A Rotary Travelling Wave Oscillator Based All-Digital PLL in 65nm CMOS

by

Eric Cathcart

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the requirements for the degree of

Master of Applied Science

In

Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Electronics

Carleton University

Ottawa, Ontario

© 2017

Eric Cathcart

Abstract

This thesis presents the design and implementation of an All-Digital Phase Locked Loop (ADPLL) that uses Delta-Sigma ($\Delta\Sigma$) modulation, multi-phase outputs, and Dynamic Element Matching (DEM). The system is designed using a combination of 65nm CMOS technology and an FPGA. The frequency range of the ADPLL output is 5.48GHz to 6.22GHz. Several design techniques are used to reduce the phase noise of the ADPLL output. The ADPLL uses a rotary travelling wave-based Digitally Controlled Oscillator (DCO) with multi-phase outputs to improve quantization noise. Manufacturing variations on the fine-tuning DCO input capacitors are averaged using DEM to produce more uniform frequency steps. $\Delta\Sigma$ modulation is used on the least significant of the DCO input bits. This modulation introduces a dithering effect on the output frequency that has the effect of moving some of the phase noise away from the ADPLL carrier frequency.

Acknowledgements

First, I would like to express my gratitude to my supervisor, Dr. Ralph Mason, for all of the time he has spent on all of the various stages of this work and for sharing his knowledge and experience. This thesis would not be possible without his help in designing and testing the chip, as well as the funding that he provided to make the fabrication of the chip possible.

I greatly appreciate the contributions that my colleague Xing Zhou has made to this device. His work on the chip layout, the PCB layout, and many of the key components in the system were instrumental in the creation of our device. I would like to recognize all of his hard work and effort in both the design and testing processes.

I would like to thank Nagui Mikhail for graciously providing us with the hardware necessary to complete our design as well as the equipment and lab space required to test our device. I would also like to thank Blazenka Power and Anna Lee for all of the administrative help they kindly provided to me during my time at Carleton University.

I would like to acknowledge the support received from Rohde & Schwarz for loaning us two key pieces of equipment that greatly helped during testing. I would also like to acknowledge support from CMC Microsystems for granting us access to fabrication facilities and various test equipment loans.

Finally, I would like to thank my family for all of the support and encouragement throughout my years in the Master's program.

Table of Contents

Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Tables	vii
List of Figures	viii
List of Abbreviations	xiii
Chapter 1 : Introduction	1
1.1 Chapter Overview	1
1.2 Motivation.....	1
1.3 Thesis Focus	3
1.4 Contributions	4
1.5 Thesis Organization	5
Chapter 2 : PLL Overview	6
2.1 Introduction.....	6
2.2 Conventional Analog PLL	6
2.3 All-Digital PLL.....	14
2.4 Phase Noise in an ADPLL	20
2.5 Additional Noise Considerations	33

Chapter 3 : Design and Simulation Results **36**

3.1 Overview	36
3.2 ASIC	41
3.2.1 Counter.....	41
3.2.1.1 Johnson Counter.....	43
3.2.1.2 Synchronous Counter.....	46
3.2.2 Bubble Correction	50
3.2.3 DCO Control System	50
3.2.3.1 PVT Control.....	51
3.2.3.2 Acquisition Control.....	52
3.2.3.3 Tracking Control.....	54
3.2.3.4 $\Delta\Sigma$ Modulator.....	57
3.2.4 Digitally Controlled Oscillator.....	60
3.2.5 Clock Retiming and Delay	64
3.3 FPGA	71
3.3.1 Integer Count Decoder	71
3.3.2 Fractional Lookup Table.....	76
3.3.3 Phase Detection.....	78
3.3.4 Digital Loop Filter	81
3.3.4.1 IIR Filter.....	81

3.3.4.2 PI Controller.....	85
3.4 Additional Supporting Components	90
3.4.1 Configuration Modules	91
3.4.2 Compression Modules	95
3.4.3 Verilog DCO Simulation Model.....	96
Chapter 4 : Measurement Results	99
4.1 Overview and Test Bench Setup.....	99
4.2 IC Physical Measurements.....	102
4.3 Open Loop Measurements	105
4.4 Closed Loop Measurements.....	110
4.5 Performance Summary	117
Chapter 5 : Conclusion	120
5.1 Accomplishments.....	120
5.2 Issues in the Design	121
5.3 Future Work	122
Appendix A : Additional Figures	125
Bibliography	129

List of Tables

Table 3.1: Johnson counter clock-to-data-in delays	44
Table 3.2: Synchronous counter clock-to-data-in delays.....	48
Table 3.3: Summary of delay line simulation results	68
Table 3.4: Johnson counter decoding patterns	73
Table 3.5: A selection of fractional count output codes	77
Table 3.6: Poles and zeros of the IIR filter	85
Table 3.7: I/O pin requirement reductions	90
Table 3.8: A sample of DCO data and the corresponding compressed representation.....	96
Table 4.1: Voltages and currents of components on the IC	102
Table 4.2: Key dimensions of the IC	102
Table 4.3: Summary of ADPLL performance	118
Table 4.4: Performance comparison against other ADPLLs	118

List of Figures

Figure 2.1: Basic block diagram of a PLL	7
Figure 2.2: VCO control voltage and output frequency	8
Figure 2.3: Example of VCO clock division for $N=6$	9
Figure 2.4: Phase detector implementation.....	10
Figure 2.5: Behaviour of a phase detector when $f_{REF} > f_{DIV_VCO}$	11
Figure 2.6: Charge pump connected to phase detector	12
Figure 2.7: Implementation of a passive low-pass loop filter.....	12
Figure 2.8: Core components of an ADPLL	15
Figure 2.9: DCO control word and output frequency	16
Figure 2.10: Sample behaviour of the integer and fractional counter outputs.....	17
Figure 2.11: P and PI Controller block diagrams and the expected resulting phase errors	20
Figure 2.12: Ideal output in (A) frequency domain and (B) time domain; noisy output in (C) frequency domain and (D) time domain.....	21
Figure 2.13: Transfer function of the IIR low-pass filter.....	23
Figure 2.14: Open loop transfer function of the ADPLL.....	24
Figure 2.15: Closed loop transfer function of the ADPLL for the reference clock path ..	25
Figure 2.16: Closed loop transfer function of the ADPLL for the PDC path.....	26
Figure 2.17: Closed loop transfer function of the ADPLL for the DCO path	27
Figure 2.18: PDC quantization noise contribution	28
Figure 2.19: Reference clock noise contribution	29
Figure 2.20: Simulated DCO phase noise.....	30

Figure 2.21: DCO noise contribution.....	30
Figure 2.22: $\Delta\Sigma$ modulator noise contribution.....	32
Figure 2.23: Total estimated phase noise of the ADPLL output	33
Figure 2.24: Expanded ADPLL block diagram	34
Figure 2.25: Bubble correction operation	35
Figure 3.1: Connections between the different parts of the full system	37
Figure 3.2: Block diagram of the FPGA portion of the ADPLL system	38
Figure 3.3: Block diagram of the IC portion of the ADPLL system	39
Figure 3.4: IC block diagram illustrating digital and analog design sections.....	40
Figure 3.5: Synchronous counter using T flip-flops with parallel carry.....	41
Figure 3.6: Counter block diagram	42
Figure 3.7: Johnson counter schematic	43
Figure 3.8: Post-layout Johnson counter output pattern	45
Figure 3.9: Synchronous counter schematic	46
Figure 3.10: Master-slave JK flip-flop schematic.....	46
Figure 3.11: Synchronous counter output waveform.....	49
Figure 3.12: Block diagram of the DCO control system	51
Figure 3.13: Binary to thermometer code expansion for acquisition bits.....	52
Figure 3.14: Balanced acquisition bit distribution around the DCO ring	53
Figure 3.15: Tracking bit cyclic shifting within the active row over four clock cycles ...	55
Figure 3.16: Verilog simulation result of tracking bit control block	56
Figure 3.17: Tracking bit control system block diagram	57
Figure 3.18: Delta-Sigma Modulator with MASH-3 topology.....	58

Figure 3.19: Delta-Sigma Modulator combiner block structure	58
Figure 3.20: Simulation of the tracking part of the DCO tuning word (A) without $\Delta\Sigma$ modulation and (B) with $\Delta\Sigma$ modulation	60
Figure 3.21: RTWO consisting of 128 transmission line segments (from [24])	62
Figure 3.22: Unfolded RTWO with inverter pairs and capacitance banks (from [14])	63
Figure 3.23: A single capacitor bank and inverter pairing (modified from [24])	64
Figure 3.24: Schematic of the clock retiming flip-flops	65
Figure 3.25: Simulation of the clock retiming circuit.....	65
Figure 3.26: Block diagram of the clock retiming system including delay lines	66
Figure 3.27: Schematic of one delay line with each section marked by dashed boxes	66
Figure 3.28: Delay line simulation results illustrating delay times	67
Figure 3.29: Timing differences between data transitions and rising retime clock edges	70
Figure 3.30: Finding the zero pattern based on the Johnson bits and a synchronous bit..	73
Figure 3.31: (A) 11 bit integer counter output and (B) 12 bit integer counter output	75
Figure 3.32: Phase error output with (A) 11 bit integer counter and (B) 12 bit integer counter.....	75
Figure 3.33: Simplified DCO with a selection of output bits indicated.....	77
Figure 3.34: Phase detection block diagram	78
Figure 3.35: Phase error pre- and post-deglitching.....	80
Figure 3.36: (A) Reference and DCO phase inputs and (B) phase error	81
Figure 3.37: (A) Magnitude and (B) phase response of the IIR	83
Figure 3.38: IIR pole/zero plot.....	83
Figure 3.39: Block diagram of the PI controller	86

