CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Clock recovery algorithm in Circuit Emulation Service (CES)

Master's thesis in Embedded Electronic System Design

ANTONIOS PANAGIOTOU

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Master's thesis 2020

Clock recovery algorithm in Circuit Emulation Service (CES)

ANTONIOS PANAGIOTOU

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Programme of Embedded Electronic System Design
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2020

Clock recovery algorithm in Circuit Emulation Service (CES)

ANTONIOS PANAGIOTOU

Supervisor: Lars Svensson, Department of Computer Science and Engineering
Supervisor: Daniel Logenius, Ericsson
Examiner: Per Larsson-Edefors, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Programme of Embedded Electronic System Design
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Clock recovery algorithm in Circuit Emulation Service (CES)

ANTONIOS PANAGIOTOU
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

The clock recovery function of a communications systems was studied along with the performance requirement of that system. The thesis focused on the phase locked loop (PLL) that is used for clock recovery, specifically the low pass filter (LPF) used by the PLL. Given the lack of a frequency response specification to be used as a starting point for the LPF design, a simulation tool that simulates and predicts the system's performance in terms of maximum time interval error (MTIE) was developed. This tool was subsequently used to design alternative low pass filters for the PLL. By examining the predicted performance, a filter design was proposed, implemented and integrated in an FPGA based design. The filtering algorithm was run in the real system and evaluated. It was concluded that in a system that utilizes a packet selection algorithm (PSA), a filtering solution that combines a proportional-integral (PI) controller with an added infinite impulse response (IIR) filter can comply with the required specification.

Keywords: FPGA, clock-recovery, CES, digital filter, ADPLL

Acknowledgements

I would like to thank my supervisors, Daniel Logenius and Lars Svensson, for their invaluable guidance and advice. I would also like to thank all the people working at Ericsson for their help and support and especially Per-Arne Thorsén for the fruitful discussions we held together.

Antonios Panagiotou, Gothenburg, July 2020

Contents

List of Figures

Acronyms

ADPLL all-digital phase locked loop.

CAE computer-aided engineering.
CES circuit emulation service.
CLB configurable logic block.

DCO digitally controlled oscillator.
DUT device under test.

FIR finite impulse response.
FPGA field-programmable gate array.

HDL hardware description language.

IIR infinite impulse response.
ITU International Telecommunications Union.

LIU line interface unit.
LPF low pass filter.
LTI linear and time-invariant.

MTIE maximum time interval error.

NE network element.

OI observation interval.

PCM pulse-code modulation.
PDH plesiochronous digital hierarchy.
PDV packet delay variation.
PI proportional-integral.
PLL phase locked loop.
PSA packet selection algorithm.

RTL register transfer level.

TDC time-to-digital converter.
TDEV time deviation.
TDM time division multiplexing.
TIE time interval error.

UI unit interval.

VCO voltage controlled oscillator.

1

Introduction

This section describes communication systems in which clock recovery is used and explains which specific part of these systems induces the problem of clock recovery. Furthermore, the problem of clock recovery itself is discussed in more detail and the scope of this thesis is defined. Moreover, the implementation platform is briefly described and the research statement is presented. The chapter is finalized with an outline of the thesis report.

1.1 Background

The fundamental purpose of a communication system is the transmission of one or more signals which contain data. The most basic structure of a communication system is comprised of a source, a transmitter, a channel, a receiver and a destination for the transmitted signal. Because the channel is more often than not a scarce source, multiplexing[1] is frequently utilised.

One type of multiplexing is time division multiplexing (TDM). When TDM is applied, the transmission medium is divided into time slots. Certain time slots are allocated to each signal on a periodical basis and this is how the different signals use the same medium. This way, TDM provides constant and predictable bandwidth and latency values. Furthermore, TDM realises circuit switching since it establishes a dedicated communications channel (circuit) between the nodes that communicate during each time frame [2].

The ability to multiplex and switch several sub-channels is the main reason that TDM is widely used in applications such as 2G backhaul, broadband services, enterprise leased lines, utility communications etc. As a result, there is a large circuit based TDM network infrastructure globally. This state of affairs hinders the full transition to the newest generation packet based network infrastructure for reasons of incompatibility.

The reason that TDM infrastructure is incompatible with a packet based network is the synchronous nature of the first and the asynchronous nature of the latter.

[1] In telecommunications and computer networks, multiplexing (sometimes contracted to muxing) is a method by which multiple analog or digital signals are combined into one signal over a shared medium [1]

In the case of TDM, circuit switched equipment transmits and receives data bits continuously with fixed delay. This continuity of data transmission with a fixed delay requires synchronization between the transmitter and the receiver. On one hand, the circuit switching associated with TDM guarantees permanent transmission which is beneficial for voice-based services. On the other hand, the available bandwidth and the provided flexibility are negatively affected. In the case of IP/Ethernet and packets, transmission is not continuous and there is a variable delay between packets.

The transitional solution that allows for the implementation of TDM services over IP/Ethernet networks, which utilise packets, is a circuit emulation service (CES) [3]. A CES is a system function that is used for two purposes. The first purpose is to convert TDM traffic into packets which are then transported over packet networks. The second purpose is to convert a series of packets that arrive asynchronously into a TDM data stream. The conversion of a series of packets into TDM traffic leads to the problem of clock recovery. An abstract schematic that demonstrates a CES function is shown in Figure 1.1.

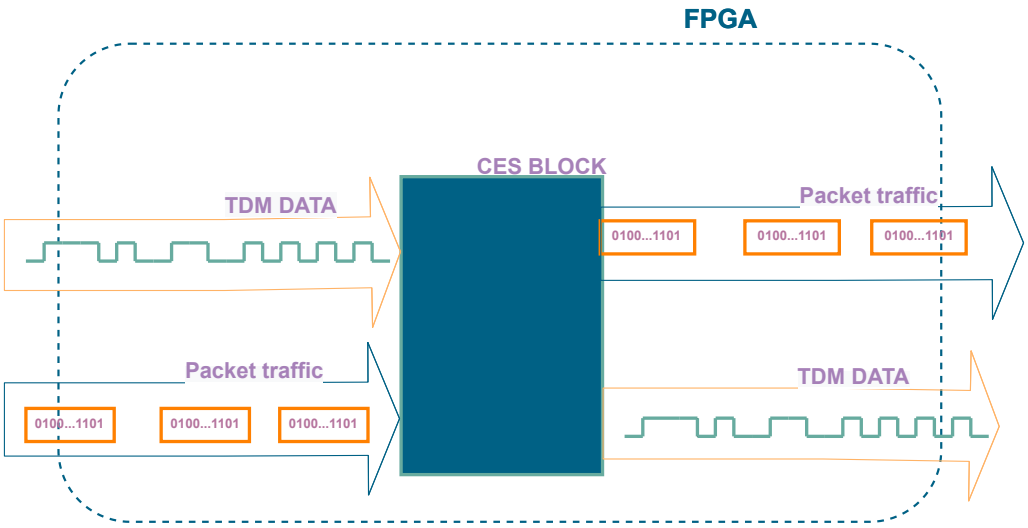

Figure 1.1: Abstract schematic of a field-programmable gate array (FPGA) based system that facilitates a block to realise a CES. The clock recovery function is necessary for the derivation of TDM data from incoming packet traffic, a process which is presented in the lower half of the Figure.

1.2 The problem of clock recovery

To better explain the problem of clock recovery, an example of two communication nodes, which both implement a CES, is considered. On the transmitter's side, a CES converts TDM traffic into packets of the same size, which are generated at defined intervals, as is demonstrated in Figure 1.2. The generated packets go through a series of network elements (NEs), such as switches and routers, before reaching their destination. In an ideal situation the packets would arrive with the exact same rate as they were generated by the transmitter. The presence of NEs, in combination with system and noise disturbances, may result in packets arriving later than they

are expected, in packets getting lost or in a lot of packets arriving simultaneously, as is demonstrated in Figure 1.3. Thus, the traffic load of the network causes a variance in the arrival time of each packet. In other words, the system and its clock recovery function is greatly affected by packet delay variation (PDV).

The incoming packets are used to extract timing information by comparing the measured arrival time against the expected arrival time. Thus, a deviation from the expected arrival causes a synchronization problem. The difference between the real and the expected arrival time results in a phase difference between the clock signal in the transmitter's side and the recovered clock in the receiver's side and this is why a phase locked loop (PLL) is used to achieve clock recovery.

Figure 1.2: Generation of packets in predefined intervals at the transmitter's side. Each orange box represents a packet each downward arrow the time instant when a packet is generated. The packets are generated at a specific rate and subsequently go through a series of NE.

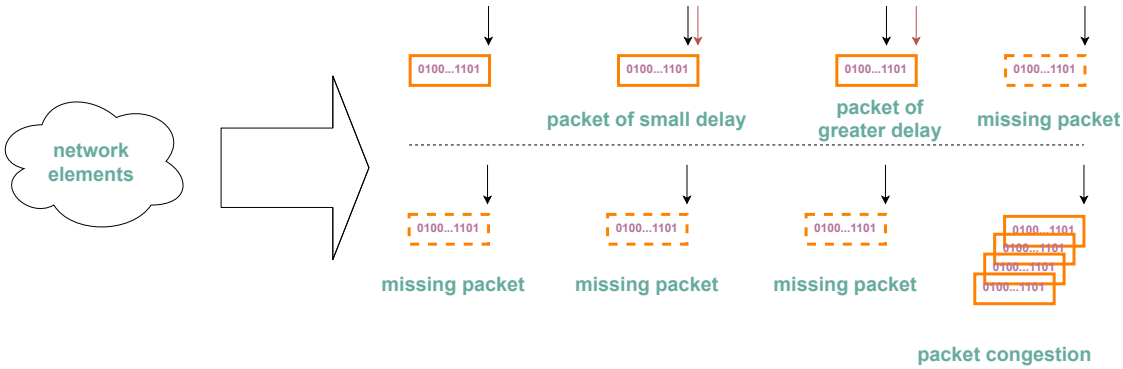

Figure 1.3: Examples of packet delay variation. Each orange box represents a packet. Each downward black arrow represents the expected arrival time. Each downward read arrow represents the measured arrival time. In an ideal situation packets arrive at the exactly expected time. In more realistic situations packets might have smaller or larger delay, packets might go missing and packets might get queued in a series of NE and arrive in batches at almost the same time.

A PLL is an electronic circuit comprised of a phase detector, a LPF, a voltage controlled oscillator. The design of the filter greatly determines the performance of the PLL since it affects the PLL's response time and bandwidth [4]. Thus, the problem of clock recovery is largely a problem of filter design.

By examining the function realised by a PLL, it is observed that a PLL changes its output frequency so that the phase difference between the receiver clock and the local clock converges to zero. This difference can be referred to as an error that needs to be kept below a certain threshold. Taking this into account, clock recovery can be seen not only as a problem of filter design but also as a problem of controller design.

For the special case of communication systems, the performance requirements for such a filter are dictated by the ITU-T[2]. More specifically, the ITU-T G.823 standard [5] specifies a series of timing requirements that need to be satisfied by communication systems that are using a CES and belong to a category of systems that utilise a plesiochronous digital hierarchy (PDH). The ITU-T G.823 standard is complemented by the G.8261 [6] standard which describes a series of test cases for the formal verification of such systems. The requirements of ITU-T G.823 are the basis for the design of all the proposed clock recovery algorithms while the test cases of ITU-T G.8261 are the basis for the final verification of the proposed designs.

At this point, it might be useful to clarify that the general problem of clock recovery is not limited to the specific problem of filter design. Missing packets, packets whose integrity has been compromised or packets which are extremely delayed are bad candidates to recover a clock signal from. Thus, clock recovery also includes the problem of deciding which packets are suitable to be used and which need to be ignored [7]. For this reason, a packet selection algorithm (PSA) decision algorithm is often used in clock recovery systems but the use and application of these algorithms are beyond the scope of this thesis.

1.3 Implementation platform

The implementation platform is an FPGA based communication system. An FPGA is a semiconductor device consisting of a matrix of configurable logic blocks (CLBs) which are connected via programmable interconnects. The programmability of interconnections between the CLBs allows for the FPGA to be used in conjunction with hardware description languages (HDLs). Hence, the device that utilises it can be reprogrammed so that any additional application and functionality requirements can be satisfied even after the physical device has been manufactured [8]. In other words, the use of an FPGA provides flexibility. The same hardware device can be re-purposed, upgraded in features on-site, used as a test bed for hardware designs or used for different iterations of a specific hardware design.

The hardware that utilises the clock recovery function is an embedded electronic system based on a Xilinx Kintex 7 series [9] FPGA. Embedded electronic system design using FPGAs is associated with the use of an HDL, as well as a series of software tools with each tool corresponding to an implementation stage. Implemen-

[2]ITU-T is the Telecommunication Standardization Sector of the International Telecommunications Union (ITU), which is a specialized agency of the United Nations which is responsible for issues that concern information and communication technologies.

tation stages include register transfer level (RTL) simulation, synthesis and 'place and route'. RTL simulations were done using Questasim. The synthesis part was done using Synopsys Synplify Pro, while the 'place and route' stage was done using Xilinx Vivado.

1.4 Research statement

The goal of this master's thesis was to design a clock recovery subsystem that supports a circuit emulation service (CES) over a packet based network, using an FPGA-based hardware platform. The subsystem that the project is dealing with is a low pass digital filter. The criterion that decides the success of the project is whether the system that utilises the proposed design can satisfy a series of tests dictated by the ITU-T G.8261 standard [6].

The main scope of this thesis was to evaluate different kinds of PLL filters. More specifically, the tasks defined for the thesis were to :

- study of the clock recovery function and the performance requirements set by the ITU-T standards
- develop an alternative filter algorithm and explore the option of a Kalman filter
- evaluate the new algorithm in a simulation environment and compare to the existing solution
- implement and integrate an alternative algorithm in an existing FPGA based design
- run the alternative algorithm in the system and evaluate the result

1.5 Limitations to mitigate risks

The scope of this thesis was mainly focused on the digital filter design part of an all-digital phase locked loop (ADPLL). It essentially deals with improving the performance of an ADPLL. This ADPLL is used in a clock recovery system and its current performance fails to comply with the required specification. In other words, the study will address the problem of replacing the currently used filter implementation with a new one.

Furthermore, the thesis does not deal with the problem of clock recovery on packet based networks from a packet selection perspective, which is an equally important aspect of clock recovery on such systems, as is indicated by the relevant literature in [10], [11], [12], [13].

Lastly, the thesis goal was reconsidered and redefined so that its focus was to achieve successful clock recovery for the CES part of the FPGA system regardless of the

filtering algorithm that is used to achieve it. In other words, possible solutions that might have been more interesting to implement, from a research standpoint, were not preferred against simpler solutions. This choice was made to satisfy the thesis research statement under the given time frame.

1.6 Thesis outline

Chapter 2 explains some basic concepts associated with the problem of clock recovery. It also contains two sections that introduce the theoretical tools that are used to attack this problem. The purpose of these sections is to present a more complete flow of the work done during this thesis, as well as to provide the reader with the theoretical background that is necessary to understand both the analysis and the results which are presented in later chapters.

Chapter 3 describes the system design process that formed the basis for the proposed design. Furthermore, the testing methodology, with regards to tools and procedures, is presented.

