Implementation of ADPLL Networks on FPGAs

Conor Dooley

This thesis is submitted to the School of Electrical and Electronic Engineering

in the College of Engineering and Architecture of University College Dublin

in partial fulfilment for the requirements for the degree of

Master of Engineering

Research Supervisors: Dr Elena Blokhina & Brian Mulkeen

Head of School: Prof Andrew Keane

April 2019

Contents

List of Figures

List of Tables

Acronyms

Acronyms

ADPLL All-Digital Phase Lock Loop

ASIC Application Specific Integrated Circuit

C2C Cycle-to-Cycle

CPU Central Processing Unit

DCO Digitally Controlled Oscillator

EDA Electronic Design Automation

FIR Finite Impulse Response

FPGA Field Programmable Gate Array

FSM Finite State Machine

GALS Globally Asynchronous Locally Synchronous

GSLS Globally Synchronous Locally Synchronous

HDL Hardware Description Language

IC Integrated Circuit

IIR Infinite Impulse Response

IOT Internet Of Things

LF Loop Filter

MSB Most Significant Bit

NCO Numerically Controlled Oscillator

Nexys4 XC7A100T-1CSG324C

NMOS N-type Metal Oxide Semiconductor

PD Phase Detector

PFD Phase Frequency Detector

PI Proportional Integral

PLL Phase Lock Loop

PMOS P-type Metal Oxide Semiconductor

RADAR RAdio Detection And Ranging

RAM Random Access Memory

RF Radio Frequency

RO Ring Oscillator

RTL Register Transfer Level

SoC System-On-Chip

SR Set-Reset

TDC Time-to-Digital Converter

TDL Tapped Delay Line

TIE Time Interval Error

UCD University College Dublin

VCO Voltage Controlled Oscillator

XOR Exclusive Or

Abstract

Low power, high frequency clock distribution systems will be in ever increasing demand in the near future as the need for high performance digital circuitry grows. At these frequencies, however, the conventional clock distribution systems are unable provide a clock signal of adequate quality without compromising on either of these problems. Many devices have turned away from using Globally Synchronous clock distribution systems in favour of those that divide the area of the chip into Globally Asynchronous but Locally Synchronous areas. However, new technologies seek to enable the use of Globally Synchronous methods at high frequencies, one of which being the use of oscillators coupled in phase, each responsible for delivering the clock to a subregion of the chip. Each oscillator forms part of a Phase Lock Loop (PLL), and in order to enable the synchronisation of the network each PLL is linked those controlling the adjacent clock regions. For a digital system it is expedient to implement these PLLs digitally as an All-Digital Phase Lock Loop (ADPLL) as this provides a number of advantages. It has been shown in theory and experimentally that this method can produce the high quality clock desired with low power consumption.

As the production of test chips is expensive and time consuming through simulation and validation of a design is vital, traditionally carried out for mixed signal circuitry using complex behavioural and theoretical models. For an ADPLL Network the consistency of a signal both with respect to itself, and to the other clock signals on the chip is a key performance attribute and much of the variation is due to random processes that may be difficult to simulate effectively.

This thesis will demonstrate that the Field Programmable Gate Array (FPGA) can be used in order to simulate, model or validate ADPLL network architectures in a cost and time effective manner, as a complement to conventional methods. The key benefit is that many system dynamics that will be seen when a network is implemented on an Application Specific Integrated Circuit (ASIC), but be overlooked in software simulations can be examined in the hardware simulation that an FPGA

provides.

This thesis implements networks using three designs of ADPLL, each using a different architecture, and highlights the use cases to which each is best suited. The performance of each design is then analysed and this compared to the suggested use cases. Additionally the impact of more minor architectural modifications is tested and documented.

Lay Abstract

With the increasing proliferation of "smart" devices the need for low power yet high speed devices has never been greater. Each smart device contains a processor to control the device, each containing complex circuitry, requiring extremely exact synchronisation. The task of this synchronisation falls to the "clock" signal which must occur at the same instant in all areas of the processor. Conventional solutions to this problem cannot satisfy both power and speed requirements simultaneously, so designers have proposed the ADPLL Network which approaches the problem from the other side. Rather than generate the clock once and send it around the chip, which consumes a large amount of power, an ADPLL network divides the chip into a grid and generates many signals that each serve a local area, only requiring non time critical control signals to be sent over large distances.

The production of chips to test designs is both expensive and time consuming so designers must ensure that mistakes have not been made, accomplished by simulating the behaviour of the designs using complex behavioural and theoretical models. This thesis will discuss the use of Field Programmable Gate Arrays (FPGA) as a hardware testing, simulation and validation platform, to be used prior to test chip production, that is both cost and time effective. The hardware nature of an FPGA enables the analysis of behaviours that would not be possible in software, without the cost and time penalties of a custom chip. This is made possible by the ability to reconfigure the chip at will, albeit with comparatively lesser capabilities.

This thesis implements networks using three designs of ADPLL, each using a different architecture, and highlights the use cases to which each is best suited. The performance of each design is then analysed and this compared to the suggested use cases. Additionally the impact of more minor architectural modifications is tested and documented.

Chapter 1

Introduction

This thesis will put forward the Field Programmable Gate Array (FPGA) as a tool in the design of All-Digital Phase Lock Loop (ADPLL) networks to bridge the gap between software simulations and implementation in custom silicon by providing a hardware-based simulation, modelling and validation platform. While an FPGA lacks the direct control over the schematic and layout that an Application Specific Integrated Circuit (ASIC) provides, the hardware nature of this platform enables the analysis of system dynamics that are not easily modelled in simulation and the testing.

The goal of this project is to design and implement an extensible platform that can be used by the research team in University College Dublin (UCD) going forward as they seek to understand the behaviour of ADPLL networks at a higher level and to serve as a hardware test-bed for proposed new architectures or system components. In order to accomplish this goal a number of potential ADPLL architectures will be investigated, implemented and tested to ensure they are function correctly. Individual ADPLLs, however, will not give sufficient insight into the behaviour of a network, so once the ADPLL designs have been established, each will be implemented as part of a network of increasing sizes. Each contrasting design will be analysed based on the results of measurements and tests, and these results will be used to corroborate claims made regarding which FPGA based ADPLL design or ADPLL network architecture is best suited for particular use cases.

The design of each block, or component part, used in the network will be discussed, starting with the reasons for their selection and an explanation of the design methodology, along with any major pitfalls encountered their creation. The impact on performance caused by modifying the design of these blocks will again be assessed on the basis of measurement results, before comparison is made to both theoretical expectations and their use case.

An FPGA based test platform is ideal for those who wish to examine system dynamics without the time delays, financial burden or expertise required to develop a complete mixed-signal system on an ASIC but retain the ability to realistically simulate the behaviour of an ADPLL network. The result of this project is such a platform, designed to be extensible, with flexibility built into each component/module used.

Chapter 2

Background Review

2.1 Brief Overview

In a world where the demand for high performance hand-held computing devices continues to grow and the prevalence of "smart" devices is increasing, there is unprecedented demand for System-On-Chip (SoC) devices to control systems as varied as medical devices and entertainment systems. As these applications become more and more demanding, with ever increasing amounts of data to process and the expectation that today's devices will outperform those of yesterday, the problem of maintaining the steady gain in performance of SoCs remains at the fore.

The main drivers of performance in SoCs are the number of transistors on a chip, which is correlated with the number of calculations that can be carried out simultaneously, and the frequency at which the device operates, which determines the number of calculations performed per second. Moore's Law, based on the famous observation by Gordon Moore in 1965 [1], predicted a doubling in the transistor count of Integrated Circuit (IC)s per year for the forthcoming decade. This behaviour has carried on to this day as a result of the ever decreasing size of transistors, and has only begun to slow down in recent years. However, as the number of transistors on a chip has increased roughly following Moore's Law, the increase in clock frequency has not been able to follow a similar linear trajectory, having remained roughly equivalent for the last number of years [2], indeed the clock speed of the Intel Core family of Central Processing Unit (CPU)s has not changed since their introduction in 2009 [3].

This plateauing of clock frequency has been caused by high power consumption due to the demands placed by the global distribution of a high frequency clock, often the single biggest consumer of
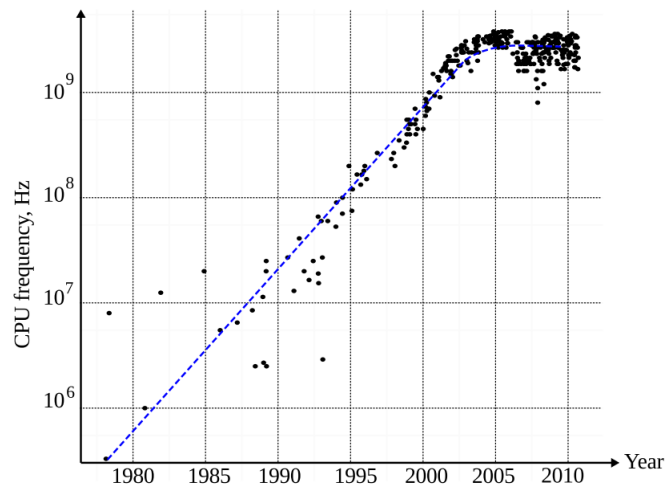
Figure 2.1: Frequency of the Intel microprocessors over past 30 years [4].

power on the chip [5]. With the growth of the Internet Of Things (IOT) market where low power devices are desirable, with many of the emerging uses of SoCs being portable and thus without a permanent power source, high power consumption goes directly against one of the key pillars of the technology. This forces many of these devices to use lower performance hardware in order to reduce the power consumption, and increase the battery life, of their devices.

In digital systems, two main approaches are used when designing the clocking system. In both cases, the chip is broken down into small areas in which all transistors are clocked synchronously, with the size constrained by the ability to deliver a quality clock signal to all transistors. The first of these methods is Globally Synchronous Locally Synchronous (GSLS), where the clock signals in each of these subregions of the chip are synchronised with one other. In practice, however, this is very difficult to achieve, as extremely high precision is required across the ever increasing number of transistors and the entire area of the chip, and doing so leads to high power consumption.

In contrast in a Globally Asynchronous Locally Synchronous (GALS) clock delivery system the "local" areas are not synchronised with other. This reduces the clocking system's complexity and thus the power consumption and chip area used, at the expense of communication speed between blocks. This disadvantage comes from the need to then somehow synchronise the messages being sent from one area to another to avoid the corruption of any messages. A GSLS, system, however has the advantages of deterministic behaviour and greater rates of communication between clocking areas and, as such, remains a desirable system design. A number of methods which deliver GSLS clocking exist at present such as clock trees as well as emerging technologies

such as ADPLL networks.