Figure 3.40: K_P sweep simulation results showing the phase error with K_I set to zero....	88
Figure 3.41: K_I sweep simulation results showing the phase error with K_P set to 2^{-4}	89
Figure 3.42: Configuration modules between the FPGA and the IC	93
Figure 3.43: Simulation waveforms of a configuration module pairing.....	94
Figure 3.44: Annotated illustration of the Verilog DCO model fork/join_none operation	98
Figure 4.1: Test bench setup	100
Figure 4.2: Voltage level shifters between the FPGA and the IC.....	100
Figure 4.3: Annotated screenshot of the IC layout	103
Figure 4.4: Micrograph of the IC	104
Figure 4.5: Open loop phase noise of the DCO at 5.7GHz.....	106
Figure 4.6: DCO output in the time domain	107
Figure 4.7: Measured phase noise of the 100MHz reference clock.....	108
Figure 4.8: Counter data with (A) incorrect delay setting and (B) correct delay setting	109
Figure 4.9: Simulated phase error	110
Figure 4.10: Measured phase error	111
Figure 4.11: ADPLL output spectrum from 4GHz to 24GHz with harmonics labeled..	112
Figure 4.12: ADPLL output spectrum from 4GHz to 14GHz with harmonics labeled..	113
Figure 4.13: ADPLL output spectrum from 5.66GHz to 5.74GHz	113
Figure 4.14: ADPLL phase noise at 5.7GHz	114
Figure 4.15: Hybrid phase noise profile combining closed and open loop measurements	116

Figure A.1: K_I sweep simulation results showing the phase error with K_P set to 2^{-5} 125

Figure A.2: K_I sweep simulation results showing the phase error with K_P set to 2^{-6} 126

Figure A.3: K_I sweep simulation results showing the phase error with K_P set to 2^{-7} 127

Figure A.4: K_I sweep simulation results showing the phase error with K_P set to 2^{-8} 128

List of Abbreviations

$\Delta\Sigma$	Delta-Sigma
ADC	Analog-to-Digital Converter
ADPLL	All-Digital Phase-Locked Loop
BB	BaseBand
BFM	Banerjee Figure of Merit
BIBO	Bounded-Input, Bounded-Output
CMOS	Complementary Metal-Oxide-Semiconductor
DAC	Digital-to-Analog Converter
DC	Direct Current
DCO	Digitally Controlled Oscillator
DEM	Dynamic Element Matching
DPLL	Digital Phase-Locked Loop
DUT	Device Under Test
FCW	Frequency Control Word
FDA	Filter Design and Analysis
FIR	Finite Impulse Response
FoM	Figure of Merit
FPGA	Field-Programmable Gate Array
GPIO	General Purpose Input/Output
HDL	Hardware Description Language
I/O	Input/Output
IC	Integrated Circuit
IF	Intermediate Frequency
IIR	Infinite Impulse Response
LSB	Least Significant Bit
OTW	Oscillator Tuning Word
PCB	Printed Circuit Board
PDC	Phase-to-Digital Converter
PI	Proportional-Integral
PLL	Phase-Locked Loop
PVT	Process, Voltage, and Temperature
RF	Radio Frequency
RMS	Root Mean Square
RPi	Raspberry Pi
RTWO	Rotary Travelling Wave Oscillator
SMA	SubMiniature version A
TSMC	Taiwan Semiconductor Manufacturing Company
USB	Universal Serial Bus
VCO	Voltage Controlled Oscillator

Chapter 1: Introduction

1.1 Chapter Overview

This chapter introduces the motivation, focus, and contributions of this thesis. The end of the chapter also outlines the thesis organization.

1.2 Motivation

In recent years, wireless devices have become increasingly common in everyday life. Wireless devices such as radios, television receivers, mobile phones, and all manner of Wi-Fi-enabled devices commonly use Phase Locked Loops (PLLs) as frequency synthesizers. Traditionally, the PLLs being used for wireless applications are analog devices; however, efforts are being made to develop low cost, low power, low voltage, fully integrated replacements for these analog devices.

As CMOS processes allow for smaller component features and lower voltages, it becomes increasingly difficult to design appropriate analog PLLs. With smaller components, small manufacturing variations become more significant relative to the overall component sizes in an integrated system. With lower voltages, small amounts of electrical noise on signal lines have a greater impact on the performance of the device as the amplitude of the noise becomes a more significant percentage of the amplitude of the desired signal. As such, certain analog components in a design may have to be moved off-chip, which increases the total cost of the PLL and prevents it from being a fully-integrated system. Another challenge of analog designs is that they are not easily portable between different process technologies, which may result in a full redesign being required if the process technology changes.

Creating an all-digital version of the PLL allows designers to take advantage of the low power and low voltage capabilities of modern CMOS while avoiding the design difficulties of integrated analog devices and the costs of off-chip analog devices. In an All-Digital Phase Locked Loop (ADPLL), all of the components of the PLL are replaced by digital equivalents and are integrated on-chip. All of the analog signals within the PLL are also replaced by buses of digital bits.

Another advantage of an all-digital implementation is that much of the system can be designed using a Hardware Description Language (HDL) which allows a designer to describe the desired behaviour of each component within the system, rather than designing the specific implementation of the components. The specific implementation, including placement and routing, can then be automated using software tools. This allows for a fast redesign of the layout if the system requirements change. In addition, since HDL syntax is independent of process technology, a digital design made in this manner may be ported to different process technologies if required, often without requiring any changes to the HDL coding.

At the core of the ADPLL is a Digitally Controlled Oscillator (DCO) based on a Rotary Traveling Wave Oscillator (RTWO) structure, which tends to have better phase noise than more conventional oscillator structures [1]. RTWO structures lend themselves well to having multi-phase outputs. This is achieved by having many output points on the ring at different locations in the layout. The multi-phase outputs are used to gain a finer phase resolution on the DCO output signal. This, in turn, reduces the quantization noise of the DCO.

The DCO fine-tuning inputs are provided by unit-weighted capacitors that can be activated or deactivated by a set of digital bits to control the output frequency. These capacitors make use of a Dynamic Element Matching (DEM) technique to reduce the impact of process variation on the frequency step size of the DCO. The DEM system reduces the impact of process variation by ensuring that the input of the DCO is impacted by capacitors from all regions of the layout, rather than concentrating the active capacitors in any one region.

The least significant bits of the aforementioned inputs are also subject to Delta-Sigma ($\Delta\Sigma$) modulated dithering. By dithering these bits, an element of randomness is introduced to the oscillator control. The $\Delta\Sigma$ modulation reduces the phase noise near the carrier frequency of the ADPLL by pushing noise away from that frequency and also provides an improved frequency resolution through its dithering effect on the control bits.

1.3 Thesis Focus

The focus of this thesis is to create an ADPLL with low phase noise using digital standard cell library components. This design attempts to improve phase noise performance by combining various noise reduction techniques implemented in the digital domain. These noise reduction techniques include a multi-phase output on the DCO, the use of Dynamic Element Matching (DEM) on the fine frequency DCO control bits, Delta-Sigma dithering on the lowest bits of the DCO control word, and a balanced activation sequence for the DCO control bits which compensates for Process, Voltage, and Temperature (PVT) variation across the layout of the oscillator. Although the focus of this thesis is on the standard cell digital components of the PLL, the basic design of the DCO will also be presented.

1.4 Contributions

The major research contributions of this thesis are as follows:

1. The implementation of the digital core of a low phase noise ADPLL. A combination of noise reduction techniques are used to improve noise performance, such as DEM, $\Delta\Sigma$ modulated dithering, and multi-phase DCO outputs.
2. A flexible implementation that combines the use of Integrated Circuit (IC) logic and an FPGA. The use of an FPGA allows the lower frequency parts of the digital logic to be moved off chip. By placing a significant portion of the logic on the FPGA, the PLL can be modified as required during testing. However, the connections between the FPGA and the IC are limited by the number of pins on the IC package. This thesis also presents systems to compress the data transfer between the two parts of the PLL to solve this problem.
3. The use of standard cell libraries with automated synthesis and layout tools for the digital components of the system so that the digital design is not restricted to the process technology used in this thesis.
4. A new high-speed counter structure which enables the design of a standard cell-based counter capable of counting at the DCO clock speed of more than 6 GHz. This counter design achieves a balance between having a fast counting speed and having a small layout area.
5. The presentation of a complete ADPLL system including the components that are not part of the digital core, such as the RTWO.

6. The creation of a written tutorial detailing how to use the software kits available at Carleton University at the time of this writing to make a complete digital design with the TSMC 65nm digital standard cell library.

This tutorial contributed to the writing of two theses.

1.5 Thesis Organization

The current chapter, Chapter 1, introduces the ADPLL and some of its applications.

Chapter 2 gives an overview of the theory of PLLs starting with an analysis of a conventional analog PLL. Following that, an analysis of the ADPLL is provided along with an analysis of the phase noise of the ADPLL.

Chapter 3 presents the design and simulation of the ADPLL and all of its components. This section first describes the components found on the IC, then the components found on the FPGA, and finally the components that are either split between or are external to both the IC and the FPGA.

Chapter 4 contains the results of tests and measurements. The results are compared with simulation results where possible.

Chapter 5 presents the conclusions of this work as well as some recommendations for future work related to this thesis.

Chapter 2: PLL Overview

2.1 Introduction

Frequency synthesizers are commonly used in radio applications as Local Oscillators (LOs) in the mixing stages of radio transmitters and receivers [2], [3]. In such applications, a synthesizer can be used with a mixer to mix signals to and from the BaseBand (BB) frequency, Intermediate Frequency (IF), and Radio Frequency (RF) within the transceiver. One way of implementing a frequency synthesizer is with a Phase-Locked Loop (PLL). Traditionally, two common types of PLL are the Integer-N PLL and the Fractional-N PLL, both of which use analog techniques in their design and implementation. More recently, as technology progresses further towards deep submicron processes, it has become more desirable to make use of an All-Digital Phase Locked Loop (ADPLL) as analog components such as filters and charge pumps become increasingly difficult to design and implement in an integrated system [3], [4]. The design of an ADPLL seeks to replace all of the analog components of a conventional PLL with their digital equivalents.