Chapter 4 presents the results of the work that was conducted. It includes simulation results as well as the results of the testing that was conducted to verify the correctness of the proposed design on actual hardware.

Chapter 5 discusses the results presented in the previous chapter. Suggestions for potential improvements and further study are also included in this chapter.

Chapter 6 presents the conclusive points of this thesis, which correspond to the research statement described section 1.4.

2

Theory

The challenge of clock recovery can be viewed as a problem deriving from mis-timing. The arrival of a packet at a later point in time than the one expected is the reason that a clock recovery method might fail. The cause of this variance of arrival times between packets is attributed to network traffic. Variances in network traffic result in different arrival times of packets to their destination either because packets need to be transmitted through longer or shorter network pathways or because a lot of packets arrive at their destination in bulk. This is quite often the case at a network system's startup or when the level of modulation drops and the network speed is greatly reduced.

Because of the reduction in network speed a lot of packets might be queued at the transmitting end and initially cause congestion on the transmitting end and starvation in the receiving end. Once the network speed is restored to nominal levels, the congested packets are transmitted back-to-back causing a congestion on the receiver. This congestion results in packets arriving at a later time than expected thus causing jitter in the recovered signal. This jitter translates into phase noise that needs to be filtered out by the all-digital phase locked loop filter.

This section clarifies the basic concepts that are associated with the problem of clock recovery and the type of network systems that utilize it. It also establishes the theoretical background that supports the methodology used in addressing this problem.

2.1 Basic concepts

The most basics concepts that are related with the problem of clock recovery are physical quantities used to describe timing and mis-timing, as well as the relation between time and phase both of which are used as means to express a difference between signals.

2.1.1 Timing

Same as the physical world, the concept of timing in digital systems is closely associated with the existence of a clock. In the case of digital systems, a clock is a square wave signal with a fixed period. This period is measured from the edge of

the clock to the next similar edge of the clock. More often than not, the rising edge of a clock is the one that is used to define the period of the clock [14]. An ideal clock signal, which has a 50% duty cycle , is presented in Figure 2.1. Other than the period, clock signals are described by their frequency. Frequency is defined as the inverse of a period.

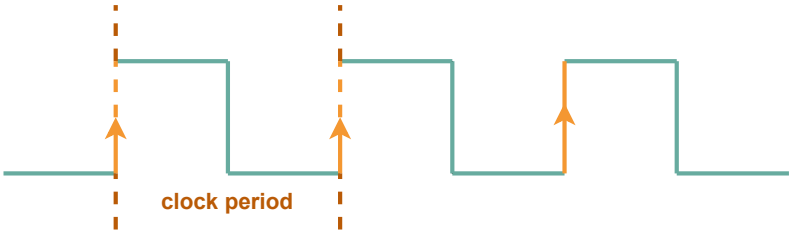

Figure 2.1: Representation of an ideal clock signal.

The need for a clock signal comes from the requirement of synchronization between the components of a system or synchronization between different systems. In the case of a communication system, the latter case is of interest. Successful communication requires that both the transmitter and the receiver use the same clock to derive their timing signals.

Inevitable disturbances, for example fluctuations in the phase of the oscillator that generates the timing signal, result in actual timing signals to be pseudo-periodic and slightly differ in phase.

In an ideal timing signal of nominal frequency f_{nom}, the total phase ϕ_{ideal} is given by equation 2.1

$$\phi(t) = 2\pi f_{nom} t \tag{2.1}$$

where f_{nom} is the nominal frequency of the signal [15].

2.1.2 Mis-timing : Jitter and Wander

Jitter and wander are closely related concepts in the sense that they are both used to describe discrepancy between the expected and the real position of a signal with relation to time. In other words, they describe mis-timing between two systems. Both of them are only useful as a means of expressing the relation between two different signals.

According to the ITU-T G.810 standard, timing jitter is “the short-term variations of the significant instants of a timing signal from their ideal positions in time (where short-term implies that these variations are of frequency greater than or equal to 10 Hz)”. The same standard defines wander as “the long-term variations of the significant instants of a digital signal from their ideal position in time (where long-term implies that these variations are of frequency less than 10 Hz)” [15].

Both jitter and wander are described using an amplitude and a frequency quantity. The amplitude quantity expresses how much a signal is shifting in phase when compared to a specific reference signal. The frequency quantity expresses how quickly the signal is shifting in phase, when compared to the same reference signal. Taking into account the definitions of the previous paragraph, the frequency quantity is the differentiating factor between the two: phase variation with a rate higher than 10 Hz is defined as jitter and phase variation with a rate lower than 10 Hz is defined as wander.

Jitter is measured in unit intervals (UIs). One UI is equal to the width of one data bit. It follows that different data rates define different unit intervals. For example, a data rate of 2048 kbps corresponds to a UI of 488.28 ns, while a data date of 565.14 Mbps corresponds to a UI of 1.77 ns.

In the specific case of clock recovery, we want the jitter/wander to remain within predefined limits. These limits are dictated by the ITU-T G823 standard [5] and they are expressed in terms of time deviation (TDEV) and maximum time interval error. Both MTIE and TDEV are derived from the time interval error (TIE). The TIE is defined as the time difference between the recovered clock signal and the reference clock signal respectively.

By measuring the TIE values in a specified observation interval (OI), we can define a magnitude that will describe the intensity of frequency offset and phase transients. By observing the peak to peak TIEs within an observation interval we can decide on the maximum of these TIEs and this is how the MTIE is defined. Thus, MTIE is a function of the observation interval and it increases monotonically with it.

An estimation formula for the MTIE is given by equation 2.2, where $x(i)$ is the i-th time error sample, τ_0 is the sampling period of the error signal, $\tau = n\tau_0$ is the observation interval, $N = \frac{T}{\tau_0} - 1$ is the size of the measurement period T expressed in number of time error samples and k, $k+n$ are the limits of the observation interval τ [15].

$$MTIE(n\tau_0) = MTIE(\tau) \cong \max_{1 \le k \le N-n} \left(\max_{k \le i \le k+n} x(i) - \min_{k \le i \le k+n} x(i) \right), n = 1, 2, ..., N \tag{2.2}$$

2.1.3 Phase and time

Taking into account equation 2.1, we can express the phase of a signal as a function of the angular frequency ω with the use of equation 2.3.

$$\phi(t) = \omega t \tag{2.3}$$

Given that the angular frequency ω is related to a frequency f by equation 2.4

$$\omega = 2\pi f \tag{2.4}$$

we get that

$$\phi = \frac{t}{T} 2\pi \tag{2.5}$$

Solving equation 2.5 for t we get

$$t = \frac{\phi}{2\pi} T \tag{2.6}$$

where T is the signal's period.

By using equations 2.5 and 2.6, we can express the mis-timing between two signals, as a fractional offset from the nominal values of the signal's phase and period, respectively. Furthermore, we can use these equations to translate a phase difference (expressed in radians) into a time difference (expressed in seconds) and vice versa.

2.2 Clock recovery and digital filter design

Since the design and implementation of a digital filter is the main goal of this thesis, it is useful to clarify what is a filter and how does it relate to the task of clock recovery.

2.2.1 Introduction to filters

A filter is a process or device that considers a signal as an input and generates an altered version of the input signal at its output. By applying the Fourier transform, a signal can be described as series of component signals, each with each own frequency, amplitude and phase. Thus, the altered version of the input signal, which is derived at the output of a filter, is a signal whose components differ in amplitude, frequency or phase.

One of the most prominent uses of these filters is to suppress the amplitudes of components of certain frequencies in which case they are called frequency-selective. The range of frequencies that a filter suppresses defines a filter as a low-pass, band-pass or high pass filter. A low-pass filter suppresses components of high frequency. A band-pass filter suppresses components outside a specified frequency range in-between low and high frequencies. Finally, a high pass filter suppresses components of low frequency.

The input signal and the processing operation that the filter applies can be either continuous or discrete-time functions, thus defining two groups of filters. The first

group of filters, that refers to a continuous time signal and a continuous time operation are called analog filters. An analog filter is implemented with the use of analog components such as capacitors and resistors. The seconds group of filters, that realise a discrete mathematical function are called digital filters. A digital filter is implemented with a digital hardware component or a software algorithm that realizes a function which is a combination of the basic operations of delay, addition and multiplication.

In the specific case of clock recovery, we are interested in a digital filter implementation since the signal that is processed to recover a clock signal is a discrete-time signal.

2.2.2 Introduction to digital filters

This project deals with frequency selective filters which are linear and time-invariant (LTI). The mathematical functions realized by LTI digital filters are discrete convolutions, that is, operations which are both linear and shift invariant. Other than linearity and shift invariance, there are other qualities that can characterize a digital filter. These qualities are causality, stability and linearity of phase [16].

Causality defines whether the output of a filter is a function of current, previous or future values of its input. A causal filter is a filter whose output is only dependent on the current and previous inputs [17]. A non-causal filter is a filter whose output is also dependent on future values of its input. Causality is a very important quality for a filter design because only causal filters can be implemented in hardware [18].

A stable filter is a filter whose output remains bounded as long as its input is bounded [19]. The linearity of phase refers to whether the phase of the output signal components changes linearly with the frequency of each component. Taking that into account, a linear phase filter is a filter that shifts all the input components by the same amount of phase [20].

Digital filters can also be characterized by examining their impulse response[1]. If the impulse respond eventually converges to zero then the impulse response is characterized as finite and the respective filter is called an FIR filter. In contrast, when a filter's impulse response doesn't become zero after a specified value of discrete time then it is called an IIR filter.

A digital FIR filter can be described by mathematical formality either in the time domain, with a difference equation, or in the Z-domain, with the use of the Z-transform[2]. In the time domain, an FIR filter's response is modeled with equation 2.7 , where $y[n]$ is the filter output at instant n, $x[n-k]$ is the output at instant

[1]The impulse response of a digital filter is the output of the filter when the input is the Kronecker delta function. The Kronecker delta function is the discrete-time equivalent of the Dirac delta function. Both functions' value are zero for every value of their dependent variables, other than n=0 and t=0, respectively.

[2]The Z-transform is the discrete-time equivalent of the Laplace transform. Both transforms convert a function of time into a function of complex frequency [21] [22]

$n-k$, M is the order of the filter and b_k is the filters impulse response the k-th instant where $0 < k < M$. If equation 2.7 is directly implemented the values of b_k are referred to as the filter's coefficients or the filter's 'taps'.

$$y[n] = \sum_{k=0}^{M} b_k * x[n-k] \tag{2.7}$$

The FIR filter can also be modeled in the Z-domain using equation 2.8 [17], where $H_{FIR}(z)$ is the filter's transfer function.

$$H_{FIR}(z) = \sum_{k=0}^{M} b_k * z^{-k}. \tag{2.8}$$

Similarly, an IIR filter can be modeled in the time domain with the difference equation 2.9, where $y[n]$ is the filter's output at instant n, $x[n-k]$ is the filter's input at instant n-k, $y[n-k]$ is the filter's output at instant n-k, b_k are the feed forward filter coefficients, a_k are the feedback filter coefficients, M is the feed forward filter order and N is the feedback filter order. In contrast to the case of the FIR filter the IIR filter equation includes previous values of the filter's output.

$$y[n] = \sum_{k=0}^{M} b_k * x[n-k] - \sum_{k=0}^{N} a_k * y[n-k] \tag{2.9}$$

The IIR filter can also be modeled in the Z-domain by its transfer function $H_{IIR}(Z)$, given by equation 2.10 [17].

$$H_{IIR}(Z) = \frac{\sum_{k=0}^{M} b_k * z^{-k}}{\sum_{k=0}^{N} a_k * z^{-k}}. \tag{2.10}$$

In both cases of filters that were described above, when the filters' coefficients are independent of the input and the output then the filters are linear. If the filters' coefficients are also fixed then the filters are LTI.

2.2.3 Digital filter design theory

As far as FIR filters are concerned, the design process refers to the determination of the filter's b_k coefficients. An FIR filter's coefficients can be calculated using various methods such as the Parks–McClellan algorithm, the Kaiser window or the least square FIR method [23].

When it comes to the design of IIR filters there is the option of the Deczky method as well as the option of adopting an analog filter design method (Butterworth, Chebyshev I and II, and elliptic).

Transitioning from an analog to a digital implementation that uses a fixed-point number representation means that the coefficients are only approximations of the

derived coefficients, which are usually transcendental numbers. This causes a discrepancy between the pole and zero locations of a digital filter and its analog equivalent. Another important issue has to do with fixed-point arithmetic. More specifically, the operations of addition and multiplication cause round-off quantization noise. Finally, another problem is that the signal might need to be scaled so that its maximum and minimum values can be represented during processing [24].

2.2.4 Filtering in a clock recovery process

As mentioned in section 2.2.1, filters are used to suppress frequency components that belong to a certain range of frequencies. The need for the suppression of certain frequencies derives from the specific application that the filtered signal is used in. For example, if we want to transmit a digitally recorded sound signal we want to filter out frequencies above the audible range of 20 kHz. On top of that, we need to filter out frequencies to deal with the problem of aliasing [3].

In the case of clock recovery, we need an LPF to filter out high frequencies in the signal that controls the frequency output of a PLL. This signal is derived from the phase detector part of the PLL and it changes its value according to the detected phase difference between the transmitted timing information and the locally generated clock of the receiver. Very rapid changes in this control signal, attributed to components of high frequencies, can cause the oscillator to constantly change its output in very short periods of time. This could render the oscillator of the loop unable to settle on a certain frequency and affect the system's ability to achieve clock recovery.

2.2.5 PLL model and implementation

A very simple model of a PLL is presented in Figure 2.2 in the form of a flow chart. Since the project is part of a digital system, an ADPLL model was used instead, as a basis for the design which extracts timing information from a transmitted input signal, comprised of a series of packets, and generates a local clock signal that is in phase with the transmitted clock.

In the case of an ADPLL, the phase detector functionality is achieved with the use of a time-to-digital converter (TDC), which converts the time difference between two clock events into a digital number [25]. Furthermore, output frequency generation is achieved with the use of digitally controlled oscillator (DCO), which generates an oscillating signal with a period which is a function of a digital input word [26]. On top of that, the use of digital parts for the implementation of the loop necessitates the use of a digital filter in the place of the analog filter of a traditional PLL. A generic flow chart of an ADPLL is presented in Figure 2.3.

[3] Aliasing describes the problem of different signals becoming indistinguishable from each other when they are sampled.

