniques for the data acquisition system's acquisition cards.

Synchronisation of the nodes in the data acquisition system is critical to establish coherent collection of data from multiple different nodes in the system. Furthermore, to

ensure compatibility with CERN's infrastructure, and allowing for comparison with other timestamped data collected at CERN, the nodes in the system should be synchronised to CERN's central time network.

Key background work into hardware design, choice of synchronisation technology, overall system setup, and the experimental setup are documented in the previous interim report [1]. This thesis builds on the foundation of the previous work by thoroughly documenting the system noise, and taking steps to implement synchronisation techniques on the data acquisition cards.

The thesis is structured as follows. First, the need for synchronisation to counter clock errors is explained in chapter 2. Subsequently, fundamental background knowledge on clock error sources is detailed in chapter 3. Chapter 4 goes on to describe practical methods that allows for the time offset between two clock to be measured. Chapter 5 introduces the existing system setup, as well as the requirements and constraints that the system must adhere to. This chapter is inspired by the contents of the previously written interim report. Following, chapter 5 goes on to identify and quantify key noise present in the system when no control is applied. This leads into chapter 6, where control is applied and the quality of the attained synchronisation is measured. Finally, chapter 8 concludes the thesis.

Chapter 2

Problem

This chapter explores how oscillators do not track time perfectly, and why synchronisation is necessary to ensure the accuracy of long term time tracking.

Initially, an ideal oscillator is considered [2, p. 1]:

$$v(t) = V_0 \cos\left(2\pi(f_0 t + \psi_0)\right) \tag{2.1}$$

where:

$v(t)$ models an ideal sinusoidal oscillator

t is elapsed time

V_0 is the amplitude

f_0 is the nominal frequency

ψ_0 is the initial phase at $t = 0$

The above model models the signal level of $v(t)$, and does not inherently track time. To obtain a metric for elapsed time, the phase of $v(t)$ must be considered:

$$f(t) = f_0 t + \psi_0 \tag{2.2}$$

where:

$f(t)$ is the phase of $v(t)$

Here $f(t)$ is continuous function of time that can take on any value between $[-\infty, +\infty]$. It represents the unwrapped phase of $v(t)$ and should not be confused with wrapped phase which is constrained to $[0, 2\pi)$ (or in this case $[0, 1)$ as $f(t)$ is normalized with respect to 2π). As phase is directly proportional with time, the two units are effectively synonymous.

The above model models an ideal oscillator, however real-life clock implementations are imperfect and subject to noise, impacting their time and frequency accuracy. These imperfections cause a variety of different deterministic and random fluctuations, modelled by substituting ϕ_0 in equation (2.1) with a random process, $\phi(t)$ [2, p. 2] [3, p. 2]:

$$x(t) = \frac{f_0 t + \psi(t)}{f_0} = t + \phi(t) \tag{2.3}$$

where

$x(t)$ is a realistic model of $f(t)$. It is normalized with respect to operating frequency f_0.

$\psi(t)$ is a random process modelling all phase deviations from ideal time source, $f(t)$.

$\phi(t) = \frac{\psi(t)}{f_0}$. I.e. it is the frequency normalized $\psi(t)$.

$\phi(t)$ causes a clock to deviate from the ideal, thus necessitating periodic re-synchronisation to maintain accuracy over long-time scales. Figure 2.1 illustrates this principle.

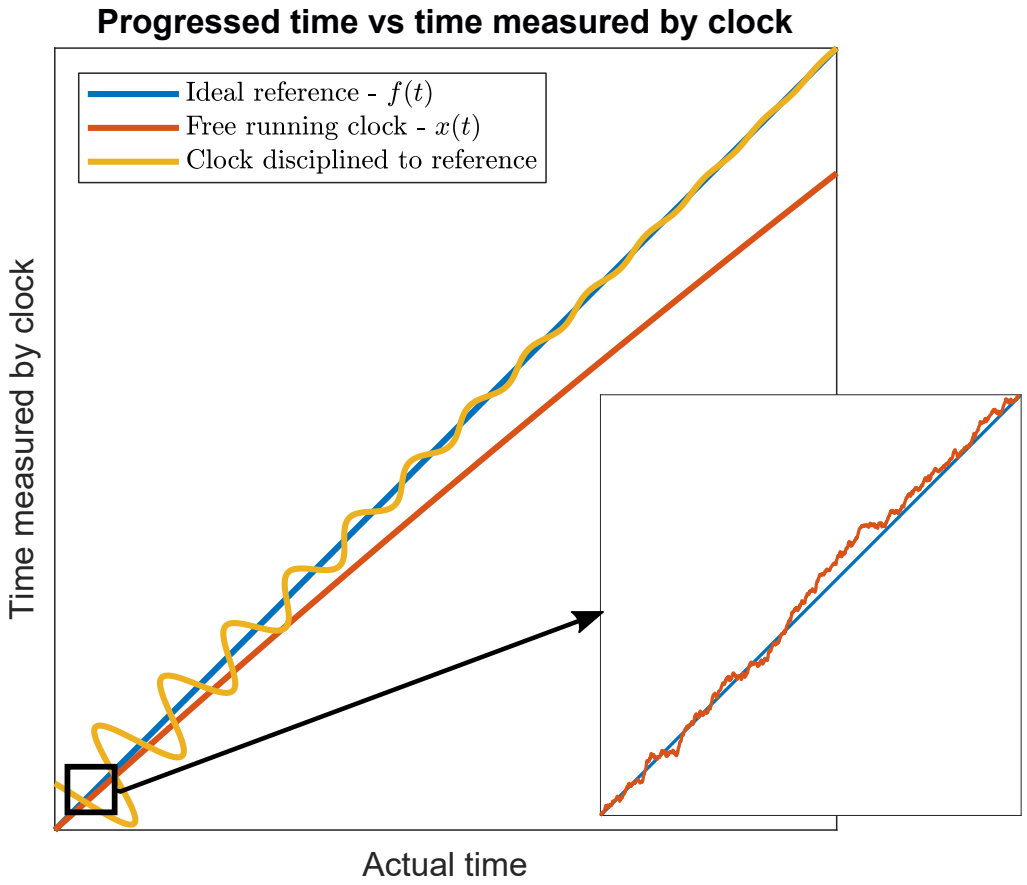


Figure 2.1: Illustration of a real clock deviating from an ideal reference, and an example of a synchronised time signal.

As seen in figure 2.1 synchronisation is achieved by continuously changing the clock frequency of the clock to try and match that of the reference. Such synchronisation requires the use of some kind of control system that can steer the clock towards the reference. This is achieved using a control loop, as shown in figure 2.2.

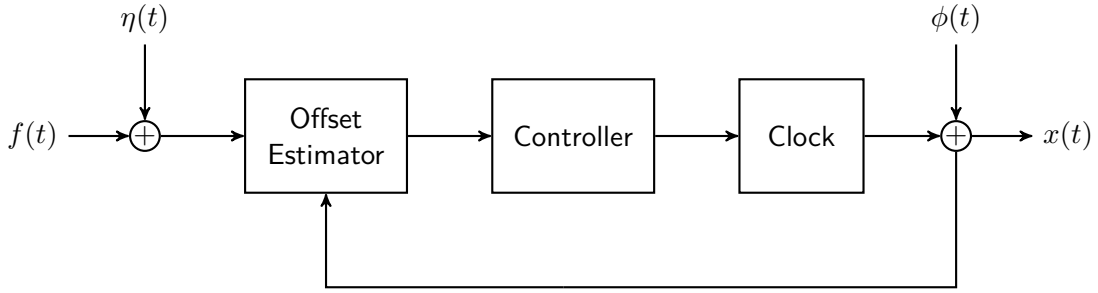


Figure 2.2: Block diagram of a controller used for synchronising a clock to a reference. $\eta(t)$ is propagation/measurement noise introduced when measuring the value of $f(t)$.

This loop is composed of three distinct parts:

1. A method to estimate how much the clock under test deviates from the reference.
2. A controller that calculates a new target frequency of the clock.
3. A clock with a variable frequency.

Theoretically, such a synchronisation scheme would make it possible to counter all the fluctuations introduced by $\phi(t)$. However, to achieve this would require the use of a control loop with infinite bandwidth. In practice, clock synchronisation uses discrete control and the controllable bandwidth of $\phi(t)$ is limited by factors such as the synchronisation period (sample frequency), the degree of control over the clock rate, as well as the accuracy of the offset estimate between the real and reference clock. The value and behaviours of $\eta(t)$ limits the offset accuracy of the offset estimate. This is the noise that is present when measuring the values of $f(t)$. The type and amount of noise primarily relies on the measurement method. This is technology dependent and varies greatly. It has previously been decided to use the Precision Time Protocol (PTP) to facilitate the timestamping and transport of $f(t)$ to the control loop logic located in the slave clock. This relies on packet based networking and as such is subject to standard traffic jitter that impacts the measurement accuracy of $f(t)$. PTP was chosen as it is a mature and well proven technology. Further, it is supported by industry and can facilitate sub microsecond synchronisation. While a PTP system is specifically considered in this thesis, all the presented theory is implementation agnostic and should apply to other technologies.

Chapter 3

Clock Noise Error Sources

In this chapter, the clock error sources are described. First, the theoretical aspects of various deterministic and stochastic error sources are covered. Following, more practical considerations regarding digital implementations and measurements noise is covered.

3.1 Theoretical Aspects

Generally it can be said that $\phi(t)$ models the frequency stability of a clock, where frequency stability is defined as "The degree to which an oscillating signal produces the same value of frequency for any interval, τ, throughout a specified period of time" [3, p. 2]. This definition as well as subsequent characterisation focuses on the frequency fluctuations of $\phi(t)$, i.e. the first derivative of $\phi(t)$, $\dot{\phi}(t)$. While this is a good and common way to describe how $\phi(t)$ varies in oscillators, ultimately it is the deviation of $\phi(t)$ itself that matters when it comes to synchronisation.

As $\phi(t)$ encompasses the errors found in the behaviour of real clocks, it is impossible to provide a comprehensive description of $\phi(t)$. However, a variety of tools and methods that allows one to characterise $\phi(t)$ to a high degree are available. Generally, $\phi(t)$ is considered as the sum of various deterministic and stochastic components that in principle allows $\phi(t)$ to take on any real value in the range $[-\infty, \infty]$ [3, p. 9]. The deterministic and stochastic components are considered separately, and of the two, the deterministic fluctuations are usually the dominant source of error.

The primary deterministic error source is usually a constant offset in operating frequency of an oscillator as compared to the designed nominal frequency [3, p. 10]. While an offset is not a frequency fluctuation, it causes a phase fluctuation in the form of a linear increase in the phase error. The predominant cause of frequency offset is from variations in the manufacturing process. For example, in crystal oscillators, variations in crystal cuts will cause the oscillator frequency to deviate from the nominal [4, p. 7]. In addition to a constant frequency offset, oscillators also undergo a linear drift in operating frequency. Aging is the prominent cause of linear drift [4, p. 11] and results in a quadratically increasing phase error. Linear drift is very slow and is usually measured in the timescale of days or even years.

Characterising the random components of $\phi(t)$ is a somewhat more complicated endeavour than its deterministic counterparts. Stochastic processes can be characterized in a variety of ways, however over time it has been found that $\phi(t)$ can be well modelled using

a combination of power law noise processes [3, p. 10]. Power law noise processes are characterized by having a power spectral density that adheres to a power law. Generally, five distinct power law processes are considered. These power law noise processes are combined through summations to form a characterization of the frequency spectrum of $\phi(t)$:

$$S_y(f) = \sum_{i=-2}^{2} h_i f^i \tag{3.1}$$

where

$S_y(f)$ is the single sided power spectral density of the frequency noise of $\phi(t)$

f is frequency

h_i is the intensity constant of the ith power law

Each power law is listed in table 3.1 along with their associated single sided frequency power spectrum, $S_y(f)$.

Table 3.1: Power Spectral Density and Allan Variance of different power law noise processes. [2, p. 31]

Noise Type	$S_y(f)$	$\sigma_y^2(\tau)$
White Phase Modulation	$h_2 f^2$	$k_1 \tau^{-2}$
Flicker Phase Modulation	$h_1 f^1$	$k_2 \tau^{-2}$
White Frequency Modulation	h_0	$k_3 \tau^{-1}$
Flicker Frequency Modulation	$h_{-1} f^{-1}$	k_4
Random walk Frequency Modulation	$h_{-2} f^{-2}$	$k_5 \tau$

To promote further intuition of the behaviour of the aforementioned power law processes, a time domain realisation of each of processes is shown in figure 3.1.

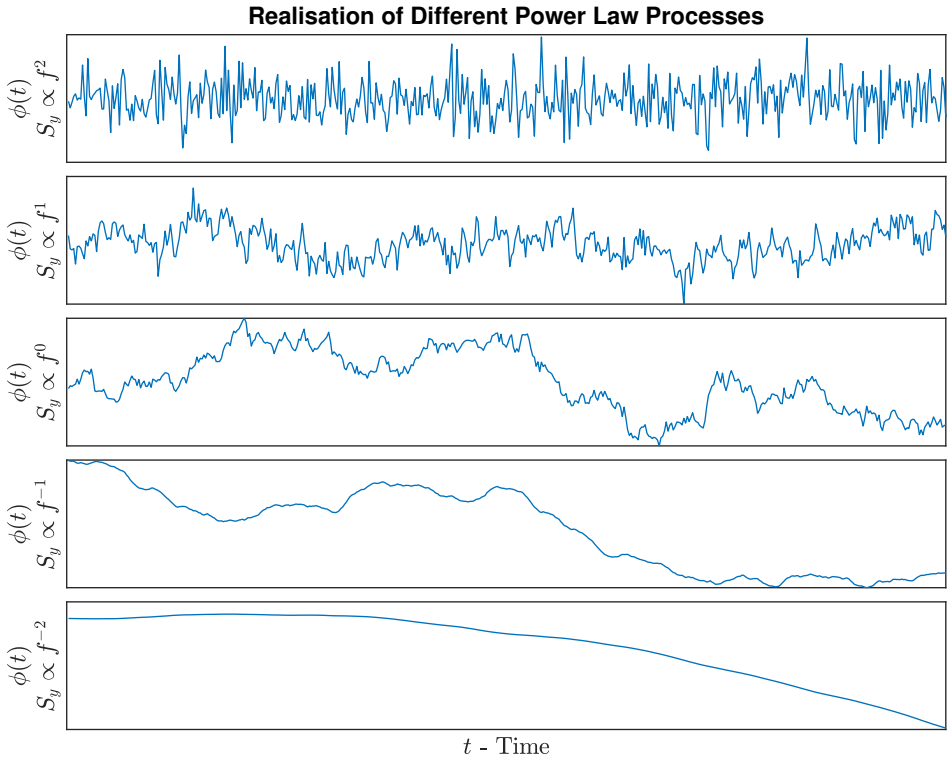


Figure 3.1: Realisation of the five different kinds of power law noise shown in table 3.1.

The exact causes of these distinct noise processes are an ongoing area of study, however certain relations are known. For example, flicker and white Phase Modulated (PM) noise appears caused by noisy electronics in the amplification stages of an oscillator, whereas random walk Frequency Modulated (FM) is often attributed to environmental factors such as vibration, temperature and humidity [3, p. 38].

When $S_y(f)$ is plotted on a log-log plot, the slope of the spectrum helps identify the dominating power law noise in each part of the spectrum. Figure 3.2 illustrates this.

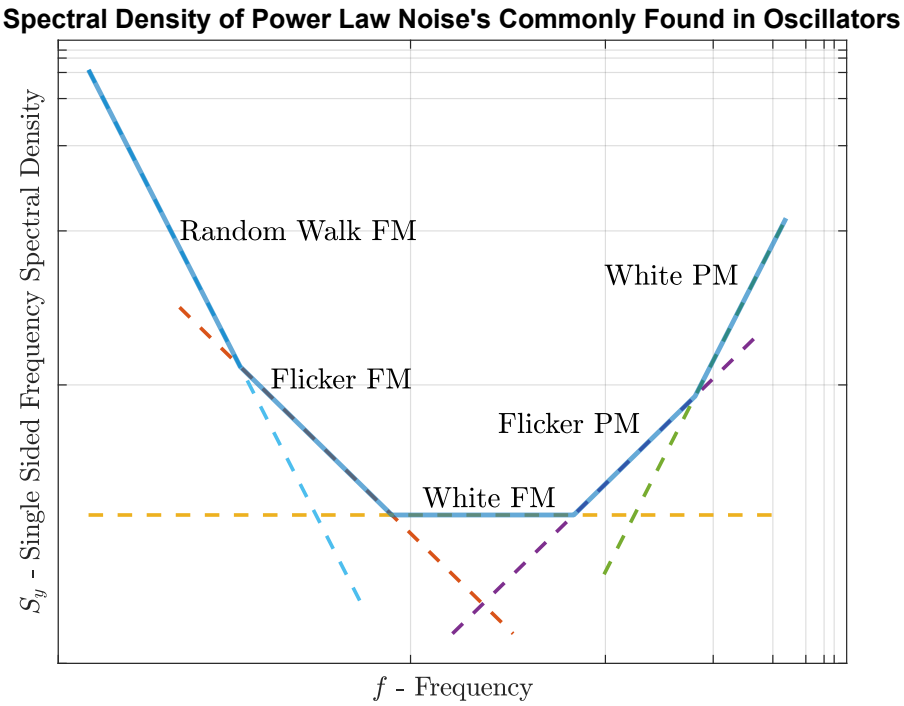


Figure 3.2: Log-log plot of the power spectral density of the frequency of $\phi(t)$. The contribution of the various power law noise processes is highlighted.

In figure 3.2, $S_y(f)$ is designed such that each of the five power law noise processes has region of bandwidth where they are the dominating process. This may not be the case for real oscillators, and it is possible only a subset of the power law noise processes is observable.

As all the above power law noise processes have a time invariant power spectrum, one may think that these are weak sense stationary processes. However, random walk and flicker FM processes clearly do not have a time invariant mean, and as such are non-stationary. Instead random walk FM and Flicker FM are part of a special class of processes that have a time-invariant spectrum and auto-correlation yet are not stationary [5].

While $S_y(f)$ completely characterises the stochastic nature of $\phi(t)$ this rarely finds use in practical characterisation of the mid to low frequency behaviour $\phi(t)$ of real oscillators. This is because the presence of non-stationary flicker and random walk processes makes it difficult to accurately measure $S_y(f)$ due to the processes diverging behaviour as $f \to 0$. Furthermore, the contributions of these noise processes also cause common statistical descriptors such as standard deviation to diverge [3, p. 12]. This diverging nature makes it difficult to rely on such descriptors when characterising oscillators as it is hard to perform accurate and repeatable measurements. To overcome this, engineers have created a variety of alternative statistical measures. The most popular of these is the Allan Variance (AVAR). The definition of AVAR is as follows [2, p. 25] :

$$\sigma_y^2(\tau) = \frac{1}{2} E\left[(y(t+\tau,\tau) - y(t,\tau))^2\right] \tag{3.2}$$

where

$\sigma_y^2(\tau)$ is the Allan variance measured at timescale τ

$E[\,]$ is the expected value over all time, t

$y(t,\tau) = \frac{\phi(t+\tau)-\phi(t)}{\tau}$ i.e. the average frequency between time t and $t+\tau$

Analogously to normal variance and standard deviation, the Allan Deviation (ADEV) is defined as the square root of the AVAR. Equation (3.2), can be interpreted as the variance of the first difference of $y(t,\tau)$, plus some bias. Differencing $y(t,\tau)$ makes the time series data stationary thus allowing $\sigma_y^2(\tau)$ to converge.

As this thesis primarily deals with phase deviations as opposed to frequency deviations, it is often convenient to express the Allan variance in terms of $\phi(t)$:

$$\begin{aligned}\sigma_y^2(\tau) &= \frac{1}{2}E\left[(y(t+\tau,\tau)-y(t,\tau))^2\right] \\ &= \frac{1}{2}E\left[\left(\frac{\phi(t+2\tau)-\phi(t+\tau)}{\tau}-\frac{\phi(t+\tau)-\phi(t)}{\tau}\right)^2\right] \\ &= \frac{1}{2\tau^2}E\left[(\phi(t+2\tau)-2\phi(t+\tau)+\phi(t))^2\right] \end{aligned} \tag{3.3}$$

$\sigma_y^2(\tau)$ is usually plotted on a log-log plot over a range of τ's. This plot is named a "Sigma-Tau" plot and is exemplified in figure 3.3. Like in the plot of $S_y(f)$, the dominating power law noise processes are identified from the curve's varying slope. Similar to $S_y(f)$, each of the five power law noise processes have a well defined impact on $\sigma_y^2(\tau)$. The relationship between $S_y(f)$ and $\sigma_y^2(\tau)$ is shown in table 3.1.

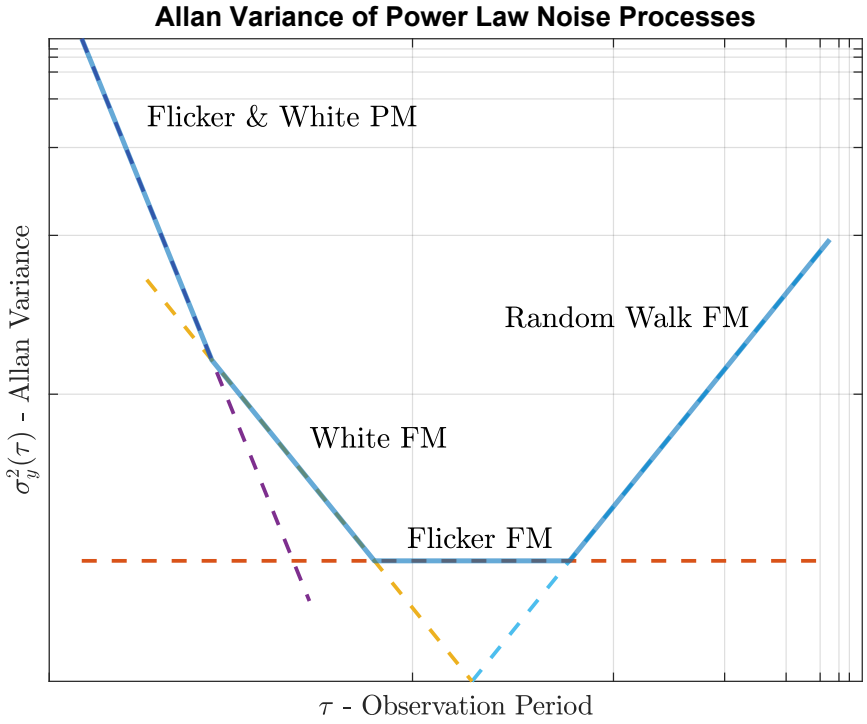


Figure 3.3: Sigma-Tau plot of the Allan Variance $\phi(t)$. The contribution of the various power law noise processes is highlighted.

As is evident from figure 3.3 and table 3.1, flicker and white PM noise have a very similar slope on a Sigma-Tau plot. This can make it hard to discern them and separate their individual noise contribution. When performing oscillator design, this can render it difficult to identify noisy components. To overcome this, different variants of AVAR are commonly used. For example, the Modified Allan Variance allows for easier separation of these two noise processes. Other variations of AVAR have also been created for purposes such as improving the statistical significance of measurement, for measuring phase fluctuations instead of frequency fluctuations, or for taking the second difference of $y(t, \tau)$ as opposed to just the first difference [2][6]. Further details of these variations of AVAR are outside the scope of this thesis.

3.2 Practical Considerations

Thus far oscillator phase, $x(t)$, and phase error, $\phi(t)$ were considered theoretically. However, as an oscillator provides no inherent way to extract the phase at a given time, additional processing is required to extract the phase of an oscillators periodic output. In digital electronic devices this is done discretely where time is tracked through periodically occurring events in the oscillator waveform, commonly called clock ticks. The time resolution of each 'tick' is dependent on the underlying oscillator's clock frequency. For example, for a 100 MHz clock, each tick corresponds to 10 ns. Mathematically, these clock ticks are related to the continuous representation phase, $x(t)$, according to:

$$\hat{x}(t) = \lfloor t + \phi(t) \rfloor_c \tag{3.4}$$

where:

$\lfloor\ \rfloor_c$ floors to the nearest value exactly divisible by c. i.e. $\lfloor x \rfloor_c = x - (x \bmod c)$

$\hat{x}(t)$ is the quantised phase

Using this model, the previous example is modelled by setting $c = 10^{-8}$.

Figure 3.4 illustrates the relationship between the oscillator's continuous behaviour and discrete clock ticks registered by digital logic.

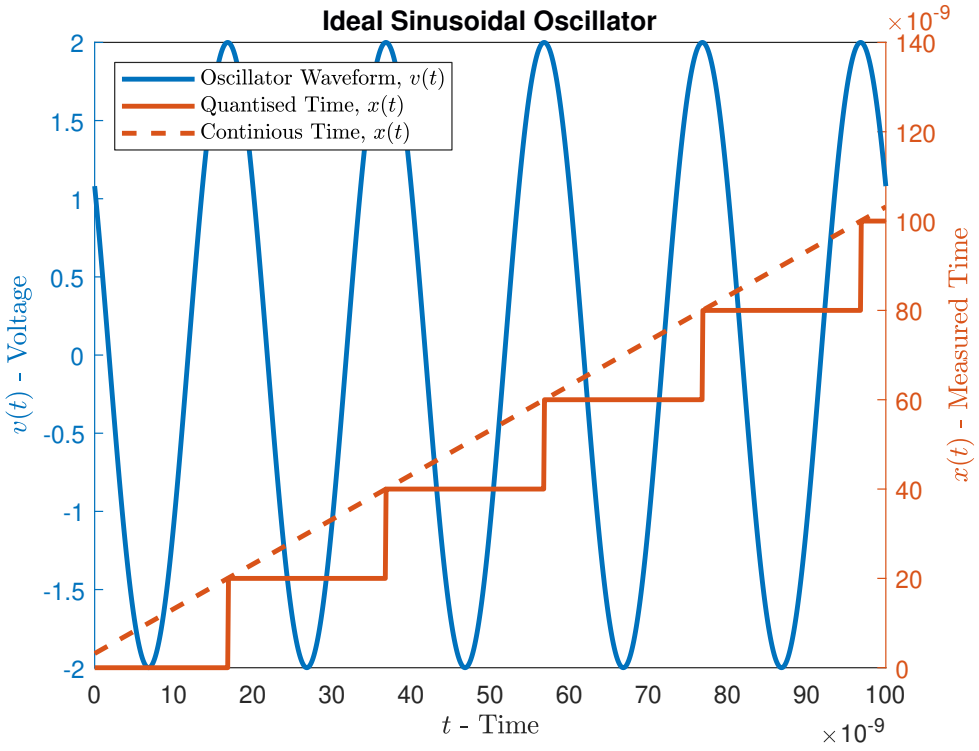


Figure 3.4: Illustration of how measured time is quantised. In this example $v(t)$ is a 5 MHz signal and $\phi(t) = 3.18 \cdot 10^{-9}$.