In this chapter, the design of a conventional PLL and its components will be discussed and analyzed. Following that, an ADPLL will be realized by replacing the components of the conventional PLL with their digital counterparts.

2.2 Conventional Analog PLL

A conventional PLL consists of a feedback loop which controls the output frequency of a Voltage Controlled Oscillator (VCO). The PLL has four basic components: a VCO, a clock divider, a phase detector, and a loop filter. These components are arranged as shown in Figure 2.1. The PLL functions by comparing a

divided version of the VCO output signal against a reference signal, typically provided by a crystal oscillator, and adjusting the VCO control signal until the two compared signals match [3]. When the loop is settled, the output frequency is given by Equation (2.1).

$$f_{OUT} = f_{REF} \times N \quad (2.1)$$

In a conventional PLL, the value of N is an integer [2], [5]. Because N is an integer in this type of PLL, the minimum output frequency step size is equal to the reference clock frequency. To achieve a finer frequency step size in an Integer- N PLL, the reference clock frequency would have to be reduced; however, this would result in a narrower loop bandwidth, longer switching times, and worse suppression of VCO noise [5].

Figure 2.1: Basic block diagram of a PLL

The VCO at the core of the PLL receives an input control voltage, V_{TUNE} , and outputs a corresponding output frequency, f_{OUT} . A VCO has a minimum and maximum frequency range and a corresponding minimum and maximum control voltage, as indicated in Figure 2.2. Although a VCO does not normally have a linear response as illustrated, it can be assumed to be linear for the purposes of a system analysis [3]. In addition, for the purposes of analysis, the frequencies and voltages in the VCO are considered in terms of their deviations from a nominal frequency, f_{NOM} , and its corresponding control voltage, V_{NOM} . The rate of change of the frequency output relative

to the voltage input is listed as K_{VCO} . The output frequency of the VCO can be written in terms of rad/s as shown in Equation (2.2).

Figure 2.2: VCO control voltage and output frequency

$$\omega_{OUT} = 2\pi f_{OUT} = (K_{VCO})(V_{TUNE}) \quad (2.2)$$

Since the output signal of the VCO will be passed to a phase detector after being divided by the clock divider, it is necessary to have an expression for the phase of the VCO signal. Given that frequency is the derivative of phase with respect to time [6], the phase of the output signal can be written as the integral of the frequency as in Equation (2.3).

$$\theta_{OUT} = \int (\omega_{OUT} \cdot dt) = \int (K_{VCO} V_{TUNE} \cdot dt) = K_{VCO} \int (V_{TUNE} \cdot dt) \quad (2.3)$$

Rearranging the above equation to give the transfer function of the VCO, $\frac{\theta_{OUT}}{V_{TUNE}}$, and converting it to the Laplace domain results in Equation (2.4).

$$\frac{\theta_{OUT}(s)}{V_{TUNE}(s)} = \frac{K_{VCO}}{s} \quad (2.4)$$

The VCO output is passed to the clock divider which produces a lower frequency signal. An example of such clock division is shown in Figure 2.3 for an arbitrarily chosen $N=6$. As mentioned previously, the value of N in a conventional PLL is an integer.

Alternately, the frequency divider in a conventional PLL can be replaced with a divider capable of using fractional division ratios to make a Fractional-N PLL. Doing so allows for output frequency step sizes that are smaller than the reference clock frequency. Several methods exist to create PLL frequency dividers [5], [7], [8], [9], [10]; however, the specific architecture of the divider does not impact this analysis. When the clock divider is connected to the VCO, the combined transfer function between the tuning voltage and the phase of the divided clock signal is as in Equation (2.5). The addition of the divider changes the transfer function in Equation (2.4) by introducing a $\frac{1}{N}$ term [3].

Figure 2.3: Example of VCO clock division for N=6

$$\frac{\theta_{DIV}(s)}{V_{TUNE}(s)} = \frac{1}{N} \cdot \frac{K_{VCO}}{s} \quad (2.5)$$

As indicated in Figure 2.1, the divided VCO output is passed to the phase detector along with a reference clock signal. The output of the phase detector indicates whether the reference clock is at a higher frequency than the divided VCO clock or vice versa. One possible implementation of a phase detector, as described in [11], is shown in Figure 2.4.

Figure 2.4: Phase detector implementation

If the reference frequency is higher than the divided VCO frequency, then the Up command will be asserted because the reference clock rising edge will arrive at the top flip-flop before the divided VCO clock rising edge reaches the bottom flip-flop. This assertion indicates that the frequency of the VCO should be increased. The Up command remains asserted until the divided VCO clock arrives at the bottom flip-flop at which time the Down command is asserted briefly. With both the Up and Down commands asserted simultaneously, the AND logic gate outputs a high signal which resets both flip-flops to low output states. The width of the Down signal before the reset is enacted, and therefore the length of time during which both commands are asserted, will depend on the delay of the circuit logic.

If the reference frequency is lower than the divided VCO frequency, then the divided VCO clock edge will arrive at the bottom flip-flop before the reference clock edge arrives at the top flip-flop. This situation results in the Down command being asserted which indicates that the VCO frequency should be decreased. When the reference clock edge reaches the top flip-flop, the Up command is asserted briefly, which results in both flip-flops being reset to low output states.

As an example, the simulation result shown in Figure 2.5 illustrates the behaviour of the phase detector for a situation in which the reference clock frequency is higher than

the divided VCO clock frequency. As shown, when the reference clock rising edge arrives before that of the divided VCO clock, the Up command is asserted. The command remains until the divided VCO clock edge arrives, at which point the Down command is asserted briefly. Both commands remain asserted until the reset signal affects the output of both flip-flops by causing them to both output low signals.

Figure 2.5: Behaviour of a phase detector when $f_{REF} > f_{DIV_VCO}$

The phase detector from Figure 2.4 can be connected to a charge pump as shown in Figure 2.6. The purpose of the charge pump is to convert the voltage pulses produced by the phase detector into current pulses that can be used by the loop filter [2]. The output of this subsystem is determined by the difference in the phases of the two input clocks multiplied by the gain of the subsystem, K_{PHASE} . From [3], the average current output of this subsystem is

$$I_{CONTROL} = K_{PHASE}(\theta_{REF} - \theta_{DIV}) \quad (2.6)$$

where K_{PHASE} is defined as

$$K_{PHASE} = \frac{I_{SRC}}{2\pi} \quad (2.7)$$

and the current, I_{SRC} , is the current produced by one of the current sources in Figure 2.6 when turned on by one of the command signals.

Figure 2.6: Charge pump connected to phase detector

Following the charge pump is a low-pass filter. The purpose of the low-pass filter is to convert the current generated by the charge pump into a voltage that can be used to control the VCO. In addition, the low-pass filter serves to filter out large ripples in the control signal [12]. One simple implementation of a low-pass filter that is suitable for this application is shown in Figure 2.7 using only passive components.

Figure 2.7: Implementation of a passive low-pass loop filter

To determine the output voltage, V_{TUNE} , as a function of the input current, $I_{CONTROL}$, it is necessary to first determine the impedance or admittance of the loop filter. The impedance of the filter, shown in Equation (2.9), is used in conjunction with the current for the charge pump from Equation (2.6) to produce Equation (2.10) which shows the tuning voltage in terms of the phase difference at the input of the phase detector.

$$Y_F = Y_1 + Y_2$$

$$Y_F = \frac{1}{R_1 + \frac{1}{sC_1}} + sC_2$$

$$Y_F = \frac{sC_1}{1 + sC_1R_1} + sC_2$$

$$Y_F = \frac{sC_1 + sC_2(1 + sC_1R_1)}{1 + sC_1R_1} \quad (2.8)$$

$$Z_F = \frac{1}{Y_F} = \frac{1 + sC_1R_1}{s(C_1 + C_2) + s^2C_1C_2R_1} \quad (2.9)$$

$$V_{TUNE} = I_{CONTROL} \cdot Z_F$$

$$V_{TUNE} = \frac{K_{PHASE} \cdot (\theta_{REF} - \theta_{DIV}) \cdot (1 + sC_1R_1)}{s(C_1 + C_2) + s^2C_1C_2R_1} \quad (2.10)$$

The closed-loop transfer function of the system takes the general form shown in Equation (2.11) where $G(s)$ represents the open-loop transfer function of the system [13]. The open-loop transfer function combines the transfer functions for the individual components and condenses the loop filter transfer function into a generalized $F(s)$ to make the transfer function shown in Equation (2.12). Combining Equations (2.11) and (2.12) results in the closed loop transfer function shown in Equation (2.13).

$$H(s) = \frac{G(s)}{1 + G(s)} \quad (2.11)$$

$$G(s) = \frac{K_{PHASE} \cdot K_{VCO} \cdot F(s)}{s \cdot N} \quad (2.12)$$

$$H(s) = \frac{\left(\frac{K_{PHASE} \cdot K_{VCO} \cdot F(s)}{s \cdot N} \right)}{1 + \left(\frac{K_{PHASE} \cdot K_{VCO} \cdot F(s)}{s \cdot N} \right)} = \frac{K_{PHASE} \cdot K_{VCO} \cdot F(s)}{s \cdot N + K_{PHASE} \cdot K_{VCO} \cdot F(s)} \quad (2.13)$$

2.3 All-Digital PLL

In contrast to the conventional PLL described previously, an All-Digital Phase-Locked Loop (ADPLL) replaces all of the analog components with their digital equivalents. From R. E. Best's definition of the different PLL types in [6], in order for this system to be considered an ADPLL, the system must use only logical devices and the signals within the system must be binary signals. This is in contrast to the analog PLL described previously, which uses analog components and voltage levels. It is also in contrast to certain hybrid systems using mostly digital systems and a selection of analog components connected by Digital-to-Analog Converters (DAC) and Analog-to-Digital Converters (ADC), which Best calls the Digital Phase-Locked Loop (DPLL). Since the system described in this work uses entirely digital systems, it can be considered an ADPLL.