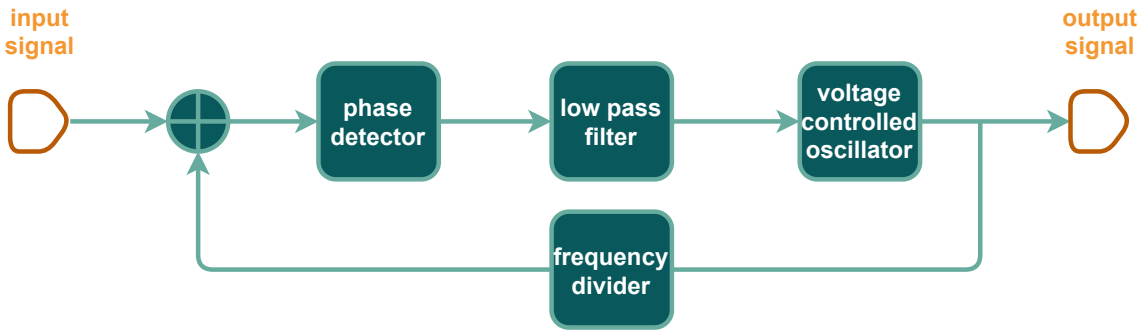

Figure 2.2: The main parts of the PLL

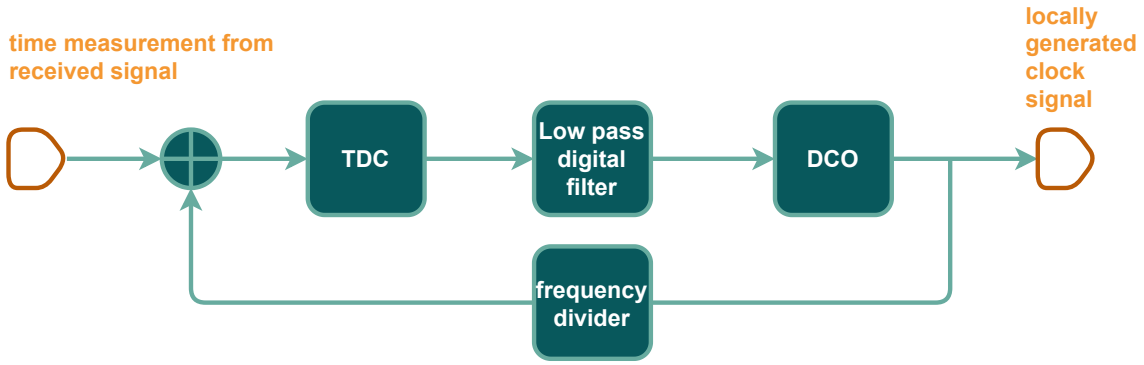

Figure 2.3: Generic diagram of an ADPLL. The phase detector is realised with the use of a TDC, the LPF is realised with a digital implementation and the voltage controlled oscillator (VCO) is realised with the use of a DCO

3

System Design Process and Testing Methodology

This chapter motivates the filter design process that was followed, explains the derivation of the different filtering algorithms that the project dealt with and describes the testing methodology that was followed to test the suggested implementation.

3.1 Filter design methodology for clock recovery

The system design process that was followed deviated from the one indicated by relevant literature [23], [27], [28], where a given frequency response requirement is the starting point of the design and the consequent steps of the design process are involved with achieving this frequency response as closely as possible .

In the case of clock recovery for communication systems, the specification is given in terms of MTIE threshold value and not in terms of a frequency response. A frequency response specification would provide specific values for the passband, the transition band and stopband of the desired filter. Consequently, the use of computer-aided engineering (CAE) tools to derive filter parameters or simulate filter performance was of limited use. By extension, the automatic generation of HDL code was not feasible so any proposed algorithm needs to take into account the problems associated with the design of digital filters and were mentioned in section 2.2.2.

Thus, a simulation algorithm was developed in order to predict the PLL's performance by producing a series of plots. These plots demonstrate the difference between the unfiltered input and the derived signal and whether the simulated loop performance, measured with regards to the MTIE, stays within the limits that are dictated by specification.

The plots produced by the simulation tool were used to compare different iterations of a filter design as well as different filter designs. Based on the derived simulation results a filter design was chosen to be implemented. The following sections describe the above process in more detail.

3.2 Initial design of a low pass filter

Before proceeding with an alternative filter design, it was useful to start with the simulation of a design that corresponds to what was already implemented in HDL. This was necessary for several reasons. The first reason is that using and improving an existing implementation, rather than starting with a totally new design, allows for a better understanding of the system. Secondly, it allows for a quick transition to an implementation attempt once the simulation results confirm the correctness of the design. Furthermore, the current filter solution was used to confirm that the simulation gives trustworthy results and that it can used to design and consequently predict the performance of alternative filters.

The filter already implemented in the system was a PI controller. If we compare the transfer functions of a second order LPF, given in equation 3.1 [29], and the transfer function of a PI controller system with negative feedback, given in equation 3.2, we conclude that the frequency response of the PI controller coincides with that of a low pass filter.

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.1}$$

$$H(s) = \frac{k_p s + k_i}{s^2 + k_p s + k_i} \tag{3.2}$$

Thus, the design process started with the assumption of a PI controller in the place of a low pass filter. The parameters of a controller of this kind are the values of its gains; the proportional gain, k_p, and the integral gain, k_i. The design of such a filter refers to the definition of these parameters.

On top of that, the implementation of the TDC and DCO implies the use of sensitivity quantities for their modeling. As such, the TDC is modelled by the definition of a sensitivity quantity k_d, that defines the range of phase difference values that it can detect. In other words, the value of k_d determines the resolution of the TDC. Similarly, the sensitivity quantity k_g, determines the range or the DCO. Furthermore, with the use of negative feedback the output frequency of the ADPLL is derived by multiplying the phase detector comparison frequency by N [30].

Taking the above into account, a more specific model of the implemented ADPLL is derived and presented in Figure 3.1.

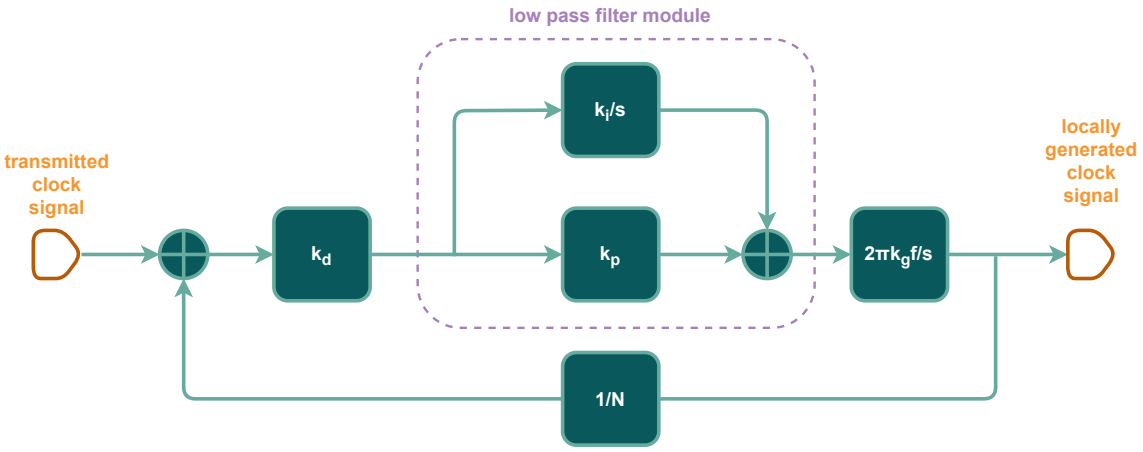

Figure 3.1: Specific diagram of an ADPLL utilizing a PI controller as a low pass filter

In order to define a PI controller's parameters, we need to express them using mathematical formality. The respective model for the controller itself is derived using the diagram shown in Figure 3.1 and is given by equation 3.3, where N is the digital gain of the loop.

$$H(s) = N\frac{\frac{2\pi k_g k_p k_d}{N}s + \frac{2\pi k_g k_i k_d}{N}}{s^2 + \frac{2\pi k_g k_p k_d}{N}s + \frac{2\pi k_g k_i k_d}{N}} \tag{3.3}$$

If we were to describe the system in terms of its natural frequency, w_n, and its damping factor, ζ, then equation 3.2 can be rewritten as equation 3.4.

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.4}$$

By comparing equations 3.3 and 3.4 and by solving for the natural frequency, ω_n, and the damping factor, ζ, we get:

$$\omega_n = \sqrt{\frac{2\pi k_g k_i k_d}{N}} \tag{3.5}$$

and

$$\zeta = \frac{k_p}{2k_i}\sqrt{\frac{2\pi k_g k_d k_i}{N}} \tag{3.6}$$

Although the equations presented above are derived by considering transfer functions in the Laplace domain, they are considered a valid approximation for the digital implementation under discussion. This is because the loop's bandwidth is a small fraction of the reference clock frequency [31].

At this point, it would be useful to clarify that the loop's bandwidth refers to the range of frequencies that the loop can accept as an input and alter its output to match that input. When, after a transitional time frame, both the input and the output of the ADPLL are in phase then the loop's state is referred to as the "locked" state.

By examining equation 3.5 it should be quite clear that the ADPLL's bandwidth is dependent on the integral gain. At the same time, the damping factor is depended on both the integral and the proportional gains of the filter. Moreover, the design needs to take into account that using high gains for the filter will allow the PLL to track the change of the input frequency but may result in it not being able to lock. At the same time, low values for the gains will allow the loop to lock but will limit its bandwidth. These conclusions were the basis for the subsequent design.

The design makes the assumption that when the system starts tracking there is a need for the loop to have higher bandwidth, which corresponds to higher values for k_i and k_p. Then, as time progresses there is a need for less bandwidth so the gain values can be lowered to allow for the loop to lock. When enough time has passed the filter's gains are lowered even further with the aim to reduce the unwanted jitter or wander. Thus, the design assumes three levels of gain values, namely high, medium and low, and changes these values from high to low as time passes. The simulation of the system is presented in section 4.

3.3 Alternative designs of a LPF

The use of any of the methods mentioned in section 2.2.2, either for designing an FIR filter or an IIR filter, assumes knowledge of the magnitude of the desired frequency response of the filter. In the case of clock recovery this kind of frequency response specification is not given.

On top of that the design needed to take into account the complexity associated with the fixed point arithmetic of a digital filter described in HDL. A way to circumvent the issues caused by fixed-point arithmetic is to implement the filter using a number of elemental building blocks comprised of filters of the smallest possible order [24]. Thus, the design proceeded with the simplest possible filters with the intention of using them as units that can be either repeated or units that can have their parameters altered, during simulation or implementation testing, so that they match the MTIE specification that was given.

If we were to take into account the equations of section 2.2.2 we could come up with two very simple designs for an FIR and IIR filter. The simplicity of the designs is attributed to the use of the smallest number of terms described by these equations which correspond to the smallest possible order for either case of filtering algorithm. A very simple FIR filter is described by equation 3.7.

$$y[n] = b_0 * x[n] + b_1 * x[n-1] \tag{3.7}$$

Similarly, a simple IIR filter is described be equation 3.8.

$$y[n] = b_0 * x[n] + a_1 * y[n-1] \tag{3.8}$$

3.3.1 Design of an FIR filter

By examining equation 2.7, we can see that each addend of the filter's formula is comprised of a coefficient and a previous sample of the signal that needs to be filtered. Moreover, each addend requires two basic arithmetic operations : multiplications and addition. As far as embedded electronic system design is concerned, both multiplication and memory elements are costly in terms of area. Other than that, the problem of design implies the determination of the filter's coefficients. As was already mentioned, determining these coefficients with no specified frequency response can be a serious hindrance to proceeding with a filter design.

In order to deal with the issues of limited logic resources and the lack of frequency response specification, the simplest possible case of an FIR filer was examined. More specifically, by setting $b_0 = b_1 = 1$ in equation 3.7 we get equation 3.9

$$y[n] = x[n] + x[n-1] \tag{3.9}$$

This very simple design requires neither a multiplication operation nor the definition of filter coefficients and it allows for a filter design that produces its output at the system clock rate. On top of that, implementation of the filter would require the filter output to grow only by one (1) bit to accommodate a possible overflow caused by the addition operation.

Such a simple filter design was not expected to be very efficient in terms of how selective it is with the filtering of higher frequencies but was examined as a building block for a higher order filter comprised of a series of this block [16].

Given the complexity of the system that the filter design needed to be integrated in, the use of a simple FIR filter as an elementary block was dismissed cause it would require multiple instances of syntheses to optimize the filter's performance after the first synthesis instance was loaded in the FPGA system. In other words, the choice of not proceeding with an FIR based solution was made for reasons related to system integration.

3.3.2 Design of an IIR filter

By considering equation 3.8 and by defining the IIR filter's coefficients as $b_0 = \alpha$ and $a_1 = 1 - \alpha$ in equation 3.8, we get a filter referred to as a recursive averager and is described by equation 3.10.

$$y[n] = \alpha * x[n] + (1 - \alpha) * y[n-1] \tag{3.10}$$

The option of the recursive average solves the problem of defining the filter's coefficients by introducing the parameter α in the filter's model. Given that both

coefficients of the filter are a function of the α parameter provides two advantages. First, it reduces the number of unknowns that need to be determined for a design. Secondly, it allows for a highly flexible design where the change of a single parameter can alter the filter's performance to match specification [16].

This flexibility of this solution isn't limited to the design and simulation of the filter. It can also be utilized to customize the filter's performance after the design has been synthesized. This customization of the proposed IIR filter's performance can be achieved with the use of a control register within the design that holds the value of α.

The difference equation 3.10 can be re-written as equation 3.11

$$y[n] = y[n-1] + \alpha * (x[n] - y[n-1]) \tag{3.11}$$

Choosing to implement equation 3.11, instead of equation 3.10, reduces the number of required multiplications to just one. Same as with the case of the proposed FIR filter the output of the filter only increases by one bit due to the addition operation that ultimately produces the filter output.

Other than that, by making the design choice of defining only α values that are negative powers of 2, the multiplication operation can be replaced by an arithmetic left shift operation. This design choice provides two advantages. First, the multiplication operation can be completed in one system clock cycle. Secondly, the multiplication with a pure fraction excludes the occurrence of overflows and eliminates the necessity to truncate the bit width of a multiplication result to the original word length of the multiplied quantity [32]. A series of simulations that demonstrate the predicted performance of the recursive averager is presented in chapter 4.

3.4 The option of state estimation filter algorithms

Another filtering option, which would be described by an algorithm whose parameters would not be static but dynamic, is a state estimation algorithm, which is known as a Kalman filter. Although there are variations of the Kalman filter, such as the Extended Kalman filter and the Unscented Kalman filter [33], they all assume a Gaussian distribution of the uncertainties included in a system's model [34]. This means that that a Kalman filter option is associated with the additional challenge of the modelling of a system's uncertainties and the definition of their distribution.

On top of that, the state space analysis associated with the application of a Kalman filter might require operations on large matrices. Taking that into account, one source of complexity for a Kalman filter implementation on an FPGA is to create a so-called systolic array, which is dedicated hardware with a parallel, pipelined architecture for matrix operations. On the other hand, if the matrices of the problem at hand are small in dimensions it may be a better approach to design application specific hardware for the fundamental functions of the Kalman equations. Some basic operations within these functions are the same which means that if they are executed simultaneously, they will cost considerable hardware resources. Given the

dependence of these, where one needs to be calculated before the other, we might reuse hardware to optimize resource usage. On top of that, for those functions that include a large number of the same operations we can apply pipelining and folding to further optimize resource usage without deteriorating throughput to a great extend [35].

Given the unpredictable and asynchronous nature of the incoming packet data, modelling the distribution of the system disturbances proved to be a great challenge. Since the systems disturbances couldn't be proved to comply with a Gaussian distribution the Kalman filter was not chosen as an option.

3.5 Implementation design

The comparison of the simulation results of the designs described in previous sections lead to the following observations.