2.2   The Impact of Clocking Errors

In Figure 2.2 the data path between two synchronously clocked registers is shown, with the circuit's function being carried out by the combinatorial network between the registers. Each register has a setup time, which represents the amount of time that the input value to a register must remain constant before the clock edge, and a hold time, the time for which the input must remain constant after a clock edge.
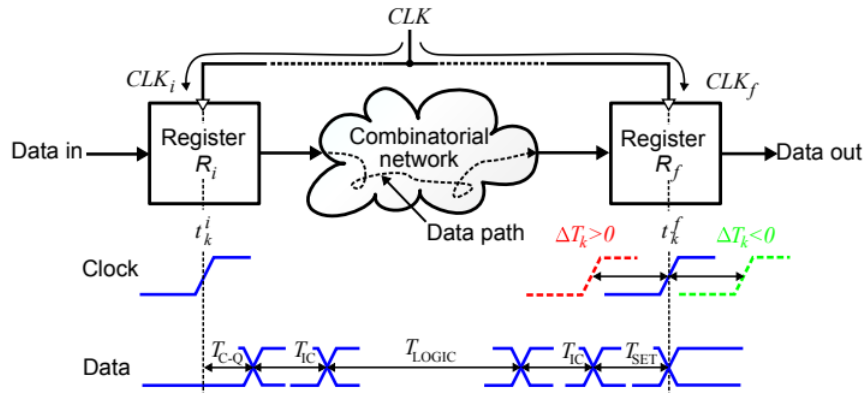
Figure 2.2: Data flow in a clocked system [4].

A lack of synchronisation between the clock edges will manifest itself as a time difference between the clocking events at both registers, $\Delta T = t_k^i - t_k^f$. ΔT is considered to be ergodic and can be described by an average deviation called skew and random process, normally modelled as a Gaussian random variable. If ΔT is negative this reduces the time available for the intervening combinatorial network thereby, having the same effect as a reduction in clocking frequency. Correspondingly a positive ΔT for depicted registers implies a negative ΔT for R_f and the subsequent register. The most common sources of clock error are caused by mismatches which usually stem from production, such as differences in the length of clocking paths, buffer delays or in the parameters of either active or passive components in the clock distribution network, which as the size of components on an IC reduces becomes more difficult to avoid. All sources of mismatch will manifest themselves in the clock distribution system as skew between transistors, while the noise in active components or the power supply system will appear as jitter in the clock signal.

2.3 Traditional Solutions

A number of traditional solutions exist which provide GSLS clocking systems, using a variety of techniques. The most simple of these implement clock distribution systems that are symmetrical in order to distribute a centrally generated clock signal to all areas of the chip at the same phase. These systems are named in accordance with their geometry, with the most common variants being branch, X or H trees.
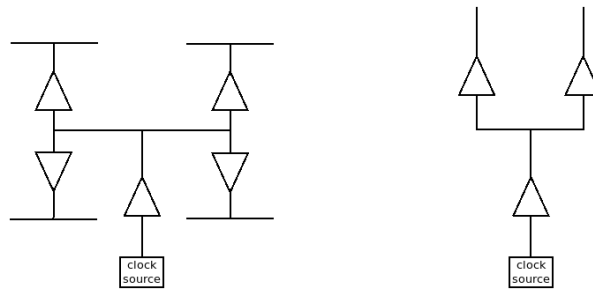

Figure 2.3: H and branch tree clock distribution systems.

While on the surface these appear simple, the task of obtaining an exact matching is, in practice, the limiting factor in this design. Even if the clock distribution system is geometrically symmetrical by design, production mismatches in either active or passive components will lead to a skew that varies from part to part. In order to minimise the impact of production tolerances, the dimensions of components in the distribution network can be increased, thus reducing the relative variation possible. However this has the impact of increasing the power consumption of the distribution network [5].

A mesh clock distribution network is an alternate design where the clock is delivered using a Cartesian grid of distribution lines. Compared to a tree type system, the variation in skew seen with a clock mesh is inversely proportional to the density of the grid while the sources of jitter remain identical. According to Abdelhadi *et al* (2010) clock meshes "*achieve low and deterministic skew, low skew variations, and low jitter*", all desirable characteristics for a clock distribution system. However they dissipate more power due to extra capacitive loading, attributable to vast number of lines required to form the grid. Similarly mesh distribution networks suffer from potential mismatch in production and alleviation through increasing of the dimensions of interconnects will, as with a tree type system, lead to higher power consumption [6]. Alternative designs replace the electrical lines used in the tree networks with waveguides for optical signals, with only the distribution
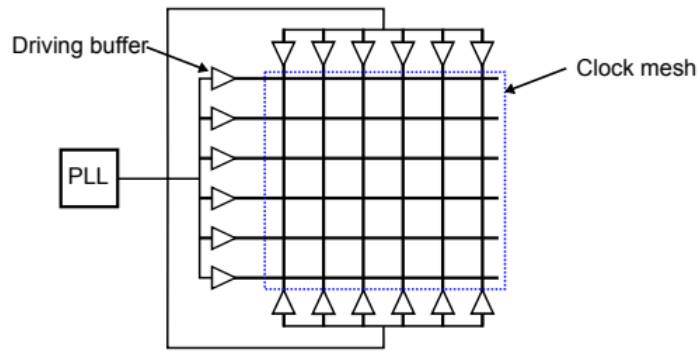
Figure 2.4: Mesh clock distribution system [4].

in the local area carried out using regular wires. This technique presents many advantages [7]: optical clock delivery is immune to the noise sources that affect electrical clock distribution systems, consume less power and do not suffer from the electrical losses present in a regular tree system.

2.4 Skew Compensation

In a tree type distribution system, skew is the main issue affecting clock accuracy and as such some effort has gone into addressing the problem. Skew due to the manufacturing process can be, at least, partly accounted for by means of active control through a skew compensator. This is a circuit, or controller, that compares the skew of each local clocking area on the chip and attempts to ensure in-phase clock delivery. Two main strategies exist to provide skew compensation, each named according to the location of the control mechanism. Designs featuring the controller located at the clock source, are known as "centralised" methods, and those with multiple controllers in the individual clocking areas known as "decentralised". Regardless of the controller placement these techniques allow for the tuning of the propagation delay between the centralised clock source and the local clocking areas.

In a centralised skew compensation circuit, the skew across the chip is calculated by the central controller which then manipulates the distribution network in order to deliver a more in-phase clock around the chip. This calculation is done by measuring the round trip time from the clock source to both the root of local clock tree, and to the individual "leaves" of the tree. The controller then has a limited ability to tune the propagation path. The downsides of this technique are the resolution of both the measurement and compensation are poor, allowing for the correction of just skew and

not of any jitter that may be present in the system, and that the extra circuitry required for both the tunability of the forward path and the two extra return paths contribute to an increased footprint and power consumption.

As the name suggest a decentralised skew compensation technique delegates the responsibility of tuning the propagation path to the individual clock regions. This strategy has the advantage of not requiring the return paths present in a "centralised" design. Instead comparison is made betweeen the leaves of different clocking areas and on this basis the propagation delay is varied. For example, Yamashita *et al* (2005) designed a system in which each clocking area or "leaf node" contains a partial clock tree. Each of these "leaves" is able to compare its clock phase to the neighbouring node, and based on the result, tune an adjustable delay buffer [8]. While this method can compensate for process, voltage and temperature variation, it does not address the power consumption due to the delivery of a high frequency clock across the entire chip area nor does it have any impact on clock jitter.

2.5   Multi-oscillator Designs

The designs described previously, are all similar in that they have a single central oscillator that provides the clock for all areas of the chip, whereas the following methods attempt to synchronise multiple oscillators, each of which provides the clock for a single clocking area. The main advantages of a multi-oscillator design, are that as each clocking area has its clock created locally, there is no degradation in the quality of the signal as it is distributed around the chip and the number of potential noise sources is reduced. In order to obtain global synchronisation some method of comparison between local clocking areas is required, and how this is done depends on the architecture. Regardless of the comparison is made, it is carried out between neighbouring clocking areas and as such the feedback network need not have a large footprint or power overhead.

One such method is a network of oscillators as in Figure 2.5 which uses coupled Phase Lock Loop (PLL) to generate local clocks. Here the output of a leaf node is compared with an external reference and the operating frequency of each Voltage Controlled Oscillator (VCO) tuned based on the result. The advantage of this method is the simplicity of the feedback network, requiring just the divided clock output from a single leaf node. The VCOs then adjusted by the control voltage, v_c, which needs delivery to all areas of the chip. However, this is a regular signal and as such
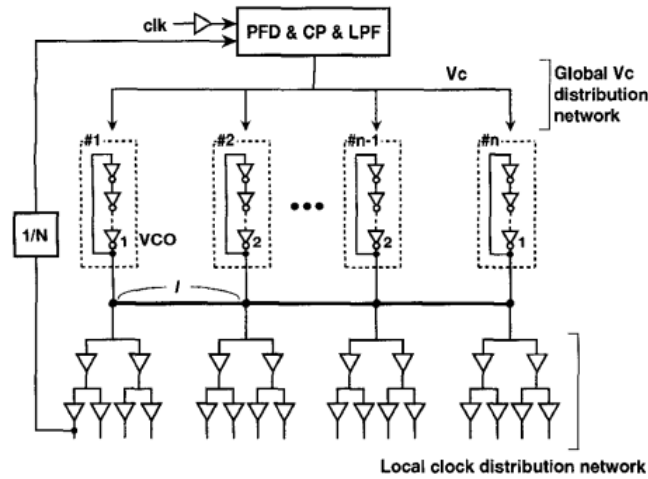
Figure 2.5: Coupled oscillator clock delivery circuit [9].

does not suffer from skew or jitter. This alleviates the need for a power hungry distribution circuit, while also being more noise-immune than the transmission of a high frequency clock. However this design still suffers from clock variation as all VCOs are fed the same control voltage, and thus the manufacturing tolerance issues present in conventional designs persists here also. This is acknowledged by the authors:

> *Unfortunately, as with the conventional ... method, distributing the VCOs over the*
> *entire chip causes the problem that jitter and skew are increased by variations in the*
> *fabrication process (static), temperature, and power supply (dynamic)* [9].

This type of multi-oscillator design is implemented by analogue circuits, and as a result not only are the clock signals, but also the control signals are liable to variation due to noise, fabrication mismatch and power supply dynamics.