The all-digital implementation of the PLL is advantageous because it allows the system to be more easily integrated with other digital components on an IC by eliminating the need for analog subsystems such as the charge pump and the RC loop filter which may require off-chip resistors and capacitors [3], [14]. Since this is now a digital system, a reference clock signal is required for synchronously clocking the digital

components. A clock for this purpose was not required when all of the components operated in continuous time in the analog PLL. Such a reference clock is often supplied by a crystal oscillator [2].

The core components of the ADPLL are as shown in Figure 2.8 and will be discussed in general terms in this section. In Chapter 3, the specific implementation of these components and the other supporting components will be discussed.

Figure 2.8: Core components of an ADPLL

In an ADPLL, the VCO is replaced with a Digitally Controlled Oscillator (DCO). A DCO uses an array of digital inputs to control a bank of binary- and unit-weighted switchable capacitors, or varactors, of varying sizes [2], [14], [15], [16]. Because of the digital nature of the tuning varactor bank, the DCO has a finite minimum step size determined by the size of the smallest varactors. The array of input bits is known as the Oscillator Tuning Word (OTW). Although the DCO is not truly linear, similar to the VCO, it can be considered sufficiently linear for the purposes of analysis. The DCO used in this design has multiple outputs distributed throughout the oscillator that give information on the fractional phase of the waveform. These outputs are captured by a Phase-to-Digital Converter (PDC) and are sampled at the reference clock frequency. This information can be used by the fractional counter to improve the phase resolution of the system. More detail on the specific design of the oscillator is presented in Chapter 3 as well as in [1], [14], [15], and [17].

The output frequency of the DCO is a function of a nominal frequency, f_{NOM} , and an offset frequency Δf [2]. The offset frequency, in turn, is a function of the DCO gain, K_{DCO} , and the oscillator tuning word, OTW. The DCO gain corresponds to the change in frequency that results from changing the least significant bit of the input oscillator tuning word. The function for the output frequency is as shown in Equation (2.14) and an illustration of the output frequency as a function of the tuning word is shown in Figure 2.9.

$$f_{DCO} = f_{NOM} + \Delta f = f_{NOM} + (K_{DCO})(OTW) \quad (2.14)$$

Figure 2.9: DCO control word and output frequency

The divider from the analog PLL is replaced in the ADPLL by a digital counter. The counter consists of an integer component and a fractional component. The integer component uses a single tapping point on the DCO and increments with every complete DCO oscillation. The fractional component of the counter uses all of the available DCO tapping points to determine how much of an oscillation has occurred since the previous integer increment, which improves the phase resolution of the system. The improvement in the phase resolution is directly impacted by the number of tapping points on the oscillator – having more tapping points will result in a better phase resolution [14]. Due

to the digital nature of the system, it is more appropriate to consider the phase information as a unitless value rather than attempting to use radians as a measurement unit [2].

The integer and fractional counter values are appended to one another and are sampled by the reference clock to give a fixed point multi-bit digital representation of the DCO phase accumulated over time. The behaviour of the two counters is illustrated in Figure 2.10. In the figure, the DCO is assumed to have only eight tapping points for clarity. In addition, it is assumed that the integer counter is triggered on the rising edge of Tap 1.

Figure 2.10: Sample behaviour of the integer and fractional counter outputs

The phase detector, as mentioned in the discussion on the analog PLL, compares the signal received from the oscillator feedback path against a reference signal. Unlike in the case of the analog PLL, however, the signal on the oscillator feedback path (i.e. the signal received from the counter) is a digital word representing the oscillator's phase information. It is therefore necessary to create a signal that can be compared against this in order to determine the phase error of the system.

The reference phase signal is created by integrating a Frequency Control Word (FCW) over time using an accumulator. The frequency control word, which may be a fractional value, represents the ratio between the desired DCO frequency, f_{DCO} , and the reference clock frequency, f_{REF} , as indicated in Equation (2.15). The FCW is represented as a fixed-point value with integer and fractional components that are the same lengths as their counterparts from the counter outputs discussed previously. Because of the finite word length of the phase signals and because these signals are continually incrementing, the reference phase signal and the DCO phase signal will both overflow periodically.

$$FCW = \frac{f_{DCO}}{f_{REF}} \quad (2.15)$$

The phase detector subtracts the counter phase from the reference phase as per Equation (2.16) and outputs a unitless signed digital word representing the phase error. The sampled nature of the signals is represented by the index n in the equation. If the reference phase is higher than the counter phase, then the phase detector output will be a positive value. This indicates that the DCO is oscillating at a frequency that is lower than desired. If, instead, the reference phase is lower than the counter phase, then the result will be a negative value. In this case, the DCO frequency is too high and needs to be lowered for the counter phase to match the reference phase.

$$\phi_{ERROR}[n] = \phi_{REF}[n] - \phi_{DCO}[n] \quad (2.16)$$

The loop filter in an ADPLL typically consists of a digital low-pass filter and a digital controller. The purpose of the loop filter is similar to the analog PLL loop filter in that it filters high frequency spurs in the phase error signal. As is the case with the other subsystems of the ADPLL, both the input and the output of the loop filter take the form of digital words.

The specific form of the loop filter may vary from one system to the next as both the low-pass filter and the controller may be implemented in different methods. In [2], the low-pass filter is implemented as an Infinite Impulse Response (IIR) filter. The work presented in [14] instead uses a finite impulse response filter. The systems in [3] and [16] omit the low-pass filter entirely. In the case of the controller, the systems in [2], [3], and [14] use a PI controller, which is short for proportional-integral controller. In contrast, [16] uses a simpler P controller. The system described in this thesis uses an IIR low-pass filter and a PI controller.

A P controller provides an output that is directly proportional to the input with a gain factor α . Using a P controller results in a Type-I PLL (DCO-Based). A PI controller builds on the P controller by adding an integral response path. This addition results in a Type-II PLL. The proportional path, as before, has a gain factor α and the new integral path has a separate gain factor ρ which is typically smaller than α [2]. The end results of both paths are added together to produce the output of the controller. Due to the digital implementation of the PI controller, these gain factors are implemented as bit shift operations. Therefore, the available values for these factors are powers of two.

A Type-I PLL in a locked state will have some amount of static phase error whereas the phase error in a Type-II PLL under locked conditions will be zero [13]. See Figure 2.11 for a block diagram of the P controller and the PI controller. Figure 2.11 also illustrates the difference between the phase error signals of a PLL as the system achieves a locked state using both controller types. In the case of the P controller and the Type-I PLL, the phase error settles to some non-zero value, whereas in the case of the PI controller and the Type-II PLL, the phase error settles to zero.

Figure 2.11: P and PI Controller block diagrams and the expected resulting phase errors

2.4 Phase Noise in an ADPLL

Phase noise is an important factor in determining the performance of an oscillator. In an ideal situation, the oscillator would produce a pure output signal with no phase noise. However, in a real oscillator, there will be some amount of noise that introduces frequency components near the desired output frequency. In the frequency domain, the ideal output would appear as a narrow spike at the desired frequency output. The addition of phase noise to the output signal is represented in the frequency domain as a skirt centered on the desired output frequency [2]. In the time domain, noise in the output signal is represented by the waveform edges occurring slightly before or slightly after the expected transition times. Figure 2.12 illustrates the frequency and time domain representations of an ideal output signal and an output signal with phase noise.

Figure 2.12: Ideal output in (A) frequency domain and (B) time domain; noisy output in (C) frequency domain and (D) time domain

In the ADPLL designed for this thesis, there are four major sources of phase noise [14]. These four sources are the quantization noise of the PDC; the frequency resolution of the DCO control system, which is modified by a $\Delta\Sigma$ modulator; the DCO noise; and the reference clock noise. To determine the impact of each of these noise sources on the output signal, each noise source is multiplied by the closed loop transfer function of the system from the noise source to the output.

The open loop transfer function is as shown in Equation (2.17) [2]. As can be seen, the function is dependent on the PI controller parameters α and ρ , as well as the transfer function of the IIR filter. As will be discussed further in Chapter 3, α will be set to 2^{-4} and ρ will be set to 2^{-11} . The IIR filter used in the system has a transfer function as shown in Equation (2.18). The filter is an elliptic 3rd order low-pass with a 10MHz cutoff frequency and a 25dB stop band attenuation.

$$H_{OL}(s) = \frac{\rho \cdot (f_{REF})^2}{s} \frac{1 + \frac{s}{\rho \cdot f_{REF}/\alpha}}{s} \cdot F_{IIR}(s) \quad (2.17)$$

$$F_{IIR}(s) = \frac{0.06262 \cdot s^3 + 1.726 \cdot 10^7 \cdot s^2 + 6.773 \cdot 10^{14} \cdot s + 1.58 \cdot 10^{23}}{s^3 + 6.100 \cdot 10^7 \cdot s^2 + 4.911 \cdot 10^{15} \cdot s + 1.58 \cdot 10^{23}} \quad (2.18)$$

Although the IIR filter is actually implemented as a two-section filter using the Matlab Filter Design & Analysis Tool (FDA), it is equivalent to the single-section form which was obtained using the “Convert to Single Section” command within the FDA Tool. In addition, the form shown in the equation is the S-domain conversion of the filter’s Z-domain form obtained using the d2c() command, which is short for “discrete to continuous”, in Matlab. The transfer function of the single-section IIR filter is illustrated in Figure 2.13.