First, the FIR filter design, as expected failed to satisfy the MTIE specification and was dismissed as an option.

Secondly, the reference design performance also fell within specification according to the simulated performance. Since the respective implementation failed to meet specification, until a packet selection algorithm was applied, lead to the conclusion that there will be a discrepancy between the designed and the real performance of the system.

Thirdly, the IIR filter's performance did fall within specification (except for some values of the alpha parameter) but didn't always perform better than the reference design. More specifically, the IIR filter simulation gave a much longer settling time although it performed with much less overshoot.

These observations lead to an implementation proposal that utilizes both a PI and a IIR filter in series. The PI filter's parameters were kept as they were initially set. When it came to the IIR filter its design was determined by the simulation results. An initial alpha value of $2^{-8} = 0.039$ was chosen as a starting point as it provided quick settling time and no overshoot.

A block diagram of the proposed filtering solution is presented in Figure 3.2. A series of simulations that demonstrate the predicted performance of the proposed design is presented in section 4.

3.6 Testing and implementation methodology

In order to test and implement the proposed design a test setup was used to conduct a series of predefined test cases.

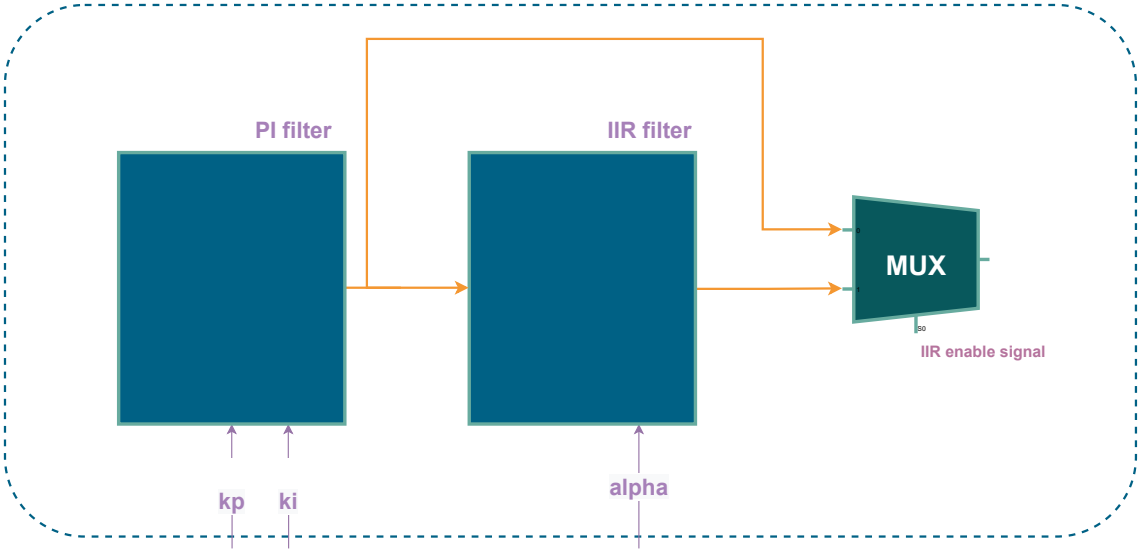

Figure 3.2: Combined filter implementation block diagram

3.6.1 Testing hardware and test setup

The test setup realizes a communication loop where data are transmitted and then received back to verify successful communication. It comprises of two FPGA network devices, which are referred to as leaf nodes, as well as two testing devices. The first device is used to generate network traffic, as well as emulate a series of network elements (NEs). Thus, it initiates the communication, by playing the role of a transmitter, and at the same time it replaces a number of NE that would have been placed in between the two leaf nodes. The second testing device extends the functionality provided by the network emulating device and is used to detect the presence of specific fault signals. An abstract diagram of the test setup is presented in Figure 3.3.

Each test session includes the following processes. To begin with, the testing equipment generates and transmits data using an E1 transmission link [1]. Then, the first leaf node converts the transmitted information into CES packets which are then fed to an Ethernet port of the test equipment. After going through an emulated network, within the test equipment, the packets are fed into a second leaf node using an Ethernet connection. This series of packets is then used by the second leaf node to terminate the CES packets and extract timing and data information. The received data are then transmitted back to the testing using an E1 transmission link.

3.6.2 The ITU-T test cases for adaptive clock recovery

The ITU-T G.8261 standard's document [6] describes a group of ten test cases that need to be conducted to verify the compliance of systems using PDH synchronization interfaces. The test cases that are most relevant to this project are the first two test

[1]E1 is a 2048 kbit/s pulse-code modulation (PCM) communication system mainly used in Europe [36]

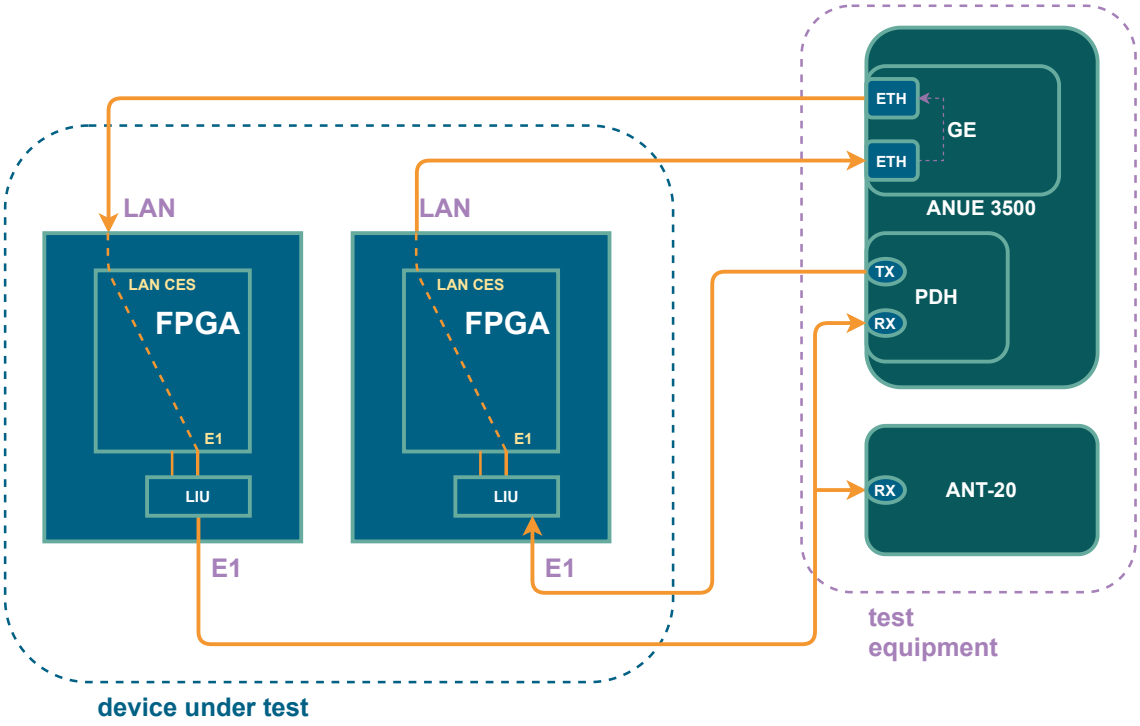

Figure 3.3: Device test setup diagram. The "ANUE 3500" device is used to emulate network traffic and NEs. The "ANT-20" device is used to detect the presence of fault signals, if any.

cases. These test cases differ from each other with regards to the system's stimulus that is the network traffic. The rest of the test cases examine system performance during network disturbances and were considered irrelevant to the testing of the proposed filter's performance. Thus, the implementation testing and the respective results didn't include all the test cases described in the ITU-T G.8261 document [6].

3.6.3 Implementation methodology

The first stage in implementing the proposed design was to run an RTL simulation that confirmed that the proposed filter's output is the expected one for both positive and negative input values. A negative input value corresponds to a measured arrival time that is earlier than the one expected. Similarly, a positive value corresponds to a measured arrival time that is greater than the one expected.

The second stage was to load the proposed design into the device under test (DUT). In contrast to the simulated system, the DUT implements a PSA as part of its clock recovery function. The implemented PSA uses a specified time window before deriving a timing measurement. This measurement is used as an input to the ADPLL. Given that the simulation results didn't consider how the use of a PSA would affect performance, a deviation between the simulated and the measured performance was expected.

Thus, before proceeding with the use of the DUT as part of the test setup described

in Section 3.6.1, some preliminary testing was conducted. More specifically, the behavior of the signal that is used as an input to the ADPLL was examined at RTL level, for different values of the low-pass filter's α value. The α values that were considered in these preliminary tests belonged to the range of values that were predicted to provide performance that doesn't violate the MTIE specification.

Finally, by examining how the filter's output was affected for different settings, three pairs of α values and PSA window sizes were tested in a real setup. By comparing the MTIE and TIE data measurements, which were provided by the test equipment, a final implementation was proposed. The proposed implementation's performance is presented along with the performance of the reference system in sections 4.3.1 and 4.3.2.

4

Results

This section presents the results of using the proposed clock recovery algorithm in an existing FPGA design. These results are organized in two sections.

The first section presents the simulation results. These simulation results are presented in the form of plots, which were derived using the developed simulation tool. The second section presents the implementation results, that is to say the results of running the system through tests which are described in the ITU-T G.8261 specification [6].

4.1 Preliminary simulations

In order to confirm that both the mathematical representation and the simulation model are correct, a series of preliminary simulations were run. These preliminary simulations consider an initial phase offset, along with a frequency offset, between the transmitted clock signal and the generated clock signal. Consequently, once these preliminary simulations were examined, the same models were used to derive simulations where a packet delay variation (PDV) is present in the simulated system.

To begin with, we examine the error signal, which expresses the phase difference between the all-digital phase locked loop (ADPLL)'s input and output. This phase difference expresses the lack of synchronization between the clock derived from a series of received packets and the locally generated clock, that is the recovered clock.

These preliminary simulations didn't use a specific model for the input of the phase locked loop (PLL), for example, one that would correspond to the packet traffic that was emulated during implementation testing. At that stage, the simulations considered a uniform distribution of the random phase difference between a packet's arrival time and the expected arrival time.

As is described in section 3.2, the design uses a filter whose performance is changing by defining different gains as time progresses. The simulation results' metrics are presented in different colors. Each color corresponds to a different gain level of the proportional-integral (PI) controller that acts as the first candidate of a low pass filter (LPF) solution. The ω_n quantity in the figure legends expresses the loop's

bandwidth, while k_p and k_i are the respective proportional and integral gains. The first metric to check was the error signal which is the input of the filter and is presented in Figure 4.1. The second metric to examine was the timing error which is presented in 4.2.

Assuming that all the packets in the receiver end arrive in time but there is a frequency offset in relation to the nominal frequency of the line interface unit (LIU)[1], we get the simulation results presented in Figures 4.3 and 4.4. Both of these Figures showcase that the loop is able to change its output so that the phase difference between its input and output is diminished. More specifically, Figure 4.3 shows that the frequency output of the ADPLL changes to match the value of the relative frequency offset, expressed in ppm. Similarly, the plot in Figure 4.3 showcases the same behavior and the loop eventually matches the value of the absolute frequency offset, expressed in Hz.

The next simulations were run to check whether the loop can handle noise that is attributed to the expected PDV which is present in an implemented system. If we consider that the packets arrive at random times relative to the expected one and that the packets' delays follow a standard deviation scheme then, considering a phase offset of 30% and frequency offset of 10ppm, we get the results presented in Figures 4.5 and 4.6. The comparison of these figures proves the ability of the simulated ADPLL to change its output to match its input. The noise present in the filter input shown in Figure 4.5 is gradually filtered out by the loop, as is demonstrated in Figure 4.6.

Moreover, we can examine the timing error by examining Figure 4.7. It is concluded that the filter allows the ADPLL to compensate for variation in the arrival time of packets, as well as the presence of phase offset and frequency offset.

[1] A LIU is a hardware unit that implements an E1 transmission link.

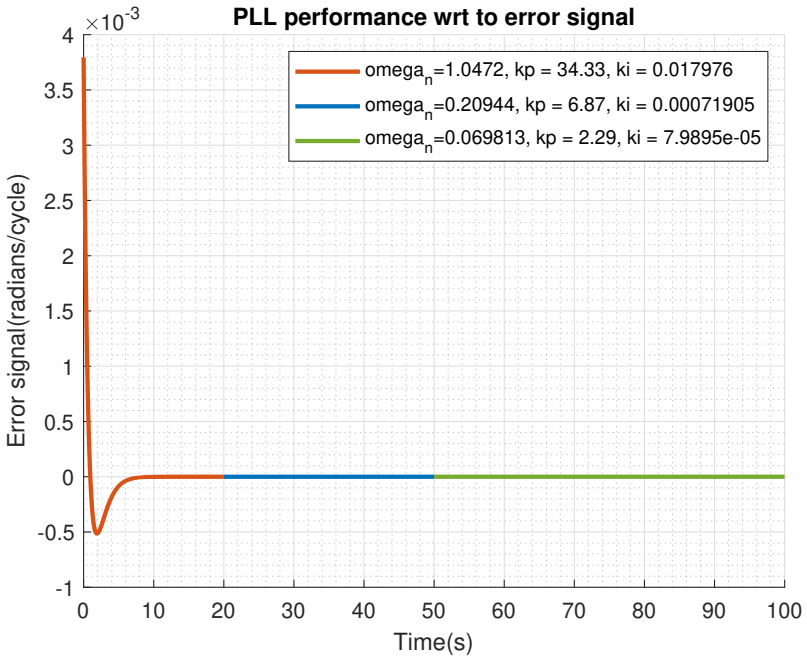

Figure 4.1: Error signal for a simulated system with a 30% phase offset. The error value is high when the system starts and converges to zero, meaning that the loop eventually locks to the desired frequency.

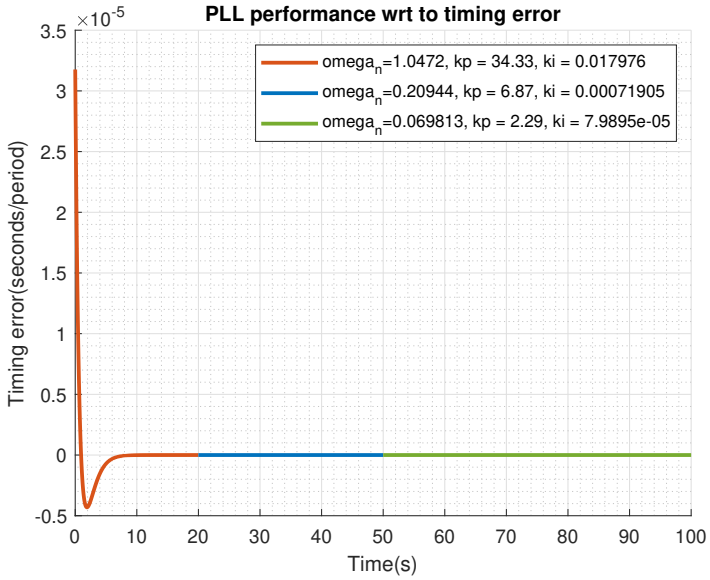

Figure 4.2: Timing error signal converging to a value close to zero, following the trend of the error signal.