If the clock time is measured in between clock ticks, the inherent discretisation of the clock phase adds quantisation noise to the measured value as compared to continuous counterpart, $x(t)$. While complex analog circuitry can be added to interpolate between two clock cycles, thereby largely eliminating the quantisation error, this is rarely used as it is expensive and the quantisation error added from a pure digital solution is generally considered acceptable for most applications. Generally, it is preferred to use $x(t)$, as opposed to $\hat{x}(t)$, when doing mathematical analysis of the phase. This is because the discontinuities in floor function of $\hat{x}(t)$ complicates mathematical derivations. Thus, all mathematical results are exclusively derived with respect to $x(t)$. Furthermore, while all measurements undertaken in this thesis are $\hat{x}(t)$, they are all treated as $x(t)$ unless specified explicitly.

While $x(t)$ is sufficient for timekeeping, it is not enough to infer the phase error, $\phi(t)$. Ideally, $\phi(t)$ is derived by comparing $x(t)$ to an ideal oscillator's phase:

$$\phi(t) = x(t) - t \tag{3.5}$$

However, such a device does not exist, and thus another clock must be used as a reference. As this clock is also affected by phase noise, this will impact the measurement accuracy. Since the clocks' phases are measured in relation to each other, the phase error between clock 1 and 2 is measured. Specifically, the relationship is:

$$\begin{aligned}\phi(t,t) &= x_2(t) - x_1(t) \\ &= (t + \phi_2(t)) - (t + \phi_1(t)) \\ &= \phi_2(t) - \phi_1(t)\end{aligned} \tag{3.6}$$

where:

$\phi(t,t)$ is the phase error between clock 1 and 2, both measured at time t.

$\phi_i(t)$ is the phase error of clock i

$x_i(t)$ is phase error of clock i

In most of the scenarios considered in this thesis, one clock, clock 1, initiates at time t_1 the reading of another clock, clock 2, which is sampled after a propagation delay at time:

$$t_2 = t_1 + \eta_{1,2}(t_1) \tag{3.7}$$

where

$\eta_{1,2}(t_1) \geq 0$ is the time required for the sample signal initiated by clock 1 at time t_1 to propagate to clock 2.

For example, when clock 2 is sampled from a signal emitted by clock 1, the phase difference is:

$$\begin{aligned}\phi(t_1,t_2) &= x_2(t_2) - x_1(t_1) \\ &= (t_2 + \phi_2(t_2)) - (t_1 + \phi_1(t_1)) \\ &= \phi_2(t_1 + \eta_{1,2}(t_1)) - \phi_1(t_1) + \eta_{1,2}(t_1)\end{aligned} \tag{3.8}$$

In its simplest form the model of $\eta_{1,2}(t)$ is a constant representing the signal propagation delay imposed by the speed of light. However, in more complex implementations, variations in the systems emitting, receiving, and processing of the sample signal all contribute to $\eta_{1,2}(t)$, and ultimately provides it with stochastic properties. Furthermore, it is not guaranteed that the propagation delay is symmetrical and that $\eta_{1,2}(t) = \eta_{2,1}(t)$.

Chapter 4

Offset Estimation Methods

Having previously described the source of clock noise, this chapter goes into further detail on how to practically measure $\phi(t_1, t_2)$. As an introduction to the topic, the widespread case of using electrical pulses is covered to find $\phi(t_1, t_2)$ is covered. Subsequently, packet based method are covered in general alongside details specific to the PTP.

4.1 Pulse Based

Perhaps the simplest method to estimate the temporal offset between a reference, clock 1, and a slave clock, clock 2, is to configure the reference clock to emit periodic pulses at a pre-determined phase and interval. For example, most GPS clocks emit a pulse at the beginning of every second, known as 1 Pulse Per Second (PPS). The slave clock measures the arrival time of this pulse and based on a priori knowledge of the pulse's phase and interval; it is possible to infer how much the slave lags/leads the reference. figure 4.1 illustrates the basic concept:

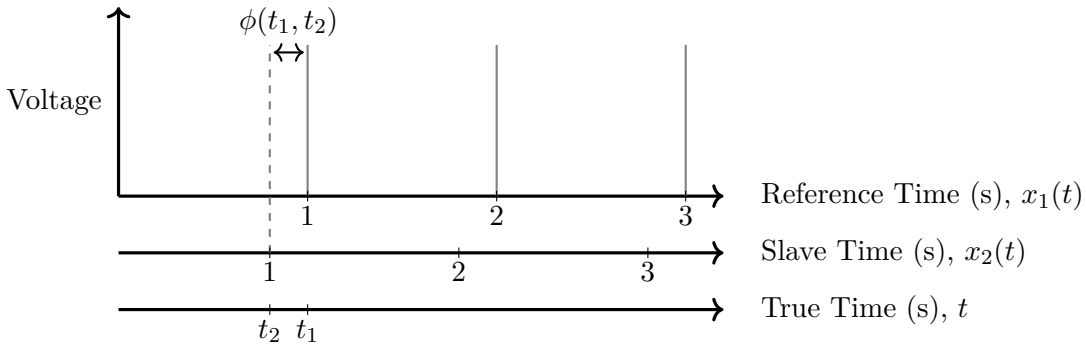


Figure 4.1: Illustration of offset found using pulse based synchronisation.

This concept is now described using the mathematical framework established in the prior sections. We start by defining a pulse, with an emission time in relation to clock 1 being an exact multiple of some initial phase:

$$x_1(t_1^k) = t_1^k + \phi_1(t_1^k) = k\tau_0 \tag{4.1}$$

t_1^k is the k'th pulse's actual emission time. (superscript does not refer to exponentiation)

τ_0 is a predetermined pulse period and initial phase.

k is some integer representing the kth pulse

In the following, the constant k will be omitted as only a single pulse will be considered.

Given that t is rewritable in terms of the difference between $x_1(t)$ and its phase error:

$$\begin{aligned} x_1(t) &= t + \phi_1(t) \\ &\Downarrow \\ x_1(t) - \phi_1(t) &= t \end{aligned} \tag{4.2}$$

It follows that clock 2 can be written in terms of $x_1(t)$:

$$\begin{aligned} x_2(t) &= t + \phi_2(t) \\ &= x_1(t) - \phi_1(t) + \phi_2(t) \end{aligned} \tag{4.3}$$

The pulse is detected by clock 2 at time t_2 and this is timestamped according to:

$$\begin{aligned} x_2(t_2) &= t_2 + \phi_2(t_2) \\ &= t_1 + \eta_{1,2}(t_1) + \phi_2(t_2) \\ &= x_1(t_1) - \phi_1(t_1) + \eta_{1,2}(t_1) + \phi_2(t_1 + \eta_{1,2}(t_1)) \end{aligned} \tag{4.4}$$

As the electronic device containing clock 2 has a-priori knowledge of the mathematical relationship of the pulse emission time in relation to $x_1(t)$ it estimates its value as:

$$\widetilde{x}_1(t_1) = \lfloor x_2(t_2) \rceil_{\tau_0} \tag{4.5}$$

where

$\widetilde{x}_1(t_1)$ is the estimated value of $x_1(t_1)$

$\lfloor \rceil_{\tau_0}$ rounds to the nearest integer exactly divisible by τ_0

If $x_2(t_2)$ is in the range $(x_1(t_1) - \frac{\tau_0}{2}, x_1(t_1) + \frac{\tau_0}{2})$ then this estimate will be correct, and the total phase error between x_1 and x_2 is found as (equivalent to equation (3.8)):

$$\begin{aligned} \phi(t_1, t_2) &= x_2(t_2) - x_1(t_1) \\ &= \phi_2(t_2) - \phi_1(t_1) + \eta_{1,2}(t_1) \end{aligned} \tag{4.6}$$

If on the other hand $x_2(t_2)$ is not within this range, the estimate will be off by an integer multiple of τ_0.

The error in the offset estimation caused by propagation delay must also be considered. For small installations where the two sources are connected by short cables, this delay is usually of negligible significance. In these situations, pulse based offset estimations offers excellent performance. However, if the two sources are far apart, the propagation delay may become significant. Additionally, since approximate initial synchronisation is necessary to ensure high performance another mechanism is necessary for setting the initial time of the slaves' clocks. Finally, since pulses are distributed to every slave clock in a synchronisation network, pulse based synchronisation is often too complex and costly to rely on for larger installations.

4.2 Precision Time Protocol

As an alternative to pulse based synchronisation, packet based synchronisation was developed to allow for synchronisation using the existing infrastructure build for network communication. Perhaps the simplest way to implement offset estimation through a network would be for the reference clock to periodically broadcast its time to all the slave clocks in the network. Each slave can then compare the received time with their own time to derive their offset. Compared to pulse based synchronisation, such a scheme has the advantage that the initial synchronisation between clock 1 and clock 2 does not matter as the exact value of $x_1(t)$ can be transmitted, and x_2 does not need to rely on an estimate.

While simple to implement, this method has a major flaw. Compared to pulse based synchronisation the propagation delay, $\eta(t)$, is rarely considered insignificant as the addition of complex networking equipment adds significant processing overheard. Thus, depending on the network configuration $\eta(t)$ can easily be in the order of 10's or 100's of microseconds. Here $\eta(t)$ is refers to propagation delay in general, and the direction is ambiguous.

To overcome this modern packet based synchronisation networks allow the reference and slave clocks to exchange an additional set of packets from the slave to the master. This is known as a two-way time transfer scheme and allows slave clocks to estimate the network propagation delay. Figure 4.2 illustrates these transfers:

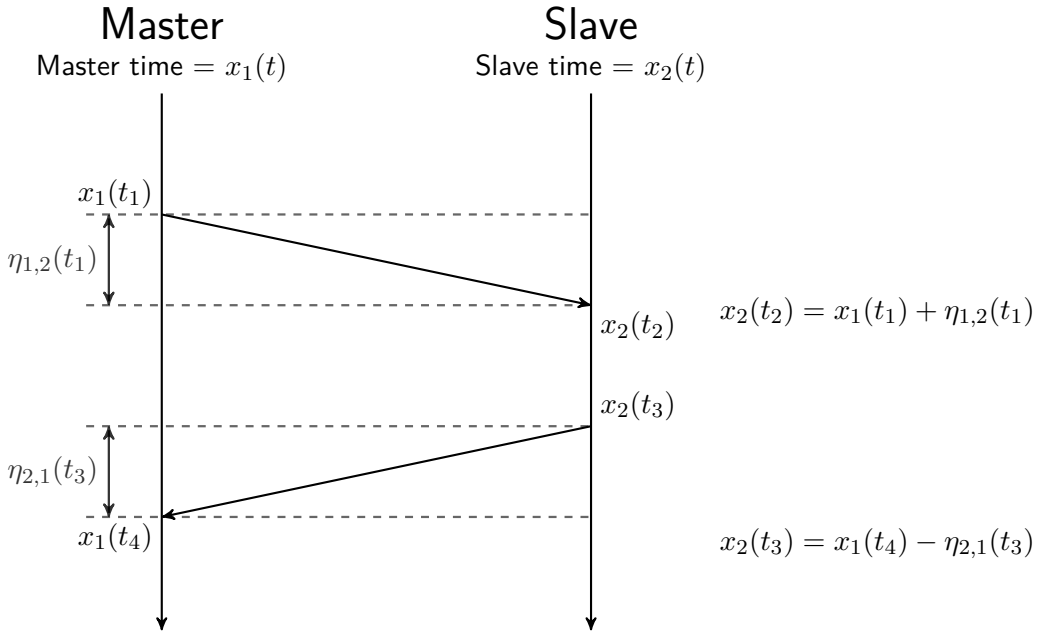


Figure 4.2: Visualisation of a two-way time transfers and the four timestamps acquired from the exchanges. [1].

In our implementation it has been decided to use the PTP [1], as specified by IEEE 1588. This protocol defines the packets needed to carry out the two-way time transfer illustrated in figure 4.2.

To calculate the offset between the master and slave device, a total of four timestamps are required. Using timestamps $x_1(t_1)$ and $x_2(t_2)$, the phase offset plus the propagation

delay, $\eta_{1,2}(t)$, can be found.

The second pair of timestamps, $x_2(t_3)$ and $x_1(t_4)$, allows the phase offset minus the propagation delay in the opposite direction, $\eta_{2,1}(t)$, to be determined.

If $E[\eta_{1,2}(t)] = E[\eta_{2,1}(t)]$ the propagation delay is said to be symmetrical. When this is the case, the four timestamps allows the mean propagation delay, $E[\eta(t)]$, to be estimated:

$$\begin{aligned}\widetilde{\eta} &= \frac{[x_2(t_2) - x_1(t_1)] - [x_2(t_3) - x_1(t_4)]}{2} \\ &= \frac{[\phi_2(t_2) - \phi_1(t_1) + \eta_{1,2}(t_1)] - [\phi_2(t_3) - \phi_1(t_4) - \eta_{2,1}(t_3)]}{2} \\ &= \frac{\phi(t_1, t_2) - \phi(t_4, t_3) + \eta_{1,2}(t_1) + \eta_{2,1}(t_3)}{2}\end{aligned} \tag{4.7}$$

where

$\widetilde{\eta}$ is an estimate of the mean system propagation delay based on timestamps $x_1(t_1)$, $x_2(t_2)$, $x_2(t_3)$ and $x_1(t_4)$.

For this estimate to be valid, clock phase noise must be low, i.e. $\phi(t_1, t_2) \approx \phi(t_3, t_4)$. If the phase noise of clock 1 and clock 2 in the time interval between t_1 and t_3 is low, then it is often an adequate approximation. Furthermore, it is necessary that the jitter/Packet Delay Variation (PDV) is low compared to the mean value. Otherwise it is not fair to assume that $\eta_{1,2}(t_1) \approx \eta_{2,1}(t_3)$, nor that the estimate, $\widetilde{\eta}$, is an adequate approximation of $E[\eta_{1,2}(t)]$. If the PDV is high, more samples are usually combined using advanced processing algorithms to derive a superior estimate of $E[\eta_{1,2}(t)]$.

In addition, to short term PDV, long term consistent asymmetry must also be considered, where asymmetry is defined as the mean difference between $\eta_{1,2}(t)$ and $\eta_{2,1}(t)$, i.e. $E[\eta_{1,2}(t) - \eta_{2,1}(t)]$. If asymmetry is present, it is not possible to accurately compensate for propagation delay without further knowledge of the asymmetry. Thus, it is critical to minimize asymmetry if the propagation delay is to be accurately estimated.

Using the propagation delay estimate, $\widetilde{\eta}$, phase error is estimated as:

$$\widetilde{\phi}(t_1, t_2) = x_2(t_2) - x_1(t_1) - \widetilde{\eta} \tag{4.8}$$

Thus, once $\widetilde{\eta}$ has been found, the phase error can be derived using only one set of transfers between clock 1 and clock 2. However, to account for dynamic changes in networks, the estimate is usually recalculated periodically.

As highlighted earlier, asymmetry and PDV degrade the estimate of the offset between clock 1 and clock 2. These effects mainly originate from delays in the software/hardware of the receiver/transceiver's networking stack as well from packet queuing in networking switches. Furthermore, the instantaneous PDV is not constant and factors such as networking load or the computational load put on the slave/reference clock nodes all affects the PDV. In addition to specifying the PTP used to transfer timestamps, IEEE 1588 also specifies multiple strategies that can be employed to minimize PDV as well as the

effect of changing network/processing loads. Principal of which is the use of hardware timestamping immediately following packet ingress and egress. figure 4.3 illustrates this:

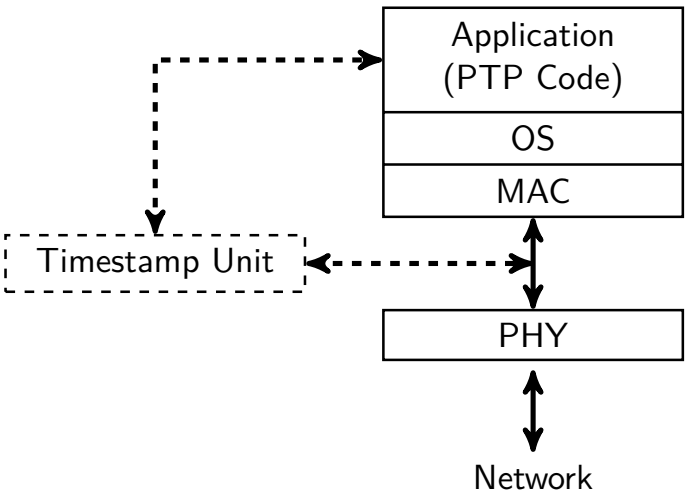


Figure 4.3: Visualization of where hardware timestamping may be used in PTP [7].

The Time Stamping Unit (TSU) is special built hardware that scans incoming/outgoing packets. When it detects a PTP packet it timestamps the packets ingress/egress time. The PTP synchronisation application can then transfer these timestamps to slave devices, or alternatively use them to estimate time and frequency offset. To minimize PDV the TSU hardware is located as close to the physical layer as possible. This largely eliminates timestamping variation.

In addition to timestamping, the IEEE 1588 specifications specifies a variety of switch types designed to minimize PDV introduced by network switches. In our setup we have chosen to use E2E (End to End) transparent switches, however, note that other types exist. These switches measure PTP packet residence time and inserts this time in the packet before it is re-transmitted. Once received by a device, this residence time can then be accounted for when calculating $\widetilde{\eta}$ and $\widetilde{\phi}(t_1, t_2)$.

Chapter 5

System Overview

To aid development of the synchronisation network, a development and testing setup has been deployed. The creation of this setup was a key subject in the previous interim report [1]. For the sake of completeness, the key elements are reiterated. Following the overview of the system setup, the expected use cases, networking scenarios, and system requirements are detailed. To facilitate understanding, the setup is illustrated in figure 5.1. Going from left to right in the figure 5.1, reference time is initially received by a GPS receiver somewhere at CERN. This timing information is processed by CERN's Central Time Management system. This system distributes reference time to CERN's various equipment, laboratories, and experiments. The central time management system is maintained by other sections at CERN and is not discussed further, however it is still relevant as the acquisition system ultimately derives its time from it. The time from the central time management system is fed into a Central Time Receiver (CTR) card. This card is a PCI expansion card and is placed and controlled from a Front End Computer (FEC) in our laboratory. The CTR card serves as the reference time source in the synchronisation network. However, as the goal is to synchronise data acquisition nodes using packet based synchronisation through Ethernet, the CTR cannot directly be used as it lacks a network interface. Thus, the CTR is used to synchronise the clock of a 1 Gbit/s Network Interface Card (NIC) within the FEC. This synchronisation is performed using pulse based synchronisation, as described in section 4.1. Specifically, the CTR is configured to emit 1PPS signals, and the NIC then proceeds to align its internal clock to the received pulses. This synchronisation process is controlled by the `ts2phc` application from the *Linux PTP Project* [8]. Following synchronisation of the NIC, the data acquisition nodes can be synchronised through PTP as described in section 4.2. On the FEC side, PTP is managed by the `ptp4l` application, another program from the *Linux PTP Project*. Meanwhile the Data Acquisition Boards (DQAMEs) rely on a custom implementation of PTP. Finally, note that the PTP test network under development is flat and only contains a single PTP compliant switch (Juniper QFX5110-48S). Thus far the complete synchronisation chain has been accounted for. This resembles the system that will be deployed. However, for development purposes, functionality has been added to simplify logging, debugging, and testing. This includes port mirroring on the switch to monitor PTP traffic and the possibility to configure each synchronisation node in the chain to emit periodic pulses which is to quantify the synchronisation between the nodes.

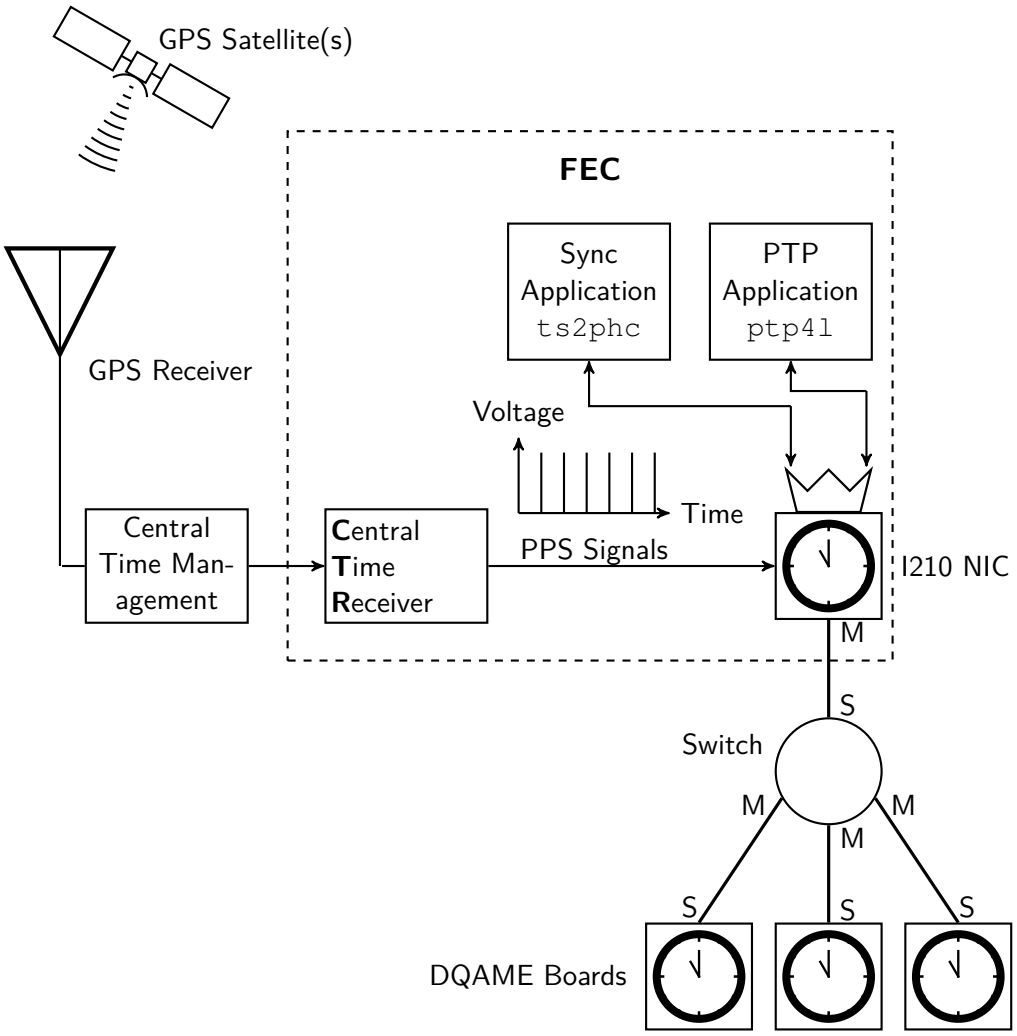


Figure 5.1: Detailed illustration of the lab setup [1].

As the setup does not allow for direct interaction with the GPS time or central time management time, these are ignored and only the CTR, NIC, and DQAME nodes are further considered. The synchronisation between these three entities can be modelled as two cascaded control loops. In the first loop, the FEC is synchronised to the central timing receiver through the use of pulse per second signals. The FEC then uses the PTP to synchronise the various DQAMEs in the network. This cascaded control system can be conceptualised as the block diagram in figure 5.2.

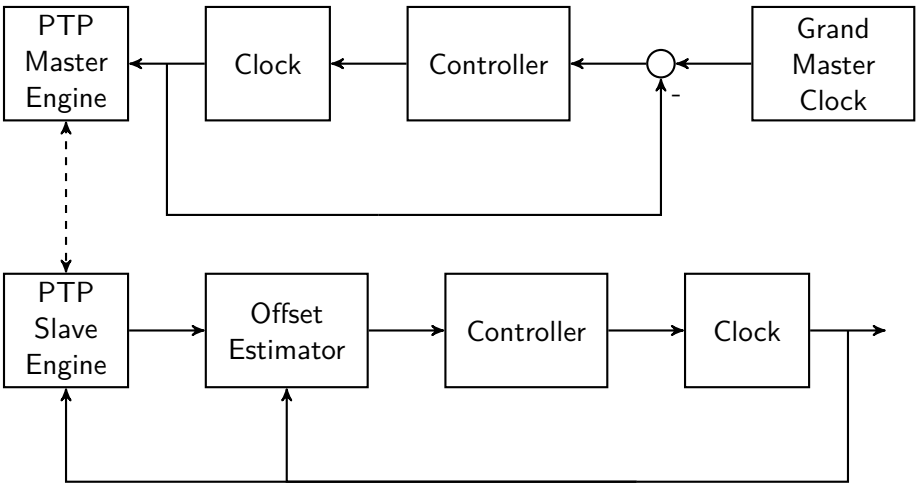


Figure 5.2: Conceptual block diagram of the cascaded control loops in the setup.

The first control loop is fully managed by the aforementioned `ts2phc` application. To simplify and accelerate development, this off the shelf solution is kept as is and not studied or optimized further. As no similar readily available solution exists for the DQAMEs, the second control system used to synchronise the NIC and DQAMEs through PTP is the primary topic of interest in this thesis.

To allow for proper characterisation of the synchronisation sub-system, the data acquisitions system's requirements and expected operational behaviour must be considered. When deployed, it is anticipated that up to 47 DQAMEs will be associated with a NIC. It is expected that each of these are able to transmit a maximum of 20 Mbit/s of data to the NIC. In the extreme scenario where all DQAMEs transmit at their peak rate, a NIC will receive 940 Mbit/s. On the other hand, the NIC only transmits control and synchronisation packets to devices in the network. Thus, the network load going from the NIC to its DQAMEs will be minimal. Furthermore, during operation the network remains static. This constraint greatly simplifies synchronisation, as changes in the propagation delay caused by dynamic packet routing can be ignored. Finally, while there are no definitive requirements for the synchronisation performance, development is carried out with the expectation that the phase error between clock 1 and clock 2, $\phi(t_1, t_2)$, shall not exceed 1 µs.

Chapter 6

Noise Characterisation

In this chapter the stochastic properties of the system noise are investigated. First the clock noise of the NIC and DQAME, $\phi_1(t)$ and $\phi_2(t)$,is quantified by measuring their AVAR. Secondly, the nominal PDV, i.e. $\eta_{1,2}(t)$ and $\eta_{2,1}(t)$, in the PTP network is found. Finally, PDV behaviour in non ideal networking conditions is studied by measuring the frequency and magnitude of outliers.