Figure 2.13: Transfer function of the IIR low-pass filter

The open loop transfer function of the system is shown in Figure 2.14. From the function, the phase margin of the system can be determined. The phase margin is given as in Equation (2.19), where Φ_{UG} is the phase of the transfer function at the unity gain frequency. If the phase is greater than -180° at the unity gain frequency, then the system is stable [6]. From the open loop transfer function, the phase margin of the system is 54.8° .

$$\phi_M = 180^\circ + \phi_{UG} \quad (2.19)$$

The loop bandwidth is given in Equation (2.20) [2]. Using 2^{-4} for α and 2^{-11} for p as before, the bandwidth of the system is 1.1MHz. The bandwidth can also be found visually as the 0dB crossing point of the open loop transfer function.

$$f_{BW} = (f_{REF}) \cdot \left(\frac{\alpha}{2\pi}\right) \cdot \left(0.5 + 0.5 \cdot \sqrt{1 + \frac{4 \cdot \rho}{\alpha^2}}\right) \quad (2.20)$$

Figure 2.14: Open loop transfer function of the ADPLL

Using the open loop transfer function, it is possible to derive the closed loop transfer function for the noise sources. The closed loop transfer function for the reference noise is a low-pass function as shown in Equation (2.21) and in Figure 2.15. As before, N is the ratio between the DCO output frequency and the reference clock frequency.

$$H_{CL,REF}(s) = N \cdot \frac{H_{OL}(s)}{1 + H_{OL}(s)} \quad (2.21)$$

Figure 2.15: Closed loop transfer function of the ADPLL for the reference clock path

The closed loop transfer function for the PDC quantization noise is a low-pass function shown below and illustrated in Figure 2.16. The bandwidth of the system can be found visually using the closed loop transfer function in addition to the methods discussed previously. In this case, the bandwidth is the 3dB cutoff point of the transfer function.

$$H_{CL,PDC}(s) = \frac{H_{OL}(s)}{1 + H_{OL}(s)} \quad (2.22)$$

Figure 2.16: Closed loop transfer function of the ADPLL for the PDC path

The closed loop transfer function for the DCO noise and for the DCO control system is a high pass function given in Equation (2.23) and shown in Figure 2.17.

$$H_{CL,DCO}(s) = \frac{1}{1 + H_{OL}(s)} \quad (2.23)$$

Figure 2.17: Closed loop transfer function of the ADPLL for the DCO path

According to [14], the phase variance for the PDC used in this design is

$$\sigma_{\phi}^2 = \frac{(\Delta\phi)^2}{12} \quad (2.24)$$

In the above equation, $\Delta\Phi$ is the phase resolution and is given by

$$\Delta\phi = \frac{\pi}{N_{TAP}} \quad (2.25)$$

The equation for $\Delta\Phi$ is derived from the number of physical tapping points on the oscillator and from the fact that one lap of a waveform edge around the ring corresponds to one half of an oscillation period. Assuming a Gaussian distribution of the phase noise, the PDC quantization noise is given as

$$L_{PDC} = \frac{\sigma_{\phi}^2}{f_{REF}} = \frac{(\Delta\phi)^2}{12 \cdot f_{REF}} = \frac{\pi^2}{12 \cdot f_{REF} \cdot (N_{TAP})^2} \quad (2.26)$$

As can be seen from the equation, the PDC noise can be reduced by increasing the reference frequency of the system and by increasing the number of tapping points on the oscillator. To get the total noise contribution of the PDC on the output of the system, the above noise equation is multiplied by the PDC closed loop transfer function from Equation (2.22). The PDC quantization noise is illustrated in Figure 2.18.

Figure 2.18: PDC quantization noise contribution

The reference clock used in this design is provided by a Crystek CCHD-575 oscillator. The phase noise profile of this device can be found at [18]. The phase noise of the reference clock is multiplied by the closed loop transfer function in Equation (2.21) to get the phase noise contribution illustrated in Figure 2.19.

Figure 2.19: Reference clock noise contribution

Simulation results for the DCO used in this design are shown in Figure 2.20 using a lumped element model. The design of the DCO, including schematics, will be presented in Section 3.2.4. The design and simulation of the DCO were done by another designer working on this system.

At a frequency of 5.7GHz, the phase noise of the free running DCO at a 1MHz offset is expected to be -112dBc/Hz. The DCO noise is multiplied by the DCO closed loop transfer function to generate the phase noise contribution shown in Figure 2.21.

Figure 2.20: Simulated DCO phase noise

Figure 2.21: DCO noise contribution

The noise contribution of the DCO control system results from the finite step size of the DCO control. This noise contribution consists of two parts: the contribution

derived from the smallest step size achievable by the least significant bit (LSB) of the oscillator tuning word and the impact of the $\Delta\Sigma$ modulation acting on five sub-LSBs [19]. The first contribution is given in Equation (2.27) and uses the step size of the smallest varactor, which is 19kHz, and the reference clock frequency, which is 100MHz. The second contribution is given in Equation (2.28). This contribution uses the frequency resolution provided by the $\Delta\Sigma$ dithering effect as given by Equation (2.29) [16]; the dithering clock frequency; and the order of the $\Delta\Sigma$ modulator, which is $n=3$ in this system. The frequency resolution provided by the dithering effect in Equation (2.29) uses the frequency resolution of 19kHz from the smallest varactor step size and the number of input bits to the $\Delta\Sigma$ modulator, which is $W=5$. The dithering clock frequency is set to one eighth of the approximately 6.5GHz DCO output frequency, which is 0.81GHz. When these two noise components are combined and multiplied by the DCO closed loop transfer function, they produce a noise contribution at the output shown in Figure 2.22.

$$L_{\Delta\Sigma 1}\{\Delta f\} = \frac{1}{12} \cdot \left(\frac{\Delta f_{RES}}{\Delta f}\right)^2 \cdot \frac{1}{f_{REF}} \cdot \left(\text{sinc}\left(\frac{\Delta f}{f_{REF}}\right)\right)^2 \quad (2.27)$$

$$L_{\Delta\Sigma 2}\{\Delta f\} = \frac{1}{12} \cdot \left(\frac{\Delta f_{RES, \Delta\Sigma}}{\Delta f}\right)^2 \cdot \frac{1}{f_{DTH}} \cdot \left(2 \cdot \sin\left(\frac{\pi \cdot \Delta f}{f_{DTH}}\right)\right)^{2n} \quad (2.28)$$

$$\Delta f_{RES, \Delta\Sigma} = \frac{\Delta f_{RES}}{2^W} \quad (2.29)$$

Figure 2.22: $\Delta\Sigma$ modulator noise contribution

When put together, the noise sources produce an expected phase noise plot as shown in Figure 2.23. The integrated Root Mean Square (RMS) jitter of the output is calculated by converting the single-sided phase noise plot to linear terms, integrating the area under the linearized curve, and applying Equation (2.30) [20]. From the plot of the total expected phase noise in Figure 2.23, the expected integrated RMS jitter of the output signal is 176.3fs.

$$RMS\ Jitter_{SECONDS} = \frac{RMS\ Jitter_{RADIANS}}{2 \cdot \pi \cdot f_{DCO}} = \frac{\sqrt{2 \cdot Area}}{2 \cdot \pi \cdot f_{DCO}} \quad (2.30)$$

Figure 2.23: Total estimated phase noise of the ADPLL output

2.5 Additional Noise Considerations

In addition to the phase noise considerations above, a number of design decisions were made with regards to specific subsystems in an effort to reduce the overall noise in the system. These subsystems are the DCO control system and the bubble corrector. Figure 2.24 expands on the block diagram shown in Figure 2.8 by adding the bubble corrector and the DCO control system. The $\Delta\Sigma$ modulator is included as part of the DCO control system. The figure also splits the counter into its fractional and integer components.

Figure 2.24: Expanded ADPLL block diagram

The DCO control system contains four groups of varactors that control the output frequency of the DCO. These groups are the PVT, acquisition, tracking, and fractional varactor groups. More details on the design and operation of the DCO control system are provided in Section 3.2.3. The PVT varactors are binary weighted and control the coarse tuning of the DCO. The acquisition varactors are unit weighted and control frequency steps that are smaller than those of the PVT bits. Being unit weighted means that each varactor in the group is the same size. The tracking varactors are also unit weighted and control finer tuning steps than the acquisition bits. The fractional varactors are controlled by the $\Delta\Sigma$ modulator and are also unit weighted.

The design of the acquisition and tracking varactor banks seeks to minimize the impact of the manufacturing variation on the output frequency by distributing the varactors throughout the DCO structure. The acquisition and tracking varactors are distributed in such a way that as the OTW requires more varactors to be activated, the locations of the active varactors are not concentrated in any one specific location of the physical layout. If manufacturing variations were to cause the varactors in one region of the layout to have a different impact on the frequency step than those in another region, then spreading the varactor activation sequence throughout the layout would serve to average the effects of the imperfections on the frequency step sizes.

For the tracking bit varactors, a second feature is included in addition to the layout distribution which also seeks to reduce the effects of manufacturing variation. These bits use a rotation technique that ensures that even if the OTW is held constant, a small selection of the active varactors will toggle with every reference clock period, as discussed in [16]. The group of toggling varactors is also distributed evenly throughout the layout. The effect of this toggling is that if the varactors are not uniform throughout the layout, then their differing effects on the frequency output will be averaged over time.

After the output of the PDC and before the fractional counter, an extra system is in place for basic error checking. This error checker is a bubble corrector and its purpose is to remove single bit errors from the PDC output [14]. It does so by comparing each bit with its neighbours in order to find bits that do not match either adjacent bit. If, for example, a high bit is surrounded by two low bits, this indicates a bubble in the PDC output. The bubble corrector would flip the high bit to a low bit. This action is demonstrated in Figure 2.25. By reducing or eliminating errors coming from the PDC, the fractional counter should be able to provide better information to the phase detector with regards to the fractional phase of the DCO.