Another potential multi-oscillator clock distribution system uses the phase relationship between the oscillators driving neighbouring clock areas in order to obtain synchronisation. Once again, this negates the requirement for a global distribution structure and the signals used for comparisons need only be sent between neighbouring clocking areas. As a PLL is being used it is again possible to perform the phase comparisons using a divided version of the generated clock. This in turn means the hardware transporting the divided clock signal to the phase comparator, has significantly lower requirements placed on it, thus lowering the power consumption due to electrical losses. Pratt and Nguyen initially proposed method of clock distribution in their 1995 paper entitled "*Distributed Synchronous Clocking*" in which they propose a Cartesian grid of clocking areas, each

with their own PLL, which has become known as a PLL Network [10]. In this design any given

node is synchronised with its neighbours and one of the corner nodes is additionally synchronised

with the reference. According to the authors this is "a simple, effective way to achieve low cost,

high quality, low skew clock generation in a synchronous parallel processor". They did, however,

note the presence of a phenomenon called "mode locking", which is setting of the network into a

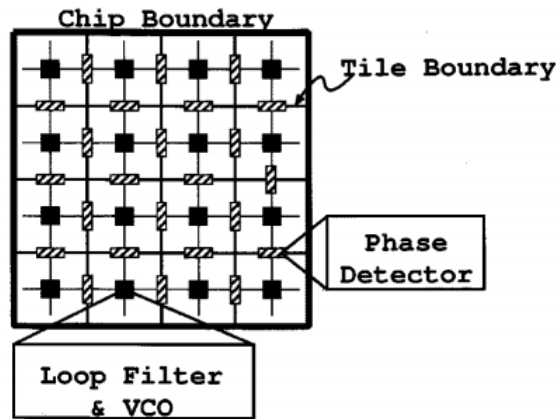stable equilibrium where there are non-zero relative phases.

Figure 2.6: PLL network topology [11].

This architecture of clock distribution network was then implemented by Gutnik *et al* (2000)

who fabricated a 4x4 array of oscillators, operating at a centre frequency of 1.2 MHz. The oscillator

was implemented as a voltage controlled "nMOS-loaded differential ring oscillator", and in order

to mode locking the phase detector was implemented as a highly non-linear circuit. The design

was a success and the authors concluded:

> *Design and measurements on this chip confirm that generating and synchronizing mul-*
>
> *tiple clocks on chip is feasible. Neither the power nor the area overhead of multiple*
>
> *PLLs is substantial compared to the cost of distributing the clock by conventional*
>
> *means* [11].

The remaining benefits of such a clock distribution system are: As the individual oscillators

have their own control signal mismatch between different oscillators is not a factor as they will also

receive different control signals. Secondly, and unlike the conventional methods, sources of jitter

in the system such as power supply dynamics can be accounted for. Finally symmetry between the

different oscillators is not required, once again attributable to the individual control signals in use.

2.6   ADPLL Networks

As a PLL network is an analog circuit, its integration in a modern IC is a barrier to usage, and as such it has not been used in any commercial designs [12]. An alternative design that is more suitable for current fabrication techniques eschews from using analogue components and instead implements the network of controlled oscillators using only digital circuitry, hence the name All-Digital PLL. A 4x4 All-Digital Phase Lock Loop (ADPLL) network was designed and prototyped in 65 nm CMOS by Zianbetov and Shan in order to test the suitability of the technique as a clock distributor [4, 13].

In this design the oscillators are once again laid out in a Cartesian grid, with each node coupled to their neighbours in phase. As this is now a digital system the coupling is carried out using digital phase comparators, which attempt to measure the phase difference between two oscillators. Figure 2.7 shows high level detail of the architecture of both the entire clocking system and that of an individual node in the design. The digital nature of this architecture brings with it a number of advantages over traditional analogue implementations, as it can benefit from advancements in digital circuit design suites, be reconfigurable and programmable and has a significantly greater immunity to perturbations inherent to its digital nature, as the exact voltage of signals is of no importance [4]. This last advantage is of particular use in a digital environment, as otherwise there is potential for clock degradation resulting from switching of transistors. The drawback of the switch to a digital architecture however is the presence of quantisation. Analogue designs both deliver continuous control signals to the oscillators and have a continuous phase detection capability, unlike a digital system where these actions are carried out with fixed resolution.

Looking at the design of a given node it is notable that the function carried out by the Error Combiner is akin to a average, therefore as mentioned by Pratt and Nyugen, there is potential to a mode locked equilibrium in which the oscillators are not synchronised. In their paper, they presented a method where initial start-up was performed uni-directionally and, once all nodes are close to alignment, full connectivity could be restored, however this was not viable in an analogue system as reconfigurability was not an option [10]. Javidan *et al* (2011) found that this was in fact the case, and stated:

> [Mode-locking] is solved in a simple and original way, by a dynamic reconfiguration
> of the network interconnection topology at the starting stage.
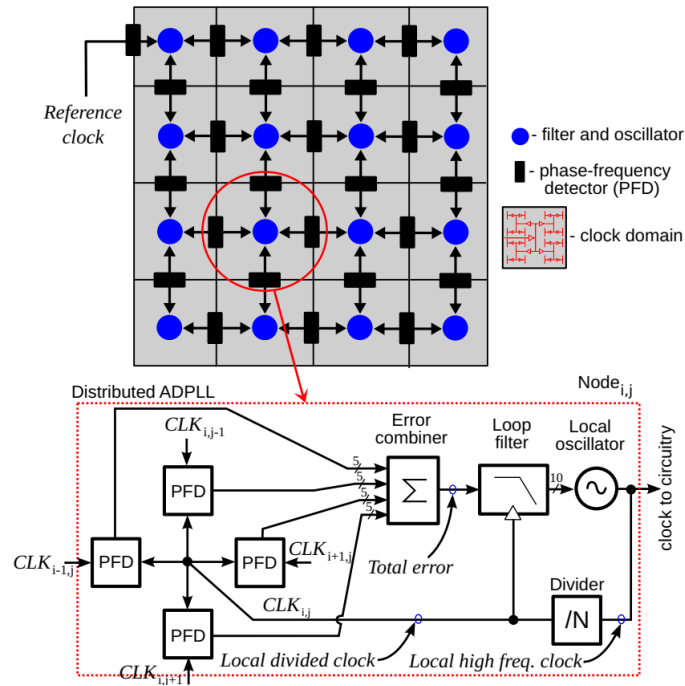
Figure 2.7: Architecture of the ADPLL network and of a single node [12].

In creating an entirely digital system, Zianbetov and Shan could easily reconfigure the network topology and by implement a uni-directional start-up avoid the problem of convergence into a mode locked state, without having to design a non-linear phase detector.

2.7 All-Digital Phase Lock Loop Architecture

As indicated in Figure 2.8 the three main building blocks of a conventional PLL are the Phase Detector (PD), Loop Filter (LF) and VCO. In an ADPLL these blocks are then replaced by their digital counterparts, necessitating quantisation in order to remain physically realisable. The "All-Digital" moniker is a misnomer as the oscillator and Phase Detector are usually both implemented by mixed signal circuits.
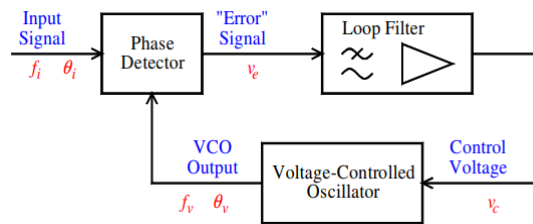
Figure 2.8: Block fiagram of a Phase Lock Loop, *Wireless Systems Notes*, B. Mulkeen (2017).

2.7.1 Digitally Controlled Oscillator

In a digital system there are a very limited number of voltages representable, most commonly just two, so using a voltage to control the oscillator is not a viable strategy. Instead a fixed bit width signal is used to control the oscillator's period, selecting the number of inverters in a ring oscillator or the varactor configuration of a travelling wave oscillator [14]. The decisions made in the design of the Digitally Controlled Oscillator (DCO), or Numerically Controlled Oscillator (NCO), determine many of the other ADPLL parameters. While tuning range and centre frequency, as well as linearity, carry over from the analogue counterpart, a DCO also has a frequency step which in combination with the bit width of the control signal determines the range over which the oscillator can be tuned. Figure 2.9 illustrates a basic ring oscillator design. A ring oscillator is an inherently unstable circuit composed of an odd number of inverters connected in a circle, which allows a signal to propagate infinitely, with the signal at any point in the circuit appearing as a square wave. The half-period of this oscillator is the time taken for the signal to propagate once through the chain, n times the propagation delay through one inverter. The frequency of operation can then be set by modulating the length of the chain, in steps of two inverters to maintain an odd number, by means of `f_select`. The main impact of output frequency quantisation is that only frequencies which are integer multiples of the frequency step away from the centre frequency can be easily reproduced, with intermediate values only obtainable in a manner akin to Fractional-N synthesis with the control code toggling back and forth. This acts as a source of jitter in the system.

Figure 2.9: Basic ring/inverter chain oscillator.
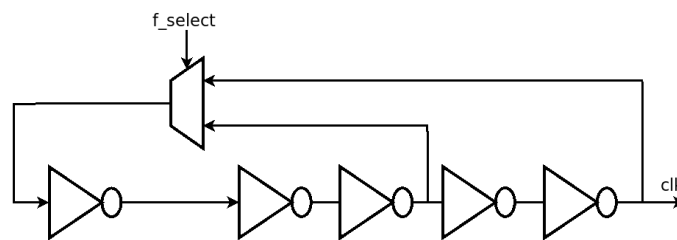
It is also possible to implement an NCO by means of a counter, in a manner that will produce either linear period or linear frequency steps. Both methods use the most significant bit of the counter's value to form the output signal. Period linearity is achieved by varying the reload value of the counter after overflow depending on the control code, thereby changing the period by a

multiple of a fixed step. Alternatively frequency linearity can be achieved if the reload value is left constant, but instead the amount added to the counter every clock cycle is changed according to the control code, once again a fixed step size is used.