6.1 Clock Noise Measurements

To quantify the clock noise, present in the synchronisation system, each clock in the system must be measured. Clock noise is measured using a timestamping counter. Such a device keeps track of time using an external oscillator. It then timestamps incoming pulses in relation to its internal time. These timestamps can then be used to infer the total phase error in relation to the arrival time of the first pulse.

The tests for this thesis were conducted using a TARP TICC time interval counter in conjunction with a 10 MHz FS725 rubidium oscillator as the reference. This reference oscillator was chosen as it has excellent performance in terms of phase noise. Its one second Allan variance is reported as approximately $4 \cdot 10^{-22}$. The TICC is a two channel time interval counter. It directly uses the 10 MHz reference clock to generate a coarse sense of time with 100 ns resolution. To increase this, it uses an internal ring oscillator with a period of 57 ps to interpolate between clock ticks and provide a finer time resolution. When a pulse is detected on one of its two channels, it is timestamped and can be extracted through a USB serial port. Due to processing overhead, the device is limited to logging about 100 timestamps per second per channel. It is critical to note that the TICC is designed and manufactured by a group of amateur timing enthusiasts. As such, it is unlikely that it has been subject to the same rigorous testing, validation, and calibration that one would expect from professional equipment. In spite of this it was chosen as it is held in high regard within the online amateur timing community, it provides gap free measurements allowing one to continuously sample over extended periods of time without dropping samples, and it is cheap.

The TICC is used to measure the stability of the PTP network's reference and slave clocks. As all stability should be found in relation to the CTR time, the reference rubidium clock is synchronised to the CTR using its PPS input. For this to be possible, both the NIC and DQAMEs are configured to output pulses at 100 Hz. This is easily achieved in the NIC, as the feature is supported by the shipped firmware. For the DQAMEs, this feature

had to be implemented from scratch. The implementation relies on the TSU's comparison register to emit a pulse for every elapsed 10 ms.

An illustration of the entire measurement setup is shown in figure 6.1:

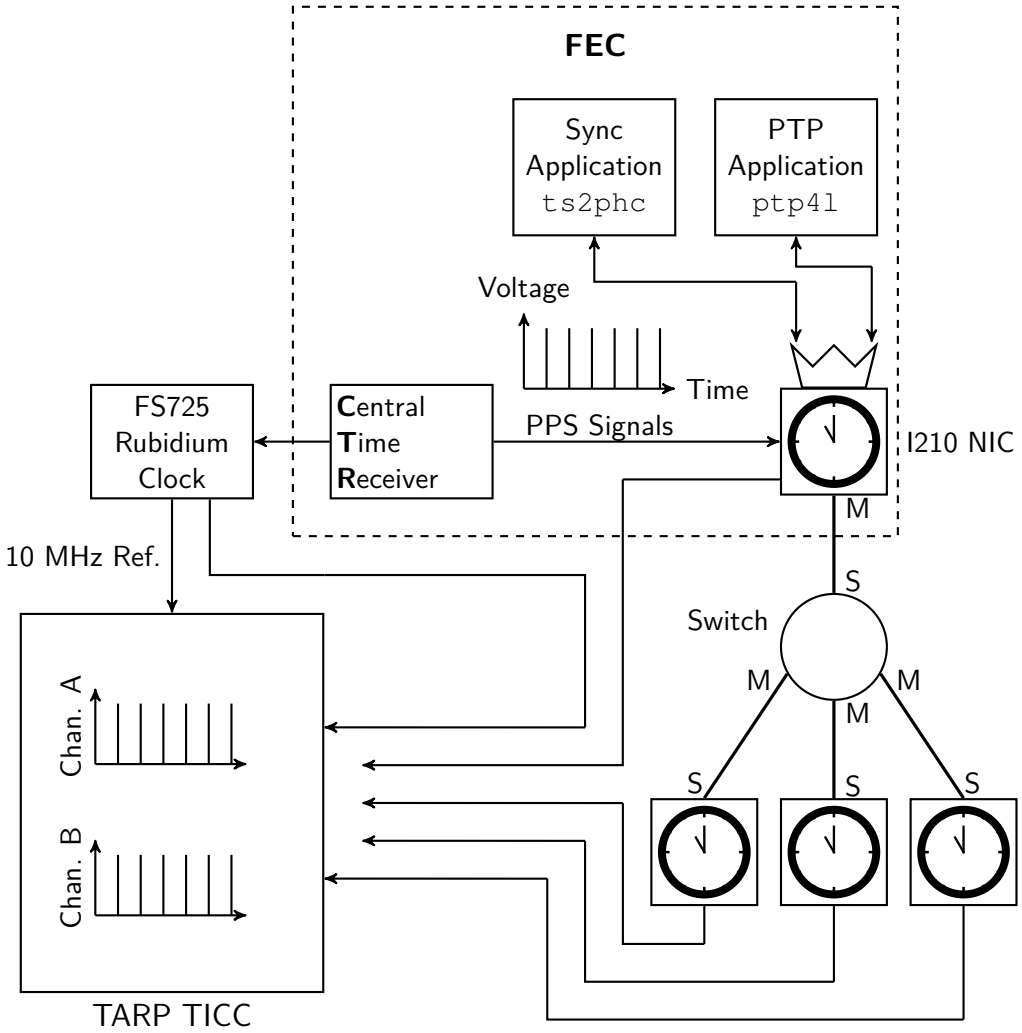


Figure 6.1: Illustration of the test setup.

As the final solution will be deployed to multiple locations, deployments will not use the same exact hardware as what is tested. Thus, operational performance may differ from the measured results due to manufacturing and environmental variations. To account for this possible spread in performance caused by equipment variations, two NICs and four DQAME cards are tested. Ideally a much larger sample size is needed to uncover the statistical nature of the variations. However due to time constraints this was not possible. Furthermore, environmental variations were not controlled during tests. As the equipment being tested needed access to the CERN timing network, all tests were conducted in a laboratory that facilitated this access. This laboratory is not temperature controlled and is open to other personnel to work in. Due to the lack of environmental control, it is expected that there will be variations in the long term behaviour of the clocks, as environmental variations impact oscillators random walk behaviour.

To obtain sufficient data about the oscillator behaviour, all tests were set to run for at least 6 hours. Some ran significantly longer than this, as they were left overnight. Furthermore, as hundredths of thousands of data points were collected per test, incidentally faulty measurements occurred. These outliers were removed manually (and subjectively) prior to further data processing. Finally, to isolate the stochastic component of the measured phase error, linear and quadratic phase drift was removed through detrending.

6.1.1 TIC Theory of Usage

Before going into further details with the noise tests, this section describes how timestamps are used to estimate the phase error of a clock. First note that because the measurements are based on pulses, the measured phases are inherently discrete. This differs from the continuous time representation covered in previous sections. To build the intuition of how a timestamping counter is used to measure phase error, we start by defining a sequence of n pulses, with known triggering time in relation to clock one, i.e. $x_1(t_1^1), x_1(t_1^2), \ldots, x_1(t_1^n)$. Where $t_1^1 < t_1^2 < \ldots < t_1^n$.

Similar to the explanation found in section 4.1, when a pulse from the first clock is detected by clock two, it is timestamped according to:

$$x_2(t_2^i) = x_1(t_1^i) - \phi_1(t_1^i) + \phi_2(t_2^i) + \eta_{1,2}(t_1^i) \tag{6.1}$$

As a discrete sequence of time, t^i, is now considered, it is appropriate to simplify notation to reflect the discrete nature of the following calculations. This new notation simply indexes a function based on the ith time and kth clock being considered, exemplified here:

$$j_k[i] = j(t_k^i) \tag{6.2}$$

where

j is some arbitrary time dependent function, such as $x_1(t)$, $x_2(t)$ or $\rho(t)$

By differencing the sequence of timestamps, the phase difference between each occurring pulse, $\Delta x_2[i]$, can be deduced:

$$\begin{aligned}
\Delta x_2[i] &= x_2[i+1] - x_2[i] \\
&= (x_1[i+1] - \phi_1[i+1] + \phi_2[i+1] + \eta_{1,2}[i+1]) - (x_1[i] - \phi_1[i] + \phi_2[i] + \eta_{1,2}[i]) \\
&= x_1[i+1] - x_1[i] + \phi_1[i] - \phi_1[i+1] + \phi_2[i+1] - \phi_2[i] + \eta_{1,2}[i+1] - \eta_{1,2}[i] \\
&= \Delta x_1[i] - \Delta\phi_1[i] + \Delta\phi_2[i] + \Delta\eta_{1,2}[i]
\end{aligned} \tag{6.3}$$

where:

$$\Delta x_1[i] = x_1[i+1] - x_1[i]$$

$$\Delta\phi_1[i] = \phi_1[i+1] - \phi_1[i]$$

$$\Delta\phi_2[i] = \phi_2[i+1] - \phi_2[i]$$
$$\Delta\eta_{1,2}[i] = \eta_{1,2}[i+1] - \eta_{1,2}[i]$$

The total phase difference between the first pulse, and pulse k is then found as the cumulative sum of $\Delta x_2[i]$:

$$\begin{aligned}\bar{x}_2[k] &= \sum_{i=0}^{i=k-1} \Delta x_2[i] \\ &= \sum_{i=0}^{i=k-1} (\Delta x_1[i] - \Delta\phi_1[i] + \Delta\phi_2[i] + \Delta\eta_{1,2}[i]) \\ &= \bar{x}_1[k] - \bar{\phi}_1[k] + \bar{\phi}_2[k] + \bar{\eta}_1[k] \end{aligned} \tag{6.4}$$

where:

$$\bar{x}_1[k] = \textstyle\sum_{i=0}^{i=k-1} \Delta x_1[i]$$
$$\bar{\phi}_1[k] = \textstyle\sum_{i=0}^{i=k-1} \Delta\phi_1[i]$$
$$\bar{\phi}_2[k] = \textstyle\sum_{i=0}^{i=k-1} \Delta\phi_2[i]$$
$$\bar{\eta}_1[k] = \textstyle\sum_{i=0}^{i=k-1} \Delta\eta_{1,2}[i]$$
$$k \leq n-1$$

The phase error between clock 1 and clock 2 at time t_1 and t_k, $\bar{\phi}[k]$, can now be estimated as:

$$\begin{aligned}\bar{\phi}[k] &= \bar{x}_2[k] - \bar{x}_1[k] \\ &= -\bar{\phi}_1[i] + \bar{\phi}_2[i] + \bar{\eta}_1[i] \end{aligned} \tag{6.5}$$

In practice a pulse sequence is usually defined with a constant period such that $\Delta x_1[i] = \tau_0, \quad \forall i$ where τ_0 is a predetermined pulse period. With this condition, the measured phase error simplifies to:

$$\bar{\phi}[k] = \bar{x}_2[k] - k\tau_0 \tag{6.6}$$

Given that we now have a method to find a discrete sequence of the relative phase error between two clocks, statistical estimators can be used to estimate the underlying stochastic properties of the process. For example, to estimate the Allan variance of $\bar{\phi}[k]$, the sample mean can be used directly [6, p. 14]:

$$\sigma_y^2(\tau_0) = \frac{1}{2\tau_0^2} E\left[(\phi(t+2\tau_0) - 2\phi(t+\tau_0) + \phi(t))^2\right]$$
$$\Downarrow$$
$$\sigma_y^2(\tau_0, M) \approx \frac{1}{2\tau_0^2(M-2)} \sum_{i=0}^{M-3} (\bar{\phi}[i+2] - 2\bar{\phi}[i+1] + \bar{\phi}[i])^2 \tag{6.7}$$

where

M is the number of samples of $\bar{\phi}[k]$ to use in the estimate.

Using the same set of $\bar{\phi}[k]$ measurements, it is straightforward to conceptualise how to estimate the Allan Variance for integer multiples of τ_0. If $\sigma_y^2(j\tau_0)$ is to be estimated simply discard every jth sample of $\bar{\phi}[k]$ and use the above equation again. This approach allows one to estimate the Allan variance of $\bar{\phi}[k]$ for a large range of τ, however the statistical significance of the estimate quickly drops as the majority of samples are discarded for larger τ values. Modern processing tools instead use the overlapping Allan variance estimator, which allows more samples to be used in the estimate, thus increasing the statistical reliability of the measure [6, p. 15]. The mathematical details of the overlapping Allan variance estimator are not discussed further here, however in the subsequent tests the estimator is used to estimate the Allan variance of the measured processes.

6.1.2 TICC Noise Floor

Consider figure 6.2. The manufacturers of the TICC claim that its noise floor at the 1-second Allan Variance is at $\approx 4.9000 \cdot 10^{-22}$. But as the TICC is not a professional device it is appropriate to validate its performance before starting extensive testing. To do this, the noise floor of both its channels is quantified by measuring the rubidium reference oscillator with respect to itself. The rubidium reference has a port that emits a 1PPS signal. These pulses are used as input to the TICC's two channels. To see how this configuration allows the noise floor of the TICC to be quantified, consider equation (6.5) for the case where $\phi_1(t) = \phi_2(t)$

$$\begin{aligned}
\bar{\phi}[k] &= \bar{x}_2[k] - k \\
&= -\bar{\phi}_1[i] + \bar{\phi}_2[i] + \bar{\eta}_1[i] \\
&= -\sum_{i=0}^{i=k-1} \Delta\phi_1[i] + \sum_{i=0}^{i=k-1} \Delta\phi_2[i] + \bar{\eta}_1[i] \\
&= -\sum_{i=0}^{i=k-1} (\phi_1(t_1^{i+1}) - \phi_1(t_1^i)) + \sum_{i=0}^{i=k-1} (\phi_2(t_2^{i+1}) - \phi_2(t_2^i)) + \bar{\eta}_1[i] \\
&= \bar{\eta}_1[i]
\end{aligned} \tag{6.8}$$

where:

$\phi_1(t) = \phi_2(t)$

$t_1^i = t_2^i \quad \forall i$

$\bar{\eta}_1[i]$ is the measurement noise added by the TICC

This holds because the same oscillator is used for both inputs. For this to be valid in practice, it is critical that all of the cables used to connect the rubidium clock's PPS and 10 MHz output to the TICC are of identical length. Otherwise, the difference in propagation delay will reduce the correlation between the two signals, and it can no longer be assumed that $\bar{\phi}_1[i] = \bar{\phi}_2[i]$. Figure 6.2 illustrates the exact test setup.

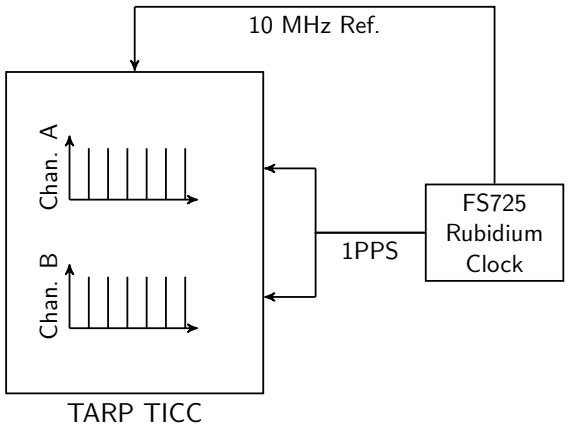


Figure 6.2: Illustration of the test setup for measuring the TARP TICC's measurement noise. Note that the illustration is not to scale, and identical cable lengths were used in the physical setup.

Measuring over 6 hours, the following Allan variances of both channels were found:

Table 6.1: TICC's estimated Allan Variance. Four decades of observation periods, τ, shown.

τ (s)	Chan A	Chan B
1	7.66e-21	7.80e-21
10	8.01e-23	8.24e-23
100	8.58e-25	8.71e-25
1000	1.21e-26	1.26e-26

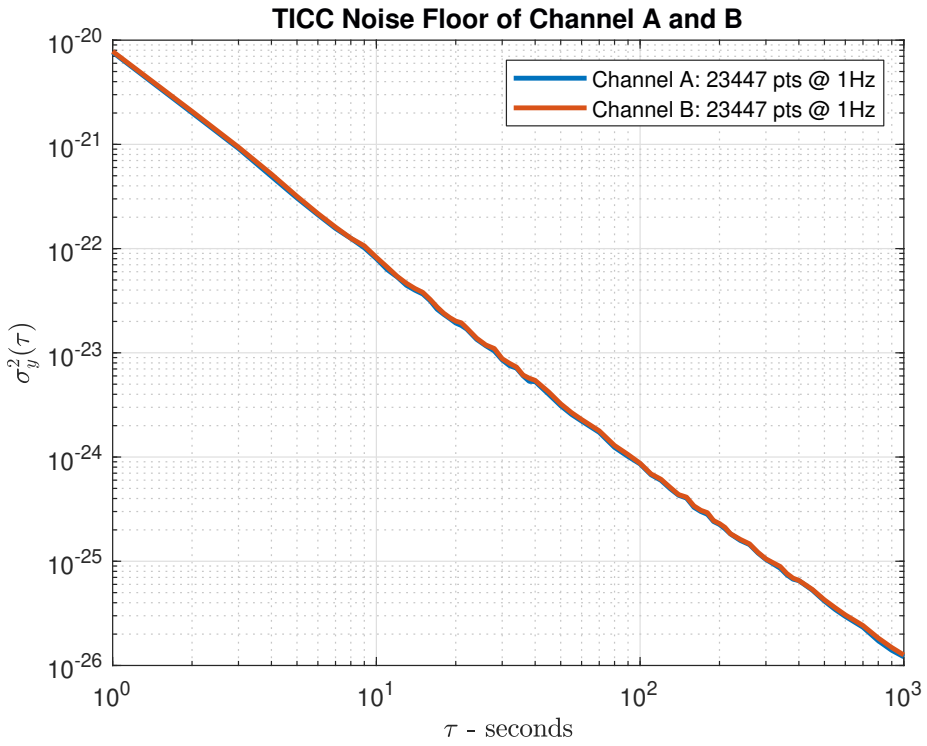


Figure 6.3: Sigma-Tau plot of the TICC's Allan Variance.

The plot and table show a curve with an approximate slope of -2, indicating that the noise is either White or Flicker PM noise (likely white). Note that in the last τ decade between 100 and 1000, the estimated AVAR does not quite decease by a decade. This is likely because these final measurements are less statistically significant. Overall, the measured AVAR is in line with the noise characteristic one would expect from the measuring equipment. The values themselves are a bit higher than the advertised Allan variance. However, it is close enough that the discrepancy could be caused by an imperfect measurement setup where $\bar{\phi}_1[i]$ and $\bar{\phi}_1[i]$ are not perfectly correlated. In any case, as the equipment noise floor has now been approximated it is appropriate to initiate measurements of the system noise sources.

6.1.3 NIC Clock Noise

The NIC's Allan variance is measured for both in the case of it being synchronised to the CTR and in the case where it is free running. This is to get a sense of how the control applied by the *ts2phc* application changes is noise characteristics. Figure 6.4 illustrates how cases are measured.

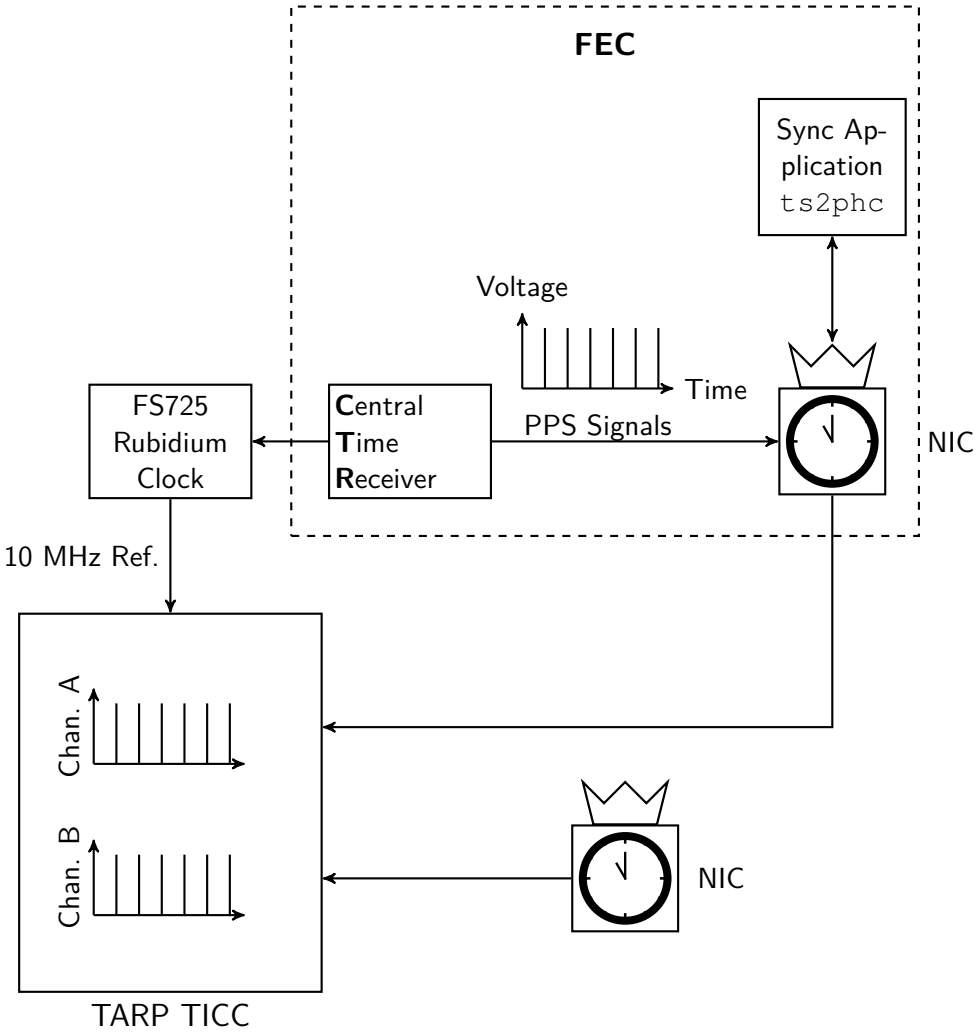


Figure 6.4: Illustration of test setup used to measure NIC noise with and without synchronisation.

The estimated Allan Variances are shown illustrated and listed in table 6.2 and figure 6.5 respectively.

Table 6.2: Allan Variance of the two NIC's (both synchronised and not synchronised). Six decades of observation periods, τ, shown.

τ (s)	NIC 1 RAW	NIC 1 Synced	NIC 2 RAW	NIC 2 Synced
0.01	4.99e-16	1.16e-13	3.23e-16	1.87e-13
0.1	6.06e-18	1.55e-15	3.40e-18	1.68e-15
1	6.14e-19	1.92e-17	3.59e-19	1.90e-17
10	1.58e-18	3.50e-19	9.78e-19	3.03e-19
100	3.00e-17	3.73e-21	2.13e-17	3.28e-21
1000	2.93e-16	4.97e-23	4.55e-16	4.16e-23

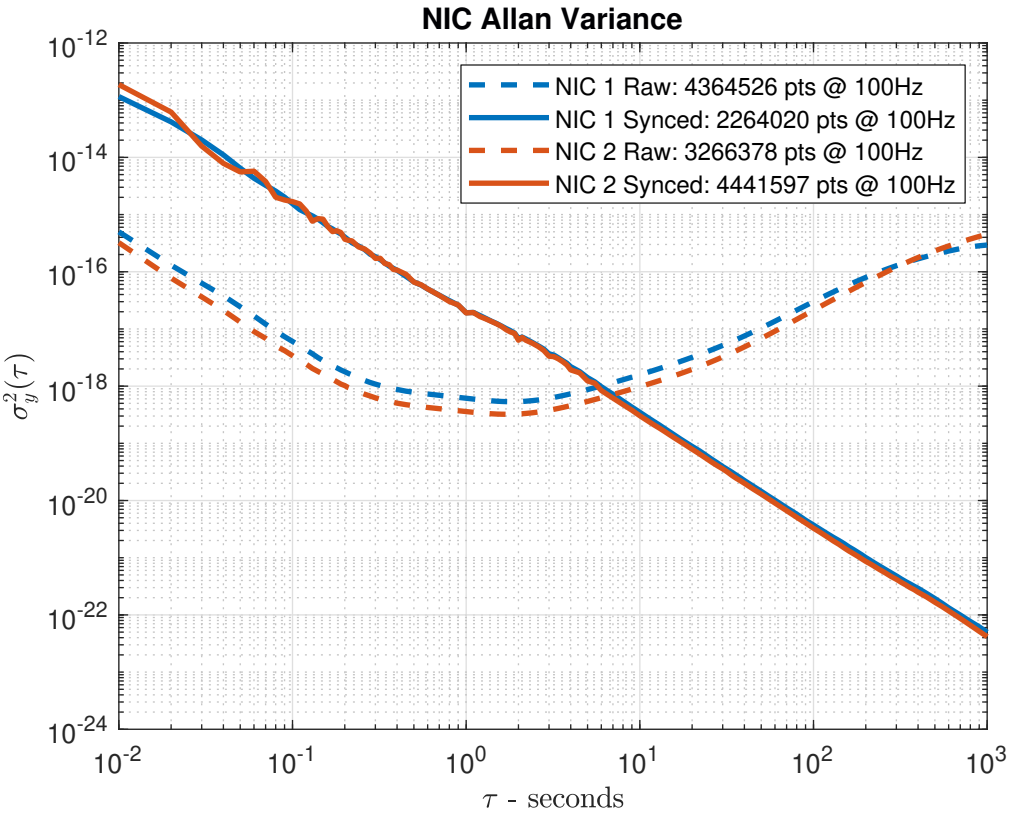


Figure 6.5: Sigma-Tau plot of the NIC's Allan Variance.

As seen in the above chart and table, it is clear that the synchronisation drastically improves the NIC's long term phase noise behaviour, at the price of an increase in short term instability. These results are partially explained by considering the long and short term time domain behaviour of the processes. These are illustrated in figure 6.6 and figure 6.7.