Figure 2.25: Bubble correction operation

Chapter 3: Design and Simulation Results

3.1 Overview

The ADPLL design is split between two different technologies: an integrated circuit using TSMC 65nm CMOS, and a Xilinx Virtex 6 FPGA on an ML605 evaluation board. By having this split, all of the modules on the FPGA can be changed as needed for prototyping and testing purposes. In addition, a Raspberry Pi 2 Model B computer, abbreviated as RPi, is connected to the FPGA to provide user-programmable data to the FPGA. An external signal generator is used to provide a clean reference clock to the IC. The block diagram in Figure 3.1 illustrates the connections between the different parts of the system.

Block diagrams for the FPGA and IC components of the system can be found in Figure 3.2 and Figure 3.3, respectively. These two figures expand on the system block diagram from Figure 2.24 by adding all of the supporting modules that are necessary for the system to operate.

The IC portion of the design can be divided into the portions that were designed using digital design techniques and those designed using analog techniques. Figure 3.4 illustrates this division. Additionally, within the digitally designed section of the IC, modules are divided into two distinct layout areas. The high speed counter comprises one of these layout areas and all of the other digital IC modules make up the second area.

Figure 3.1: Connections between the different parts of the full system

Figure 3.2: Block diagram of the FPGA portion of the ADPLL system

Figure 3.4: IC block diagram illustrating digital and analog design sections

3.2 ASIC

The modules discussed in this section are all located on the IC. They have been fabricated using the TSMC 65nm design kit including the TCBN65GPLUS digital standard cell library. The tools used for the design and verification of this section of the system are as follows: Icarus Verilog for behavioural simulation, Synopsys Design Vision for synthesis, Cadence Encounter for placement and routing, Cadence Virtuoso for schematic and post-layout simulation, and Mentor Graphics Calibre for physical verification and signoff.

3.2.1 Counter

The integer counter, located on the IC, receives a signal directly from the DCO output latches and increments with every rising edge. As such, the counter needs to be capable of handling speeds of up to approximately 6.5GHz. A standard synchronous counter, shown in Figure 3.5, in the technology used for this thesis would be overburdened if it were to try to count at the required speed.

Figure 3.5: Synchronous counter using T flip-flops with parallel carry

Instead of using a synchronous counter on its own, this design employs a hybrid counter similar to that in [21]. A four bit Johnson counter, which is equivalent to three binary encoded bits, operates at DCO speeds up to 6.5GHz. An eight bit synchronous

counter operates at one eighth of the DCO speed. All of the counter bits are sampled using the retimed clock. A block diagram of the counter system is shown in Figure 3.6.

For the counter block, the precise topology of the design is very important. All of the code for the counter was written at the gate level using specific gates from the 65nm standard cell library. This is in contrast with all of the other digital modules in the PLL system, which were programmed using behavioural code and a synthesizer. It was found that the synthesizer consistently made design choices that added too much delay to the signal paths. For example, in the Johnson counter portion of the design shown in Figure 3.7, the synthesizer would tend to place pairs of buffers between adjacent flip-flops. As such, the counter code was written at the gate level in order to bypass the synthesizer entirely; however, the full counter layout, including the Johnson and synchronous counters, was still implemented using automated place and route software.

Figure 3.6: Counter block diagram

Two programmable delay elements are placed on the retimed clock path to provide two slightly different clocks to each of the counter segments. In order to ensure that the sampling stage has stable data inputs, the delay elements are fine-tuned using post-layout simulations. The fine-tuning of the retimed clocks ensures that no counter data transitions

occur within at least 50ps of the rising edge of the retimed clock. The retimed clock delay tuning is discussed further in Section 3.2.5.

Since there are four bits that are encoded using a Johnson counter and eight bits that are already binary weighted from the synchronous counter, a decoder module is placed on the FPGA to decode the counter bits and combine them into a form that can be used by the phase detector.

3.2.1.1 Johnson Counter

The Johnson counter consists of four D-type flip flops configured in a ring with an inversion between the final stage and the first stage, as shown in Figure 3.7. Due to the configuration of the counter and due to all of the Q outputs being set to zero on reset, the Johnson counter outputs are limited to the following patterns: 0000, 0001, 0011, 0111, 1111, 1110, 1100, and 1000. A dummy cell is placed after the last register so that the loading on the Q outputs of each register is the same in the layout.

Figure 3.7: Johnson counter schematic

The output waveform of any one of the Johnson counter bits is equivalent to one eighth of the input DCO clock frequency. Figure 3.8 shows a post-layout simulation

result demonstrating the Johnson counter outputs. The Johnson counter is used to generate the divide-by-8 clocks required by the $\Delta\Sigma$ modulator and the synchronous counter. The inverted output of the first bit was chosen to be the divide-by-8 clock. The non-inverted outputs and the inverted output of the last bit were avoided for this purpose so as to not introduce additional loading on those pins; however, the inverted outputs on the two middle bits would have also been valid choices.

Using RC extracted post-layout simulations, the delay time between the positive clock edge at one flip-flop and the data input signal changing at the next flip-flop can be determined. In order for the Johnson counter to work at high speeds, the delay is required to be less than the period of the input signal, taking into account a safety margin for jitter, setup time, and other unexpected delays. In the case of this system with a 6.5GHz clock, the total delay on every bit needs to be less than 150ps. Table 3.1 lists the results of the delay measurements in simulation. The delays for bits one through three are consistently within a few picoseconds of 85ps, but the delay to bit zero is considerably higher. This is due to the fact that the zero bit receives its data from the inverted output of the previous bit, whereas all of the other bits receive their data from the non-inverted output of the previous stages – bit zero is unique in this regard.

Table 3.1: Johnson counter clock-to-data-in delays

Flip-flop bit number	Time delay between clock edge and data edge
[0]	106ps
[1]	88ps
[2]	84ps
[3]	87ps

Figure 3.8: Post-layout Johnson counter output pattern

3.2.1.2 Synchronous Counter

The eight bit synchronous counter is shown in Figure 3.9. It consists of eight T flip-flops and uses a combination of parallel and serial carry logic. The T flip-flops are designed as positive-edge-triggered master-slave JK flip-flops, similar to [22], with the J and K inputs tied together to make the T input. Figure 3.10 shows a schematic of the JK flip-flop used in this system.

Figure 3.9: Synchronous counter schematic

Figure 3.10: Master-slave JK flip-flop schematic

In order to minimize the delay along the carry bit paths, parallel logic is used as much as possible; however, due to the length of the counter and the limitations of the standard cell kit, a serial carry element is present in the middle of the counter. The delay time between the positive edge of the clock and any change in the data inputs to each flip-flop are shown in Table 3.2. These results originate from a post-layout RC extracted simulation.

It can be seen from the data that the delays become longer as the data passes through larger logic gates – for example, the data at the input of bit two passes through a two-input AND gate, whereas the data at the input of bit four passes through a four-input AND gate. In addition, a significant increase in the delay times can be seen at the point where the data passes through a second logic gate. Bits five through seven receive data that passes through a four-input AND gate in addition to another two-, three-, or four-input AND gate. The delay values found in the simulation support the idea that the synchronous counter on its own would not be suitable for counting a 6.5GHz DCO signal; the delays beyond the first few bits are longer than the DCO clock period.

For each flip-flop in the design, the inverted outputs are used as the counter outputs. This design choice was made in order to spread the loading between the two outputs of each flip-flop. Rather than having the carry logic and the sampling stage connected to a single output, this design connects the carry logic to one output and the sampling stage to the other output. The sampling stage performs a second inversion to return the count to a non-inverted state. The output of the synchronous counter can be seen in the post-layout simulation results shown in Figure 3.11. The values shown in this figure are directly viewed from the inverted outputs of the synchronous counter, so each of the values shown would be inverted again by the sampler to give the desired output data.

Table 3.2: Synchronous counter clock-to-data-in delays

Flip-flop bit number	Time delay between clock edge and input data edge
[0]	N/A
[1]	87ps
[2]	110ps
[3]	123ps
[4]	149ps
[5]	204ps
[6]	220ps
[7]	229ps

Figure 3.11: Synchronous counter output waveform

3.2.2 Bubble Correction

The bubble corrector is designed to eliminate single bit bubbles in the output code of the DCO latches. That is to say, a zero surrounded on both sides by ones would become a one, and a one surrounded on both sides by zeroes would become a zero. The behaviour of the bubble corrector was previously illustrated in Figure 2.25 of Section 2.5. The bubble correction module is designed on the IC and is similar to that in [14].

For each bit $[N]$ in the 64 bit DCO output code, the corrected output is derived by taking a majority vote of bits $[N-1]$, $[N]$, and $[N+1]$. At the start and the end of the bit array, the majority vote wraps around to include the bit at the other end of the array. To perform the majority vote, three AND operations are performed between each of the three pairs of bits being considered for the vote. Then, an OR operation is performed between the three results of the AND operations. The operation is shown in Equation (3.1). The result is that if zero or one of the three bits are high, then none of the AND operations will result in a one, resulting in a vote result of zero; whereas if two or three of the bits are high, then at least one of the AND operations will return a high result which would cause the OR operation to give a high output for the majority vote.

$$Out \leq (In[N] \& In[N - 1]) | (In[N] \& In[N + 1]) | (In[N - 1] \& In[N + 1]); \quad (3.1)$$

3.2.3 DCO Control System

The on-chip DCO is controlled by four groupings of bits. In general, each bit controls a single varactor on the DCO. A block diagram of the DCO control system can be seen in Figure 3.12. The first group comprises the three binary weighted PVT bits, which control coarse tuning of the DCO. The second group comprises the 512 thermometer encoded acquisition bits, which are generated from a binary-to-thermometer

encoder. The third group comprises the 512 pseudo-thermometer encoded tracking bits, which are similar to the acquisition bits with the addition of a dynamic element matching technique [16]. Thermometer encoding is used in the acquisition and tracking control banks to ensure monotonicity and linearity in the varactor switching [14]. The fourth and final group, the fractional bits, consists of eight bits controlled by a $\Delta\Sigma$ modulator. All of the control bit blocks are designed on the IC.