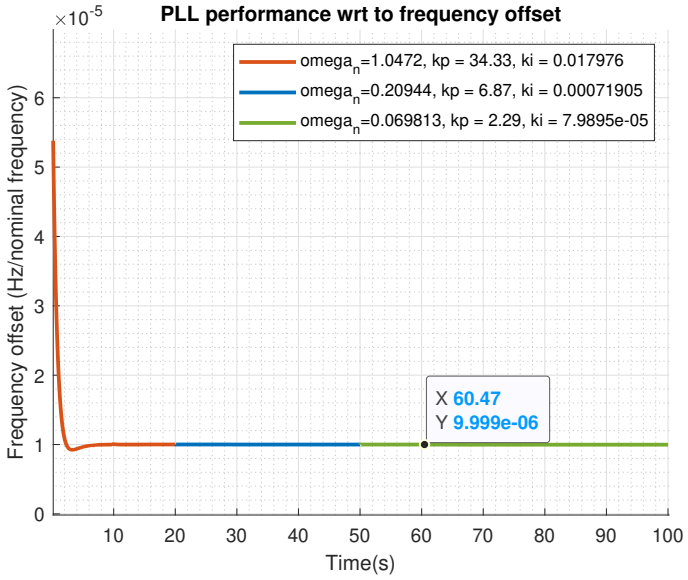

Figure 4.3: Simulated frequency offset in a system where a frequency offset of 0.00001, or 10ppm, in relation to the LIU's nominal frequency of 2048 MHz

.

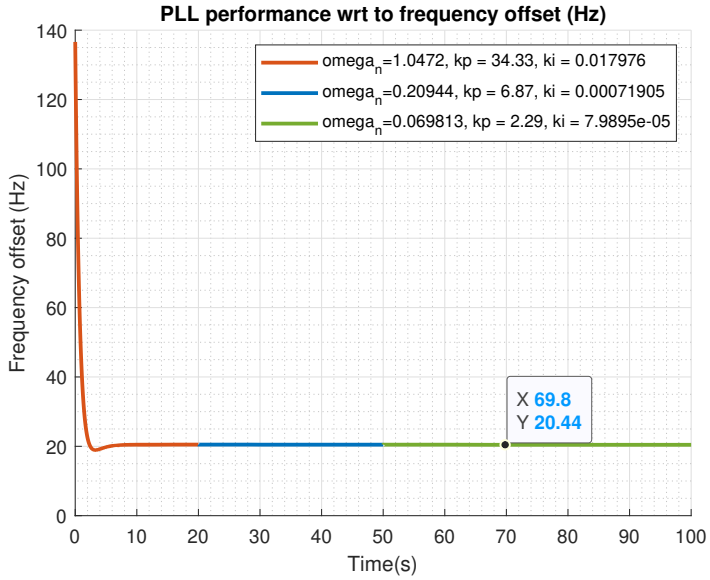

Figure 4.4: The simulation result of Figure 4.3 expressed in Hz. The frequency offset value eventually converges to the specified value of 10 ppm of 2048 MHz, which is approximately equal to 20 Hz.

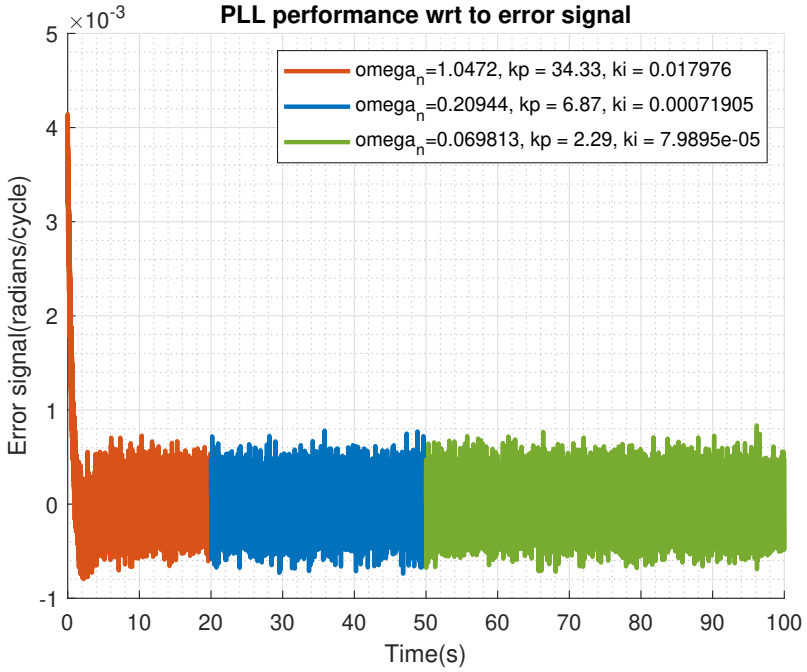

Figure 4.5: The input of the TDC for a system affected by a PDV and a frequency offset. It is the observed that initial error is compensated for by the ADPLL's operation. It is also observed that the output of the TDC is quite noisy, thus indicating the necessity of a low pass filter before using the signal as an input to the DCO.

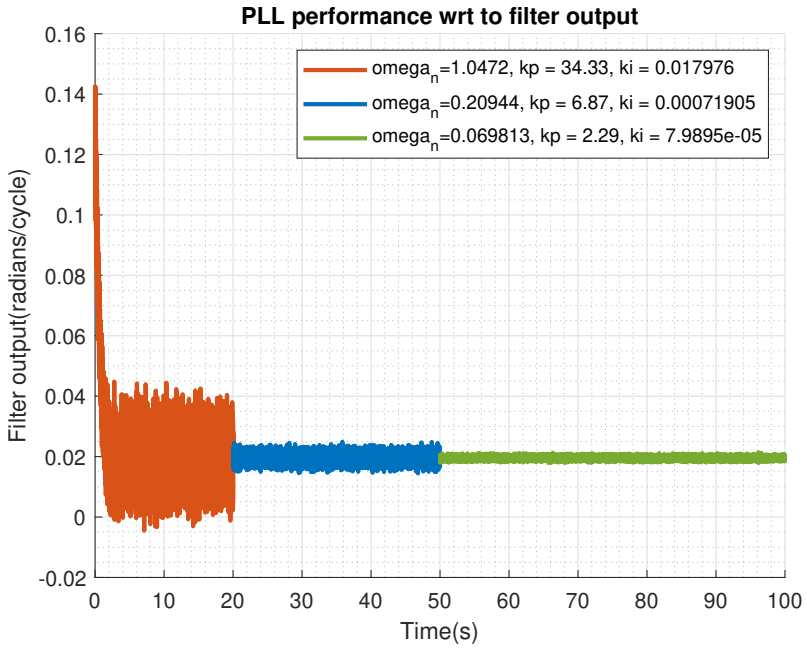

Figure 4.6: The plot shows the signal of Figure 4.5, after the signal has been processed by the suggested filter. It is observed that the ADPLL's is gradually zooming in and eventually locks in the desired phase.

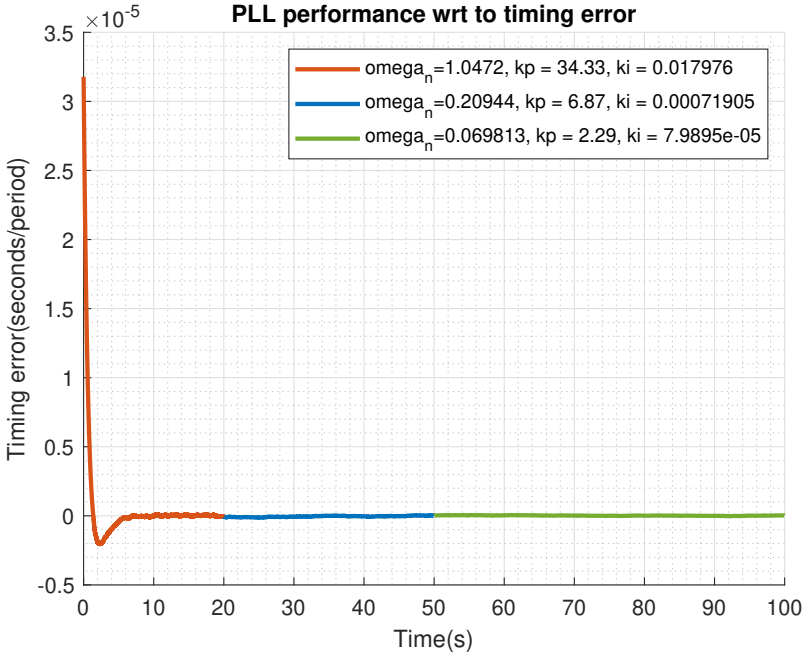

Figure 4.7: Timing error value converging to zero as time progresses.

4.2 Simulation of different filters

After the preliminary simulations, which were used as a means of diagnosing any possible problems with the tool by comparing the generated results with the expected results, the project proceeded with the simulation of the performance of different filter algorithm alternatives. In contrast to the simulation results presented in the previous section, these simulations considered a model of the incoming packet traffic that corresponds to the packet traffic defined by the implementation test cases.

4.2.1 PI controller filter simulations

A series of simulations was conducted anew, using PI controller filtering solution, to confirm that the ADPLL's performance falls within specification even when considering a specific model for the loop's stimuli. In Figures 4.8 and 4.9 the filter's input and output are shown, respectively. The reference filter's performance in terms of MTIE is shown in Figure 4.10.

4.2.2 FIR filter simulations

The series of plots that were presented in section 4.2.1 were repeated for the case of the FIR filter. When compared, Figures 4.11 and 4.12 showcase that this filter's output signal is actually greater in magnitude than the filter's input. Based on conclusions drawn from Figure 4.10 the filter would provide worse performance than the PI controller filter but would still comply with specification.

4.2.3 IIR filter simulations

In contrast to both the case of the PI controller and the case of the FIR filter, in which the filters' parameters were derived analytically or decided arbitrarily, the α parameter of the suggested IIR needed to be decided by predicting the system's performance for each different value α. Thus, a series of simulations that would predict the performance of the suggested IIR filter were conducted. By using Figures 4.14, 4.15 and 4.16 to compare the performance of the filter for each different value of α, it was concluded that very small values of α cause the filter's performance to fall out of specification.

For reasons of completion, the filter's performance was simulated for a small value of α to produce plots which are similar to the plots presented for the filtering solutions that were presented in the previous sections. These plots, corresponding to an α value of 0.038, are presented in Figures 4.17, 4.18 and 4.19.

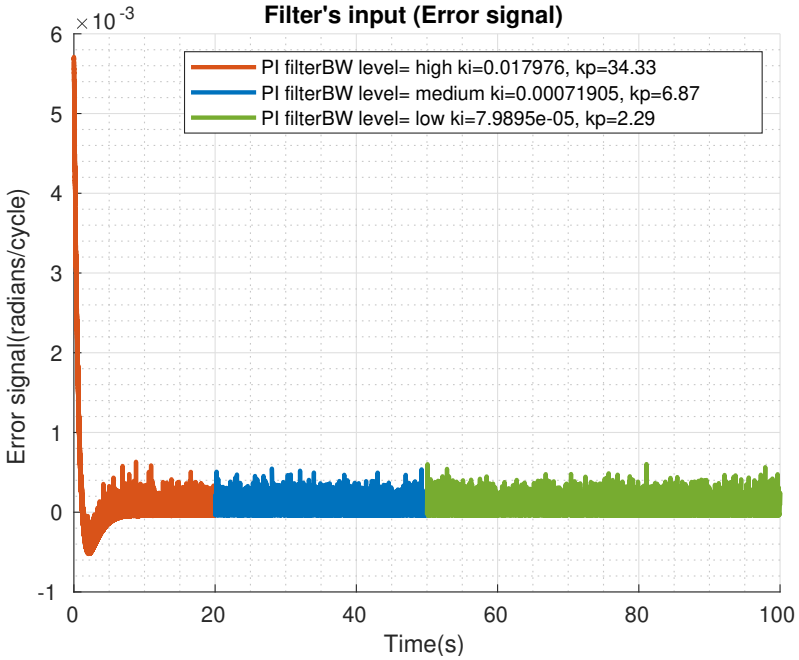

Figure 4.8: Simulated filter input of PI filter.

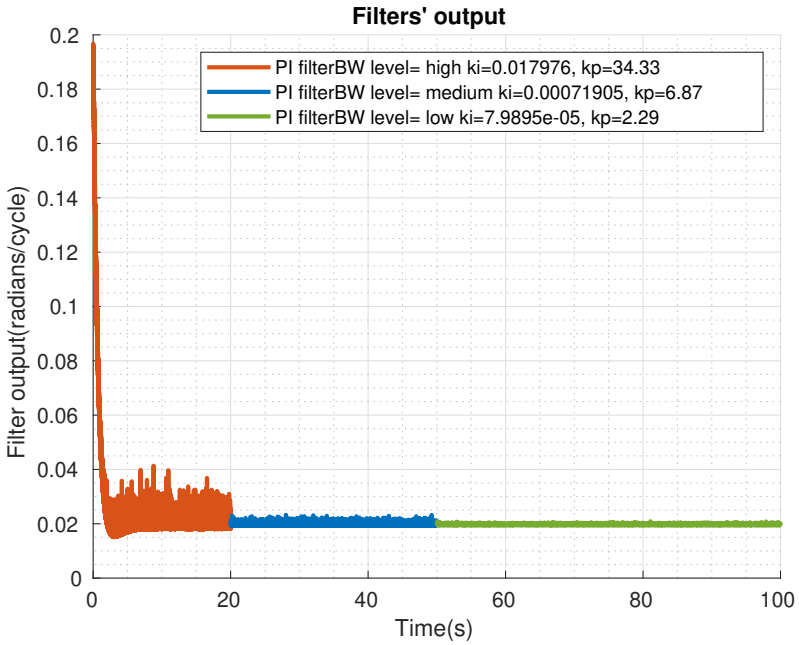

Figure 4.9: Simulated filter output of PI filter. Same as with preliminary simulations, the ADPLL is predicted to lock in the desired frequency as time progresses.

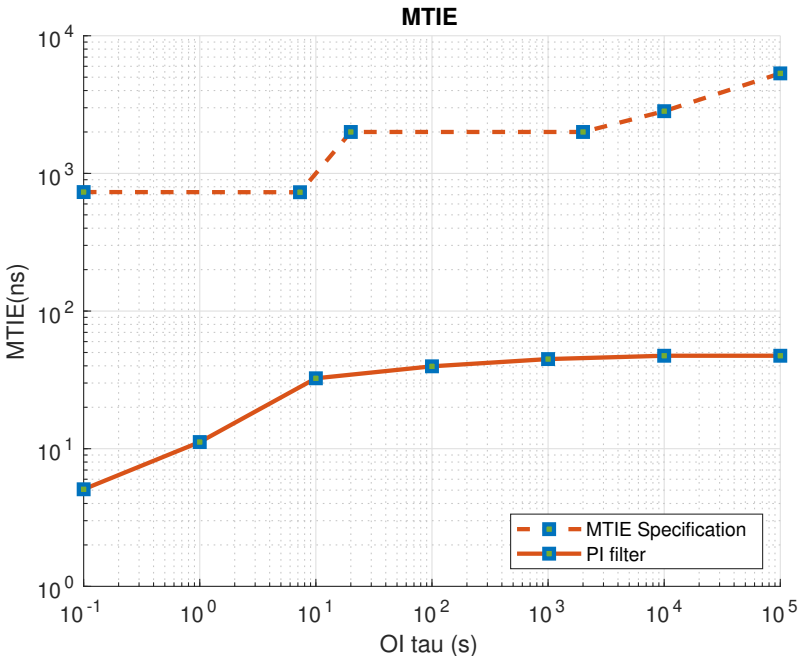

Figure 4.10: Simulated MTIE of PI controller showing that the system complies with specification.