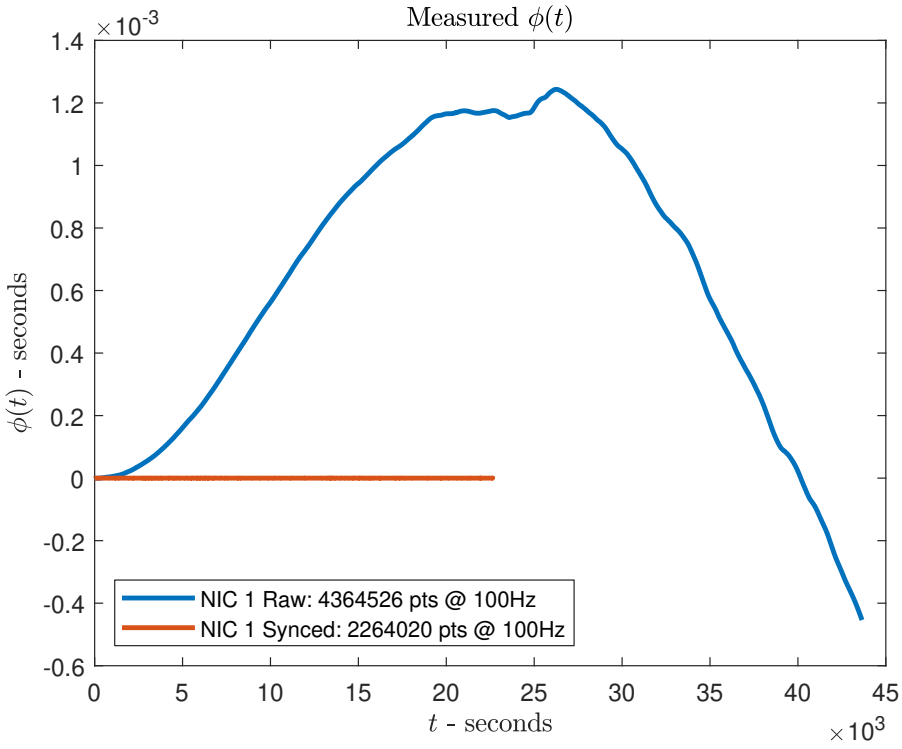


Figure 6.6: Long term evolution of the phase error of a synchronised and non synchronised NIC.

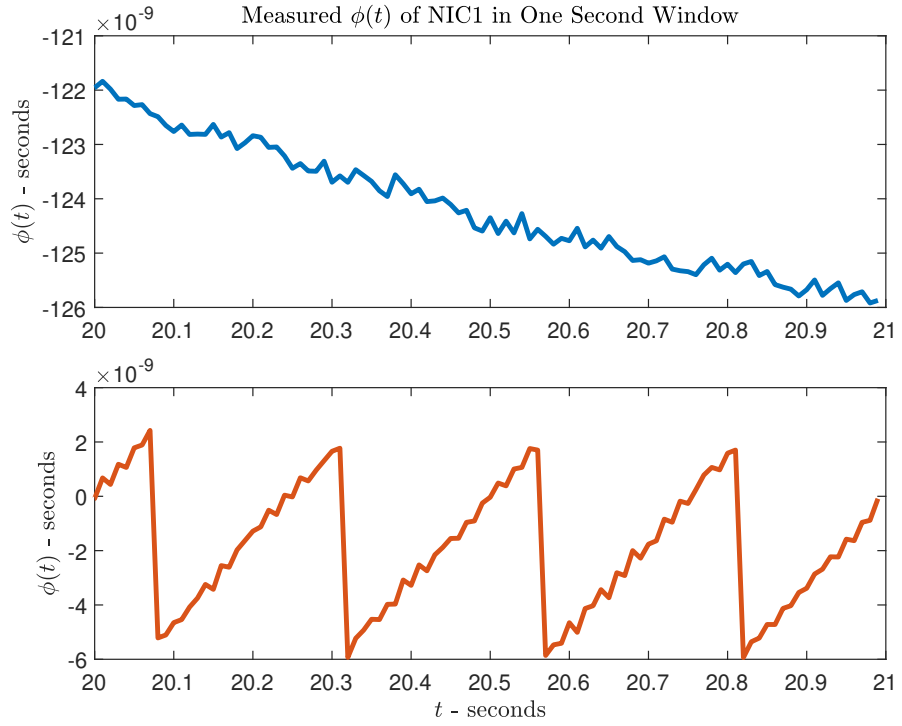


Figure 6.7: Short term evolution of the phase error of a synchronised and non synchronised NIC.

The plot of the long term behaviour clearly shows how the unsynchronised NIC's phase

drifts over time, while the synchronised counterpart does not. On the other hand, when considering a more zoomed in view of the two processes in figure 6.7, it is clear that the synchronised NIC has far more high frequency variations due to its control loop periodically altering the signal.

6.1.4 DQAME Clock Noise

Given that the NIC was an off the shelf component, it was easy to test as it was not possible to change or modify its behaviour. However, we have full control over the firmware and part of the software being used on the DQAMEs. This creates options for optimising the phase noise. The DQAMEs use an ATSAME54 MCU clocked by an external 40 MHz TCXO. To minimize the effect of clock phase noise, this TCXO would ideally be directly used to clock the timestamping systems. However, as a higher system clock is required, a Fractional Digital Phased Locked Loop (FDPLL) is used to increase the system frequency above 40 MHz.

The MCU's FDPLL has a DIGITAL FILTER setting. This setting provides 16 different configurations for the FDPLL loop filter. This filter changes the FDPLL settling time and phase noise. Further details of the other aspect of the system FDPLL are presented in section 7.1. To find the best performing configuration in terms of phase noise, each of the 16 possible filter configurations were tested. Unfortunately, these tests were performed before the TICC and stable rubidium reference was acquired, and thus the collected results can realistically only be used for comparing the phase noise performance of the different filters configurations. Thus, the extensive results of these measurements have been omitted. Instead the TICC was used to obtain accurate noise measurements for the chosen FDPLL configuration, the worst performing configuration, and the raw clock with no FDPLL configured. Note that all these measurements are made in open loop, and as such the DQAME boards are not synchronised to any external reference. Figure 6.8 illustrates the measurement setup for these tests.

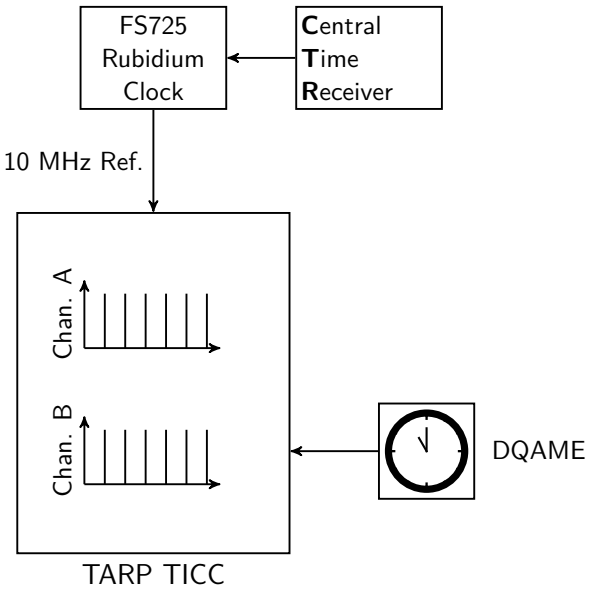


Figure 6.8: Illustration of test setup used to measure DQAME clock noise.

The estimated AVAR of these measurements are shown in table 6.3 and figure 6.9.

Table 6.3: Estimated Allan Variance of the best performing FDPLL, no FDPLL, and the worst performing FDPLL. Six decades of observation periods, τ, is shown. Note that the phase noise measurements of the configuration with no FDPLL were performed at 50 Hz due to a firmware limitation. Therefore, the ADEV for $\tau = 0.01$ is not estimated for this specific configuration.

τ (s)	DQAME2	DQAME2 Noisy FDPLL	DQAME2 No FDPLL
0.01	2.4523e-13	7.8969e-12	-
0.1	2.4490e-15	7.9042e-14	7.8202e-19
1	2.4515e-17	7.9164e-16	1.7845e-20
10	2.7187e-19	7.9686e-18	1.0526e-20
100	1.5265e-19	3.9465e-19	3.5456e-20
1000	1.1580e-18	1.4197e-18	1.3019e-18

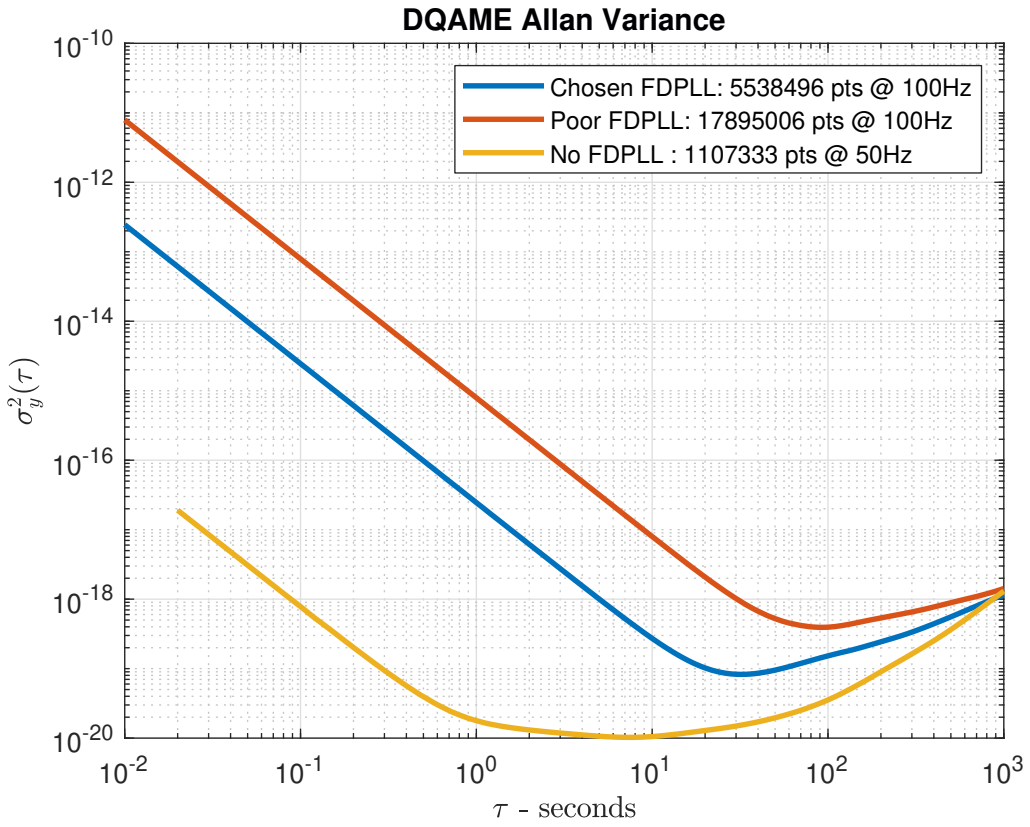


Figure 6.9: Sigma-Tau plot of the DQAME Allan Variance for the best performing FDPLL, no FDPLL, and the worst performing FDPLL. Note that the phase noise measurements of the configuration with no FDPLL were performed at 50 Hz due to a firmware limitation.

Based on the above data, it is clear that the addition of a FDPLL contributes a significant amount of clock noise. However, given that a FDPLL must be used to reach the target clock frequencies the effort to optimize the FDPLL configuration has resulted in a notably decrease in clock noise as compared to the worst case.

The above tests used to compare raw clock and FDPLL noise were conducted on a single

DQAME. To obtain more representable data of the noise across a range of devices, the tests were repeated on three other boards configured with the chosen FDPLL configuration:

Table 6.4: Estimated Allan Variance of four different DQAMEs using the best identified FDPLL configuration. Six decades of observation periods, τ, shown.

τ (s)	DQAME 1	DQAME 2	DQAME 3	DQAME 4
0.01	1.86e-12	2.45e-13	2.56e-13	3.03e-13
0.1	1.87e-14	2.45e-15	2.56e-15	3.03e-15
1	1.87e-16	2.45e-17	2.57e-17	3.05e-17
10	1.94e-18	2.72e-19	5.33e-19	3.90e-19
100	3.71e-19	1.53e-19	1.10e-18	1.95e-19
1000	2.33e-18	1.16e-18	4.83e-18	1.34e-18

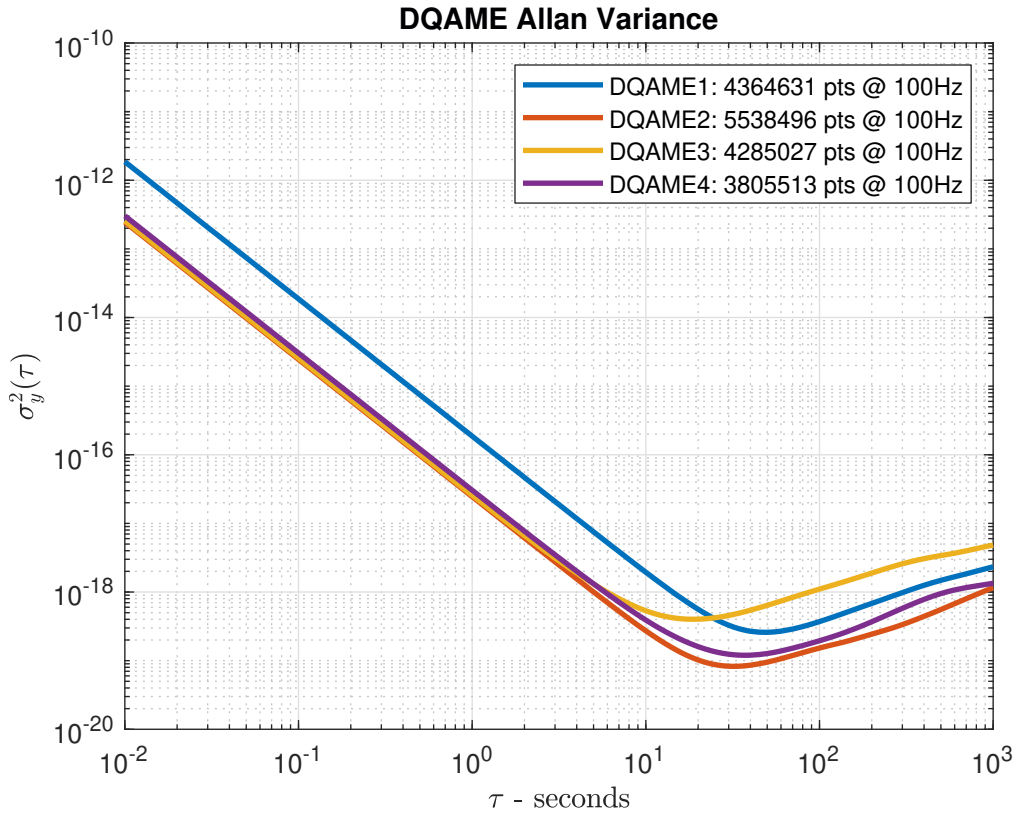


Figure 6.10: Sigma-Tau plot of the Allan Variance for four different DQAMEs using the best identified FDPLL configuration.

As seen in the plots, three of the units have very similar behaviour, while unit 1's performance differs. The reason for this difference is unknown. Ideally the different FDPLL configurations would be tested on all available DQAMEs to determine if the chosen configuration truly is the optimal configuration across all devices. However, due to time constraints this was not possible.

6.2 Nominal PTP Noise

Having quantified the phase noise of the systems two clocks, the next characterisation step is to quantify the measurement noise introduced by the PTP network. This section concerns itself with measurement noise under ideal network conditions where the synchronisation packets are the only traffic present in the network.

To measure measurement noise the FEC/NIC was configured to transmit 32 synchronisation packets per second along with its corresponding $x_1(t_1^i)$ timestamps to a DQAME. Upon reception of the synchronisation packets, the DQAME then timestamps them with $x_2(t_2^i)$. Immediately following this, the DQAME creates a new timestamp $x_2(t_3^i)$ and transmits a delay request to clock 1 instructing it to create timestamp $x_1(t_4^i)$ and transmit it back to clock 2. The measured bidirectional phase errors between the two clocks at the ith sample are then found as

$$\phi(t_1^i, t_2^i) = x_2(t_2^i) - x_1(t_1^i) \tag{6.9}$$

$$\phi(t_3^i, t_4^i) = x_2(t_3^i) - x_1(t_4^i) \tag{6.10}$$

The phase error is measured in both directions, to help inform whether the PDV is symmetrical.

In parallel to the PTP network measurements, the noise of the NIC and DQAME in use by the PTP network was measured using the TICC. This addition allows for direct comparison between the devices clock noise, and the additional noise introduced by the PTP network. This test is performed in open loop, and no synchronisation is implemented in the DQAME. Figure 6.11 illustrates the measurement setup.

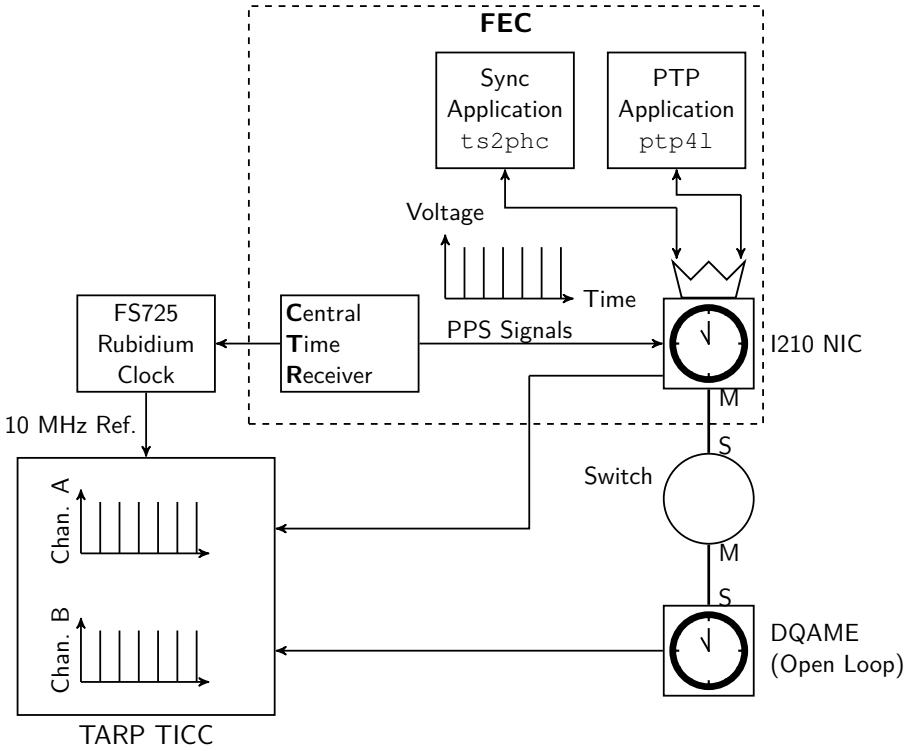


Figure 6.11: Illustration of the measurement setup used to quantify the phase noise when measured through the PTP network.

The estimate of the AVAR is found in the same way as described previously in section 6.1.1. Likewise, prior to calculating the AVAR, $\phi(t_1^i, t_2^i)$ and $\phi(t_3^i, t_4^i)$ is first detrended and any outliers are removed. The estimated AVAR from one of these tests is shown and illustrated in table 6.6 and figure 6.13 below:

Table 6.5: Allan Variance estimated using data from the PTP network and simultaneously from the NIC and DQAME using data from the TICC. Six decades of observation periods, τ, is shown. PTP M2S (Master to Slave) refers to the $\phi(t_1^i, t_2^i)$ estimate and S2M (Slave to Master) refers to the measurements in the opposite direction, i.e. $\phi(t_3^i, t_4^i)$

τ (s)	PTP M2S	PTP S2M	NIC 1 Synced	DQAME1
0.01	-	-	1.1555e-13	1.1349e-12
0.1	5.7939e-14	1.8533e-13	1.5477e-15	1.1344e-14
1	5.1079e-16	1.5841e-15	1.9216e-17	1.1370e-16
10	5.9573e-18	1.6268e-17	3.4952e-19	1.6883e-18
100	2.0050e-18	2.1078e-18	3.7260e-21	1.9482e-18
1000	2.4787e-17	2.4788e-17	4.9665e-23	2.4720e-17

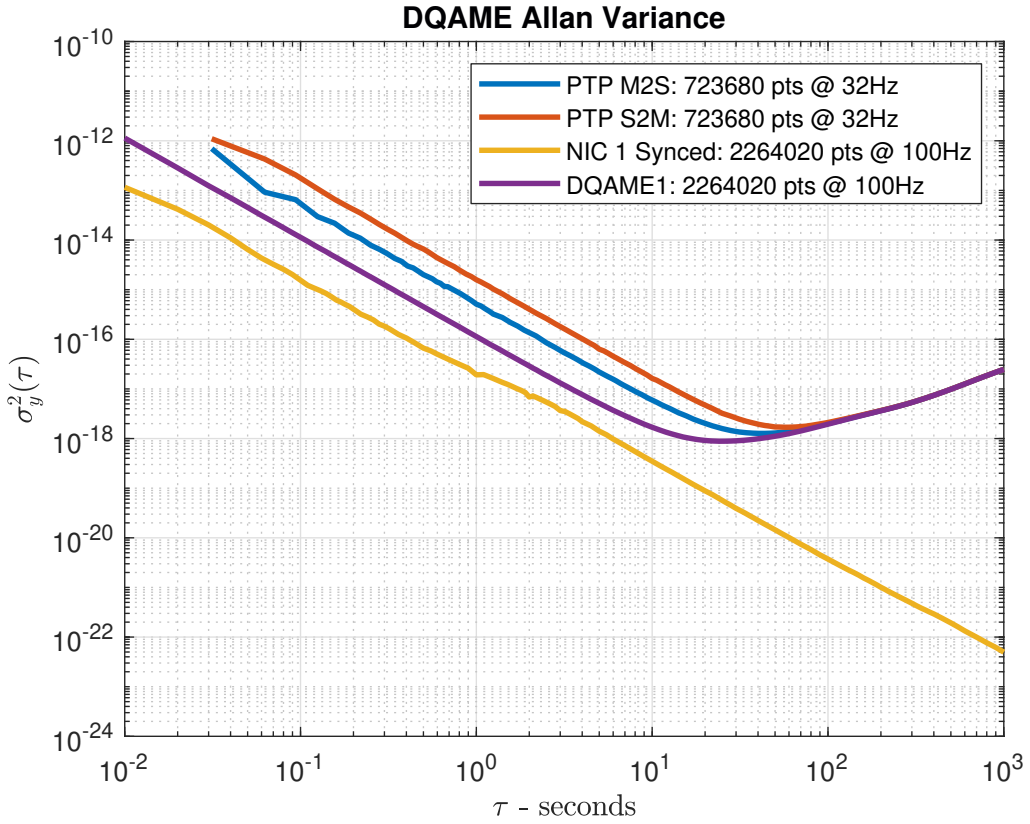


Figure 6.12: Sigma-Tau plot of the Allan Variance estimated using data from the PTP network and simultaneously from the NIC and DQAME using data from the TICC.

The estimated Allan variance of the measured PTP noise follows the expected trend. In the short term, the noise introduced by propagation delay and timestamping is the dominating source. Then over longer timescales, the DQAMEs random walk noise process

becomes the dominating contributor. Unexpectedly, however, the PDV is asymmetrical. It appears that there is approximately introduced twice as much measurement noise when initiating timestamping from the DQAME as opposed to initiating from NIC. This indicates that measurements initiated by the NIC are more reliable. As measurements in both directions are needed to estimate the mean propagation delay, this opens possibilities for using statistical method to find superior propagation delay estimates, as compared to the averaging method introduced in equation (4.7).

These measurements were repeated with different combinations of DQAMEs and NIC's. The resulting estimate of the PTP noise's Allan variance was remarkably consistent with the only major deviation being the τ at which the DQAME's random walk noise would start to dominate.

In an effort to enhance the PTP network's contribution to the measured noise, the above measurements were repeated, but this time using an unsynchronised NIC and a DQAME relying exclusively on the on-board 40 MHz TCXO for its internal clocking (i.e. no FD-PLL). These changes should lower the clocks contribution to the measured noise. This should allow for more accurate isolation of the propagation/measurement noise component. The results of these measurements are shown below:

Table 6.6: Allan Variance estimated using data from the PTP network and simultaneously from the NIC and a DQAME using data from the TICC. In these measurements the clock noise was reduced by using an unsynchronised NIC, and the DQAME was configured to be clocked directly by its TCXO. Six decades of observation periods, τ, is shown.

τ (s)	PTP M2S	PTP S2M	NIC 1 Synced	DQAME1
0.01	-	-	3.9444e-16	-
0.1	5.0311e-14	1.8938e-13	4.0457e-18	7.8202e-19
1	6.2418e-16	1.6746e-15	3.8167e-19	1.7845e-20
10	7.0751e-18	1.8641e-17	1.0395e-18	1.0526e-20
100	6.3245e-18	6.4370e-18	6.6885e-18	3.5456e-20
1000	1.2869e-16	1.2869e-16	1.5368e-16	1.3019e-18

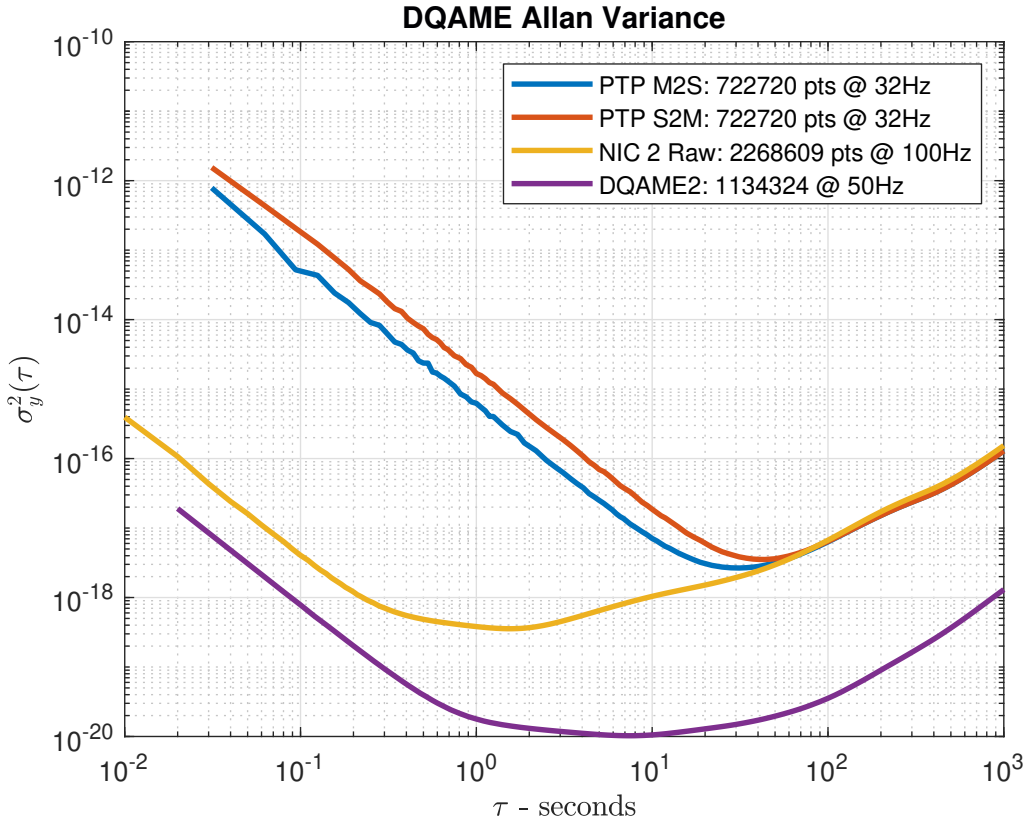


Figure 6.13: Sigma-Tau plot of the Allan Variance estimated using data from the PTP network and simultaneously from the NIC and DQAME using data from the TICC. In these measurements the clock noise was reduced by using an unsynchronised NIC, and the DQAME was configured to be clocked directly by its TCXO.