2.7.2 Digital Phase Detector

Once again quantisation impacts the Phase Detector, as rather than a continuous output the phase detector in an ADPLL has a finite number of output values, thus limiting the accuracy of the phase detector. A second form of quantisation is also present, as unlike an analog system, a digital phase detector does not provide continuous data in the time domain either, instead relying on sampling. At its most basic, a digital phase comparator may only output an indication of which signal is leading, a design known as a Bang-Bang Detector, which can be constructed using a single D Flip Flop with one the generated signal connected to the "D" input and the reference signal acting as the clock. As the output only has two levels the resultant word is only 1 bit wide and as such, limits the range over which the output frequency can be controlled. More complex designs such as that in Figure 2.10, implemented by Shan, build on this by measuring the time difference between edges of the signals using a Time-to-Digital Converter (TDC) in his case using a Tapped Delay Line (TDL) [13]. A TDL is constructed by a chain of elements of a fixed delay, and the signal to be timed is applied to this start of this chain. After the timing interval elapsed the values at each point in the chain are examined, and using temperature coding, these are converted to a digital signal, the width of which is the bit width of the PD's error signal. This mimics a time measurement and allows for the phase difference to be recorded in a non binary manner.
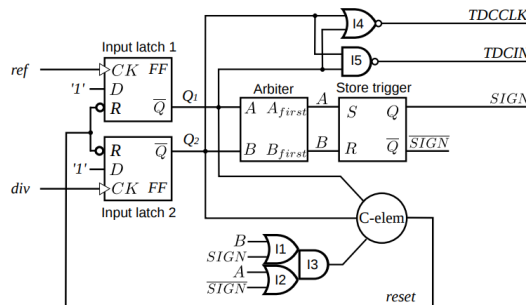

Figure 2.10: Bang-bang phase/frequency detector architecture [13].
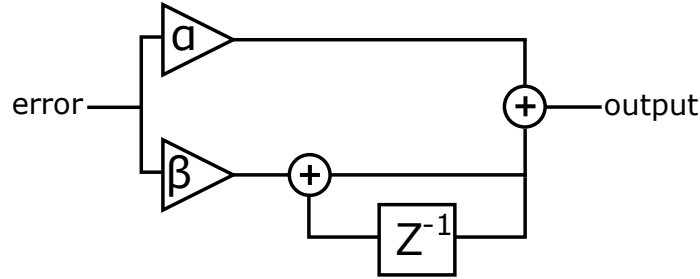
2.7.3 Digital Loop Filter

Figure 2.11: Basic PI controller architecture.

The Loop Filter in an ADPLL can be implemented as a PI controller, as only a low-pass filter is required, such as that in Figure 2.11. In the case of a node in an ADPLL network the input of this filter is a weighted of the phase difference relative to the neighbouring local clocking areas. In one example topology a digital system the proportional section can be implemented by a simple multiplier, whereas the integral path is constructed by adding the result of a multiplication by the proportional gain to an accumulator. This delayed summation can be easily implemented by an accumulator to which the current value of the multiplication is added each cycle, and as such the system has an infinite impulse response. The value of these gains determine the response and stability of the ADPLL network. The transfer function of such a controller is given by [13]:

$$H(z) = \alpha + \beta \frac{1}{1 - z^{-1}} = \frac{(\alpha + \beta) - \alpha z^{-1}}{1 - z^{-1}}$$

It has been found by Koskin *et al* (2018) that stable operation can be achieved when the integral gain, k_i, is less than half the proportional gain, k_p [15]. In the same study a range of values was found which would produce low jitter operation of the network. As these values are all less than one, the filter must implement fixed point arithmetic in an effort to maintain the simplicity of the clock distribution network, rather than incurring the complexity penalty of floating point calculations.

2.7.4 Error Combiner

The ADPLLs used in a network need to combine the error signals from multiple neighbours to determine what the average difference from its neighbours is, and this necessitates the addition of the Error Combiner. In a digital system this can be implemented by a weighted average of the

different error signals, with the weight being modifiable at run-time. This configurability is what permits the system to implement uni-directional mode and also allows for the weighting applied to certain signals, such as the external reference, to be modified. The ease of implementation of a configurable error combiner is one of the main advantages of an ADPLL over an analogue system.

2.8   The Role of the FPGA

A Field Programmable Gate Array is a type of IC that is designed to be configured by a designer after the chip itself has been manufactured. An FPGA contains a large number of logic elements that can be connected together in order to perform complex logic, written using the same Hardware Description Language (HDL)s used to design the digital blocks of Application Specific Integrated Circuit (ASIC)s. They may also implement started modules such as adders, multiplexers and Random Access Memory (RAM) as a fundamental element. More complex logic is often implemented using multiplexed lookup tables rather than true logic elements. High end devices such as the Xilinx Zynq Ultrascale even implement Multi-Processor SoCs. Compared to an ASIC the designer does not have direct control over the layout of the system but rather describes its behaviour, possibly down to the basic logic elements of inverters or other gates. Limited control is possible over the placement of the individual modules, but Electronic Design Automation (EDA) tools are responsible for the exact placement of elements. As a result, it is not possible to have precise control over the delays experienced as signals propagate through the design. To assist with the resolution of any issues EDAs provide tools to analyse timing behaviour.

Prototyping on an FPGA is a common verification stage for conventional ASIC designs as it allows for a hardware validation of any digital circuitry, and the detection of any potential flaws or errors made by the design before the expensive of an ASIC implementation. In their 2013 ADPLL network implementation Zianbetov and Shan used an FPGA in order to validate their programming interface, the design of the error processing block, ensure they had eliminated mode locking behaviour and to ensure phase synchronisation was possible [4, 13]. However they experienced two main limitations, they were not able to implement the mixed-signal Phase Frequency Detector (PFD) and DCO, and the maximum frequency of operation possible was orders of magnitude lower than the GHz range of their ASIC implementation. These issues were circumvented by implementing an alternative PFD and DCO designs which were driven by the clock distribution network

provided by the FPGA and every clock frequency scaled by the same amount such that the results of testing would remain indicative. One of the main advantages they saw was that the hardware description used for the digital blocks of their ASIC could be directly ported over to the FPGA.

It is however possible to implement limited mixed-signal circuits on an FPGA through the use of primitive logic elements, however, as control over the implementation is restricted to the module level it is not possible to mirror the implementation of a design intended for an ASIC. As these implementations are analogous to a mixed-signal circuit on an ASIC, the verification of theoretical behaviours, as done by Koskin *et all*, in order to test the findings of his PhD thesis [16], is made possible without the expense of ASIC fabrication. An FPGA based mixed-signal ADPLL network is also seeing use here in University College Dublin (UCD) as a initial prototyping platform for the validation of new modules for use in ASIC based ADPLL networks.

FPGAs have been used in other fields to simulate and experiment with new technologies, although not all of these attempt to implement the mixed-signal circuitry using primitive elements. Fernandez-Alvarez *et al* (2016) proposed a method suited to higher end FPGAs in which a co-processor on the FPGA simulates the mixed-signal circuitry while the digital section of the design is implemented on the FPGA itself [17]. While the interfacing between hardware and software remains a challenge they found:

> Obtained data are compared to the data obtained by means of using PSIM and ModelSim co-simulation. The proposed solution speeds up the evaluation in around one order of magnitude keeping the accuracy. The output signal differs in less than 0.6 mV (RMSD).

Mixed-signal circuits were, however, simulated in hardware by Óscar Lucía *et al* (2011) on an FPGA in order to overcome the excessive time penalty imposed by software based simulations that required the behaviour of both a digital and mixed-signal peripheral and that of a micro controller running code in C to be simulated side by side [18]. They concluded

> ... the proposed system provides a versatile and fast method to develop ad hoc control architectures, avoiding the need for time-consuming mixed-signal simulations and the risk of damaging the actual power converter implementation.

By carrying out simulations on an FPGA, Guanhua Wang *et al* (2013) achieved a 3000 times decrease in run-time when compared to an identical simulation in MATLAB used for the verification

of a calibration algorithm for successive approximation analogue-to-digital converters [19]. Many other examples exist of FPGAs used for the simulation of mixed-signal or Radio Frequency circuits in literature.

2.9 ADPLL Performance Characterisation

As already stated, goal of a clock distribution system is to synchronise clocking events in all areas of the chip. The effectiveness of this synchronisation is characterised by two main metrics, jitter and skew, which together describe the distribution of clocking events. Skew is the average time delay between a clocking event in one area of the chip and that of a reference event. It can be easily measured by computing the average value of this delay. Jitter has a number of definitions depending on how it is measured, but at its most basic it is the standard deviation of the time delays with respect to the reference clocking edge.

The simplest form of clock performance characterisation is on a Cycle-to-Cycle (C2C) basis, in which the reference is the signal under test itself. Here the standard deviation of the individual periods gives the jitter of the signal, while skew has no meaning with a self reference the mean period can be used to compute the centre frequency over the time interval. For a more informative measurement the Time Interval Error (TIE) can be computed, which takes into account another signal as the reference. TIE is calculated by comparing the delay between the signal in question and a reference that is treated as ideal. The mean value gives the relative skew between the two signals and once more the standard deviation of the measurements gives the jitter with respect to this reference.

Both above forms of measurement assume a large number of sequential measurements, however, there are other ways that jitter and skew can be calculated [20]. Phase jitter is an important characteristic for communications systems as it can be used to calculate phase noise, important to ensure spurious emissions are within regulations. Long term, or accumulated, jitter represents the cumulative effect of jitter on the signal over several cycles. Long term jitter in particular affects RADAR as it will manifest itself as a Doppler shift in the return signal. For an PLL network, with an appropriately designed filter, the long term jitter should be zero so long as the reference signal is stable.

Chapter 3

ADPLL Designs for FPGAs

3.1 Chapter Overview

The first step in creating an Field Programmable Gate Array (FPGA) based network of ADPLLs is the design of the ADPLL itself, which will be addressed in this chapter. The nature of an FPGA necessitates a number of compromises in the design of a given block which limits transferability to Application Specific Integrated Circuit (ASIC) designs. In this chapter the potential designs for each individual block, or module, investigated will be explained and the case for their selection in an FPGA based ADPLL made. A number of blocks have implement purely digital circuitry and as such can be transferred in their entirety from an FPGA and vice versa. However, those that will be used to emulate mixed-signal circuitry, such as the Digitally Controlled Oscillator (DCO) and Phase Frequency Detector (PFD), will be examined in greater detail. A Register Transfer Level (RTL) diagram of the ADPLL topology that will be assumed in the following sections is shown in Figure 3.1.
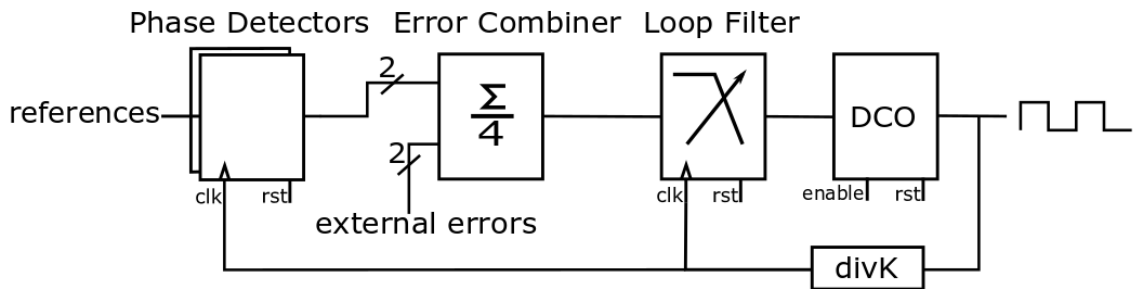

Figure 3.1: ADPLL designed for use in a Cartesian grid network.