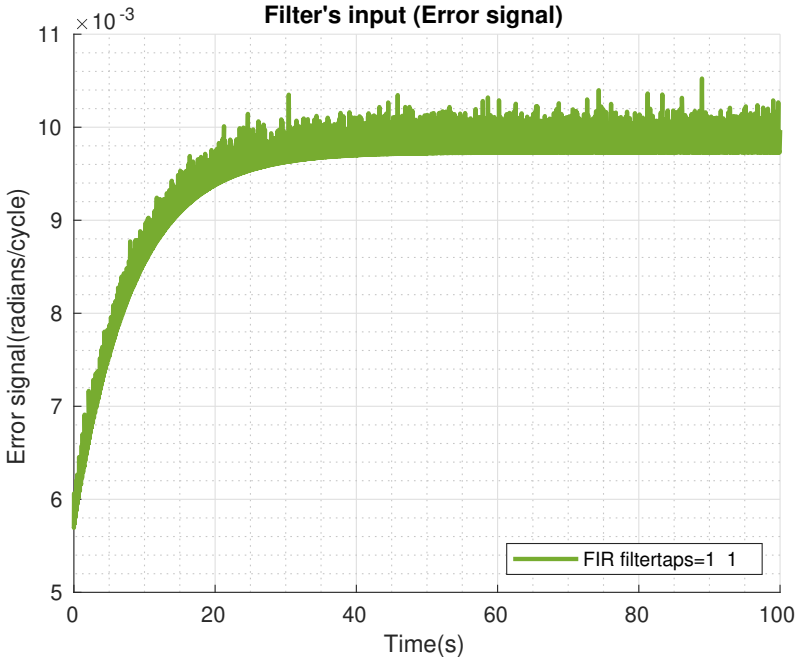

Figure 4.11: Simulated filter input of FIR filter at which an increasing trend for the filter input. This comes in contrast with the case of the PI controller in which the input followed a decreasing trend.

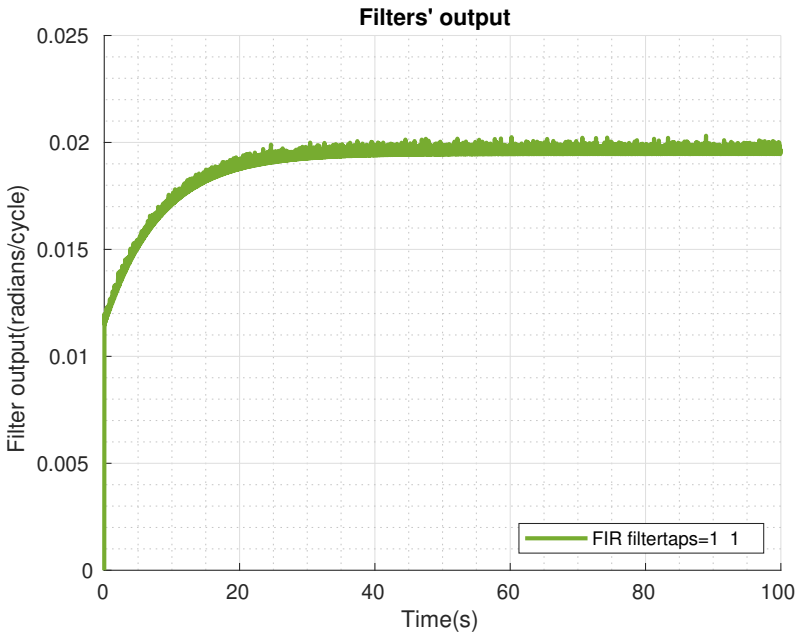

Figure 4.12: Simulated filter output of FIR filter. The filter's output follows the trend of the input and eventually settles to a specific value.

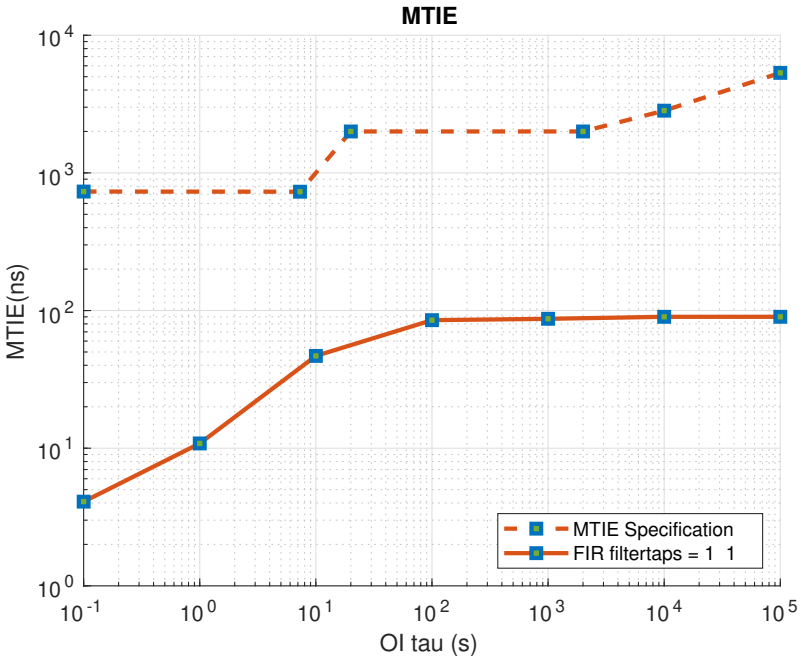

Figure 4.13: Simulated MTIE of FIR filter. The predicted performance of the filter complies with specification.

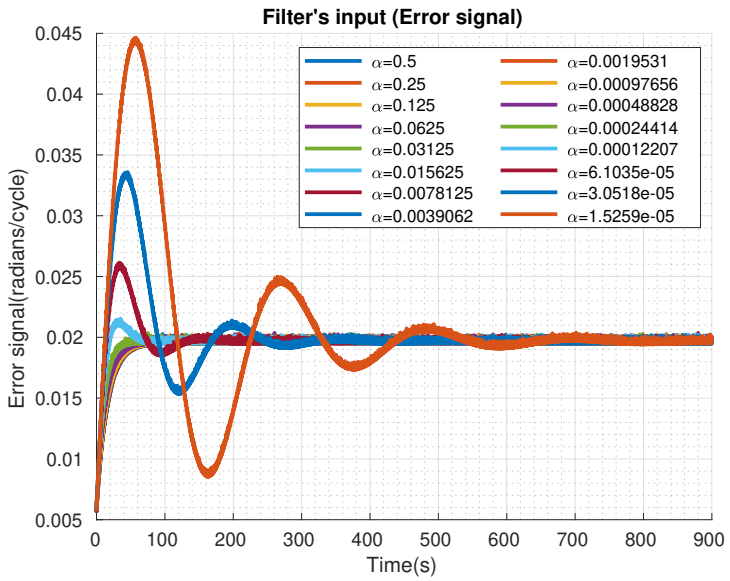

Figure 4.14: Simulated filter input of IIR filter for different values of the α parameter ranging from 2^{-1} to 2^{-16}.

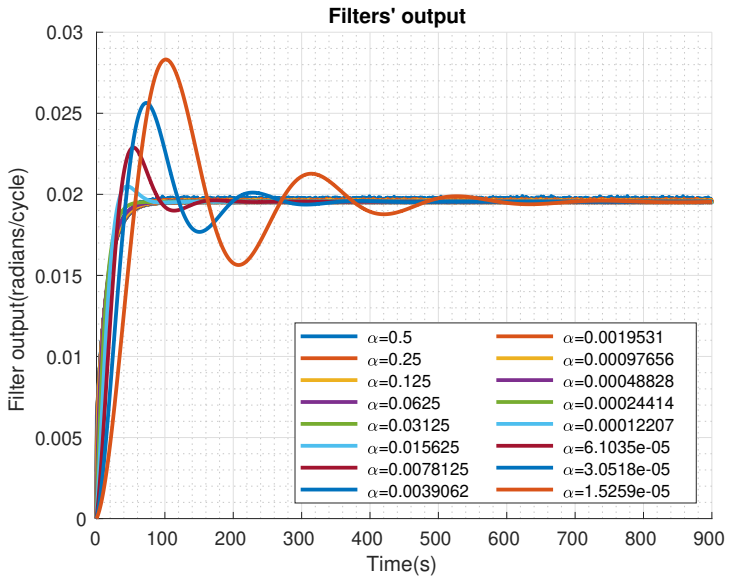

Figure 4.15: Simulated filter output of IIR filter for different values of the α parameter. The values of α range from 2^{-1} to 2^{-16} and they derive from arithmetic right shift of the original binary value of α. It is observed that the smallest values of α give the highest overshoot and settling time values.

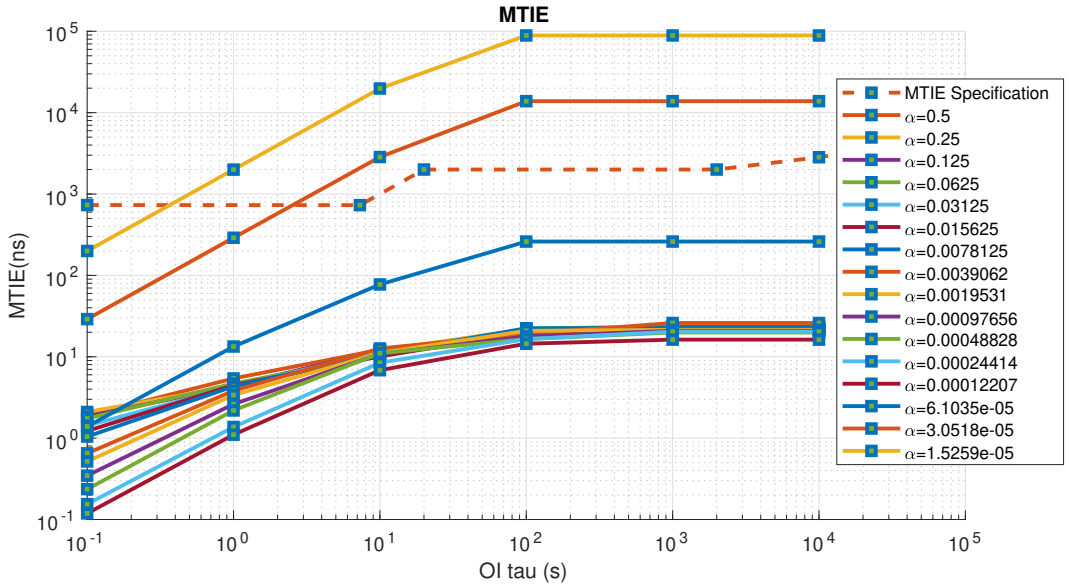

Figure 4.16: Simulated MTIE of IIR filter for different values of the α parameter ranging from 2^{-1} to 2^{-16}. As was expected from the results shown in Figure 4.15, the smallest values of α, specifically $\alpha = 2^{-15}$ and $\alpha = 2^{-16}$, provide performance that doesn't comply with the filter's specification.

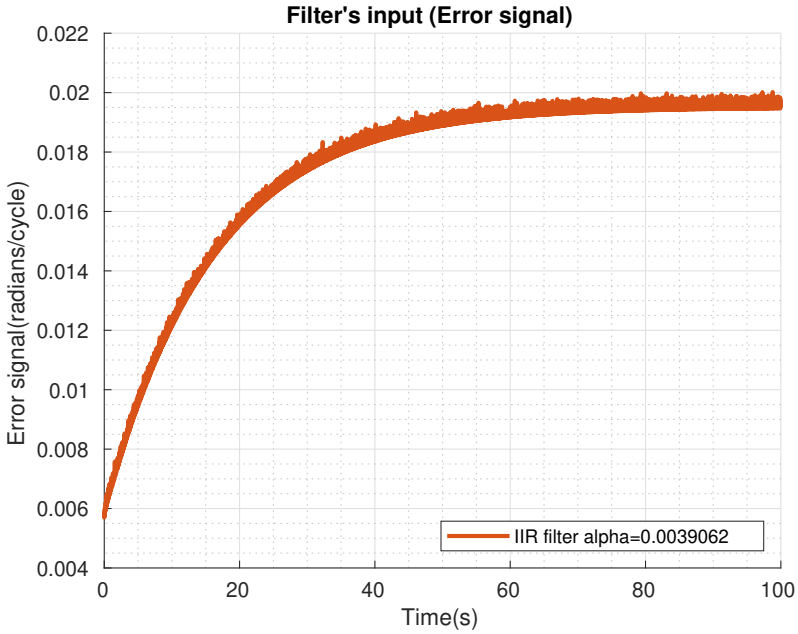

Figure 4.17: Simulated filter input of IIR filter. Same as the FIR filter, the IIR filter input follows an increasing trend before settling around a specific value.

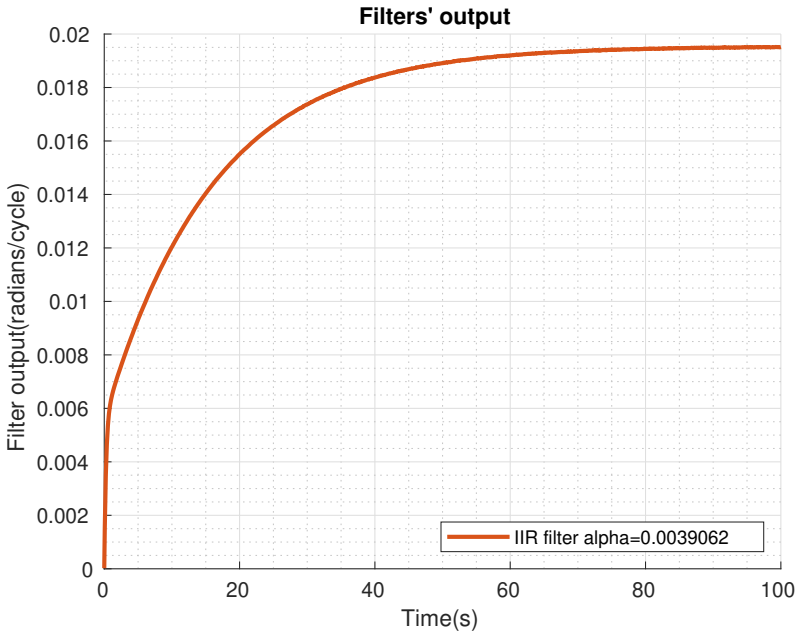

Figure 4.18: Simulated filter output of IIR filter. It is observed that the filter's output magnitude is not increased compared to the input and that the noise content of the output is reduced.