The above results initially appear rather strange. When comparing the PTP noise between table 6.6 and table 6.5, it seems that the noise is either unchanged or possibly even slightly increased for the case of using low noise clock sources. This is the opposite of the expected. One potential explanation of this result involves consideration of quantisation noise.

Thus far the effect of quantisation noise has mostly been ignored, as it is hard to describe mathematically, and its effect was thought negligible. However, the above measurements suggest that it is in fact the dominant source of short term noise in the system. To express the phase error where quantisation is incorporated, the timestamping resolution of the NIC and DQAME must be known. These values are hard to known definitively, as neither of the devices' datasheets express this directly. In lieu of a more definitive information it is assumed that the devices timestamping resolution is equivalent to the systems' clock period. Thus, the quantised expression for phase difference becomes:

$$\begin{aligned}\hat{\phi}(t_1^i, t_2^i) &= \hat{x}_2(t_2^i) - \hat{x}_1(t_1^i) \\ &= \left\lfloor t_2^i + \phi_2(t_2^i) \right\rfloor_{\frac{1}{f_2}} - \left\lfloor t_1^i + \phi_1(t_1^i) \right\rfloor_{\frac{1}{f_1}}\end{aligned} \tag{6.11}$$

where:

$\lfloor\ \rfloor_c$ floors to the nearest value exactly divisible by c. i.e. $\lfloor x \rfloor_c = x - (x \bmod c)$

f_i is the clock frequency of clock i

If the noise contributions of $\phi_1(t)$, $\phi_2(t)$, and $\eta(t)$ is sufficiently low, then based on the above equation, it is clear that their contribution is effectively filtered by the floor function. In this case, the variation caused by flooring will then become the dominant source of noise. Quantisation noise also helps explain why the measured noise appeared to increase when the TCXO on the DQAME was directly used. While the TCXO does have lower phase noise, its frequency is only 40 MHz. The other tests were conducted using a FDPLL configured to output 100 MHz. Thus, the TCXO version had a comparatively lower timestamping resolution. Ironically, while the use of a FDPLL to increase the clock frequency was found to dramatically increase the phase noise of the clock, the added timestamping resolution may in fact have lowered the overall system noise.

While a possible and seeming plausible explanation, quantisation does not directly explain why the PDV is asymmetrical. It is speculated that the timestamping resolution of the devices may differ depending on whether they are receiving or transmitting, however, at this time it is unknown. Ideally the noise contribution of this resolution limitation could be calculated exactly to verify that the measured noise does in fact stem from quantisation. However, due limited knowledge of the hardware behaviour as well as the inherent discontinuities in the floor function, it is difficult to model quantisation in the system.

6.3 PTP Load Tests

In the previous sections, measurements were performed under ideal conditions, and outliers were filtered in post processing to obtain an accurate view of the time and timestamping stability. However, during normal operations, networking conditions and load can change dynamically. High load can cause the PDV of PTP packets, i.e. $\eta_{1,2}(t_1)$ and $\eta_{2,1}(t_3)$, to change. Furthermore, filtering outliers only serves to help determine the underlying measurement noise and does not provide an accurate picture of the performance during operation, as any synchronisation scheme will have to account for such event. This section documents the behaviour of $\eta_{1,2}(t_1)$ and $\eta_{2,1}(t_3)$ for a variety of networking conditions.

To generate network traffic, a Spirent TestCenter 2000 is used. This device is equipped with two 10 Gbit/s interfaces that can be configured to send/receive custom network traffic. Two distinct networking scenarios are investigated in this section. First, constant data rate traffic going into the NIC is considered. Three levels of constant load are tested: 0, 10, and 100 Mbit/s. It is important to test the behaviour of $\eta(t)$ when the NIC receives heavy traffic, as this is a feasible operating scenario. Conversely, measuring timestamping behaviour when the NIC transmits heavy network load has not been tested, as it is not expected to occur. Furthermore, the effect of random burst traffic is also not investigated. While bursty traffic is expected to occur during operation, in the interest of time, these tests have been omitted. Secondly cross traffic in the switch between devices not related to the PTP network is examined. For this test, each of the two Spirent TestCenter's 10 Gbit/s interfaces were configured to transmit 10 Gbit/s to the other interface. Thus resulting in a total of 20 Gbit/s per second of cross traffic load. For all the aforementioned tests, the network traffic generator was configured to send a constant stream of 1024 byte ethernet frames to the NIC. Furthermore, like the tests in the previous sections, each scenario was

tested for at least 6 hours and PTP was configured to exchange synchronisation packets at 32 Hz.

The tests described in this section are performed using the test setup illustrated in figure 6.14.

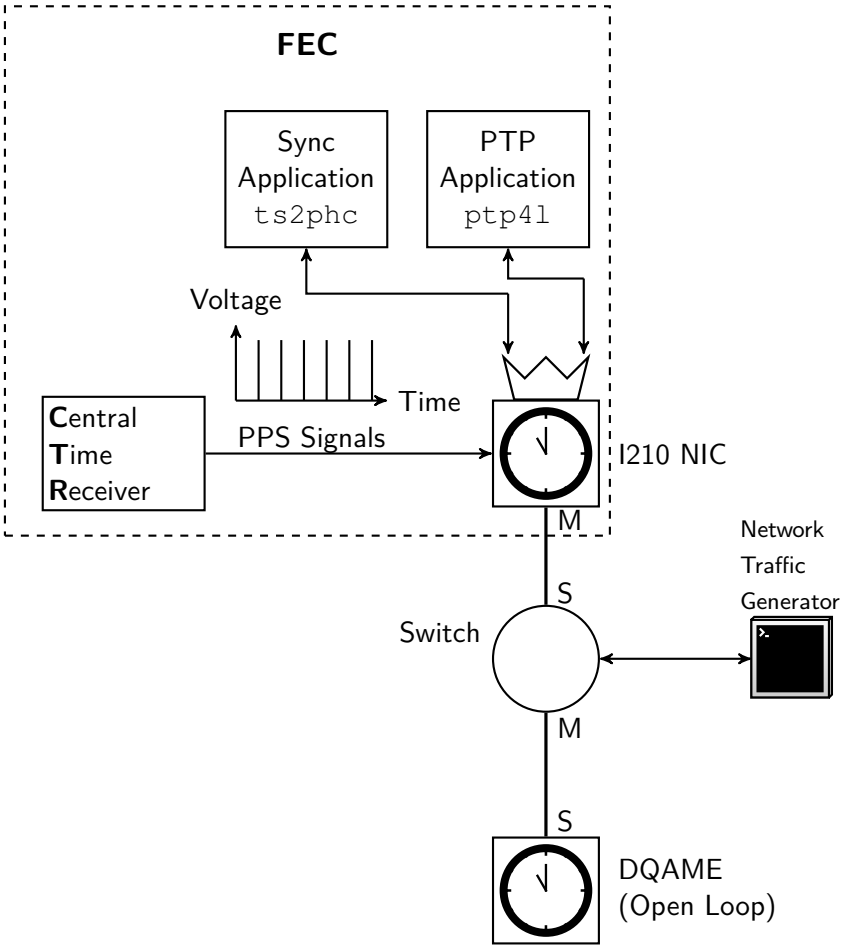


Figure 6.14: Illustration of the measurement setup used to measure outliers in the PTP network. The network traffic generator is used to create different network conditions.

To continue the trend of the last two section, the AVAR of the measurement is initially considered. The Sigma-Tau plots of all measurements are shown in figure 6.15

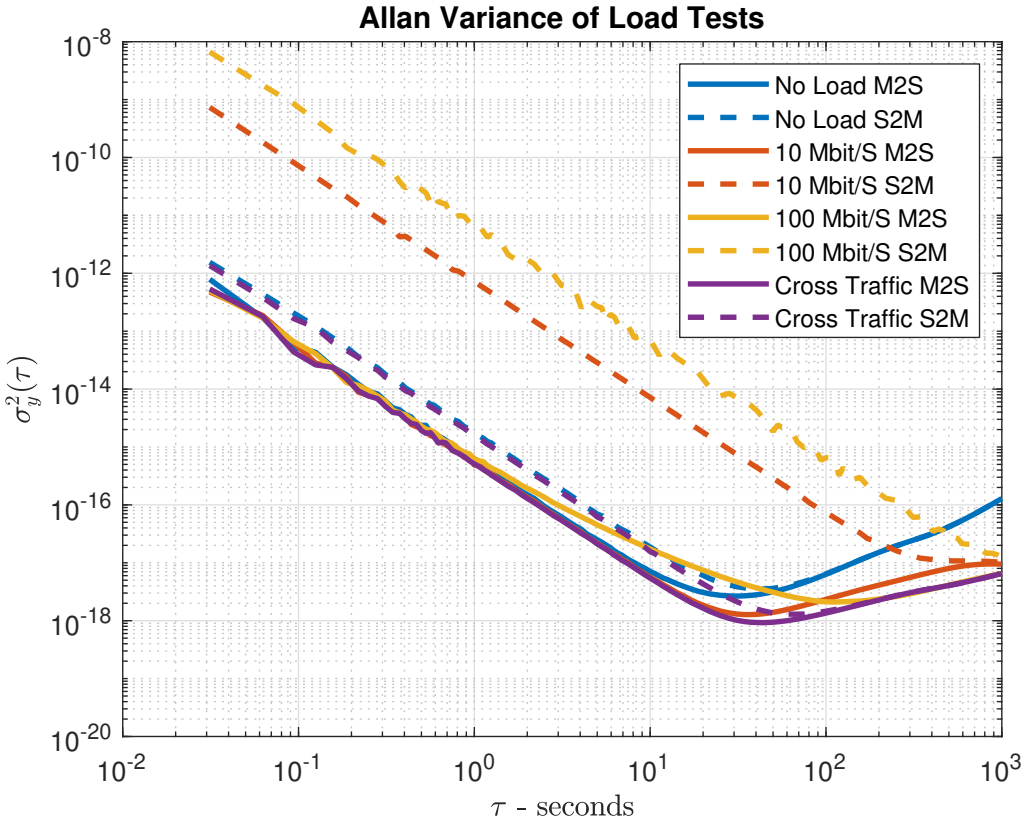


Figure 6.15: Sigma-Tau plot of the Allan Variance of the load tests.

The graphs clearly show that significantly more noise is present when the NIC is under load and simultaneously timestamping incoming packets from the DQAME.

To gain further insight into how and why this additional noise manifests itself, the 10 Mbit/s load scenario is further investigated. However, note that similar results are found for the 100 Mbit/s, though these have been omitted for the sake of brevity.

First, to obtain a high level overview of the load behaviour, time domain plots of the phase error estimates are shown in figure 6.16.

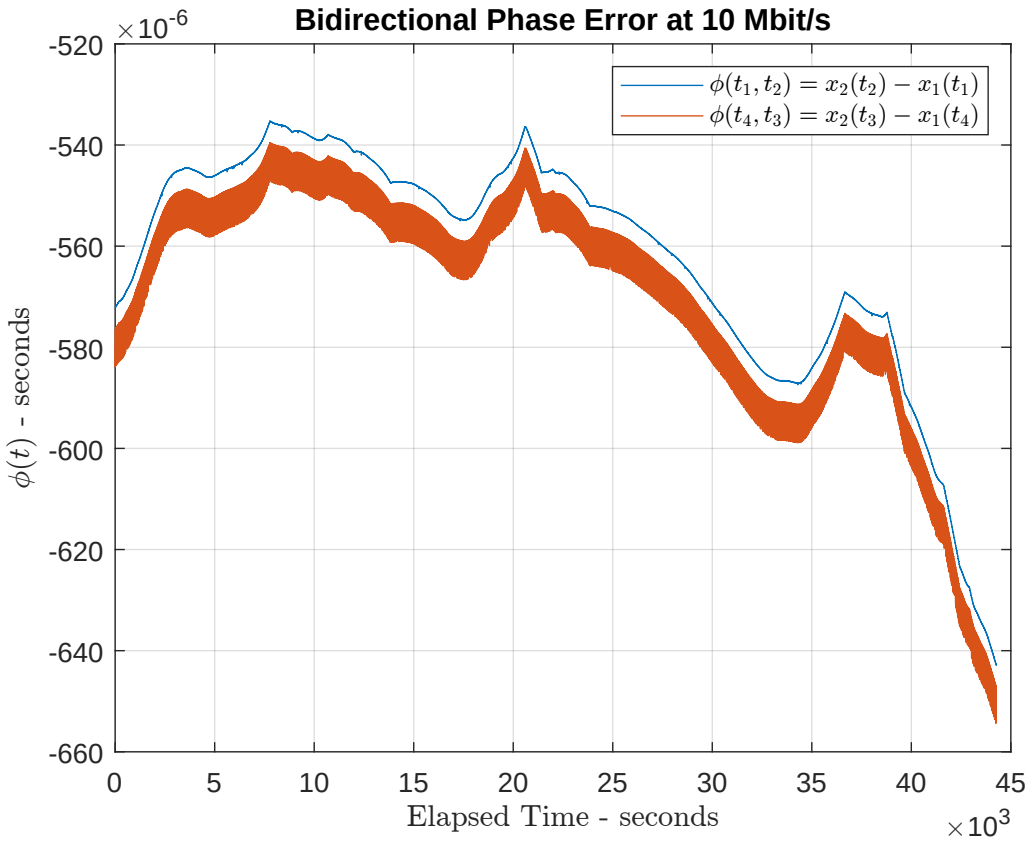


Figure 6.16: Line plot of $\phi(t_1, t_2)$ and $\phi(t_4, t_3)$ for all samples collected during the 10 Mbit/s load test. Outliers have not been filtered.

From this graph it is obvious that the estimate made between the slave and master are significantly noisier than the measurements made in the other direction. However, due to the high number of samples in this global representation of the data, it is hard to see exactly what is going on between each sample. To obtain a more detailed view of the noise, we zoom in on a subset of the data. Furthermore, to eliminate the random walk behaviour of the data the first of difference of the phase error is considered:

$$\Delta\phi_{j,k}[i] = \phi(t_j^{i+1}, t_k^{i+1}) - \phi(t_j^i, t_k^i) \tag{6.12}$$

Figure 6.17 and figure 6.18 graphs $\Delta\phi_{1,2}[i]$ and $\Delta\phi_{4,3}[i]$ respectively. To get sense for the how the network load modifies the samples, data from the scenario with no network load is also shown. To allow for easier visual comparison an offset has been added to the no network load data. Further, note the two graphs differing x and y axis scales.

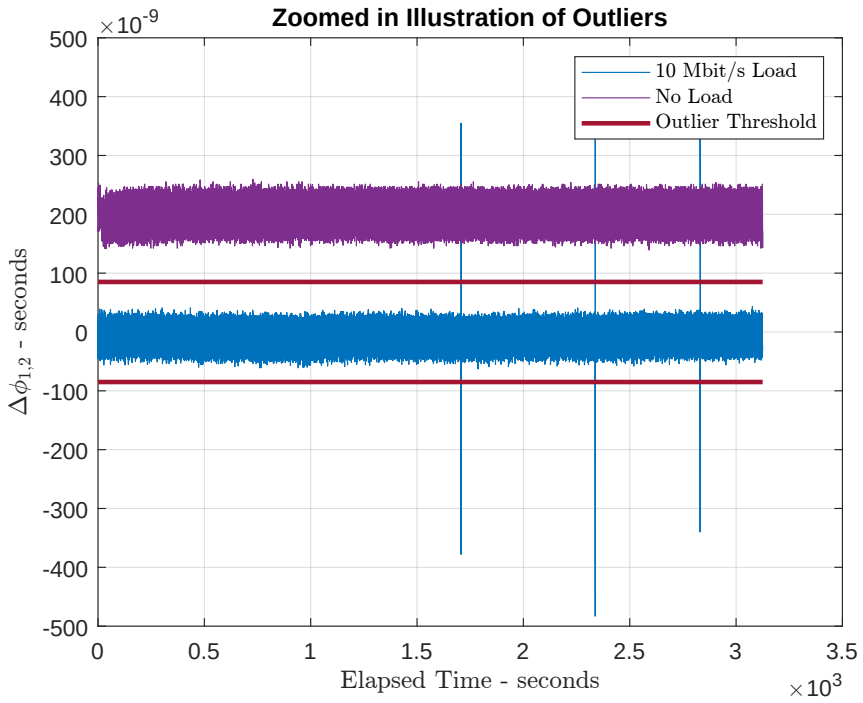


Figure 6.17: Line plot of $\Delta\phi_{1,2}[i]$ of the samples collected within the first 3125 seconds of the 10 and 0 Mbit/s load test. Samples outside the region enclosed by the red lines are considered outliers.

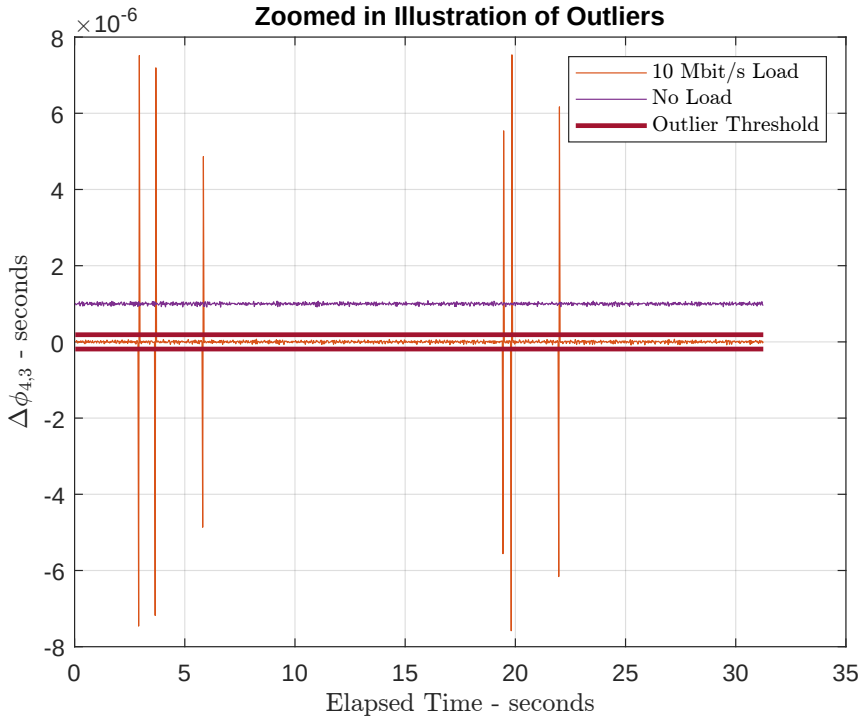


Figure 6.18: Line plot of $\Delta\phi_{3,4}[i]$ of the samples collected within the first 31 seconds of the 10 and 0 Mbit/s load test. Samples outside the region enclosed by the red lines are considered outliers.

In both of the two graphs, it is clear that the measured $\Delta\phi_{j,k}[i]$ is incidentally polluted by the occurrence of outliers. Though it is also clear that outliers occur much more frequently

and with higher amplitude in $\Delta\phi_{4,3}[i]$ as opposed to $\Delta\phi_{1,2}[i]$. Outside of these outliers it seems that the data varies approximately as much as the non loaded scenario.

Until now the AVAR has been used to characterise system noise. However, this was used under the assumptions that the noise spectrum adhered to a power law. When outliers are considered, it is clear from the time series that this is not the case. Thus, we turn our attention to exclusively characterising the outliers using other tools. To do this, the data's outliers are extracted and the frequency of their occurrences as well as their magnitude are partially described.

To quantify outliers, it is first necessary to exactly define what an outlier is. Generally, outliers are data points that are substantially different from the remaining data set. However, what 'substantially different' means is somewhat subjective. One common approach is to define outliers as data points with values that a more than some number of standard deviations away from the sample mean of the data. In this thesis it was instead decided to use the Median Absolute Deviation (MAD) since it was found to provide more consistent performance for different data sets as compared to standard deviation. The mean and standard deviation main weakness is that the contribution of each individual outliers directly changes their values. In this regard the Median Absolute Deviation (MAD) is more robust, as it relies on the median data point, and thus half the observed data points must be outliers before the estimator is corrupted [9]. Mathematically Median Absolute Deviation (MAD) is defined as [9]:

$$\text{MAD} = \text{median}\left(|x - \text{median}(x)|\right) \tag{6.13}$$

Outliers are then defined as any data-point of $\Delta\phi_{i,j}[i]$ that is more than six MAD away from the median of the dataset. Six was chosen empirically and appears to be a good compromise between over and under sensitivity. Examples of outlier thresholds are shown in figure 6.17 and figure 6.18. Due to the differencing each outlier in $\phi_{i,j}[i]$ becomes two consecutive outliers in the resulting $\Delta\phi_{i,j}[i]$. To account for this, only the first occurrence is counted.

Given that the system relies on discrete samples the fraction of outliers occurrences with respect to the total number of data points are found to help inform about the frequency of outlier occurrences. For each of the tested scenarios, this fraction is found for the both the case when the NIC transmits a synchronisation packet to the DQAME, $p_{1,2}$, and in the opposite direction, $p_{2,1}$. Table 6.7 presents the fractions:

Table 6.7: The fraction of detected outliers compared to the total number of data points. In the fraction columns the numerator and denominator of the fraction shows the exact number of outliers and the total number of samples respectively.

Network Scenario	$p_{1,2}$	$p_{3,4}$
0 Mbit/s NIC RX Load	$\frac{4}{722719} = 6.25e-06$	$\frac{0}{722719} = 0$
10 Mbit/s NIC RX Load	$\frac{31}{1416648} = 2.1883e-05$	$\frac{14475}{1416648} = 0.0102$
100 Mbit/s NIC RX Load	$\frac{56}{2862664} = 1.96e-05$	$\frac{262052}{2862664} = 0.092$
20 Gbit/s Cross Traffic	$\frac{8}{2128456} = 3.76e-06$	$\frac{0}{2128456} = 0$

The table clearly shows that the fraction of outliers occurring between the DQAME and NIC depend on the NIC RX load. Interestingly, the fraction seems to almost have a 1:1 relationship to the relative link saturation. At 10 Mbit/s the 1G NIC is under 1% load, which roughly corresponds to the error rate. Similarly, at 100 Mbit/s the NIC is under 10% load which is not far off from the resultant outlier rate. Intuitively this relationship makes sense as the probability of a PTP packet being transmitted at the same time as a load packet equals the relative load. When multiple packets are destined for the same destination the packets are queued in the switch thus creating additional propagation delay. The data also shows that the other direction, i.e. $p_{1,2}$ also experiences occasional outliers and thus these must also be accounted for when implementing synchronisation. However, as these occurrences are exceedingly rare, their occurrence rate is not further commented.

If the outliers are assumed to be independent and thus adhere to the Bernoulli distribution, then then the number of samples between each outlier will fit a geometric distribution with parameter p. To help verify if the prior hypothesis is valid, a histogram of the number of samples between each outlier found in $\Delta\phi_{4,3}[i]$ is shown in figure 6.19. Like previously, only the data from 10 Mbit/s scenario is shown. Along with the histogram, the probability mass function of a geometric distribution parameterised with the measured fraction of outliers, $p_{4,3} = 0.0102$, is also plotted.

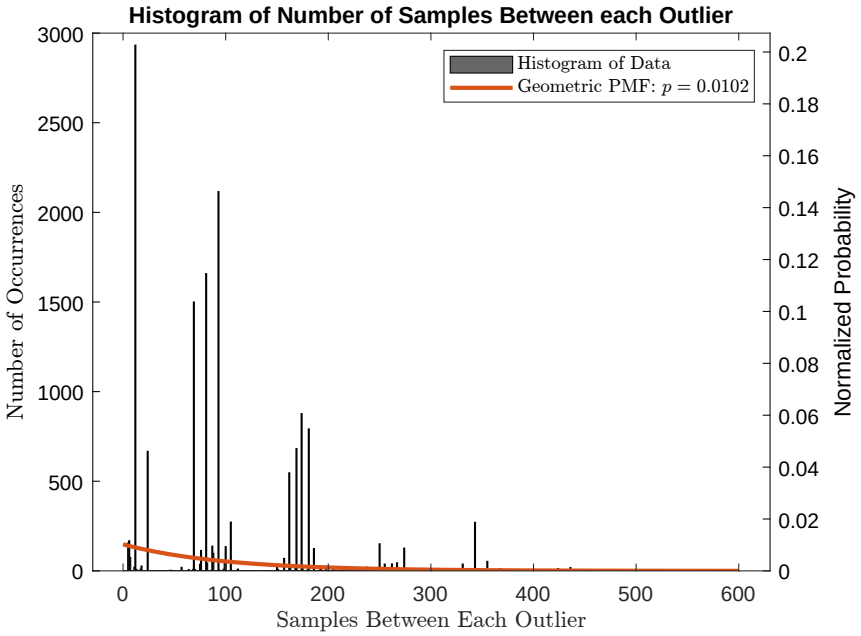


Figure 6.19: Histogram of the number of samples between each outlier in $\Delta\phi_{4,3}[i]$ for the 10 Mbit/s load test. The histogram is plotted with maximum resolution, i.e. each bin only groups the outliers with the exact same number samples between them. In addition to the histogram, the Geometric Probability Mass Function (PMF) for the estimated p is also shown.

Based on the above histogram, it is clear that the data does not adhere to the expected geometric distribution. Rather it seems there are a number delays that are more frequent than others. For example, twelve sample delays between an outlier had 2933 occurrences while the neighbouring options of eleven and thirteen sample delays only had 20 and 8 occurrences respectively. In spite of the lack of geometric fit, the envelope of the histogram roughly corresponds to the exponential decay that is expected from the geometric distri-

bution. Furthermore, the number of samples between events is not the only way to judge the frequency of outliers. Another approach is to determine how many outliers occur in a given interval. For a Bernoulli process, this is described by the binomial distribution where n is the number of number of samples and p is the probability of an outlier occurring for a given sample. Figure 6.20 and figure 6.21 shows the histogram of the number of outliers for $n = 32$ and $n = 1024$. Along with the histogram, the equivalent binomial probability mass function is also displayed.