Figure 3.12: Block diagram of the DCO control system

3.2.3.1 PVT Control

The three PVT bits control three binary weighted varactors on the DCO. The PVT control implementation is simply an on-chip buffer register; since it is binary weighted, no modifications need to be made with regards to encoding. The three varactors themselves are actually split into 64 components each which are distributed evenly throughout the oscillator so as to balance the capacitive load across the entire DCO ring.

3.2.3.2 Acquisition Control

The acquisition bit control system uses a binary-to-thermometer encoder to transform nine binary weighted input bits into 512 unit weighted output bits as indicated in Figure 3.13. These 512 bits are split into 64 groups of eight bits each due to the design of the varactor banks in the DCO.

It is worth noting that, whereas the nine binary weighted inputs represent a total of 512 patterns, the 512 thermometer encoded bits have a total of 513 possible patterns ranging from all zeroes to 512 ones. Therefore, a thermometer expansion of the nine input bits may be chosen to cover the range of values from all zeroes to 511 ones while omitting the all ones pattern, or it may cover the range of values from a single one to 512 ones while omitting the all zeroes pattern. For this design, the thermometer expansion was chosen to cover the range from all zeroes to 511 ones.

Figure 3.13: Binary to thermometer code expansion for acquisition bits

When implemented in Verilog code, the thermometer code transformation can be condensed into a single equation as follows:

$$Thermometer <= \sim(\sim(0) \ll (Binary)); \quad (3.2)$$

In the equation above, a string of zeroes is first inverted to become a string of ones. The ones are then left shifted by an amount dictated by the binary input value and this result is subject to a second unary inversion. The end result is a string of bits wherein all of the ones are grouped starting from the lower bits and all of the zeros are grouped starting from the higher bits.

The positioning in the physical layout of the acquisition control pins was chosen in such a way as to help alleviate the effect of process variation on the varactors. Rather than starting the pin placement at one position on the DCO ring from zero and placing the pins sequentially around the ring, the pin assignments are placed around the ring to achieve a somewhat balanced distribution of active varactors for any given input code. This distribution is illustrated in Figure 3.14. In the illustration, a sample of acquisition bit numbers are shown in their approximate locations around the DCO ring layout. The acquisition bits continue in the indicated pattern up to bit 511, which would be in the area on the left side of the top edge of the ring.

Figure 3.14: Balanced acquisition bit distribution around the DCO ring

3.2.3.3 Tracking Control

The tracking control bits are conceptually very similar to the acquisition control bits in that they both use 512 unit weighted control varactors. The key distinction to be made with the tracking bits is that they make use of a cyclic shifting technique described in [16] to continually toggle a small number of active bits. This toggling action serves to reduce the impact of process variation by time averaging any mismatch in the capacitance of some of the active varactors.

With this technique, the unit weighted tracking bits are split into rows, with one row being considered the active row. The active row is the one containing the transition point between the zero bits and the one bits. The total number of active bits is determined by the value of the binary weighted input, much like in the case of the acquisition bits. However, with the tracking bits, the active row is subject to a cyclic shift to toggle which specific bits within the row are high with every passing clock cycle. An example of this is shown in Figure 3.15. In the example given, the active row – the fourth from the top – has five active bits and three inactive bits. During clock cycle A, the active bits start at the 3rd bit from the left. With each passing clock cycle, the starting bit moves to the right; as the right-most bits are shifted beyond the last available position, they wrap around to the left-most bit. All rows below the active row are entirely filled with ones, and all rows above the active row are entirely filled with zeroes.

Figure 3.15: Tracking bit cyclic shifting within the active row over four clock cycles

A block diagram illustrating the design of the tracking bit control system is shown in Figure 3.17. In this implementation, the 512 tracking bits are split into 64 rows of eight bits each. The input binary weighted value is split into a six bit row select value and a three bit column activation value. The row select portion of the input determines which one of the 64 rows is the active row. The column activation portion of the input determines how many bits within the active row will be set high during any given clock cycle. A three bit counter which increments by one at each clock cycle is used to determine the starting position of the high bits within the active row. A lookup table is used to combine the starting position and the desired quantity of active bits to generate a pattern to be used for the active row. A second lookup table makes use of the row select portion of the input to generate the necessary rows of ones and of zeroes and places the active row in the appropriate position within those rows.

A Verilog simulation result showing the behaviour of the tracking bit control system is shown in Figure 3.16. The simulation results demonstrate the rotation scheme affecting groups of eight bits at a time with varying numbers of active bits depending on the specific input code. Due to the very large number of output bits, hundreds of extra bits are not shown in the image. All of the omitted output bits would be to the right of the displayed output bits and they would all be ones.

```

sim time: 296.495us, binary_in: 484, rotating_thermo: 000000000000000000000000111100111111
sim time: 296.505us, binary_in: 489, rotating_thermo: 000000000000000000000000111101111111
sim time: 296.515us, binary_in: 489, rotating_thermo: 00000000000000000000000111111111111111
sim time: 296.525us, binary_in: 489, rotating_thermo: 000000000000000001000000011111111111111
sim time: 296.535us, binary_in: 489, rotating_thermo: 00000000000000000100000011111111111111
sim time: 296.545us, binary_in: 489, rotating_thermo: 00000000000000000100000111111111111111
sim time: 296.555us, binary_in: 489, rotating_thermo: 00000000000000000001000011111111111111
sim time: 296.565us, binary_in: 489, rotating_thermo: 00000000000000000001000111111111111111
sim time: 296.575us, binary_in: 489, rotating_thermo: 00000000000000000000100111111111111111
sim time: 296.585us, binary_in: 489, rotating_thermo: 00000000000000000000001011111111111111
sim time: 296.595us, binary_in: 489, rotating_thermo: 00000000000000000000000111111111111111
sim time: 296.605us, binary_in: 494, rotating_thermo: 00000000000000000100000011111111111111
sim time: 296.615us, binary_in: 494, rotating_thermo: 00000000000000000110011111111111111111
sim time: 296.625us, binary_in: 494, rotating_thermo: 00000000000000000111001111111111111111
sim time: 296.635us, binary_in: 494, rotating_thermo: 00000000000000000111001111111111111111
sim time: 296.645us, binary_in: 494, rotating_thermo: 00000000000000000111100111111111111111
sim time: 296.655us, binary_in: 494, rotating_thermo: 00000000000000000111110011111111111111
sim time: 296.665us, binary_in: 494, rotating_thermo: 00000000000000000111111011111111111111
sim time: 296.675us, binary_in: 494, rotating_thermo: 00000000000000000111111111111111111111
sim time: 296.685us, binary_in: 494, rotating_thermo: 00000000000000000100111111111111111111
sim time: 296.695us, binary_in: 494, rotating_thermo: 00000000000000000110011111111111111111
sim time: 296.705us, binary_in: 499, rotating_thermo: 00000000000000001110011111111111111111
sim time: 296.715us, binary_in: 499, rotating_thermo: 00000000011100001111111111111111111111
sim time: 296.725us, binary_in: 499, rotating_thermo: 00000000001110001111111111111111111111
sim time: 296.735us, binary_in: 499, rotating_thermo: 00000000000111001111111111111111111111
sim time: 296.745us, binary_in: 499, rotating_thermo: 00000000000011101111111111111111111111
sim time: 296.755us, binary_in: 499, rotating_thermo: 00000000000001111111111111111111111111
sim time: 296.765us, binary_in: 499, rotating_thermo: 00000000100000111111111111111111111111
sim time: 296.775us, binary_in: 499, rotating_thermo: 00000000110000011111111111111111111111
sim time: 296.785us, binary_in: 499, rotating_thermo: 00000000111000001111111111111111111111
sim time: 296.795us, binary_in: 499, rotating_thermo: 00000000011100001111111111111111111111
sim time: 296.805us, binary_in: 504, rotating_thermo: 00000000001110001111111111111111111111

```

Figure 3.16: Verilog simulation result of tracking bit control block

In addition to the cyclic shifting described above, the tracking bits also use the distributed pin placement used by the acquisition bits as illustrated in Figure 3.14. The distributed pin placement results in every row having one bit in every octant of the DCO ring, which means that the bits being toggled by the cyclic shifting will be evenly distributed throughout the DCO.

Figure 3.17: Tracking bit control system block diagram

3.2.3.4 $\Delta\Sigma$ Modulator

The $\Delta\Sigma$ modulator, which imparts a dithering effect on the DCO tuning signal to increase frequency resolution, is implemented using a MASH-3 architecture as described in [16]. As shown in Figure 3.18, the $\Delta\Sigma$ modulator consists of three accumulators and a combiner block, the latter of which is depicted in Figure 3.19. The $\Delta\Sigma$ modulator receives five bits at its input and it outputs eight bits, one of which is tied low. This eighth bit is included in the design due to the structure of the DCO varactor banks; the varactor bank controlled by the $\Delta\Sigma$ modulator is identical to one of the 64 varactor banks controlled by the tracking bits. For uniformity and simplicity, the varactor bank for the $\Delta\Sigma$ modulator was not redesigned to eliminate the one extra varactor.