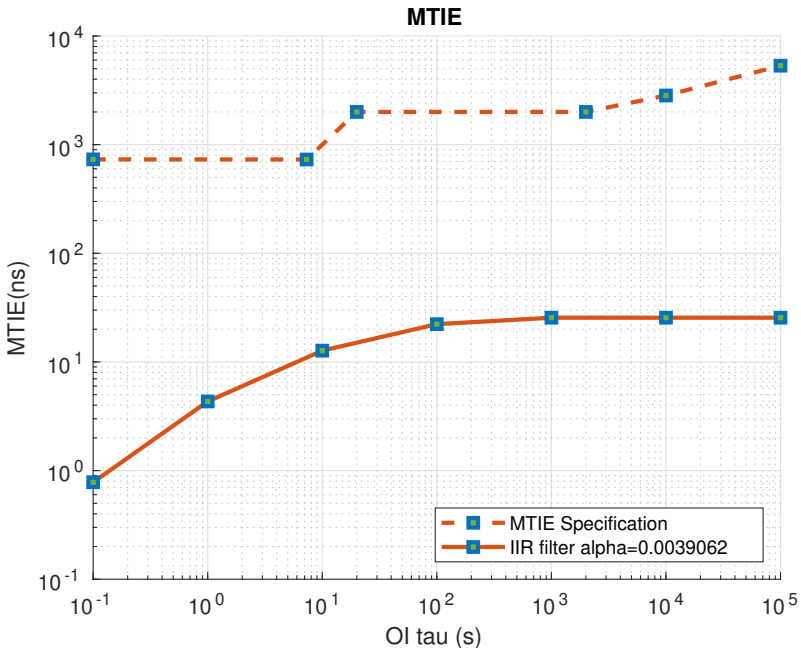

Figure 4.19: Simulated MTIE of IIR filter for $\alpha = 0.0039 = 2^{-8}$. The predicted performance falls within specification.

4.2.4 Comparison of different filters using simulation

In order to determine which algorithm was going to be implemented in FIR, it was necessary to compare the simulation results in a combined plot. The first metric that was used to compare performance was the MTIE. A combined MTIE plot for all the different filtering solutions is presented in Figure 4.20. Another metric to compare different filters was the comparison of the filters' output presented in Figure 4.21, which was used to compare the different filters' in terms of their overshoot and their settling time.

Although the FIR filter should provide better dynamic performance, its MTIE performance was much worse than that of the existing solution. On the other hand, the IIR's predicted MTIE is better than that of the existing solution but suffers from a relatively high settling time which is comparable but worse that the PI controller solution. For reasons of flexibility after synthesis and in order to achieve better performance than the current one, the FIR option was dismissed and a combined filter consisting of a PI controller in series with an IIR filter was chosen to be implemented. The simulated performance of the proposed filter is also demonstrated in Figures 4.21 and 4.20.

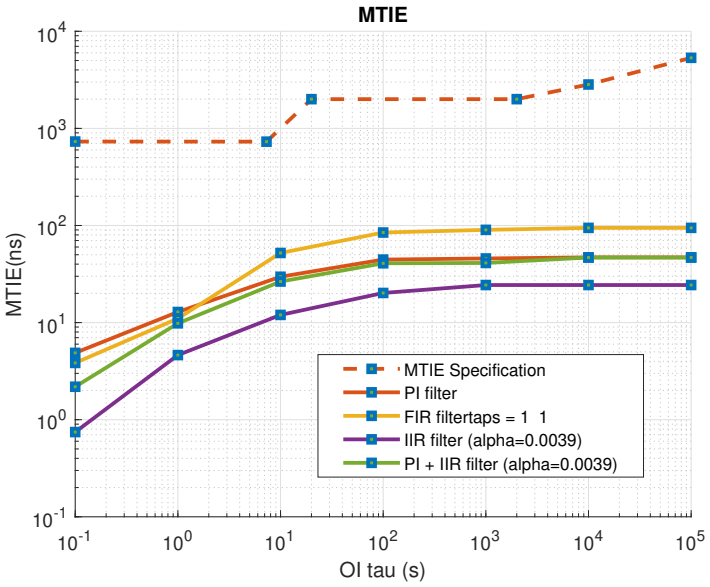

Figure 4.20: Combined simulation of different filters' MTIE performance. The FIR's performance was approximately 50% worse than that of the existing solution. On the other hand, the IIR's predicted MTIE is approximately 30% better than that of the reference (PI controller) filter. The proposed combined filter is predicted to give comparable or better MTIE performance when compared to the existing solution.

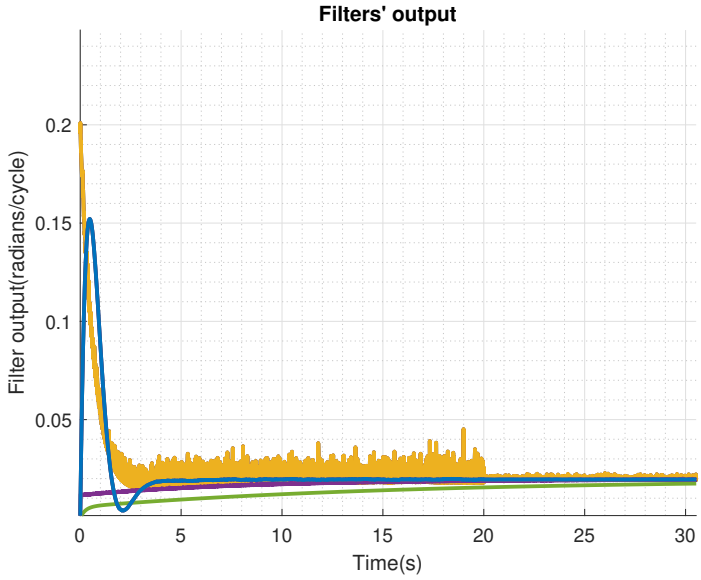

Figure 4.21: Combined simulation of different filters' output. The yellow plot corresponds the PI filter, while the purple, green and blue plots correspond to the FIR, the IIR and the combined filter, respectively. The FIR filter's simulated performance gives low overshoot and the lowest settling time, while the IIR filter gives the lowest overshoot but performs almost identically to the PI filter as far as settling time is concerned. The combined filter shows an approximately 80% smaller settling time and a 25% smaller overshoot than the existing solution.

4.3 Implementation results

This section presents a series of implementation results which were derived by running the first two out of a total of ten test cases, which are described in the ITU-T G.8261 standard.

The results are presented in two subsections, which correspond to the two test cases mentioned above, and compare the difference in performance between the reference and the proposed solution. The comparison is done with the use of plots. The data presented in these plots were derived from measurements of the TIE and MTIE quantities which were taken by the test equipment.

4.3.1 Performance results for test case 1

By running the first test case and consequently comparing the proposed system against the reference system, we confirm that the implementation results agree with the simulation results. Thus, the performance of the proposed system both achieves the required performance and performs marginally better than the reference system. The two systems were compared with regards to their MTIE using the results presented in Figures 4.22 and 4.23. Similarly, the comparison in terms of TIE is done by examining Figures 4.24 and 4.25.

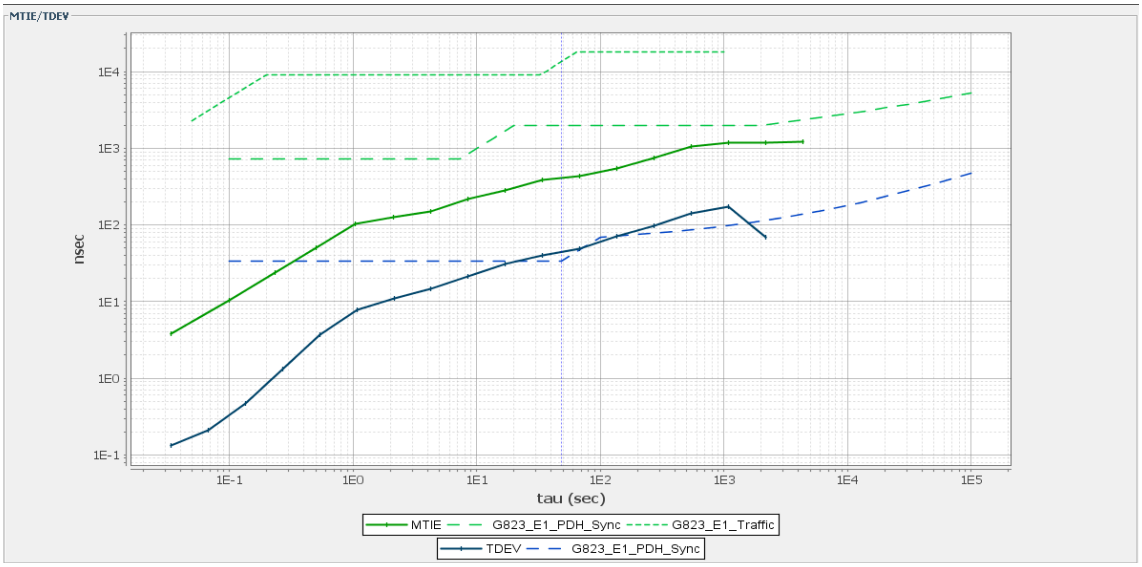

Figure 4.22: Reference system performance for test case 1. The dashed green and blue lines represent the specification on the maximum allowed values for MTIE and TDEV, respectively. The upper dashed green line is the the MTIE requirement that needs to be taken into account for successful the data synchronization. The lower dashed line is a stricter requirement that needs to be satisfied in order to use the recovered clock signal as a system clock. The stricter requirement is useful as a reference but needs not be satisfied for the implementation of a CES. Lastly, the solid green and blue lines present the measured values for MTIE and TDEV with respect to the size of the OI. The TDEV performance can be useful as a comparison metric but its requirement needs not be fulfilled.

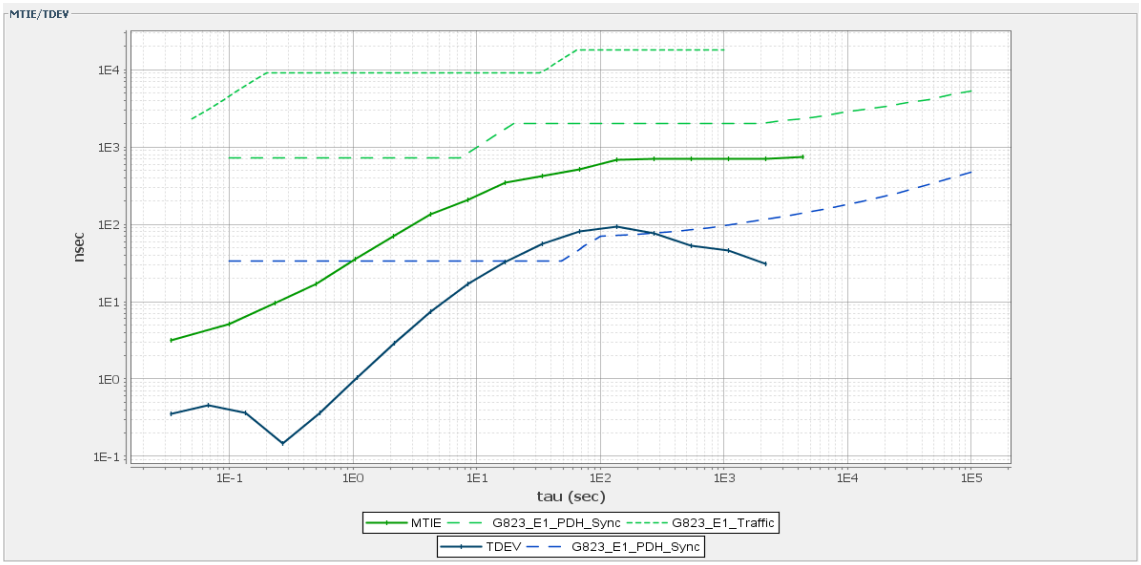

Figure 4.23: Proposed system's performance for test case 1. It is observed that the proposed system falls within the required specification and achieves marginally better MTIE performance when compared to the reference system.

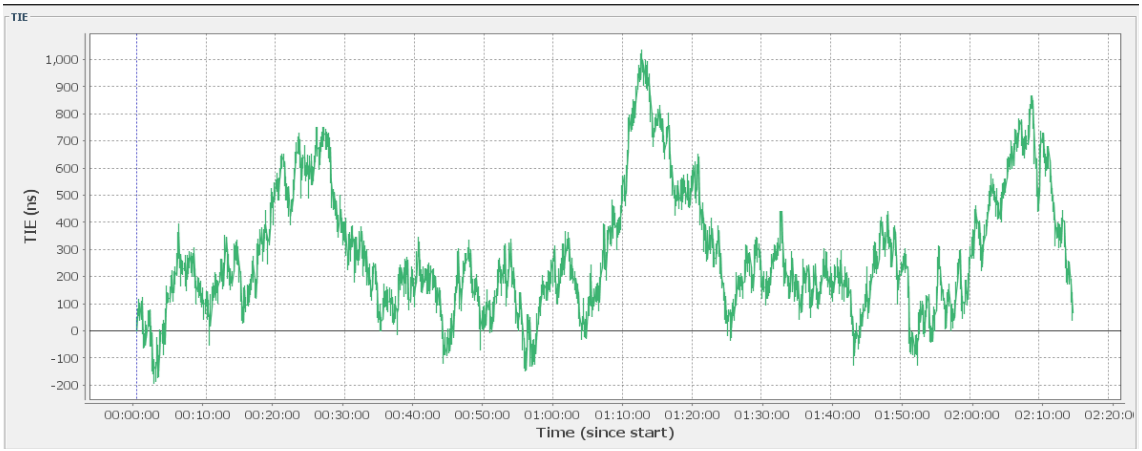

Figure 4.24: Reference system performance for test case 1 in terms of TIE. It is observed that the majority of the measured values are positive with the absolute error reaching the value of 1000 n sec.

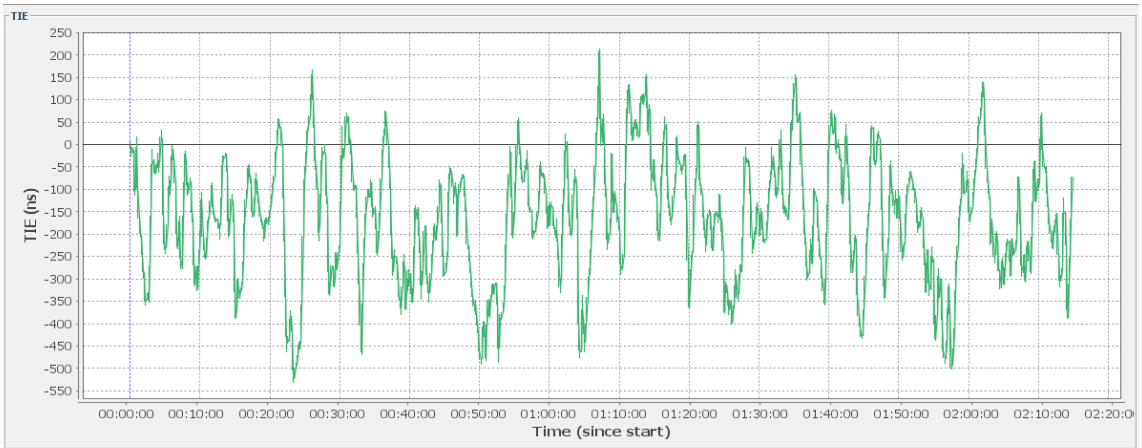

Figure 4.25: Proposed system performance for test case 1 with regards to the TIE. It is observed that the majority of the measured values are negative with absolute error reaching a maximum value of approximately 750 n sec.