Important to note is that despite a Cartesian grid leading to four neighbours, only two phase de-

19

tectors are required per oscillator. Borrowing matrix indexing, and choosing the oscillator located at index $i, j, i, j \neq 1, N$ where N is the number of oscillators in each direction, comparison need only be done with the signals generated by oscillators located at $i-1, j$ and $i, j-1$. The comparison with oscillators $i + 1, j$ and $i, j + 1$ will be carried out in those ADPLLs, and the negation of the errors measured used in the ADPLL located at i, j.

3.2 Digitally Controlled Oscillators

The choices made in the design of the DCO have the greatest impact on the effectiveness of the overall platform and which use cases the ADPLL containing it are suitable for, as the key performance benchmarks are all done using the waveform this block generates. This project will address three distinct designs of ADPLLs suitable for implementation on an FPGA, two derived from the clocks generated by the FPGAs own distribution network and one generated independently of this clock, using a chain of inverters. These are not the only ways in which an oscillator could be synthesised on an FPGA, however other designs were deemed to be unsuitable for extensible and portable implementations.

A prime example of this is the use of Xilinx proprietary `IODELAY` blocks to create an oscillator, as detailed in Xilinx Application Note XAPP872 [21]. The key idea here is that the bulk of the period is made up by the propagation time through one of the `IODELAY` blocks, which can be set at implementation time. This is combined with a section of an inverter chain, and a multiplexer used modify the length of this segment, the output of which is fed back into the `IODELAY` block. This method was discarded as the number of `IODELAY` blocks is very limited, so expanding to a larger network would be impossible, and they are all located around the edge of the chip, not suited to the construction of a Cartesian grid.

The main issue with the creation of DCOs on an FPGA is the inability to create mixed-signal circuits, such as those that would be intended for use on an ASIC. As such the FPGA based oscillator must emulate the behaviour of a mixed-signal circuit in some way.

3.2.1 FPGA Driven, Linear Period DCO

The first design of DCO to be examined is of the type used by Zianbetov in his ADPLL network test bed and relies on a counter driven by the clock manager on the FPGA [4]. At each event on the
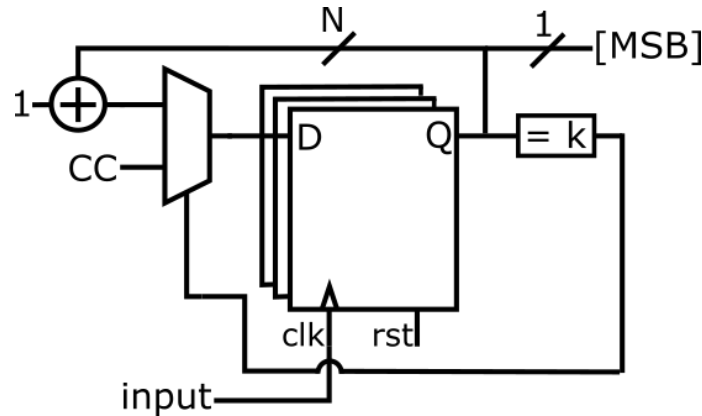
Figure 3.2: FPGA driven, linear period DCO RTL Diagram.

FPGA provided clock a counter is incremented, overflowing upon increment past the maximum possible value. The Most Significant Bit (MSB) of this counter forms the waveform generated by this oscillator, the period of which is controlled through an adjustable value that is loaded into the counter once overflow is reached, and forms the starting points for the counter. The period of oscillation is given by:

$$T_{osc} = \left(2^{width} - \left(BIAS + CC\right)\right) \times T_{FPGA} \tag{3.1}$$

where T_{FPGA} is the clock period of the FPGA, CC is the control code input, $BIAS$ centres the oscillator in the middle of the tuning range in the event that the control code is zero and $width$ is the width of the counter used. As the only variable here is the control code, period step of this design is T_{FPGA}, thereby providing period linearity with respect to the control code. This is the key advantage of this design, as most ASIC implementations of a DCO are also linear in period. The other main reason to choose this design is that its FPGA clocked nature allows for exact control over the frequency of operation, and the number of tunable parameters make it possible to configure multiple ways to achieve the same frequencies of operation. Combined these attributes make it very easy to create an oscillator that emulates the behaviour of a design intended for an ASIC, however, at a greatly reduced frequency. This restriction on the frequency of operation arises out of the period step size, which in order to obtain a good resolution must be orders of magnitude smaller than the intended period to be generated. As the output waveform is taken from the counter's MSB, the reload value of the counter must never go beyond $2^{width-1}$, as otherwise the output waveform will become a constant 1. As the reload value varies the low time of the MSB, if the desired output

waveform is a square wave this design will not be suitable.

In being FPGA clocked this design has pseudo-deterministic characteristics, with each period step being almost identical across oscillators and over the entire tuning range, unlike an ASIC where process variation will impact the layout of a high frequency oscillator. The only variation in this design will come, ironically, from jitter or skew in the FPGA's clock distribution network, which as the frequencies will typically be in the low hundreds of MHz is very minor. In the case of the Xilinx XC7A100T-1CSG324C this is at most 100 picoseconds, or 0.05% of the period of an intended output clock at 5 MHz. To put this value into perspective, on this board the minimum value of T_{FPGA} that could be used to drive this oscillator is 3.87 nanoseconds, 1.935% of the period.

The resulting DCO is best suited to applications that do not seek to gain a better understanding of oscillator performance, but rather those focused on validating the entirely digital blocks in the system, the role in which Zianbetov and Shan used this type of oscillator [4, 13].

3.2.2 FPGA Driven, Linear Frequency DCO
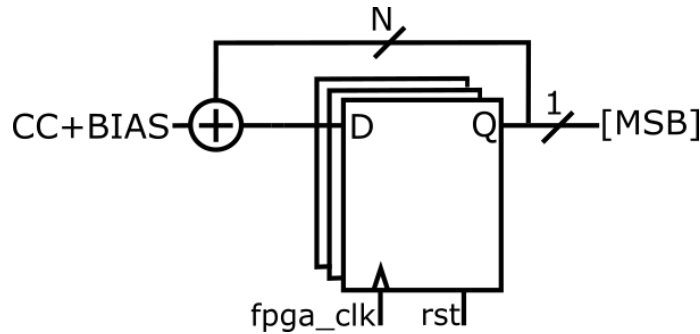
Figure 3.3: FPGA driven, linear frequency DCO RTL Diagram.

The second FPGA clocked oscillator is similar in most attributes to the above design but eschews period linearity for frequency linearity. Again the overflow property of a counter is used with the counter's MSB as output of the block, however, this time it forms a square wave. Rather than setting the reload value of the counter, instead the increment is adjusted depending on the control code, thus requiring $\frac{2^N}{BIAS+CC}$ increments to overflow. Accordingly the frequency of operation is set by:

$$f_{osc} = f_{FPGA} \times \frac{BIAS + CC}{2^{width}} \tag{3.2}$$

Here the control code CC and bias are added to the value stored in the accumulator at each event

of the FPGA, clock until overflow is reached. This occurs at 2^{width} where, as before, $width$ is the bit width of the counter, thus valuing each control code increment at $\frac{f_{FPGA}}{2^{width}}$ Hz. As with the previous design, this oscillator is better suited to frequencies where the output of the DCO is orders of magnitude lower than the clock signal driving it, as this ensures that the incremental change due to the control code remains a small fraction of the period.

This design is just as configurable as its linear-in-period counterpart, and well suited to the low frequency emulation of ASIC based oscillators that are themselves linear in frequency. In sharing the FPGA as a clock source again the pseudo-deterministic characteristics return, once more meaning this oscillator is better used for testing, simulating or verifying other blocks in the system.

3.2.3 Ring Oscillator

The best approximation of the non-idealities of an ASIC DCO implementation can be obtained using a Ring Oscillator (RO), also know as an inverter ring, as the DCO. In an ASIC an RO is constructed by connecting a number of NOT gates back in a chain, with the output of the last NOT gate fed input the input of the first. As NOT gates are not available on an FPGA, they are replaced by inverter primitives. The inverter ring is an inherently unstable circuit that leverages the propagation delay through an odd number of inverters to create a waveform that, when viewed at a fixed position, is a square wave.
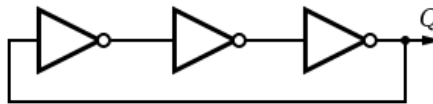
Figure 3.4: Basic ring oscillator RTL diagram [22].

The output value of the inverter at the end of the chain changes a non-zero amount of time after the input to the first element is set. It can be trivially shown that if the number of inverters is odd, this will be the inverse of the value initially applied to the first element. This sets in motion an oscillation, the period of which is governed by the number of inverters in the chain. The formula to calculate the oscillator's period is simple:

$$T_{osc} = 2 \times \tau_{inv} \times N \tag{3.3}$$

Here, τ_{inv} is the propagation time through one inverter, N is the number of inverters in the chain and the multiple of two arises from the necessity to invert twice to produce a square wave.

The main advantage of this type of oscillator stems from its similarity to that used on an ASIC which leads to many non-idealities carrying over from the truly mixed-signal implementation. Rather than being just a tool to validate that other modules, the performance of this oscillator can be analysed more deeply as well as the performance of networks containing it. Koskin *et al* used such a design to test theories proposed regarding the synchronisation of oscillators during his PhD using this type of oscillator implemented on an FPGA [16, 23]. Also in this design's favour is the period step size, four times the inverter propagation delay, which was found experimentally to be than in the region of a nanosecond on Xilinx Artix-7 and Zynq FPGAs.

While it is advantageous for the purposes of analysis that this design is affected by variation in temperature, implementation and other non-idealities this brings with it challenges not present in a true mixed-signal design. In a design intended for an ASIC the design has near total control over the placement of circuit elements, however, the FPGA based counterpart control is available over just the area in which the oscillator may lie, and the Electronic Design Automation (EDA) is responsible for the exact placement. Unlike the ASIC implementation, in which the propagation delay through each element of the ring is well bounded, the lack of control over placement means that the propagation time through each inverter is not well bounded and may vary wildly. This makes it time consuming to ensure that the frequency tuning ranges of each oscillator match that desired, and use of this oscillator less straight-forward than that of the previous two designs.