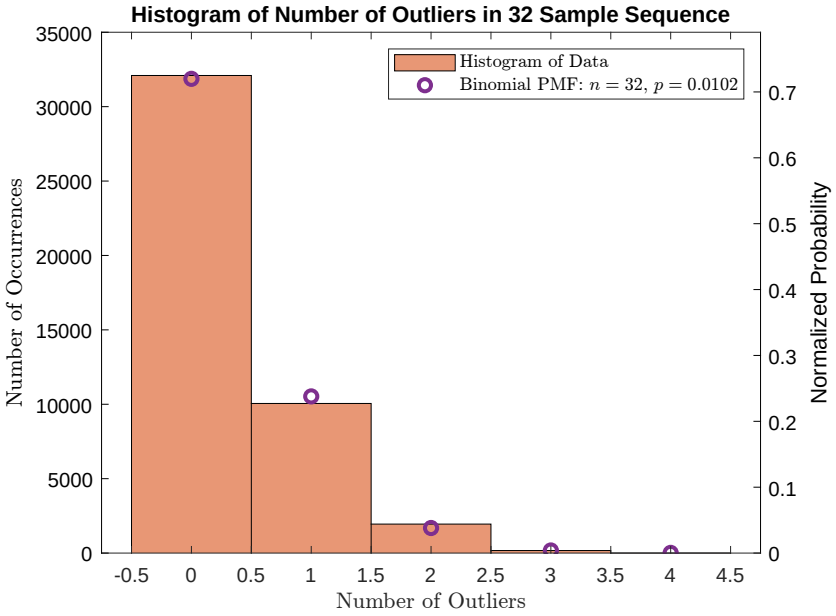


Figure 6.20: Histogram of the number of outlier occurring in a 32 sample interval for the 10 Mbit/s load test. In addition to the histogram, the Binomial Probability Mass Function (PMF) for the estimated p and chosen n is also shown.

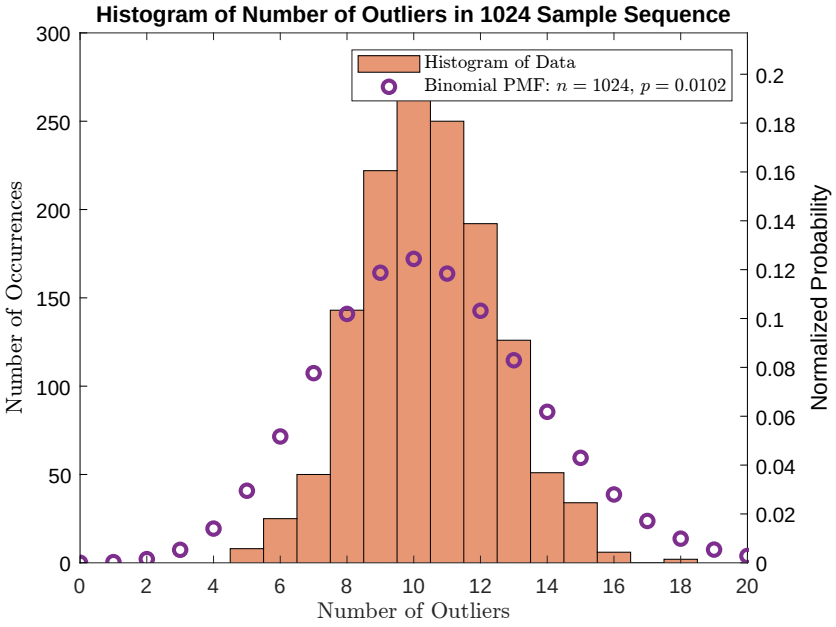


Figure 6.21: Histogram of the number of outlier occurring in a 1024 sample interval for the 10 Mbit/s load test. In addition to the histogram, the Binomial Probability Mass Function (PMF) for the estimated p and chosen n is also shown.

From the figures it appears that the data closely approximates the binomial distribution for low n. For the case where $n = 1024$ the data has still generally retained the same shape as the binomial probability mass function, albeit more concentrated towards the center.

Having considered the frequency of outlier occurrences, the outlier magnitude must also be characterised. First, simple statistics, such as the maximum and mean value of the outliers are collected in table 6.8. Statistics regarding the outlier of $\Delta\phi_{4,3}[i]$ in the no load and cross traffic scenario are omitted, as no outliers were observed.

Table 6.8: Simple statistics of the various network scenario's outlier magnitudes.

Network Scenario	Max	Mean	Median
0 Mbit/s NIC RX Load, $\Delta\phi_{1,2}[i]$	$537e-9$	$498e-9$	$498e-9$
20 Gbit/s Cross Traffic, $\Delta\phi_{1,2}[i]$	$520e-9$	$505e-9$	$509e-9$
10 Mbit/s NIC RX Load, $\Delta\phi_{1,2}[i]$	$625e-9$	$413e-9$	$369e-9$
100 Mbit/s NIC RX Load, $\Delta\phi_{1,2}[i]$	$884e-9$	$495e-9$	$495e-9$
10 Mbit/s NIC RX Load, $\Delta\phi_{4,3}[i]$	$7.60e-6$	$4.25e-6$	$4.29e-6$
100 Mbit/s NIC RX Load, $\Delta\phi_{4,3}[i]$	$7.60e-6$	$4.15e-6$	$4.14e-6$

The low quantity of observed outliers for the majority of the test cases makes it difficult to make any definitive conclusions. However, it appears that the magnitude of the sporadic and rarely occurring outliers in the first four scenarios of the table are of significantly lower magnitude than that of the outliers in $\phi_{4,3}[i]$ in the 10 and 100 Mbit/s load tests.

To obtain a better understanding of these outliers' magnitudes, a histogram of the $\phi_{4,3}[i]$ outliers' magnitude of the 10 and 100 Mbit/s load tests is shown in figure 6.22 and figure 6.23. Histograms have not been plotted for the remaining data, as there are too little available data to justify it.

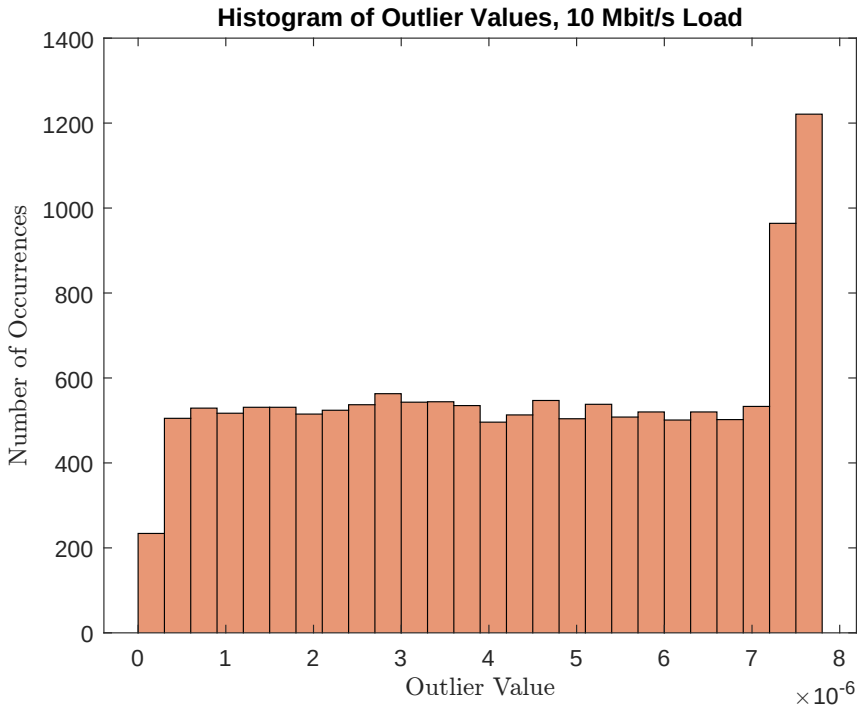


Figure 6.22: Histogram of the magnitude of all outliers in $\Delta\phi_{4,3}[i]$ for the 10 Mbit/s load test.

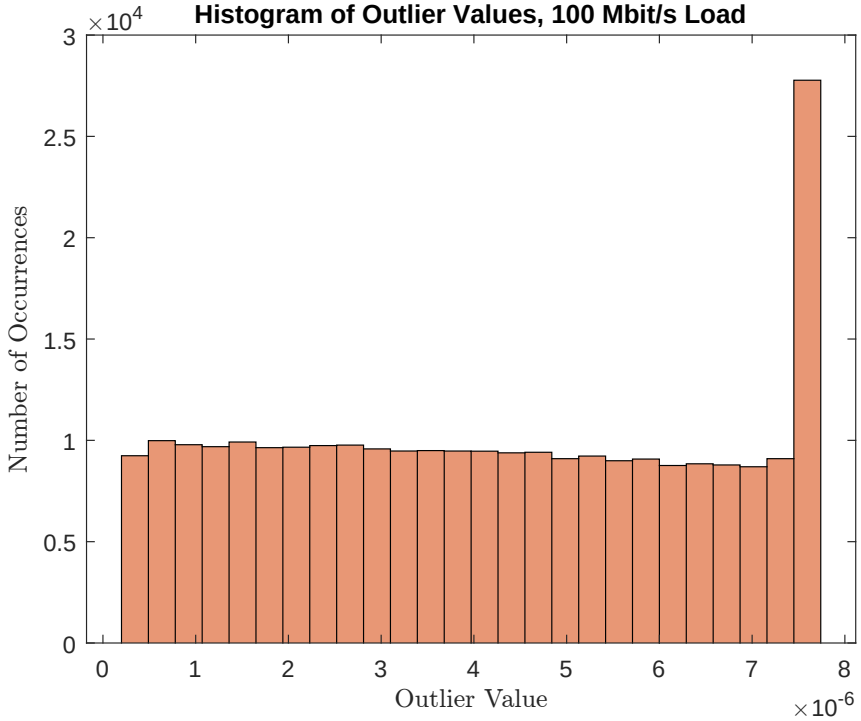


Figure 6.23: Histogram of the magnitude of all outliers in $\Delta\phi_{4,3}[i]$ for the 100 Mbit/s load test.

The above histograms show that the outliers are approximately uniformly distributed with a somewhat higher preference for the maximum of around 8 µs. While currently unknown, this distribution along with its maximum value is likely related to the 1024 byte frame length used as the load test packets. The longer the frame of the colliding packet, the

longer a queuing delay should be expected in the switch. Ideally, this test should be repeated for the average and maximum frame length used in the data acquisition network. Though as this is currently unknown this has not been investigated further.

Following the results from above, it is evident that outliers have a significant likelihood of occurrence during operation. If these are not accounted for either through removal or the use of robust control, synchronisation accuracy will suffer as any outlier can cause the offset estimate to wander off by several microseconds. It is evident that outliers are not independent, and thus if outliers are to be removed, further research into their distribution and correlation is warranted as this can possibly support the development of detection methods.

Furthermore, load tests were only conducted up to 100 Mbit/s which represents 10% of the maximum link speed. As a load up to 940 Mbit/s is be expected, this extreme scenario should also be investigated. If the outlier fraction continues to scale with the relative load, this will result in more than 90% of the data samples categorised as outliers. In this extreme case the MAD is no longer a suitable estimator for detecting outliers in post processing. This scenario must also be considered in real time implementations.

In addition to a wider range of constant load tests, it would also be relevant to test the outlier behaviour during random or bursty traffic. Since packet collisions have identical likelihood of occurrence during random burst loads, occupying the same relative link speed as an equivalent constant load, it is expected that the fraction of outliers will remain similar. However, the underlying probabilistic properties of the process may be different and thus it is worth investigating further.

Finally, it is noteworthy that both the largest and frequent occurrence of outliers occurred in the packets transferred from the DQAME to the NIC. In the opposite direction outliers occurred with significantly lower frequency. This is positive, as the packets going from the NIC to the DQAMEs are the packets that are used to estimate the sample to sample phase difference, $\phi[n]$. The packets in the opposite direction are used in support of the estimation of the propagation delay, $\hat{\eta}$, and thus outliers in this direction only affect the estimate of the propagation delay. This can introduce a constant offset in the calculation of $\phi[n]$, however, no additional sample to sample noise will be added.

Chapter 7

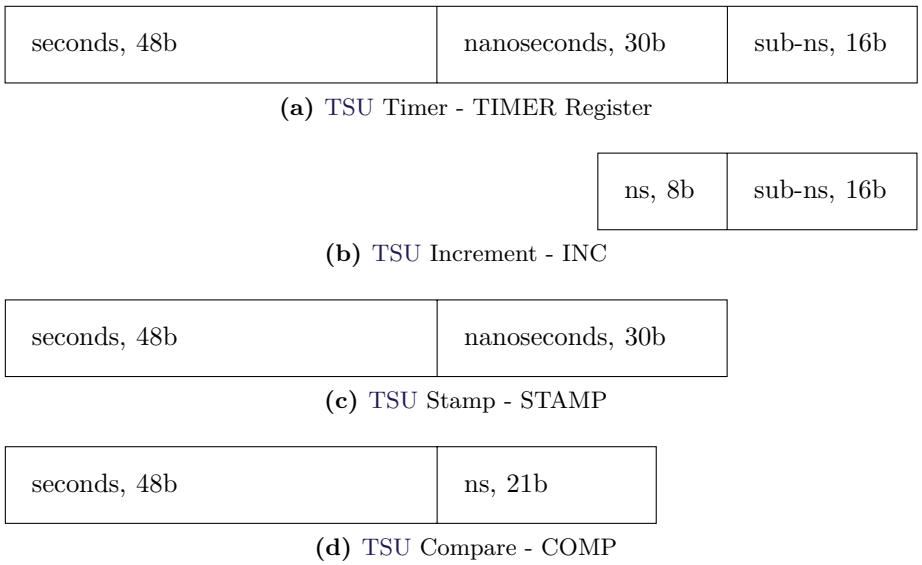
Control

Having characterised the systems open loop noise performance; the next step is the implementation of control and subsequently to evaluate the controlled system's steady state and noise behaviour.

7.1 Clock Control

To facilitate synchronisation between the master and slave, it is crucial that the slave's clock is controllable and not just a static, free running entity. To obtain an understanding of how the clock rate can be changed, the time tracking and timestamping logic of our DQAME's TSU is considered.

The TSU's four key registers are illustrated in figure 7.1



(a) TSU Timer - TIMER Register

(b) TSU Increment - INC

(c) TSU Stamp - STAMP

(d) TSU Compare - COMP

Figure 7.1: Illustration of the structure of key TSU registers.

The TSU's TIMER register is a 96-bit register that keeps track of time [10, p. 453]. The 48 bits designed to track seconds allows the chip to track time for up to 8.9 million years. This is adequate for any realistic time tracking need. Furthermore, the sub-second part of the register is slightly special as it is designed to overflow when the 30 nanosecond bits

are equal to 1 billion, as opposed to 2^{30}. Thus, the value of the nanosecond part of the register translates directly to the conventional base 10 SI perception of nanoseconds.

The TIMER register is directly linked with the TSU increment (INC) register, which controls how much the TIMER register is incremented every clock cycle. To track time exactly, the INC register must be set equal to the clock period. Thus, if the input frequency is, say, 10 MHz the INC register should be set to 100 ns. In addition to being incremented every clock cycle, the TIMER register's upper 78 bits can also be modified directly. This allows for initialisation of the TIMER register to a starting value, or to be incremented/decremented by some other value than the one in the increment register.

Outside of tracking time, the TSU's principal task is the timestamping of PTP packets. Such timestamps are stored in the TSU STAMP Register. The register stores the upper 78 bits of the timer registers value at the occurrence of the event.

Finally, the compare (COMP) register is a versatile register used to activate a hardware event or software interrupt whenever its value equals the value in the timer register. This allows for the activation of events at exact times, a feature which has proven invaluable during testing and development.

The rate at which the TIMER register is incremented depends on both the input tick rate and the value in the INC register dictating how much the TIMER register changes per clock tick. Thus, the effective clock rate can effectively be changed by changing either of these parameters.

The tick rate is controlled by the system's main clock. This clock can either be derived directly from the crystal oscillator used on the PCB, or from a FDPLL multiplying the crystal oscillator's frequency to some other value. On our boards, a 40 MHz TCXO is used as the external oscillator. This clock rate is insufficient as it severely limits the maximum performance of the MCU. Instead an embedded FDPLL is used to multiply the 40 MHz to a higher rate (maximum 120 MHz). The FDPLL along with its IO, and configuration registers is illustrated in figure 7.2.

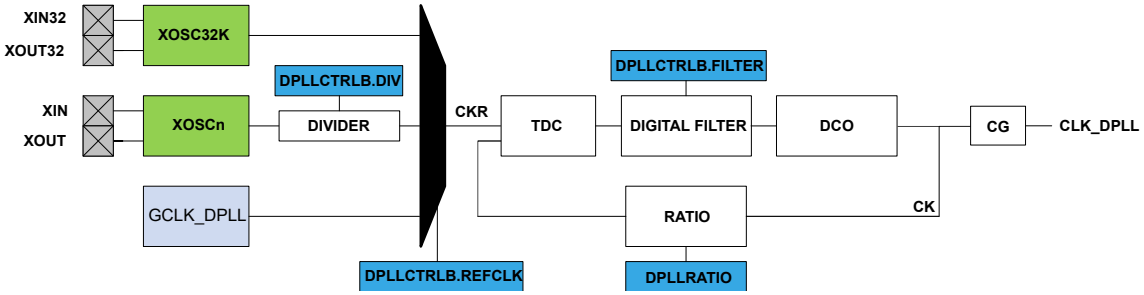


Figure 7.2: Fractional Digital Phased Locked Loop (FDPLL) used in the chip [10, p. 702].

As seen in the diagram, the FDPLL is highly configurable. When the external oscillator (XOSC) is used as an input to the FDPLL the output frequency is controlled by the DIVIDER and RATIO registers. The DIVIDER register divides the input clock frequency to a lower frequency. This is necessary as the raw clock rate is much higher than the maximum supported FDPLL input frequency of 3.2 MHz. The RATIO register controls the Loop Divider Ratio (LDR) and determines the frequency multiplication factor. The

LDR consist of an integer part and a 5-bit fractional part (LDRFRAC). The final output frequency is found according to:

$$f_{\text{CLK\_DPLL}} = \frac{f_{\text{XOSC}}}{2(\text{DIV}+1)} \cdot \left(\text{LDR} + 1 + \frac{\text{LDRFRAC}}{32}\right) \tag{7.1}$$

Due to the presence of two highly configurable degrees of freedom (DIV and LDR) available, it was initially assumed that modifying the FDPLL parameters during run time could be a good method to achieve a high degree of frequency control. However, it was found that following an update of its parameters, the FDPLL takes too long to settle in on its new target frequency. Furthermore, during testing of this scheme it was also found the our MCU's revision had faulty silicone where the FDPLL fails when the input frequency is lower than 400 kHz (This bug is confirmed by the chip's errata sheet [11, p. 28]). This limits the maximum value of the DIV configuration and lowers the frequency control achievable with this method.

Instead, rate adjustment is achieved by modifying the TSU INC register. Thus, to change the clock rate by some factor, the value of the INC register is simply multiplied by the desired factor. However, the register only has 24 bits, and the absolute rate adjustment resolution is limited to $\pm 2^{-16}$ns. In isolation, the absolute resolution is not of particular interest. Rather, the resolution relative to the INC register value is considered. The relative resolution changes depending on the current value in the INC register. In effect, relative resolution is higher if the INC value is higher, and vice versa. As the INC value changes whenever the rate is modified, so does the relative resolution. This is not useful as a metric, and thus the resolution with respect to the nominal INC value is instead used. The nominal INC value is the clock period as this sets the timer to track time at a 1 to 1 rate. This approximation is useful as the rate adjustment is expected to stay within a few percent of the nominal rate. Furthermore, fine resolution generally is only relevant when the synchronisation is near to to its steady state. At this stage, the INC value should be very close to its nominal value, and as such the relative resolution approximation is a good estimator of the true relative resolution at the present INC value. Based on this definition, the relative resolution is found as:

$$\text{Relative Resolution} = \text{Clock Frequency}(\text{Clock Period} + \text{Absolute Resolution}) \tag{7.2}$$

where the relative resolution is a dimensionless quantity.

For example, if the clock frequency is 100 MHz then the timer's INC register must be set to the clock period, 10 ns, to track time at a 1 to 1 rate. At this clock frequency the relative rate resolution is $(100 \cdot 10^6)(10 + 2^{-16}) \cdot 10^{-9} = 1 + 1.526 \cdot 10^{-6}$. This does not allow for fine control of the clock. Consider the simple case where a controller in the slave is configured to exactly match the rate of the reference clock. In the extreme case where the reference clock rate is exactly between two possible rate adjustment steps, the two clocks' rate will differ by a factor of $1 + \frac{1.526}{2} \cdot 10^{-6}$. This results in a phase drift of $\approx$ 760 ns per second. For a controller to counteract such drift, its sample rate must either be high, or its complexity must be increased.

To minimize the demand on the controller to accommodate this low resolution, the minimum rate adjustment resolution is increased by trading rate range for more resolution.

This is achieved by multiplying the value of the rate register by some power of two, 2^r. As the time stored in the TIMER register is now a factor of 2^r larger than the actual time, any timestamps must be normalised by 2^r in post processing. As the timestamp has already been captured, the additional clock cycles spent on normalizing the timestamp does not impact the timestamp accuracy. The advantage of this procedure is that it provides an additional r bits of precision. For the case of a 100 MHz input clock, and a default INC value 10 ns, r is limited to a maximum value of 4. This increases the effective rate adjustment resolution by a factor of 16 to $95.4 \cdot 10^{-9}$. For any r value higher than 4, the nanosecond increments will not fit in the 8 bits of the nanosecond part of the increment register. Theoretically it is possible to trade more range in the increment register by allowing multiplication by an arbitrary integer. However, by limiting the options to powers of two, the implementation is simplified, as it is possible to take advantage of shift operations as opposed to having to use multiplication.

7.2 Control

To control the timing registers, a controller must be implemented. The most common controller used for synchronisation is a simple Proportional-Integral (PI) controller [12, p. 146]. Such a controller will also be implemented in this thesis. To start controller design a system model is required. In broad terms systems are either modelled as transfer functions or state space models. In this thesis, both models are created. First, a transfer function of the closed loop of the system is found. This is how classical control is usually modelled and how PI controllers are commonly modelled. Subsequently, an equivalent State space model is created. State space models are the norm in modern control theory and analysis software, and such a model facilitates the use of more complex control schemes that may be envisioned in the future. Both of the models only consider the systems deterministic properties, and thus the various noise and quantisation sources are unaccounted for.

7.2.1 Transfer Function

A system controlled by a PI controller is composed of a controller $K(z)$ and a plant $G(z)$ in a closed loop. This is visualised in figure 7.3:

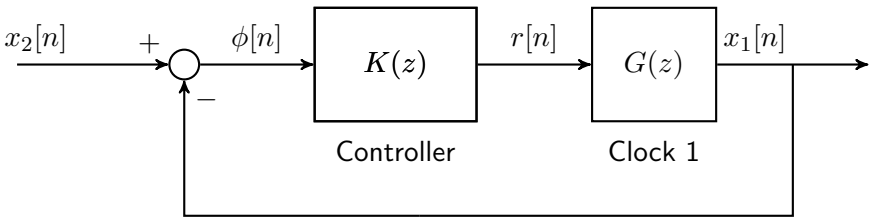


Figure 7.3: Block diagram of a closed loop transfer function model where clock 1 is synchronised to clock 2.

To find the transfer function of the closed loop behaviour, the transfer function of both the controller and the clock block must first be found. These are inferred with respect to the block's time domain linear difference equations.

To start, the clock is considered. A clock simply finds the time by integrating the previous set rate over the observation period. This is equivalent to:

$$x_1[n] = x_1[n-1] + r[n]\Delta T \tag{7.3}$$

where:

$r[n]$ is the clock rate.

ΔT is the controller sampling rate (not the clock period).

Thus, the clock is modelled as forward Euler integrator with the following transfer function:

$$G(z) = \frac{\Delta T}{z-1} \tag{7.4}$$

The PI controller can be described using the following equations:

$$r_i[n] = r_i[n-1] + \phi[n-1]\Delta T \tag{7.5}$$
$$r_k[n] = K_p\phi[n] \tag{7.6}$$
$$r[n] = r_k[n] + r_i[n] \tag{7.7}$$

Resulting in the $K(z)$ transfer function:

$$K(z) = K_p + K_i\Delta T\frac{z}{z-1} \tag{7.8}$$

Having derived transfer function for the Controller, $K(z)$ and the clock, $G(z)$ the closed loop transfer function is found as [13, p. 37]:

$$\begin{aligned} H(z) &= \frac{K(z)G(z)}{1+K(z)G(z)} \\ &= \frac{\Delta T(K_i z + K_p z - K_p)}{z^2 + \Delta T(K_i + K_p)z - 2z - \Delta T K_p + 1} \end{aligned} \tag{7.9}$$

7.2.2 State Space Model

In a state space model, each changing state of the system is modelled. For a linear time invariant discrete state space model this is generally described by the following [13, p. 79]:

$$\boldsymbol{q}[n+1] = \boldsymbol{A}\boldsymbol{q}[n] + \boldsymbol{B}\boldsymbol{u}[n] \tag{7.10}$$
$$\boldsymbol{y}[n] = \boldsymbol{C}\boldsymbol{q}[n] + \boldsymbol{D}\boldsymbol{u}[n] \tag{7.11}$$

where:

$\boldsymbol{q}[n]$ is a vector containing m states

$\boldsymbol{y}[n]$ is a vector containing l outputs

$\boldsymbol{u}[n]$ is a vector containing r control inputs

$\boldsymbol{A}$ is a $m \times m$ matrix

$\boldsymbol{B}$ is a $m \times r$ matrix

$\boldsymbol{C}$ is a $l \times m$ matrix

$\boldsymbol{D}$ is a $l \times r$ matrix

The $\boldsymbol{A}$ matrix describes how the current state, $\boldsymbol{q}[n]$, impacts the next state, $\boldsymbol{q}[n+1]]$. Similarly, the $\boldsymbol{B}$ matrix relates the current input, $\boldsymbol{u}[n]$ to the change in the next state. The system output $\boldsymbol{y}[n]$ is related to the current system state, $\boldsymbol{q}[n]$ and system input $\boldsymbol{u}[n]$ according to matrix $\boldsymbol{C}$ and $\boldsymbol{D}$ respectively.

In the model explored in this thesis, the current system state, $\boldsymbol{q}[n]$, is exactly equal to the system output, $\boldsymbol{y}[n]$. Thus only the first equation, equation (7.10), is considered further.