4.3.2 Performance results for test case 2

Same as the previous test case, the proposed system agrees with specification and performs better than the reference system. In this second test case, a much more substantial performance improvement is achieved. The respective results for MTIE are presented in Figures 4.26 and 4.27. Figures 4.28 and 4.29 were used to compare the TIE measurements of the two systems.

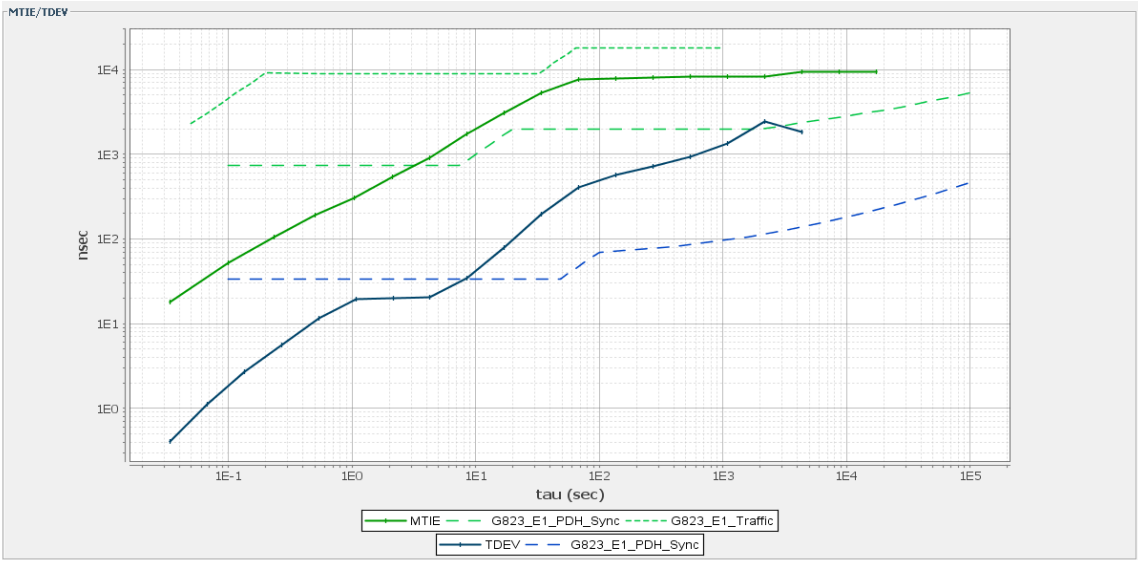

Figure 4.26: Reference system performance for test case 2. It is observed that the system's measured MTIE performance is very close to the defined specification mask, which is denoted by the upper green dashed line.

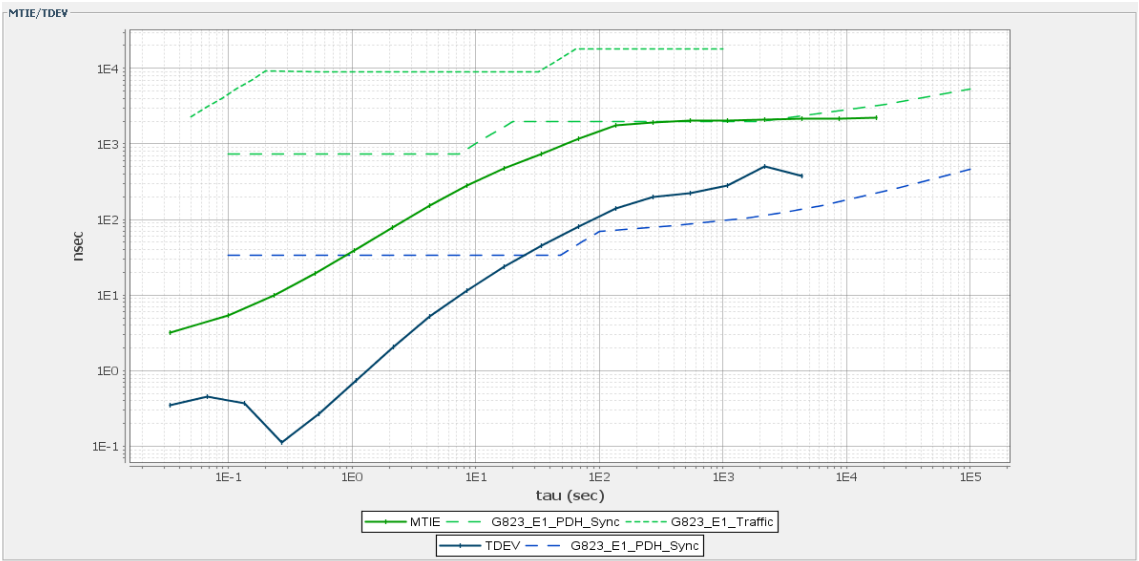

Figure 4.27: Proposed system performance for test case 2. It is observed that the measured performance is substantially better than the reference system and that the stricter MTIE specification is marginally not satisfied.

Figure 4.28: Reference system performance for test case 2. The timing error in this second test case is almost 8 times higher than the one observed in test case 1. This test case is also indicative of the overshoot performance of the reference system which is quite poor.

Figure 4.29: Proposed system performance for test case 2. As was expected from the observations made using the MTIE plots, the system's TIE performance is substantially improved. The proposed solution provides lower values for the TIE and better overshoot performance.

4.3.3 Interpretation of implementation results

The proposed system design is considered successful for two reasons. Firstly, the proposed filtering solution performs within specification. Secondly, the implemented filter achieves better performance than the previously implemented filter design. When the first test case is considered, the performance improvement is only marginal but when the second test case is examined, the system showcases much better performance.

Although the predicted performance given by the simulation results did not coincide exactly with the measurements derived by the testing of the implemented filter, the predicted performance during simulation provided an adequate estimation of the relative performance improvement over the existing filter design.

Moreover, the implementation results showcase how the system's performance is not only dependent on the filter that is utilised by the ADPLL. System performance can also be affected by network traffic. This observation emphasises the importance of modeling the ADPLL's stimuli when designing a digital filter that supports a clock recovery function.

5

Potential Improvements

Given that the proposed solution was derived by examining different settings for the filter's parameter and the implemented PSA, it is suggested that further testing should be conducted to derive even better performance.

As far as simulation is concerned, the first suggested improvements would be the inclusion of a PSA in the simulation tool. This inclusion would improve the accuracy of the simulation model and provide better simulation results, closer to real system's performance. A second improvement would be to enhance the simulation algorithm with a better statistical model for the ADPLL stimuli, that is, a better model of the PDV and the system's noise.

With regards to the filter design process, an alternative strategy than the one followed would be the "translation" of the MTIE specification into a frequency response specification. A starting point for this alternate methodology would be to arbitrarily define a cut-off frequency, choose a filter type and derive its parameters using a CAE tool and then iteratively change the filters parameters so that the system's performance falls within specification. Once this kind of specification is derived, alternative and more sophisticated filter designs can be examined to achieve that specification.

6

Conclusion

The filter design process for clock recovery, which is used in packet based communication systems, includes the additional challenge of defining a desired frequency response. This is because the performance requirement that these systems need to meet is given in terms of maximum time interval error (MTIE).

Since circuit emulation service (CES) systems need to comply with a specified MTIE requirement, the MTIE quantity is an essential concept and metric when it comes to designing a filter for the phase locked loop (PLL) that is utilized for the clock recovery function.

One of the main issues when designing a clock recovery solution is dealing with the packet delay variation (PDV) that is associated with the use of network elements (NEs) in packet networks. At simulation level, predicting and mitigating the effects of PDV requires the modelling of the PLL's stimuli to generate more accurate simulation results which can be used to predict a design's performance. At implementation level, the effects of PDV can be mitigated by the use of a packet selection algorithm (PSA). Given that the presence of PSA affects the PLL's performance, the filter design needs to be customizable after synthesis to compensate for any discrepancy between the measured and the expected performance.

By taking into account how does a PSA affect the system's input and by altering a filter's parameters accordingly, it was concluded that,in a system that utilizes a PSA, a filtering solution that combines a proportional-integral (PI) controller with an added infinite impulse response (IIR) filter can comply with specification.

Bibliography

[1] A. Gyasi-Agyei, *Multiplexing and Multiple Access Methods*. McGraw-Hill Education, 5 2020. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789811201332_0008

[2] M. L. Sanderson, "Telemetry," in *Instrumentation Reference Book*, 4th ed., W. Boyes, Ed. Boston: Butterworth-Heinemann, 2010, ch. 40.3, pp. 677–697. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780750683081000401

[3] Cisco Systems, "An Introduction to Circuit Emulation Services - Cisco," 2019. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/asynchronous-transfer-mode-atm/circuit-emulation-services-ces/10423-ces.html

[4] T. B. Mills, "Phase-locked loops," *Access Science*, 2020. [Online]. Available: https://www.accessscience.com/content/505610

[5] ITU-T, "G.823 : The control of jitter and wander within digital networks which are based on the 2048 kbit/s hierarchy," ITU-T, Tech. Rep., 2000. [Online]. Available: https://www.itu.int/rec/T-REC-G.823-200003-I/en

[6] ITU, "G.8261 Timing and synchronization aspects in packet networks," ITU-T, Tech. Rep., 2013. [Online]. Available: https://www.itu.int/rec/T-REC-G.8261

[7] D. T. Bui *et al.*, "Packet delay variation management: For a better IEEE1588V2 performance," in *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, ISPCS '09 - Proceedings*, 2009, pp. 75–80.

[8] Xilinx Inc., "Field Programmable Gate Array (FPGA): What is an FPGA?" pp. 1–2, 2017. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html

[9] Xilinx, "7 Series FPGAs Data Sheet: Overview (DS180)," pp. 1–18, 2010. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds175-xa-7k160t-overview.pdf

[10] I. Hadžić and D. R. Morgan, "On packet selection criteria for clock recovery,"

in *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, ISPCS '09 - Proceedings*, 2009, pp. 35–40. [Online]. Available: https://www.infona.pl//resource/bwmeta1.element.ieee-art-000005340211

[11] I. Hadzić and D. R. Morgan, "Adaptive packet selection for clock recovery," in *ISPCS 2010 - 2010 International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication, Proceedings*, 2010, pp. 42–47.

[12] Z. Chaloupka *et al.*, "Packet selection technique for clock recovery over packet networks," in *ISPCS 2014 - Proceedings: 2014 International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. Institute of Electrical and Electronics Engineers Inc., 11 2014, pp. 108–111.

[13] T. Murakami and Y. Horiuchi, "Improvement of synchronization accuracy in IEEE 1588 using a queuing estimation method," in *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, ISPCS '09 - Proceedings*, 2009, pp. 12–16.

[14] National Instruments, "Instrument Fundamentals: Digital Timing," National Instruments, Tech. Rep., 2019. [Online]. Available: http://www.ni.com/gate/gb/GB_INFOINSTFUNDGUIDE/US

[15] ITU-T, "G810 : Definitions and terminology for synchronization networks," ITU-T, Tech. Rep., 1996. [Online]. Available: https://www.itu.int/rec/T-REC-G.810/en

[16] D. Gisselquist, "Two of the Simplest Digital filters," 2017. [Online]. Available: https://zipcpu.com/dsp/2017/08/19/simple-filter.html

[17] G. D. Baura, "System Theory and Frequency-Selective Filters," in *System Theory and Practical Applications of Biomedical Signals*. IEEE, 5 2015.

[18] D. Gisselquist, "Building a high speed Finite Impulse Response (FIR) Digital Filter." [Online]. Available: http://zipcpu.com/dsp/2017/09/15/fastfir.html

[19] R. C. Dorf and G. O. Beale, "Control systems," *Access Science*, 2014.

[20] National Instruments, "Linear Phase Filters (Digital Filter Design Toolkit)," National Instruments, Tech. Rep., 2011. [Online]. Available: https://zone.ni.com/reference/en-XX/help/371325F-01/lvdfdtconcepts/linear_min_filters/

[21] L. Tan and J. Jiang, *Digital signal processing: Fundamentals and applications*. Academic Press, 2018.

[22] A. V. Oppenheim *et al.*, *Discrete-time signal processing*, 2nd ed. Prentice Hall, 1999.

[23] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 4th ed. Springer, 2014. [Online]. Available: http://www.amazon.com/Digital-Processing-Programmable-Communication-Technology/dp/3540726128

[24] S. A. White, "Digital filter," *Access Science*, 2014.

[25] G. W. Roberts and M. Ali-Bakhshian, "A brief introduction to time-to-digital and digital-to-time converters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 3, pp. 153–157, 3 2010.

[26] Y. B. Kim and J. Zhao, "A low-power digitally controlled oscillator for all digital phase-locked loops," *VLSI Design*, vol. 2010, p. 946710, 2010. [Online]. Available: https://doi.org/10.1155/2010/946710

[27] J. J. Shyu and Y. C. Lin, "A New Approach to the Design of Discrete Coefficient FIR Digital Filters," *IEEE Transactions on Signal Processing*, vol. 43, no. 1, pp. 310–314, 1995.

[28] A. Chandra and S. Chattopadhyay, "Design of hardware efficient FIR filter: A review of the state-of-the-art approaches," *Engineering Science and Technology, an International Journal*, vol. 19, no. 1, pp. 212–226, 3 2016.

[29] B. Jiang *et al.*, "PLL low pass filter design considering unified specification constraints," *Analog Integrated Circuits and Signal Processing*, vol. 80, no. 1, pp. 113–120, 4 2014.

[30] C. Barrett, "Fractional/integer-N PLL basics," Texas Instruments, Tech. Rep. August 1999, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.9566

[31] M. Sayadi and E. Farshidi, "A fast locked and low phase noise all-digital phase locked loop based on model predictive control," *Analog Integrated Circuits and Signal Processing*, vol. 88, no. 3, pp. 401–414, 2016.

[32] D. Schlichthärle, *Digital Filters*, 2nd ed. Springer, 2011. [Online]. Available: www.springer.com

[33] P. Pichlík, "Comparison of different Kalman filters types performance for a locomotive slip control purposes," *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1–4, 2017. [Online]. Available: http://radio.feld.cvut.cz/conf/poster/proceedings/Poster_2017/Section_PE/PE_005_Pichlik.pdf

[34] M. Dozza, "Frequency Domain & Digital Filters - Presentation, TME192 Active Safety course, Chalmers University of Technology, Gothenburg, 09 Sep 2019, p 63," Gothenburg, 2019.

[35] C. Wang *et al.*, "Real-time FPGA-based Kalman filter for constant and non-constant velocity periodic error correction," *Precision Engineering*, vol. 48, pp.

133–143, 4 2017.

[36] T. John, “Pocket Guide to The World of E1,” Plytmouth. [Online]. Available: https://web.fe.up.pt/~mleitao/STEL/Tecnico/E1_ACTERNA.pdf