3.3   Frequency Divider

Many Phase Lock Loop (PLL)s, be they digital or analogue, incorporate a frequency divider in the feedback path so that comparison can be made with a reference operating at a lower frequency. This is a feature that is advantageous for ADPLL networks, as comparison using this divided clock allows the synchronisation signal sent between modules to be of a frequency orders of magnitude lower than that of the clock distribution network and thus require circuitry that consumes significantly less energy. This division can be obtained trivially on an FPGA by a number of means, of which this thesis will mention just two. The first of these methods allows for integer division, by inverting a signal each time a counter reaches a certain value, k. When this condition is true the

value stored in the counter is also reset to zero, thus implementing division of the generated clock frequency by $2k$. By limiting to division by powers of two, the divider can be simplified with the
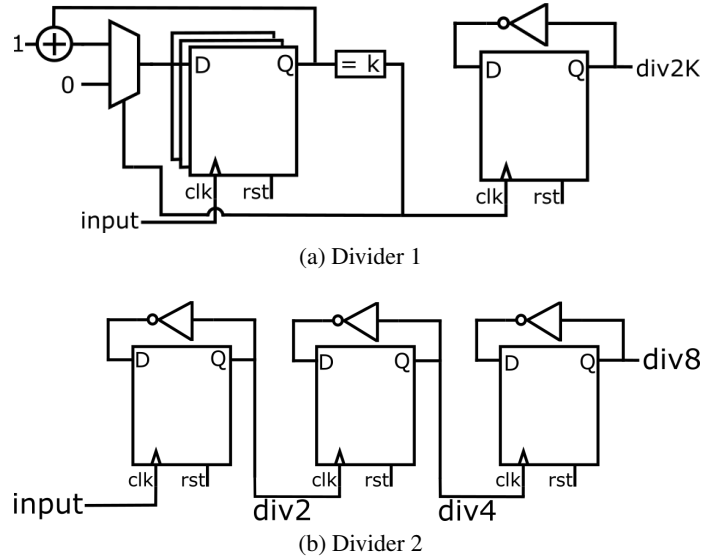

(a) Divider 1

(b) Divider 2

Figure 3.5: Frequency divider RTL diagrams.

removal of any comparison logic. One flip-flop and an inverter are required per power of two. As in Figure 3.5 (b), the inverted $\overline{Q}$ output of each flip-flop is inverted and connected to the D input, with the non-inverted Q of the previous flip-flop acting as the clock. The first flip-flop in the chain is clocked by the DCO output.

3.4 Phase Detector

Three potential Phase Detector designs will be addressed in this thesis, sign-only detection with a Bang-Bang Detector, FPGA clocked detection using a Finite State Machine (FSM) and an emulation of a Tapped Delay Line (TDL) using inverter primitives to create a delay, similarly to the RO seen previously. Both designs will have to have a representation for both situations where the generated signal leads and lags the reference signal, and for the magnitude of this difference, a problem most obviously solved by using a signed binary representation of the difference. Most easily implemented in hardware is two's complement, as operations can be carried out by the same hardware as unsigned arithmetic. The sign convention for phase error assigns the positive values to the situation where the generated signal lags the reference, and vice versa for the leading case. Apart from the Bang-Bang Detector, each design features two blocks, one to determine the sign of the error and a second which estimates the magnitude. These blocks could theoretically be

mixed-and-matched, but as they operate on different paradigms no benefit would be obtained.

3.4.1 Bang-Bang Detector

A Bang-Bang Detector is a sign-only detector, thus outputting a phase error just a single bit wide. This can be accomplished using a single flip-flop, with the reference signal acting as the clock, in this case using the rising edge, a design verified by Predrag in his 2011 thesis [**?**]. The generated signal is applied to the D input, thus when a rising edge is seen on the reference the value of the generated signal is held as the Q output until the next rising edge of the reference. If the generated signal has gone high prior to the reference, and leading it, the value of Q will be 1, thereby matching the expected sign behaviour. Trivially it can be seen that the case where the generated signal lags the reference the Q will show 0.
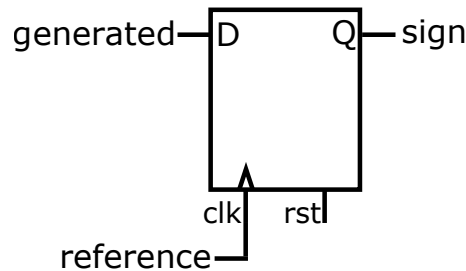

Figure 3.6: Bang-bang phase detector RTL diagram.

This design is simple but effective, and can even be used without a loop filter, albeit with a very limited oscillator tuning range. This is especially valuable an attribute at an early stage of platform development, as it may be used in order to test that other aspects of the system work as intended. The binary resolution of this detector, however, restricts its usefulness as the ADPLL cannot react more quickly to larger phase differences.

3.4.2 FPGA Clocked Phase Detector

The easier to implement of the two designs uses a combination of a state machine to determine which edge occurred first, and to control a two's complement up-down counter, which acts as the Time-to-Digital Converter (TDC) in this detector, and estimates the magnitude of the error.

3.4.2.1 State Machine Based Detection

While sign detection clocked by the FPGA can be performed in a variety of ways, a state machine is a straightforward way in which to accomplish this. A Finite State Machine consists of three main components: the states themselves, the next state logic and the output logic. In this case the next state logic is the generated and reference signals, and the output logic sets the count instructions for the up-down counter as well as ensuring the phase error output is held constant between measurement periods. The FSM will have a state representing the case where either the reference or generated signals occurred first, in addition to a wait state used between the end of one measurement interval and the commencement of the next. Two types of FSM exist, Moore and Mealy Machines, the difference being that Moore Machines base their output only on the current state while Mealy Machines do so on both the current state and the inputs to the system. If a Moore Machines is used an extra state is required in order to hold the value of the phase error at the end of the measurement interval.

As this design is clocked by the FPGA, the phase error is quantised to the the FPGA clock periods, thus setting the resolution of the design. While, depending on the oscillator design, the generated clock may be synchronous to the FPGA, the reference signal arrives from off-chip and thus will not be synchronised to the measurement clock. In order to avoid any metastability events arising from an input signal changing within the set-up and hold times of the registers sampling them, synchroniser circuits are used to ensure the value sampled is either logic high or low at the instance of the measurement. As these synchronisers are the first FPGA clocked elements in the measurement chain, they also act as the system's quantisers. There is no degradation in detection resolution, as the signals would be quantised to the FPGA clock anyway.

There are three situations that may occur while the FSM is in the waiting state. Firstly the case where the reference signal sees a rising edge before the generated clock, and thus the generated clock is said to be lagging the reference. As previously mentioned the convention dictates that the phase difference have a positive sign, so the counter is incremented to count in the positive direction. Similarly if the generated signal is leading the reference the sign of the error is negative and thus the counter is instructed to count downwards. The third potential scenario occurs when both clock signals have a rising edge within the same period of the FPGA clock, which due to quantisation will be interpreted as both signals occurring at exactly the same time. In this case the

counter receives no instruction and a zero value is stored until the next measurement interval. Once the FSM is in a measurement state, and the counter either incrementing or decrementing, a rising edge occurring on the opposite signal to that which initiated the measurement will end the interval.
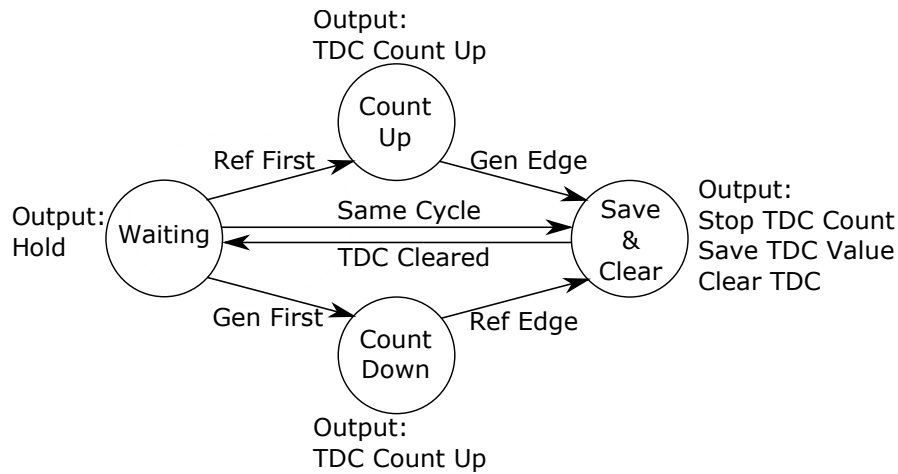

Figure 3.7: Example State Transition Diagram for a Moore Machine.

3.4.2.2 Up-Down Counter

The up-down counter in this design performs estimates the magnitude of the phase difference between the reference and generated signals. The state machine provides control instructions, indicating whether the count should be in the increasing or decreasing directions, and and holds the counter in reset once the measurement interval has been completed. The counter counts in two's complement, so counting in the downward direction can be carried out by adding on the representation of -1, the binary signal containing all 1s. The other advantage of counting in two's complement is that count value is already in the correct format for the follow block, the phase detector in a regular ADPLL or the error combiner in this instance, therefore the count value at the end of the measurement interval can be stored in a register, which maintains the output value of the phase detector between measurements. Two's complement is an advantageous way to represent signed numbers in hardware as the mathematical operations are identical to their unsigned counterparts, thus allowing use of any adder or carry hardware that may be present on the FPGA. An important attribute of this counter is the ability to freeze the count at the minimum and maximum count values in order to avoid a situation where the counter overflows, thus at high phase offsets

potentially alternating between positive and negative differences which may restrict the locking range of the ADPLL. Disallowing overflow also aligns the behaviour of this phase detector with that of those implementing delay lines, allowing better interchangeability and comparison, which for a testing platform may be valuable.

This counter is best suited for use with the aforementioned FSM sign detection circuitry, and would see no benefit from a combination with the un-clocked sign detection circuit that will be presented in the following section. This is the case as if quantisation to the FPGA clock had not already occurred in the sign detection circuit it would occur in this counter, as increments are only possible at FPGA clocking events, thereby enforcing quantisation to the FPGA clock. The coarseness of this measurement, restricted to time delay increments equal to the FPGA clock period means that it is an ideal detection method for the clocked oscillators, as the minimum phase difference that can be detected matches exactly with the minimum step by which the oscillator period can be tuned. This design can, however, be used with the Ring Oscillator, provided the Loop Filter gains are adjusted to maintain the oscillator to detector step relationship.

3.4.3   SigNum Detector

The SigNum detector is based on the methodology used in the Bang-Bang Phase Detector seen previously, but rather than just performing sign detection, the magnitude of the phase difference can also be measured. In the case of the FPGA clocked design the counter handled the representation of both sign and magnitude, but as the name SigNum (Sign Number) suggested both are handled separately in this design. The modified Bang-Bang detector establishes which signal occurred first and generates a control signal that is analogous to the time delay between a rising edge on each input signal. A TDL then uses this control signal to convert the phase difference between the signals to a digital signal. This magnitude is combined with the sign detected by the bang-Bang detector to form a two's complement error signal.