To model closed loop feedback with reference tracking, the control vector, $\boldsymbol{u}[n]$, is set equal to some reference input, $\boldsymbol{p}[n]$, subtracted from the current state :

$$\boldsymbol{u}[n] = \boldsymbol{q}[n] - \boldsymbol{p}[n] \tag{7.12}$$

Figure 7.4 illustrates the interaction between the variables making up the state space model:

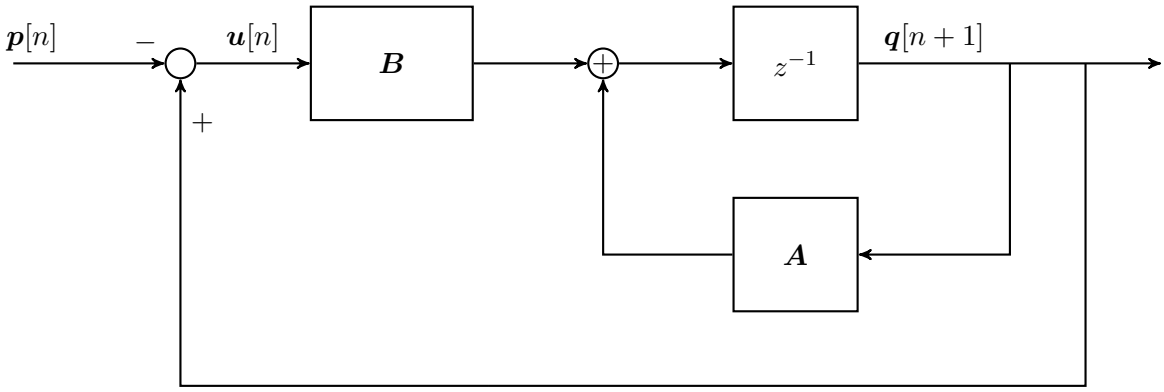


Figure 7.4: State Space Block Diagram.

To create a state space model of the system, the system states must first be defined. The clock is defined by two parameters. Its current phase, $x[n]$ and its rate, $r[n]$. Therefore the state vector is:

$$\boldsymbol{q}[n] = \begin{bmatrix} x_1[n] \\ r[n] \end{bmatrix} \tag{7.13}$$

The reference input vector $\boldsymbol{p}[n]$ only contains the phase information of the reference, clock 2, and the rate information is zero:

$$\boldsymbol{p}[n] = \begin{bmatrix} x_2[n] \\ 0 \end{bmatrix} \tag{7.14}$$

Thus the control vector is defined as:

$$\boldsymbol{u}[n] = \boldsymbol{p}[n] - \boldsymbol{q}[n] \tag{7.15}$$

$$= \begin{bmatrix} \phi[n] \\ -r[n] \end{bmatrix} \tag{7.16}$$

To model the difference equations of clock 1 synchronised to clock 2 with a PI controller, both the states of $\boldsymbol{q}[n]$ are modelled as integrators according to:

$$x[n+1] = x[n] + (r[n] + (k_p + k_i)\phi[n])\Delta T \tag{7.17}$$

$$r[n+1] = r[n] + k_i\phi[n] \tag{7.18}$$

In matrix form this system is written as:

$$\begin{aligned} \boldsymbol{q}[n+1] &= \boldsymbol{A}\boldsymbol{q}[n] + \boldsymbol{B}\boldsymbol{u}[n] \\ &= \boldsymbol{A}\boldsymbol{q}[n] + \boldsymbol{B}(\boldsymbol{p}[n] - \boldsymbol{q}[n]) \\ &= (\boldsymbol{A} - \boldsymbol{B})\boldsymbol{q}[n] + \boldsymbol{B}\boldsymbol{p}[n] \\ &= \boldsymbol{F}\boldsymbol{q}[n] + \boldsymbol{B}\boldsymbol{p}[n] \end{aligned} \tag{7.19}$$

where

$$\boldsymbol{q}[n] = \begin{bmatrix} x_2[n] \\ r[n] \end{bmatrix}$$

$$\boldsymbol{p}[n] = \begin{bmatrix} x_2[n] \\ 0 \end{bmatrix}$$

$$\boldsymbol{u}[n] = \begin{bmatrix} \phi[n] \\ -r[n] \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} (K_p + K_i)\Delta T & 0 \\ K_i & 0 \end{bmatrix}$$

$$\boldsymbol{F} = \boldsymbol{A} - \boldsymbol{B} = \begin{bmatrix} 1 - (K_p + K_i)\Delta T & \Delta T \\ -K_i & 1 \end{bmatrix}$$

The derived state space model is converted into an equivalent transfer function, to verify that it is equivalent to the previously derived transfer function. To do this the following equation is used [14, p. 350]:

$$\boldsymbol{H}(z) = (z\boldsymbol{I} - \boldsymbol{F})^{-1}\boldsymbol{B} \tag{7.20}$$

where $\boldsymbol{H}(z)$ is a matrix of transfer functions describing each of the states, $\boldsymbol{q}$, response to each of the control states.

When $\boldsymbol{H}(z)$ is evaluated, it is found that the transfer function of the $x_1[n]$ state is exactly equivalent to the derived transfer function:

$$\boldsymbol{H}_{1,1}(z) = \frac{\Delta T(K_i z + K_p z - K_p)}{z^2 + \Delta T(K_i + K_p)z - 2z - \Delta T K_p + 1} \tag{7.21}$$

where $\boldsymbol{H}_{1,1}(z)$ is the transfer function of state $\boldsymbol{q}_1[n] = x_1[n]$ subject to control signal $\boldsymbol{u}_1[n] = \phi[n]$.

7.2.3 Transient Response

To validate the developed model, the PI controller is implemented on the DQAMEs and the performance is compared with the derived model. In the tested implementation the DQAMEs exclusive rely on the $x_1(t_1)$ and $x_2(t_2)$ timestamps, and no propagation delay compensation is performed using the other two timestamps, $x_2(t_3)$ and $x_4(t_4)$. Further, prior to activating the PI control the DQAME sets its internal time equal to the first $x_1(t_1)$ timestamp received. This coarsely synchronises the two clocks. If this is not done, the DQAME's clock would start at $t = 0$ and would thus be more than a billion seconds behind the reference clock's UNIX time. Such a large phase difference between the two clocks will drastically increase settling time, and likely cause the controller to fail due to excessive integral windup past the rate control systems capabilities. The initial phase equalisation resolves these two issues. Furthermore, the clock rate is initialised as 1.0, i.e. $r[0] = 1$. As it is assumed that the clock's default rate closely resembles the reference clock rate, this initialisation further helps reducing the settling time.

The PI controller test setup is illustrated in figure 7.5.

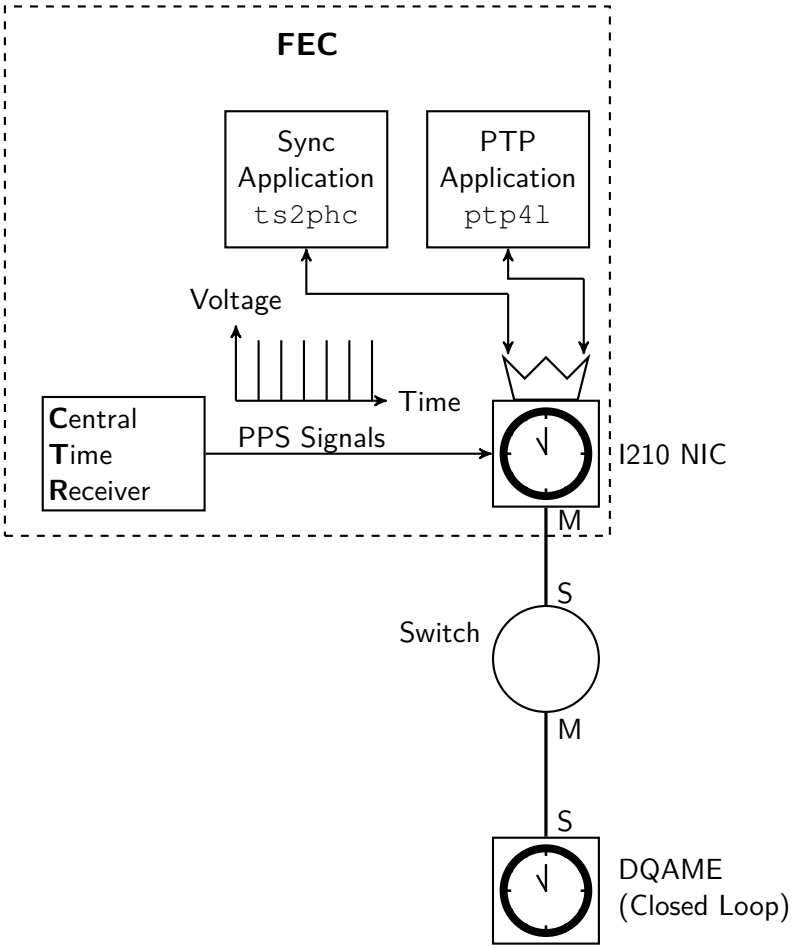


Figure 7.5: Illustration of the measurement setup used to measure the closed loop behaviour.

As time increases linearly, the system response is tested using the equivalent of a ramp signal.

Two different PI controller configurations were tested at a sampling rate of 32 Hz:

- $K_p = 0.5$, $K_i = 2.134$
- $K_p = 0.188$, $K_i = 0.0136$

Both of these configurations were determined empirically. The first configuration is purposely underdamped while the second configuration is less damped and represents a more sensible and robust configuration. These two configurations allow for testing the model for two very different operating points. The systems response results are shown in figure 7.6 and figure 7.7.

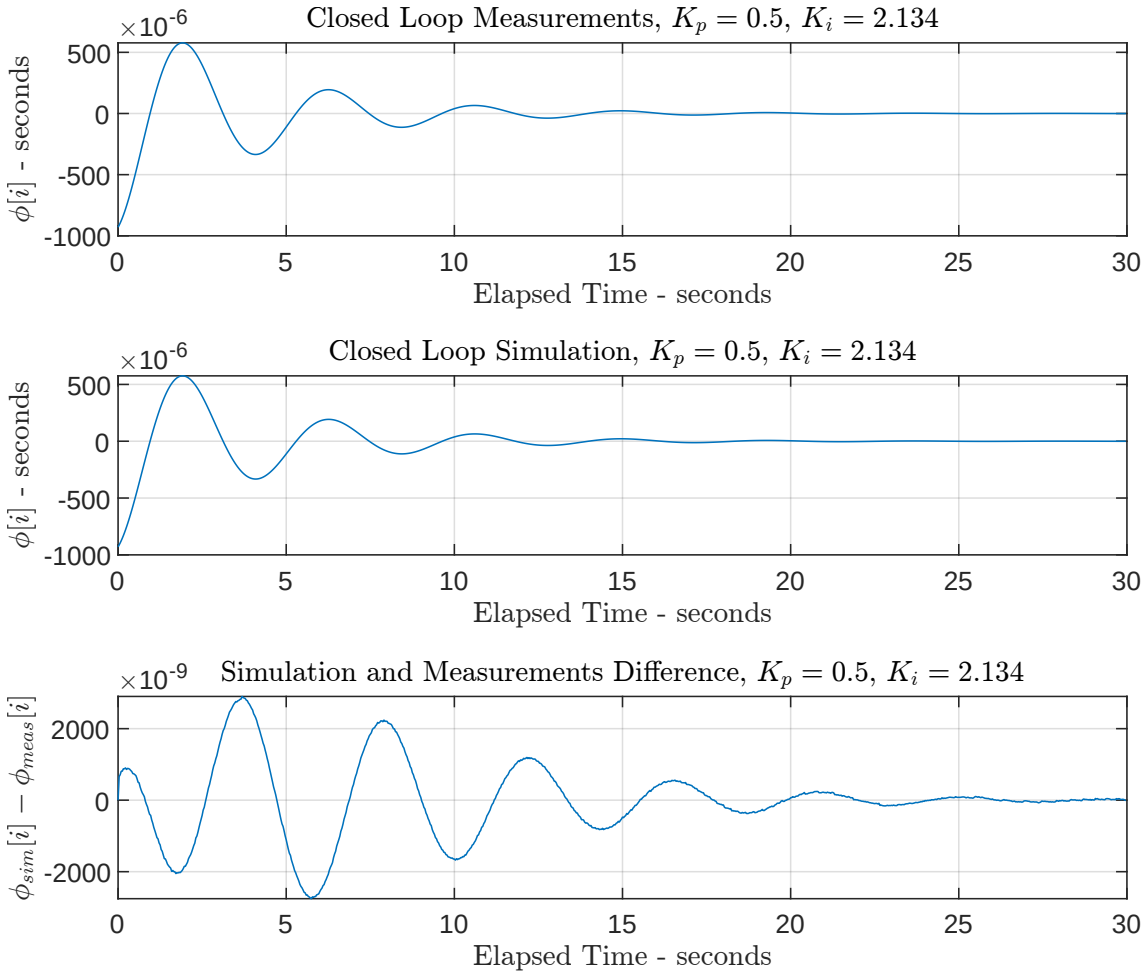


Figure 7.6: Measured and modelled system ramp responses. Performed with control parameters: $K_p = 0.5$, $K_i = 2.134$, $\Delta T = 0.03125$.

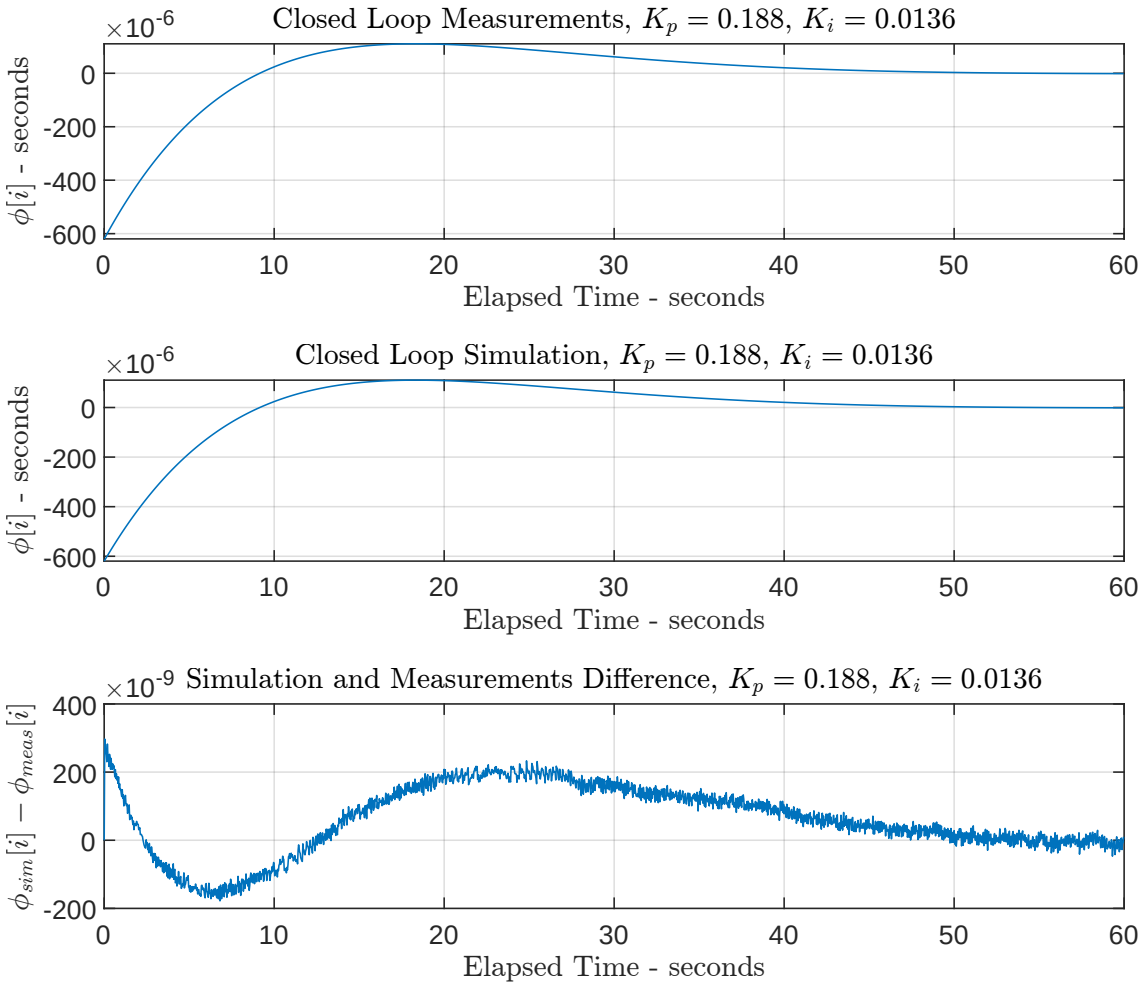


Figure 7.7: Measured and modelled system ramp responses. Performed with control parameters: $K_p = 0.188$, $K_i = 0.0136$, $\Delta T = 0.03125$.

From the figures, it is clear that the derived model models the system behaviour well. This implies that the system dynamics are well understood. Thus, future PI parameters can easily be tested through simulation before being deployed to hardware. Furthermore, if necessary, the created state space system model can be expected to facilitate the implementation of more advanced control schemes.

7.3 Steady State Error

The previous section detailed the PI controller implementation and showed the system response to the control. However, the previous measurements only considered $\phi[i]$ as measured through the PTP network's $x_1(t_1)$ and $x_2(t_2)$ timestamps. As established earlier in section 4.2, propagation delay must also be accounted for. Through the use of the $x_2(t_3)$ and $x_1(t_4)$ timestamps it is possible to estimate the propagation delay. However, this estimate assumes the delay is symmetrical. Thus, any asymmetry will result in the controlled system having a constant bias/steady state error as compared to the true value.

To test the validity of the propagation delay estimate, $\hat{\eta}[n]$ and quantify any present asymmetry, the PTP offset estimates, $\phi[i]$, is compared to the offset measured through pulse differences. The test setup is illustrated in figure 7.8.

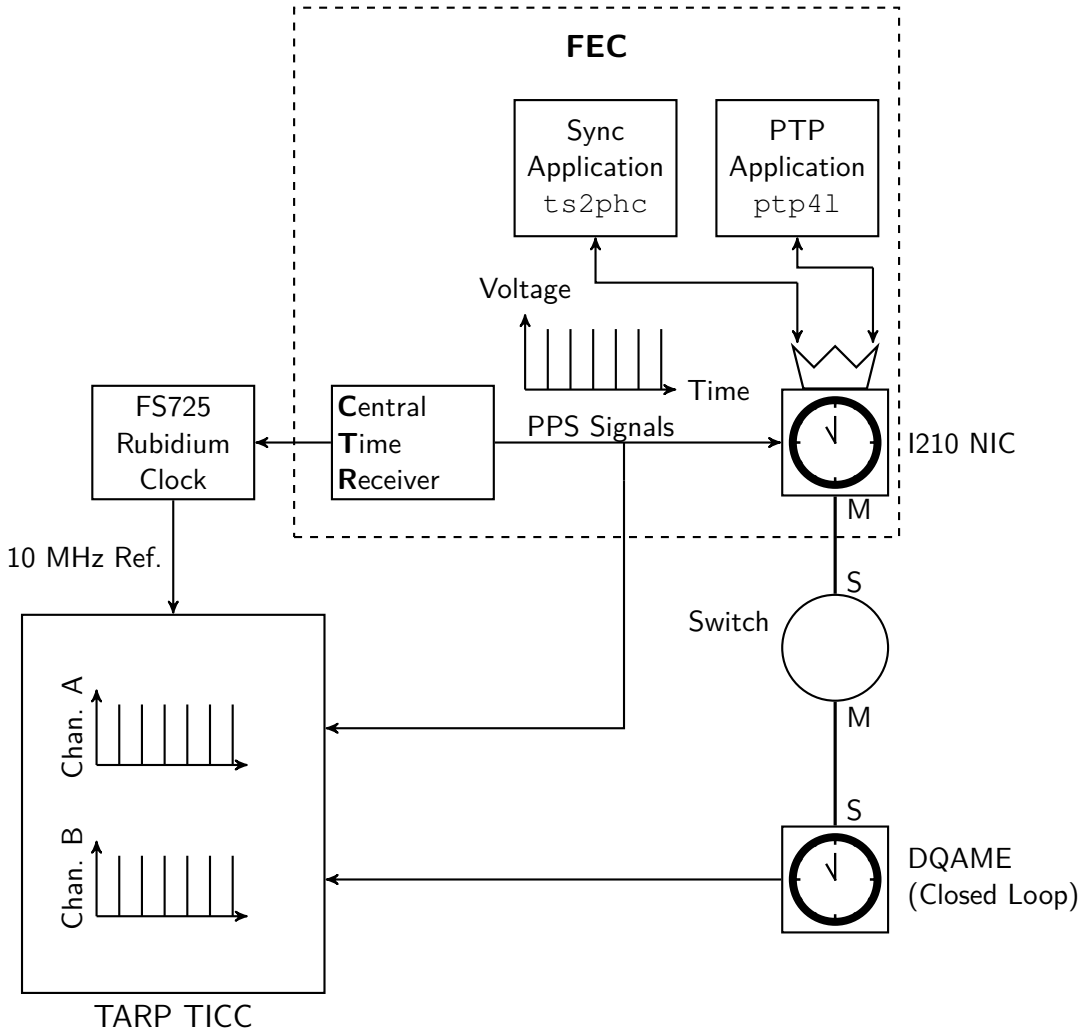


Figure 7.8: Illustration of the measurement setup used to compare the system response as measured through the PTP network and through the TICC.

Unlike in previous test, the timestamps collected by the TICC are not simply considered in isolation, but rather the timestamps from the two channels are subtracted to find their phase difference. This measure is referred to as the Time Interval Error (TIE), and its principle is shown in figure 7.9:

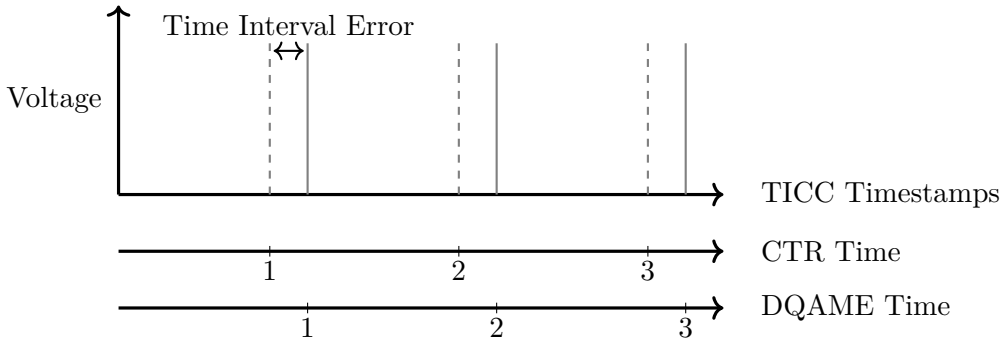


Figure 7.9: Illustration of measuring time error using pulses.

The DQAME pulses are compared to the CTR Pulses as this represents the ultimate source of time the DQAME should be synchronised to. Thus it would not be as compelling to find the TIE between the NIC. Each pulse source is configured to emit pulses at 1 second intervals, as this is the fastest the central time receiver allows. The $\phi[n]$ is estimated using both the pulse TIE and through the $x_1(t_1)$ and $x_2(t_2)$ PTP timestamps. The results of these two estimates of $\phi[n]$ is shown in figure 7.10

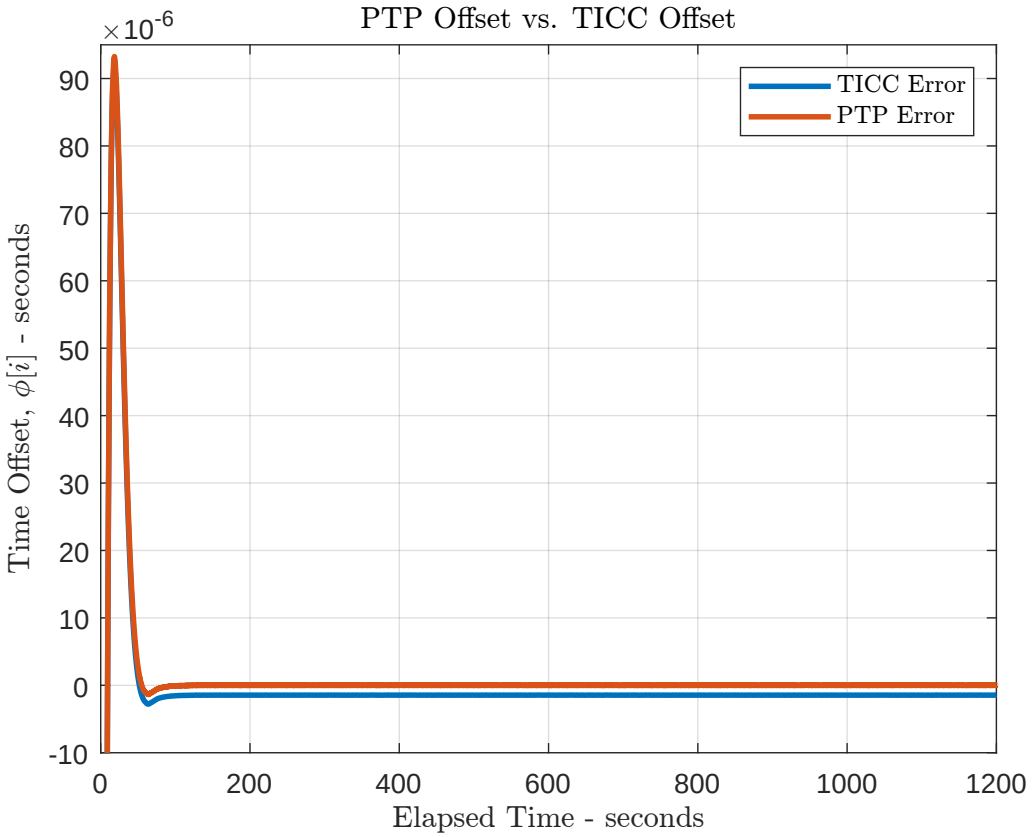


Figure 7.10: Plot of $\phi[n]$ when estimated through pulses and through the PTP network. Note that since these measurements were not synchronised, the plots alignment is only approximate as it was made manually.

From figure 7.10 it is clear that the $\phi[n]$ estimate follows the same general trajectory.

Upon reaching steady state it is seen that the TIE estimate of $\phi[n]$ differs from the PTP measurement by a couple of microseconds. This is expected, as the plotted PTP estimate does not account for propagation delay.

To see how well the estimated propagation delay matches the measured difference between the TIE and PTP estimate of $\phi[n]$, figure 7.11 zooms in on the steady state region of the measurements and additionally plots $\hat{\eta}[n]$.

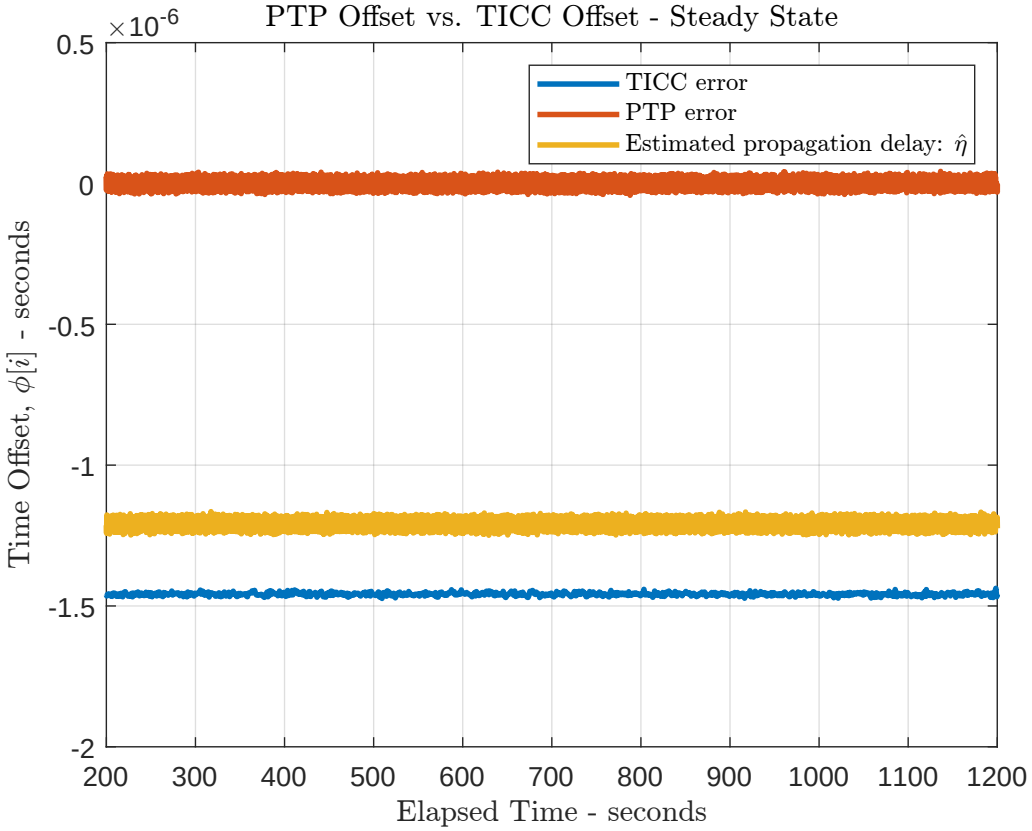


Figure 7.11: Steady state plot of $\phi[n]$ as estimated through pulses and through the PTP network.

Based on the plot in figure 7.11 it is evident that the propagation delay estimate, $\hat{\eta}[n]$, accounts for most of the discrepancy between TIE and PTP estimate. However, around ~300 ns of time difference between the two estimates remain unaccounted for. This difference may partially be attributed to asymmetry in the network propagation delay. The root course of the asymmetry is as of now unknown. From correspondence with the switch manufacturer, it is known that the used switch can contribute up to 50 ns of asymmetry. Furthermore, it is speculated that there may be a systematic error in the timestamping of packets on the DQAME, as the TSU is located in the device MAC layer and is physically disjoint from a separate PHY chip managing the physical layer. A specific example of an unexpected asymmetry source faced during development is the choice of SFP transceiver used in the switch. During initial testing, the estimated propagation delay, $\hat{\eta}$, was significantly worse than in the previously presented data - upwards of 1 µs away from the true offset. Given that the networking hardware was specifically chosen to provide accurate timestamping this seemed suspiciously high. Following extensive troubleshooting it was found that the transceiver used to connect the NIC to the switch was not PTP validated by the switch manufacturer (Juniper). After this transceiver was replaced by a validated

transceiver, the propagation delay estimate improved significantly.

Finally, other non-network related sources may also contribute to the observed offset. For example if a phase offset is present between the NIC and the CTR, this will also affect the offset between the DQAME and the CTR.

7.4 Closed Loop Noise Measurements

Additional to the propagation delay, the derived model does not consider the stochastic noise sources of the system. These could possibly be added later to further enhance the model. However in this thesis it has instead been opted to exclusively measure the closed loop noise rather than modelling it.

To measure closed loop noise the controlled DQAME clock noise is measured both using the TICC and through the PTP network. Using both these measurements, the AVAR is estimated and compared to open loop estimates presented in figure 6.10 and figure 6.12 from chapter 6. The test setup for these tests is illustrated in figure 7.12.

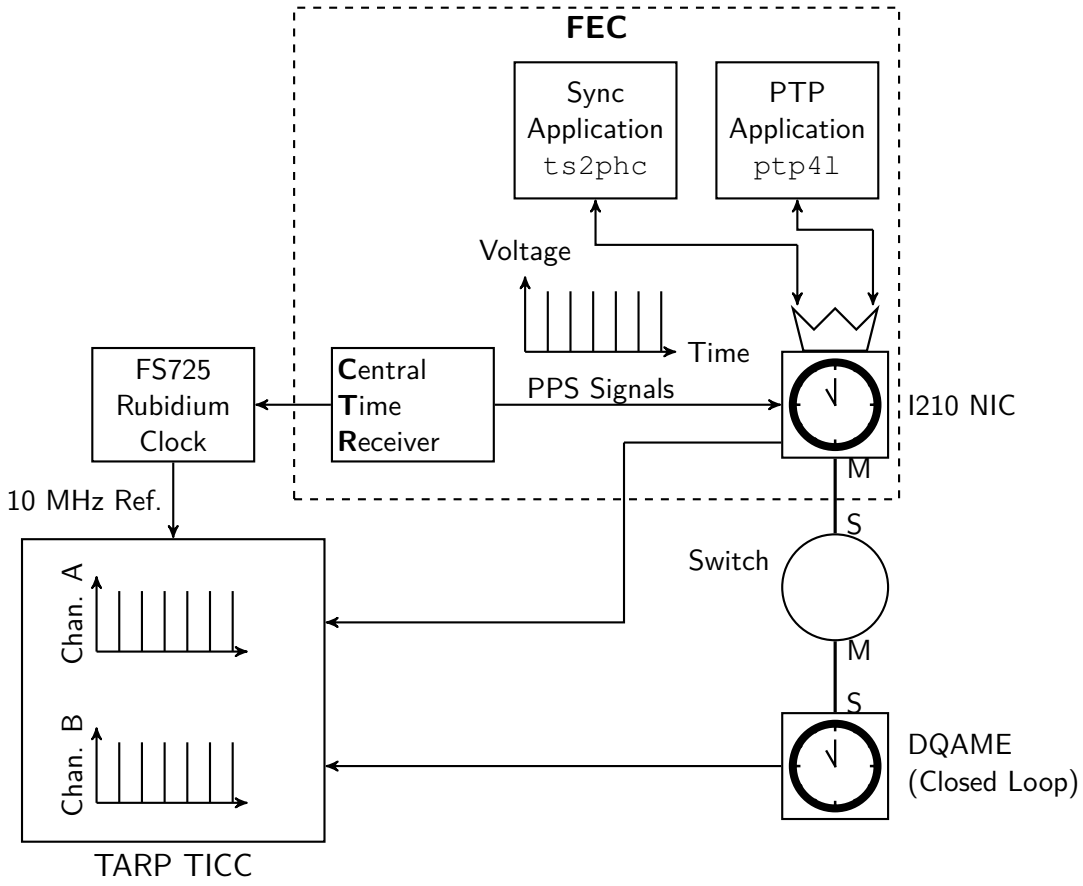


Figure 7.12: Illustration of the measurement setup used to quantify the closed loop phase noise when measured through the PTP network.

Using the above setup data was collected for the case of the DQAME2 board running

PI configuration $K_p = 0.188$, $K_i = 0.0136$. To ensure only the steady state noise is analysed, only data after the first 200 seconds was used to estimate the AVAR. Figure 7.13 illustrates the closed loop noise performance and compares it with some of the open loop measurements conducted in chapter 6.

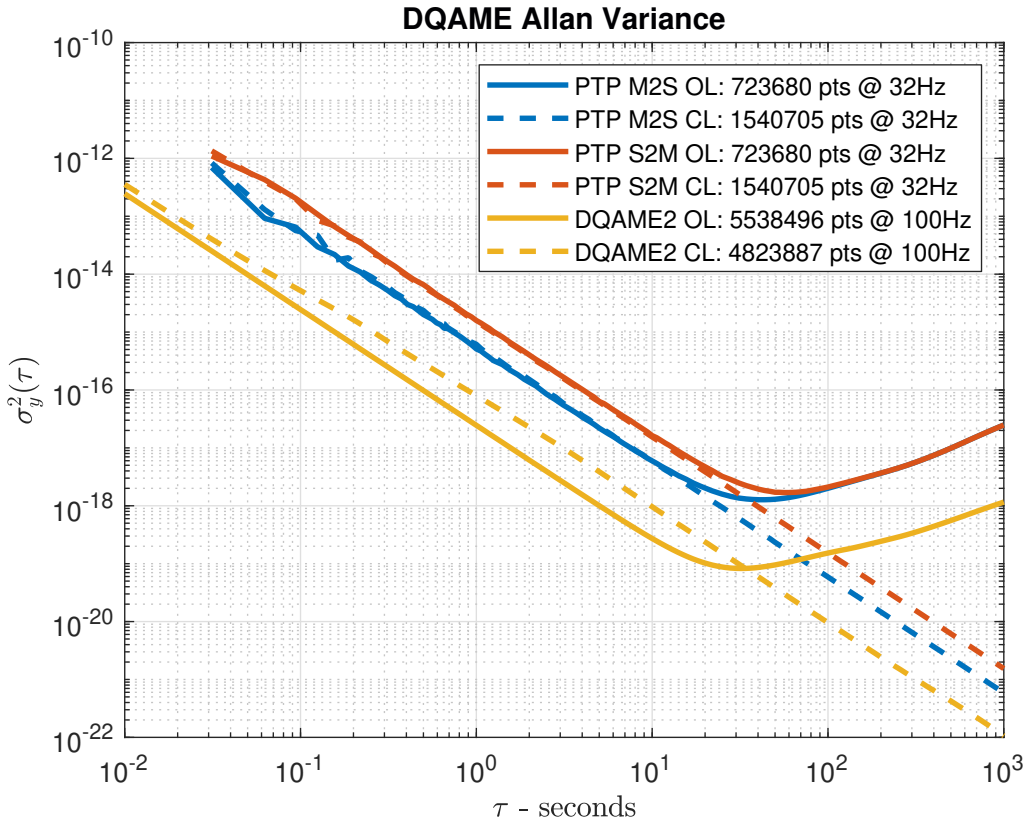


Figure 7.13: Sigma-Tau plot of the Allan Variance of open and closed loop measurements. OL is an abbreviation for Open Loop, while CL abbreviates Closed Loop.

The above data shows that synchronisation is working, as the random walk noise is eliminated. This is demonstrated by the closed loop AVAR curves having a negative slope throughout all time intervals considered. Furthermore, as the PTP closed loop noise curves overlap with their open loop counterparts, it is shown that the high frequency PTP noise is mostly unaffected by the synchronisation. This is not the case for the raw DQAME clock noise. At low time scales the open and closed loop estimates are roughly equivalent. However, as the observation period progresses past the synchronisation interval ($\frac{1}{32}$s) the closed loop noise increases slightly. This increase is attributed to the discretisation of clocks frequency control described in section 7.1. Indeed, it is found that the closed loop noise increases significantly if the resolution increasing scheme is not used. A noise comparison between the schemes is shown in figure 7.14.

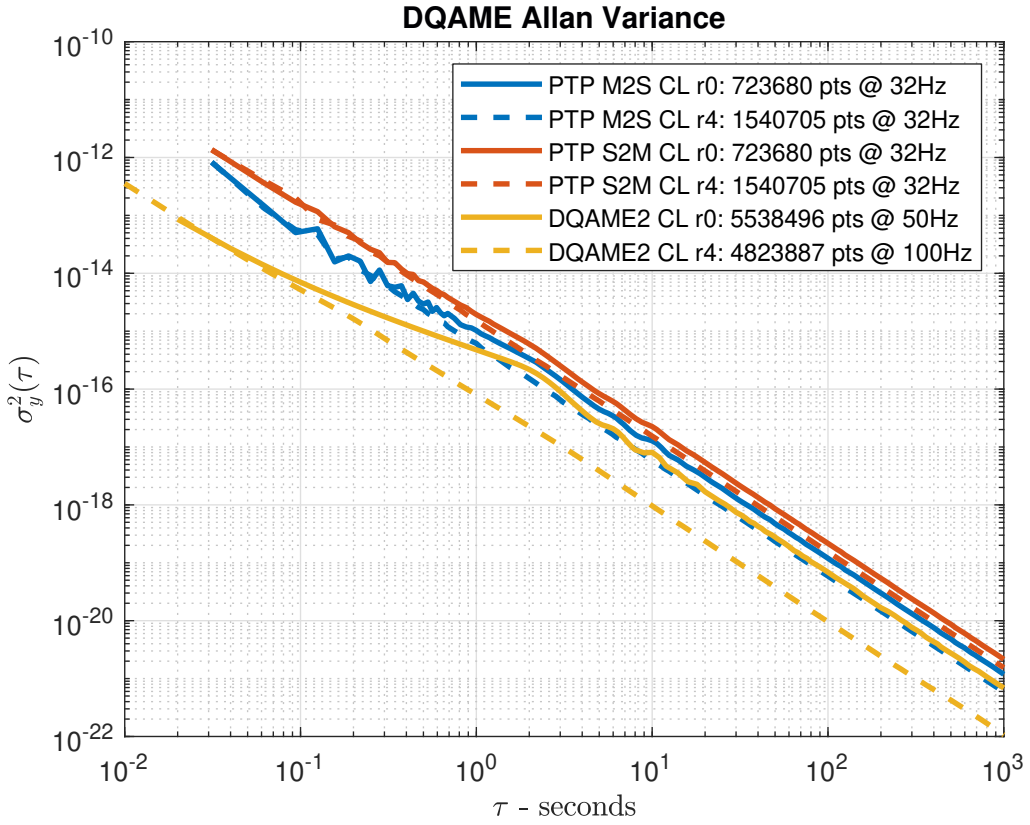


Figure 7.14: Sigma-Tau plot of the Allan Variance of closed loop measurements. The plot compares a configuration with a high degree of clock control ($r = 4$) to another configuration with more coarse control ($r = 0$). r represents the number of additional bits of precision.

Figure 7.14 shows that while the random walk noise remains eliminated in the case where clock frequency control is limited, its low frequency noise is further offset than in the high resolution case. This scenario has sufficient noise that the quantisation effect can be seen in a phase error plot. Figure 7.15 shows this.

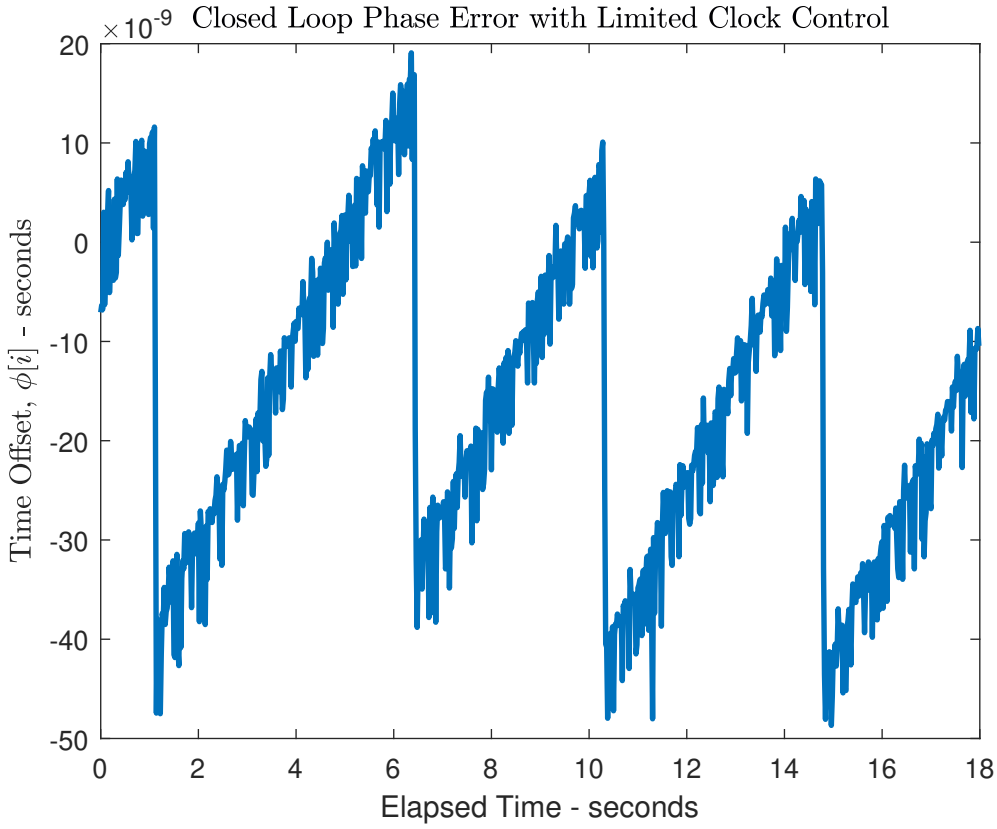


Figure 7.15: Phase error of configuration with coarse clock control. Measured at steady state.

Each of the sudden jumps in the shown phase error represents the moment the controller integral control has accumulated sufficient error to overcome the clock's frequency discretisation threshold. The relatively high threshold in the default, case where resolution is not raised, result in the rapid phase change. This also occurs in the modified high resolution case. However, due to the higher resolution, the effect is much less pronounced.

Thus far, this thesis has mostly relied on frequency noise to describe phase error. However, frequency noise only indirectly informs us of the phase error. Thus phase deviation is now considered more directly. Several methods are available for the quantification of phase error, with one of the most common being the Time Deviation (TDEV). TDEV is closely related to the ADEV and quantifies how much the phase varies within an observation period τ. Instead of focusing on TDEV, the Maximum Time Interval Error (MTIE) is instead investigated. This measure quantifies a measurements maximum time deviation within observation period τ. MTIE sheds light on the maximum time error variation that can be expected during operation. This is useful as it helps defining the upper bound of the synchronisation phase error.

The MTIE in a single observation period of duration τ starting from time t_0 is defined as [15]:

$$\mathrm{MTIE}(\tau) = \max_{t_0 \leq t \leq t_0+\tau} [\phi(t)] - \min_{t_0 \leq t \leq t_0+\tau} [\phi(t)] \tag{7.22}$$

The MTIE over all possible observation periods for a measurement of duration T is further

defined as [15]:

$$\mathrm{MTIE}(\tau, T) = \max_{0 \leq t_0 \leq T+\tau} [\mathrm{MTIE}(\tau)] \tag{7.23}$$

Using this definition, the $\mathrm{MTIE}(\tau, T)$ of the synchronised system is plotted and shown in figure 7.16. The MTIE in figure 7.16 is calculated using the $\tau[i]$ values found using the TICC.

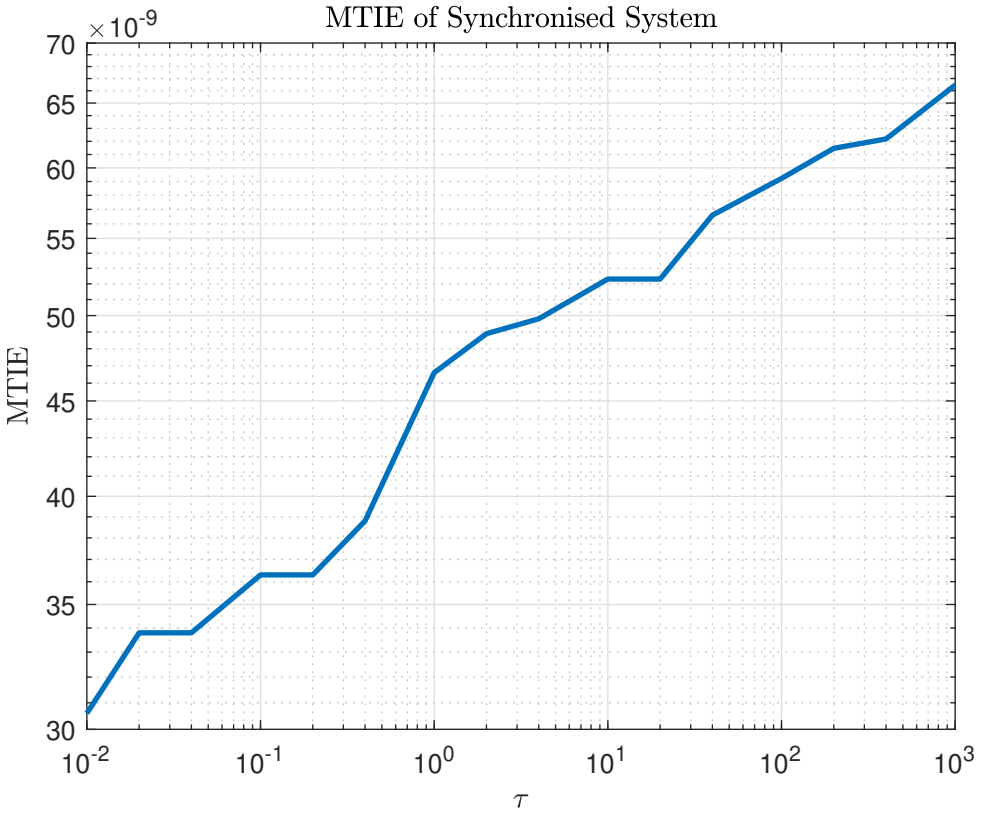


Figure 7.16: Plot of closed loop $\mathrm{MTIE}(\tau, T)$.

The MTIE plotted in figure 7.16 are promising and show an MTIE of less than 100 ns for observation periods up to 1000 seconds. Granted, the calculated MTIE for the longer observation periods are of less statistical significance than the smaller period, and thus the MTIE may increase significantly if larger data sets are considered. Furthermore, if longer intervals are considered the MTIE will also rise, as there is simply a higher chance that two phase errors will be further apart. The MTIE does not define hard upper limit for the time error. Instead it is useful in the evaluation of the maximum error that can be expected in some percentage of measurements (e.g. 99.9%). The calculations required to find these confidence intervals are not undertaken.

Having evaluated the closed loop synchronisation steady state error and noise performance separately, some general observation of the overall system performance can be made. It was found that when the propagation delay estimate $\hat{\eta}$ was accounted for, the steady state offset was around 300 ns. When this is combined with the MTIE measurements, it is inferred that the total error will generally remain below 400 ns. While more extensive

tests and calculations are required to make stronger and more definitive conclusions, the current results are promising. It seems highly probable that the goal of achieving sub microsecond synchronisation performance is attainable.

Chapter 8

Conclusion

Throughout this thesis, both theoretical and practical aspects of synchronisation were thoroughly analysed and developed. First, the underlying deterministic and stochastic error sources inherent in clocks was investigated. Following this, methods allowing for the distribution of clock phase were analysed in detail. Based on these, the earlier developed synchronisation test setup was presented. The system noise was first tested in open loop without the use of synchronisation. Subsequently control was developed, and similar test were carried out.

The resulting measurements are encouraging and strongly suggest that sub microsecond synchronisation is attainable. It is likely that even higher precision is achievable within the current setup, however, further testing is required to make definitive conclusions.

Currently the main source of error is the constant bias present in the phase offset measurements. For further research into enhanced precision, it is advised to investigate the source of this bias. If it is found that systematic error is the primary culprit, it should be possible to address, eliminating the extra several hundredths of nanoseconds of error currently present. If the bias can be accounted for, noise will be the dominating source of error. Thus, based on the results substantiated throughout this thesis, it would be possible to reduce the synchronisation error below 100 ns.

However, for such results to materialise in practise, outliers resulting from networking load must be addressed. In this thesis outliers were only analysed and removed during post processing and no attempts were made to identify real time methods to detect and remove outliers. Thus, further work into this topic is critical for any real world implementation of the synchronisation system aiming to achieve the previously claimed results.

Furthermore, practical real time methods for finding good estimates of the propagation delay, $\hat{\eta}$, has also been omitted in this thesis and needs to be addressed in a final implementation.

Finally, as mentioned previously in chapter 7, it was discovered that the SFP transceivers between the NIC and switch was not validated for PTP. This was discovered following the completions of the open loop noise and load tests in chapter 6. Thus, it is possible that the collected results are erroneous. Therefore these must using validated SFP transceivers to validate the results. However, given that the newly acquired transceiver is PTP validated, it seems plausible a new measurement round with PTP validated equipment will not worsen compared to the earlier results.

In summary, the project's results are considered a success. While finishing touches are still required to finalise a deployment ready implementation of the synchronisation system, the results obtained throughout this thesis are considered excellent. It is expected that in the coming months the synchronisation system will be implemented and provide the new data acquisition system being developed by TE-MPE-EP with accurate timestamping.

Bibliography

[1] M. Christensen, "QPS Data Acquisition: The Synchronisation Sub-System - Interim Report," Aalborg University, Tech. Rep., 2020.

[2] E. Rubiola, Phase Noise and Frequency Stability in Oscillators. Cambridge University Press, 2009.

[3] D. Howe, D. Allan, and J. Barnes, "Properties of Signal Sources and Measurement Methods," in Proceedings of the 35th Annual Symposium on Frequency Control, no. February 1981, 1981, pp. 669–716.

[4] Hewlett-Packard, "Fundamentals Quartz Oscillators," Hewlett-Packard, Tech. Rep., 1997.

[5] V. Solo, "INTRINSIC RANDOM FUNCTIONS AND THE PARADOX OF 1/f NOISE," SIAM Journal on Applied Mathematics, 1992.

[6] W. J. Riley, Handbook of Frequency Stability Analysis. National Institute of Standards and Technology, 1994, vol. 31, no. 1.

[7] "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," Institute of Electrical and Electronics Engineers, Standard, 2008.

[8] R. Cochran, "The Linux PTP Project." [Online]. Available: http://linuxptp.sourceforge.net/

[9] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," Journal of the American Statistical Association, vol. 88, no. 424, pp. 1273–1283, 1993.

[10] Microchip, "SAM D5x/E5x Family Data Sheet DS60001507G."

[11] ——, "SAM D5x/E5x Family Silicon Errata and Data Sheet Clarification DS80000748L."

[12] J. C. Eidson, Measurement, Control, and Communication Using IEEE 1588. Springer, 2006.

[13] W. Brogan, Modern Control Theory, 3rd ed. Pearson, 1990.

[14] S. Skogestad and I. Postlethwaite, Multivariable Feedback Control: Analysis and Design, 2nd ed. John Wiley & Sons, 2001.

[15] S. Bregni, "Measurement of maximum time interval error for telecommunications clock stability characterization," IEEE Transactions on Instrumentation and Measurement, vol. 45, no. 5, pp. 900–906, 1996.