This method of phase detection is advantageous as similarly to the Ring Oscillator, the TDL can be implemented using inverters, and therefore is not synchronised to the FPGA clock and the propagation time through inverters sets the resolution, thus allowing a smaller step than that of the clocked designs. The disadvantages, however, are also consistent with the RO in that the propagation time will vary significantly depending on the EDA's placement of the circuit. This is not as severe a limitation for the TDL as the RO as while the tuning range is affected by this

variation it cannot cause the centre point to vary, whereas the impact of placement on the RO caused both variation in range and centre frequency.

3.4.3.1 Sign Detection

The Bang-Bang detector structure changes significantly in order to provide the control signals required for the TDC. The TDC expects a long pulse of logic high for duration of the measurement interval, going low once the interval has ended. The original single flip-flop design maintains a value between rising edges of the reference clock and could not generate the required pulse. Instead two flip-flops are used, each clocked on an input signal with their D inputs connected to logic high. The Q outputs of these registers are connected to an XOR gate which generates a pulse as long as a rising edge has only been seen on one signal. Once a second rising edge occurs the XOR output will be set to 0, thus terminating the pulse. On completion of the measurement interval, the flip-flops must be reset in preparation for the next measurement cycle, and this can be accomplished by ANDing together the Q outputs of each signal, producing a `clear` signal that will go high when both flip-flops have a logic high output.

To determine the sign itself, which clock edge first had a rising edge must be recorded, and this can be done by setting a Set-Reset (SR) latch if the rising edge of the generated signal occurs first and resetting in the case of the reference clock having the first rising edge. Looking at the whole measurement cycle however, it is apparent that all 4 combinations possible of two inputs occur, putting the SR latch into which the output is undefined. Careful design of the system can avoid problems arising from this. Examining the truth table in Table 3.1, the combination to be avoided is the presence of a 1 on both outputs, precisely the situation seen at the end of the measurement interval. In order to avoid the undefined state occurring at the end of the measurement interval, both inputs and outputs to the SR latch can be inverted using a NOT gate. Table 3.1 also illustrates how the $\overline{Q}$ output now behaves as Q did previously, however the end of measurement state where both input registers have been triggered now causes the SR latch to enter the hold state where it maintains the previous values. The sign, $\overline{Q}$ of the SR, can then be loaded into a register using the `clear` signal generated previously, to maintain it until the next measurement completes.

This, however, ignores the potential for metastability which may arise if both signals have a rising edge within an extremely short period of time. While it may in other systems be feasible to dismiss this occurrence as a low probability event, the goal of a PLL is to reduce the time

S	R	Q	$\overline{Q}$
0	0	Q_{n-1}	$\overline{Q}_{n-1}$
0	1	0	1
1	0	1	0
1	1		undefined

Table 3.1: SR Latch Truth Table

delay between rising edges to zero which increases the possibility of this occurring, which may lead to indeterminate behaviour in other parts of the feedback network. In order to avoid this in an ASIC, an arbitration circuit is placed between the SR latch and the register storing the sign between measurements, which will default to one of the two signals in this case. This is however a mixed signal circuit, such as that proposed by Tierno *et al* (2008) and implemented by Zianbetov and Shan [4, 13, 24], which requires two P-type Metal Oxide Semiconductor (PMOS) and two N-type Metal Oxide Semiconductor (NMOS) transistors and cannot be implemented on an FPGA. As the output of the SR latch will be indeterminate prior to the detection of the first edge, the impact of both edge detection registers' output is that the SR latch attempts to hold this indeterminate value as an output. To avoid this unwanted behaviour the arbiter present on an ASIC is replaced by a second SR latch, connected as in Figure 3.8. This ensures the value presented to the register storing the sign is always determinate.

When	Gen	Ref	$\overline{Gen}/S^a$	$\overline{Ref}/R^a$	Q^a	S^b	R^b	$Sign/Q^b$
Between Measurements	0	0	1	1	undefined	01 or 10		0 or 1
Ref. First	0	1	1	0	1	0	1	0
Gen. First	1	0	0	1	0	1	0	1
Measurement Complete	1	1	0	0	Q^a_{n-1}	$\overline{Q}^a_{n-1}$	Q^a_{n-1}	$\overline{Q}^a_{n-1}$

Table 3.2: Sign Detector Truth Table

3.4.3.2 Tapped Delay Line

To complement the asynchronous sign detection circuit, an asynchronous TDC is also required, as otherwise there would be no benefit to using the above method over the more easily understood FSM based design as quantisation to the clock period would happen in the TDC rather than the sign detection circuitry. The standard method for implementing a TDC on an ASIC is through
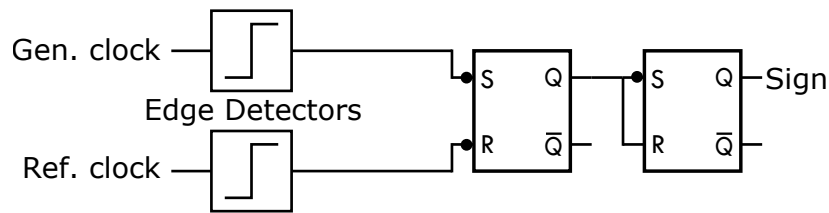
Figure 3.8: Example Sign Detector.

a tapped delay line, where a signal is propagated through a chain of fixed length delays, and at the end of the measurement interval it is recorded whether or not the signal has propagated to each point in the chain. This architecture is illustrated in Figure 3.9. As with the mixed signal circuitry previously addressed, a TDL cannot be exactly replicated on an FPGA, but through the use of inverters it is possible to replace the unrealisable well bounded delays. As with the RO, the delay though the inverters is not nearly as well bounded, and significant variation between delays, in both individual TDLs and within the same TDL, due to the implementation occurs This variation notwithstanding, the inverter pairs are a suitable replacement, and this method of time-to-digital conversion provides both resolution advantages and the presence of implementation based variation over a clocked design. These prove particularly beneficial when attempting to verify theoretical results.

Similarly to the sign at the output of the detection circuit, flip-flops store the value of each tap between phase error measurements. Prior to storage, the tap values must be converted to two's complement. As the signal propagates through the taps, magnitude of the phase error is determined by how many taps the signal has passed through. Conceptually, this thermometer coding is first converted to an unsigned integer before conversion to a signed quantity based on the sign of the error.
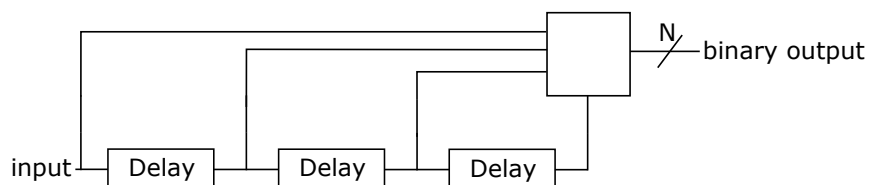

Figure 3.9: Example Tapped Delay Line Structure.

3.5 Error Combiner

In a regular ADPLL an error combining circuit is not required as there is never more than one reference signal, and as such the input of this block would be equivalent to the output. However in an ADPLL network multiple reference signals are required, one for each neighbour, thus requiring some method to combine. Simply averaging these signals was not found to be viable by Pratt and Nyugen in their original paper for analogue systems, however, the work of Javidan *et al*, showed that in a digital system this method will not result in mode locking, provided start-up is performed uni-directionally [25].

Accordingly, any intended error combiner design implementing an average should also be re-configurable, so that it satisfies the reconfigurability constraint. Given this is a digital system the most straightforward way to implement this is as a weighted sum, followed by a division. The final ADPLL network will be laid out in a Cartesian grid so a maximum of four values will possibly be combined, with only the non-corner edge nodes having a number of neighbours that is not a power of two. The simplest, and most processing time efficient, method of calculating the weighted sum restricts the weights to powers of two, as this permits their implementation as a left bit shift which is trivially achieved in hardware by a part select. Similarly this reasoning can be expanded to the division, where the complicated division by three for a non-corner edge node is instead replaced by division by four. This means the division is implemented identically by all error combiners in the network, and implemented trivially by a right shift of two bits.

3.6 Loop Filter

The Loop Filter in an ADPLL can be implemented as a Proportional Integral (PI) controller, a circuit that can be implemented in a wide variety of ways, with the exact implementation having little impact on behaviour. The first choice to be made is whether the designer wishes to implement the controller as an Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) filter. Both methods will produce a controller that fundamentally behaves the same way, but will have a starkly different implementation. In both cases the filter can be implemented using either fractional or integer representation of the data, depending on the wishes of the designer. Carrying out computations using fractional values may be conceptually easier to understand as the magnitude

of individual paths or taps will be independent of another, whereas using integer representations allows for a simpler implementation. If fractional representation is used, it will be done with fixed-point arithmetic as the time complexity of floating point calculations is too great and the potential benefits are inconsequential. The Loop Filter output may be applied directly to the oscillator, or an intermediate calculation done to set a bias point.

3.6.1 FIR Loop Filter

The FIR implementation of the Loop Filter will take the form presented in Figure 3.10 (a), and consists series of time delays, or "taps", the output of which is multiplied by a coefficient and the result of each multiplier added together to form the filter output. Various tools, Matlab for example, can be used to compute the coefficients and number of taps required to obtain a specific filter response. Using an FIR filter is advantageous as it is always stable due to the lack of a feedback path, however a large number of taps are required to obtain a frequency response akin to that of the IIR implementation, thus requiring an equal number of multipliers and $taps - 1$ adders. Compared to an IIR, which requires just a two adders, two multipliers and a flip-flop this has a much greater hardware footprint. It is possible to condense the FIR multiplication and addition to a single multiplier and adder, depicted in 3.10 (b), however this will take as many clock cycles to complete as there are taps. This, however, is not viable in this system as the filter must produce its output in a single clock cycle. In the case of any coefficient symmetry, addition can be performed prior to multiplication, thus reducing the number of multipliers for each identical coefficient.

The filter can be implemented using fixed-point arithmetic, in which the designer must ensure alignment of the radix in additions and correct radix position in multiplication results. Consequently the fractional width of the each multiplication need not be the same, most commonly used when the proportional gain, the output of the 0th tap multiplication, is significantly larger than that of the other taps. Integer arithmetic instead assumes all calculations are carried out using integers, and output of the addition stage is divided by a large number to form the module output. The integer approach simplifies the task of the designer and reduces the likelihood of incorrect calculations due to misalignment.
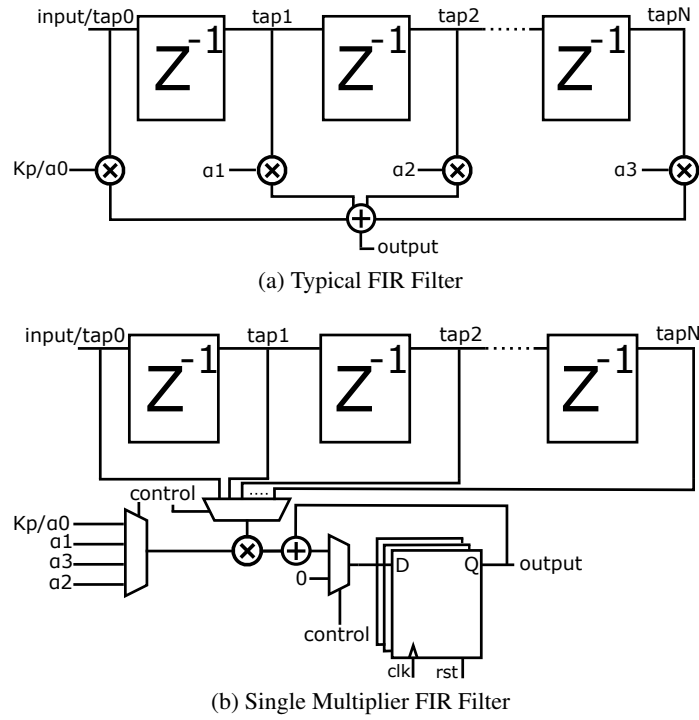
(a) Typical FIR Filter

(b) Single Multiplier FIR Filter

Figure 3.10: FIR PI loop filter RTL diagrams.

3.6.2 IIR Loop Filter

A PI controller can be easily implemented as an IIR filter in hardware, as in Figure 3.11, requiring significantly fewer circuit elements than an FIR based design as the long series of taps and multipliers is not required, as instead integration is performed by a single multiplier and an accumulator. As before the frequency response of the controller can be calculated using the following equation, where α is the proportional and β the integral gain:

$$H(z) = \alpha + \beta \frac{1}{1 - z^{-1}} = \frac{(\alpha + \beta) - \alpha z^{-1}}{1 - z^{-1}}$$

As the scaling factor used in this multiplier is the k_i gain itself, reconfiguration of the integral path of the Loop Filter can be carried out solely by varying this gain, as opposed to the FIR solution where the value of each tap must be recalculated to change the response of the system. The combination of reconfigurability and simpler hardware make the IIR the more desirable LF design.

Again both integer and fixed-point methods of LF design are possible, with the integer method requiring logic combining the two paths to preserve the relationship between the proportional and
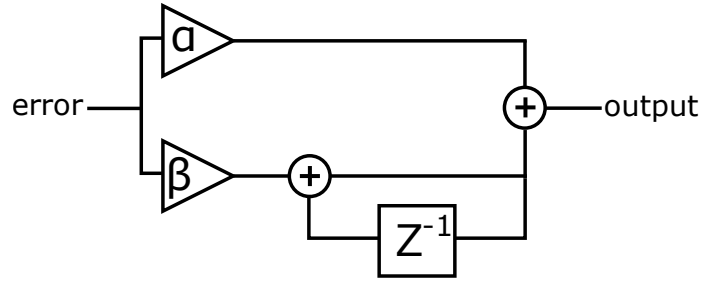
Figure 3.11: Basic IIR PI loop filter.

integral gains, which from the work of Koskin *et al* is known to be several binary orders of magnitude [15]. Failure to preserve a relationship of this order may lead to the filter becoming unstable, as in the upper-left region of Figure 3.12. In the fixed-point system the relationship between k_p and k_i is preserved during the combination of proportional and integral paths as both values have their radix point aligned before addition is carried out.
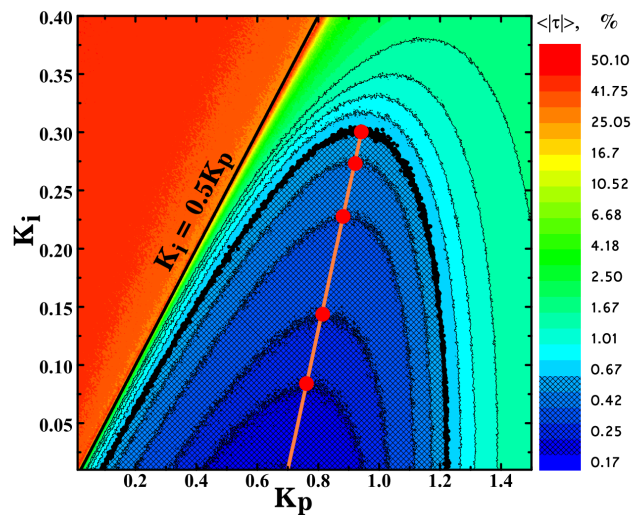

Figure 3.12: Stability of Loop Filter gains [15].

Bibliography

[1] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965. 3

[2] P. E. Ross, "Why cpu frequency stalled," *IEEE Spectrum*, vol. 45, no. 4, 2008. 3

[3] "Intel ark," https://ark.intel.com/content/www/us/en/ark.html, accessed 2019-04-06. 3

[4] E. Zianbetov, "Distributed clocking for synchronous soc," Ph.D. dissertation, Doctoral School of Informatics, Telecommunications and Electronics, UPMC, 4 Place Jussieu, 75005 Paris, France, 3 2013. 4, 5, 7, 11, 16, 20, 22, 31

[5] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez, "Reducing power in high-performance microprocessors," in *Proceedings of the 35th annual Design Automation Conference*. ACM, 1998, pp. 732–737. 4, 6

[6] A. Abdelhadi, R. Ginosar, A. Kolodny, and E. G. Friedman, "Timing-driven variation-aware nonuniform clock mesh synthesis," in *Proceedings of the 20th symposium on Great lakes symposium on VLSI*. ACM, 2010, pp. 15–20. 6

[7] G. Chen, H. Chen, M. Haurylau, N. A. Nelson, D. H. Albonesi, P. M. Fauchet, and E. G. Friedman, "On-chip copper-based vs. optical interconnects: Delay uncertainty, latency, power, and bandwidth density comparative predictions," in *2006 International Interconnect Technology Conference*. IEEE, 2006, pp. 39–41. 7

[8] T. Yamashita, T. Fujimoto, and K. Ishibashi, "A dynamic clock skew compensation circuit technique for low power clock distribution," in *Integrated Circuit Design and Technology, 2005. ICICDT 2005. 2005 International Conference on*. IEEE, 2005, pp. 7–10. 8

[9] H. Mizuno and K. Ishibashi, "A noise-immune ghz-clock distribution scheme using synchronous distributed oscillators," in *Solid-State Circuits Conference, 1998. Digest of Technical Papers. 1998 IEEE International*. IEEE, 1998, pp. 404–405. 9

[10] G. A. Pratt and J. Nguyen, "Distributed synchronous clocking," *IEEE transactions on parallel and distributed systems*, vol. 6, no. 3, pp. 314–328, 1995. 10, 11

[11] V. Gutnik and A. Chandrakasan, "Active ghz clock network using distributed plls," in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*. IEEE, 2000, pp. 174–175. 10

[12] E. Zianbetov, D. Galayko, F. Anceau, M. Javidan, C. Shan, O. Billoint, A. Korniienko, E. Colinet, G. Scorletti, J. Akrea *et al.*, "Distributed clock generator for synchronous soc using adpll network," in *Custom Integrated Circuits Conference (CICC), 2013 IEEE*. IEEE, 2013, pp. 1–4. 11, 12

[13] C. Shan, "Distributed clocking for large synchronous soc," Ph.D. dissertation, Doctoral School of Informatics, Telecommunications and Electronics, UPMC, 4 Place Jussieu, 75005 Paris, France, 10 2014. 11, 14, 15, 16, 22, 31

[14] Y. Chen and K. D. Pedrotti, "Rotary traveling-wave oscillators, analysis and simulation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 1, pp. 77–87, 2011. 13

[15] E. Koskin, D. Galayko, O. Feely, and E. Blokhina, "Generation of a clocking signal in synchronized all-digital pll networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 6, pp. 809–813, 2018. 15, 36

[16] E. Koskin, P. Bisiaux, D. Galayko, and E. Blokhina, "All-digital phase-locked loop arrays: Investigation of synchronisation and jitter performance through fpga prototyping," submitted. 17, 24

[17] A. Fernández-Álvarez, M. Portela-García, M. García-Valderas, J. López, and M. Sanz, "Hw/sw co-simulation system for enhancing hardware-in-the-loop of power converter digital controllers," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 5, no. 4, pp. 1779–1786, 2017. 17

[18] O. Lucia, I. Urriza, L. A. Barragan, D. Navarro, O. Jimenez, and J. M. Burdio, "Real-time fpga-based hardware-in-the-loop simulation test bench applied to multiple-output power converters," *IEEE Transactions on Industry Applications*, vol. 47, no. 2, pp. 853–860, 2011. 17

[19] G. Wang and Y. Chiu, "Fast fpga emulation of background-calibrated sar adc with internal redundancy dithering," in *Proceedings of the IEEE 2013 Custom Integrated Circuits Conference*. IEEE, 2013, pp. 1–4. 18

[20] "Clock jitter definitions and measurement methods, *SiT-AN10007 Rev 1.2*," https://www.sitime.com/api/gated/AN10007-Jitter-and-measurement.pdf, accessed 2019-04-10. 18

[21] M. Kellermann, *Creating a Controllable Oscillator Using the Virtex-5 FPGA IODELAY Primitive*, 1st ed., Xilinx, Internet, XAPP872, 4 2009. 20

[22] "File:ring oscillator (3-stage).svg," https://en.wikipedia.org/wiki/File:Ring_oscillator_(3-stage).svg, accessed 2019-04-10. 23

[23] E. Koskin, "Mathematical modelling and system-level design of all-digital phase-locked loop networks," Ph.D. dissertation, School of Electrical and Electronic Engineering, University College Dublin, Belfield, Dublin 4, Ireland, 2019. 24

[24] J. A. Tierno, A. V. Rylyakov, and D. J. Friedman, "A wide power supply range, wide tuning range, all static cmos all digital pll in 65 nm soi," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 42–51, 2008. 31

[25] M. Javidan, E. Zianbetov, F. Anceau, D. Galayko, A. Korniienko, E. Colinet, G. Scorletti, J.-M. Akre, and J. Juillard, "All-digital pll array provides reliable distributed clock for socs," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. IEEE, 2011, pp. 2589–2592. 33