Figure 3.18: Delta-Sigma Modulator with MASH-3 topology

Figure 3.19: Delta-Sigma Modulator combiner block structure

The extra resolution provided by the $\Delta\Sigma$ modulator can be determined based on the analysis in [16]. With the frequency step due to the unit weighted tracking bits marked as Δf^T and the number of input bits to the $\Delta\Sigma$ modulator, $N_{\Delta\Sigma}$, being 5, the time averaged frequency resolution of the system, indicated by $\Delta f^{T-\Delta\Sigma}$, is found using Equation (3.3). With a tracking bit frequency step 19kHz, as discussed in Section 4.3, the frequency resolution with $\Delta\Sigma$ modulation is approximately 375Hz.

$$\Delta f^{T-\Delta\Sigma} = \frac{\Delta f^T}{2^{N_{\Delta\Sigma}}} \quad (3.3)$$

The modulator is clocked using a divide-by-eight version of the DCO output signal. This clock is provided by the high-speed counter. With a DCO frequency of approximately 5.5GHz to 6.5GHz, this means that the $\Delta\Sigma$ modulator operates at a speed of approximately 685MHz to 815MHz. The order of the $\Delta\Sigma$ modulator is programmable up to third order, including the option to turn it off entirely if desired. Lower orders can be achieved by ignoring the third (for second order operation) or the third and second (for first order operation) accumulator outputs [16].

For demonstration purposes, Figure 3.20 illustrates the effects of the $\Delta\Sigma$ modulation dithering on the DCO control signal. In the top plot, the tracking segment of the control word is shown without any $\Delta\Sigma$ dithering. The data in the tracking word updates at the reference clock rate of 100MHz. In the bottom plot, the tracking word is merged with the signal provided by the $\Delta\Sigma$ modulator to make a dithered control word. The modulated data updates at one eighth of the DCO frequency. The tracking data is shown in the bottom plot as a white overlay to illustrate that the modulated control word averages to the value of the unmodulated control word. In order to make the $\Delta\Sigma$ modulator effect visible in the figure, the top three bits of the tracking word have been

omitted from the plot data. This has the effect of reducing the upper limit of the displayed tracking word data which in turn makes the dithering effect appear relatively larger when overlaid upon the tracking word data.

Figure 3.20: Simulation of the tracking part of the DCO tuning word (A) without $\Delta\Sigma$ modulation and (B) with $\Delta\Sigma$ modulation

3.2.4 Digitally Controlled Oscillator

The Digitally Controlled Oscillator (DCO) in the system is implemented as a Rotary Traveling Wave Oscillator (RTWO) similar to that described in [14], [1], [15], and [17]. The oscillator loop, shown in Figure 3.21, is made up of 128 RF transmission line segments which take the form of a folded pseudo differential loop. Throughout the loop, 64 inverter pairs are inserted in the RTWO ring to counteract any loss of energy and

to ensure the loop oscillates [23]. Alongside the inverter pairs are 64 identical capacitance banks to control the frequency of the oscillations. Figure 3.22 illustrates the inverter pairs and the capacitance banks in an unfolded RTWO loop. The RTWO features 64 differential tapping points to provide a phase resolution of 0.049 radians or 2.8 degrees based on Equation (3.4) [14].

The 64 tapping points are sampled by differential latches clocked at the reference clock frequency of 100MHz, and are passed to the next module, which is the bubble corrector. One of the tapping points is also passed without being sampled to the clock retiming circuit, the counter, and an output pad for probing during testing.

$$\Delta\phi = \frac{\pi}{N_{Taps}} \quad (3.4)$$

Because the RTWO signals are square waves with fast transition times [1], [23] and because the magnetic fields generated by the oscillating signals are concentrated between the differential traces [14], the RTWO phase noise performance is mostly unaffected by nearby digital switching. Therefore, it is possible to place the bulk of the digital components inside the RTWO area. By doing so, the amount of wasted area on the IC can be reduced and the 64 tapped DCO output signals can all have similar distances to travel to reach the digital logic inputs.

Simulation results for the DCO phase noise were presented previously in Figure 2.20 in Section 2.4 for an output frequency of 5.7GHz. From the simulation, the expected phase noise at a 1MHz offset is -112dBc/Hz. The frequency of the oscillator was designed to be as high as possible while having 64 tapping points using the 65nm design kit.

Figure 3.21: RTWO consisting of 128 transmission line segments (from [24])

Figure 3.22: Unfolded RTWO with inverter pairs and capacitance banks (from [14])

The capacitor banks consist of three portions: PVT bits, acquisition bits, and tracking bits. The capacitance in these banks, one of which is shown in Figure 3.23, are provided by varactors of various sizes that can change capacitance by varying a control voltage. The control voltages are provided by the DCO control system discussed previously in Section 3.2.3.

For the acquisition and tracking bits, each of the 64 capacitance banks contains eight unit weighted acquisition varactors and eight unit weighted tracking varactors for a total of 512 of each of these bits distributed throughout the RTWO. The PVT bits, by contrast, are binary weighted and each bank contains three varactors. The PVT bits in each bank are connected to the same bits in every other bank, meaning that the same three control bits control every set of three PVT bits throughout the ring. This is unlike the acquisition and tracking bits, for which all 512 can be activated and deactivated individually. Because the PVT bits are connected as they are, the PVT varactors in one individual bank actually represent 1/64th of the total desired capacitance for each corresponding PVT bit.

Figure 3.23: A single capacitor bank and inverter pairing (modified from [24])

3.2.5 Clock Retiming and Delay

A clock retiming circuit is used to compensate for the misalignment between the reference clock signal and the DCO output signal [2]. The clock signal generated from this process is referred to in this work as the retime clock. Retiming the clock has the effect of ensuring that the newly generated clock does not transition at the same time as or immediately before or after the rising edge of the DCO output signal. The rising edges of the retime clock will occur at a consistent time following the corresponding DCO output rising edge.

The clock retiming circuit is designed as a double flip-flop [14]. A block diagram of the clock retiming circuit is shown in Figure 3.24. Both flip-flops in the retiming system receive the high speed DCO output as the clock inputs. The first flip-flop takes the reference clock as the data input while the second flip-flop uses the output of the first flip-flop as its data input. The output of the second flip-flop is the retime clock which will be passed through the clock delay circuit, described below. A schematic-level simulation

result is shown in Figure 3.25 to illustrate the action of the retime circuit. The retiming circuit provides a consistent delay between the DCO rising edge and the retime clock rising edge, thereby synchronizing the two signals.

Figure 3.24: Schematic of the clock retiming flip-flops

Figure 3.25: Simulation of the clock retiming circuit

In order to meet the timing requirements of the counter modules, the undelayed retime clock is split and passed through two delay lines as shown in Figure 3.26. Each delay line receives a four bit tuning word that determines the value of the output delay. These bits originate from the configuration module on the IC.

Figure 3.26: Block diagram of the clock retiming system including delay lines

A schematic of a single delay line can be found in Figure 3.27. As can be inferred from the signal paths through the multiplexers and demultiplexers, the acceptable bit patterns for the selection bits are 0000, 0001, 0010, 0100, and 1000. If more than one bit is set high, then the output of the delay line will no longer function.

Figure 3.27: Schematic of one delay line with each section marked by dashed boxes

RC extracted post-layout simulations were used to determine suitable bit patterns for the delay lines. The results are shown in Figure 3.28. The delay times are summarized in Table 3.3. Note that when the bit patterns are written in this text, the leftmost bit is the Sel[3] bit and the rightmost bit is the Sel[0] bit.

Figure 3.28: Delay line simulation results illustrating delay times

Table 3.3: Summary of delay line simulation results

Delay bit pattern	Delay time (ps)
0000	111.9
0001	135.7
0010	162.4
0100	170.3
1000	194.3

Further testing of the retime clock delay system in combination with the counter system is done to identify bit patterns that result in a wide area around the rising edge of all of the retime clock edges in which there are no data points. If data transitions were to occur near the retime clock edges, the sampling flip-flops shown in Figure 3.6 may experience setup or hold time violations.

Post-layout simulations indicate that a bit pattern of 1000, resulting in the longest available delay time, is expected to be suitable for almost the entire DCO frequency range and for both of the counter parts. The exception is that the Johnson counter at 6.5GHz appears to be better suited for a bit pattern of 0100, resulting in the second longest delay.

Figure 3.29 shows the results of post-layout simulations which illustrate the time delays between the rising edges of the retime clock and the data transitions for both counter parts. The methodology for the collection of the data represented in the plots is as follows:

1. Locate a rising retime clock edge
2. Locate the nearest data transition before the clock edge and record the time difference
3. Locate the nearest data transition after the clock edge and record the time difference

4. Repeat steps 2 and 3 for every rising retime clock edge
5. Repeat steps 1 through 4 for both the Johnson and synchronous counters
6. Repeat steps 1 through 5 for other DCO frequencies to cover the entire DCO tuning range

The time differences in steps 2 and 3 are calculated as:

$$\text{Delay} = (\text{Retime Clock Time}) - (\text{Data Transition Time}) \quad (3.5)$$

Due to the convention established by Equation (3.5), positive delay values indicate that the retime clock edge occurs after the data transition because the timestamp of the retime clock would have a greater value than the timestamp of the data transition; negative delay values, conversely, indicate that the retime clock edge occurs before the corresponding data transition time. In addition, data points near zero indicate that a data transition has occurred near a clock edge.

Because of the periodic nature of the counter data, the data points tend to gather in groups that repeat according to the DCO period length. The data points in Figure 3.29 can be moved as a group to the left or right by adjusting the bit patterns of the delay modules; individual bits cannot be moved on their own. Increasing the delay has the effect of moving all of the data points to the right, and decreasing the delay has the effect of moving all of the data points to the left. The delay bit patterns indicated in the plot titles result in the most balanced space possible around the retime clock edges; more delay would move the clock edges closer to the data transitions that come after the clock edges, and less delay would move the clock edges closer to the data transitions that come before the clock edges. The data for the Johnson counter delays and for the synchronous counter

delays are separated vertically for visual clarity; the vertical axis has no further meaning beyond that.

Figure 3.29: Timing differences between data transitions and rising retimed clock edges

3.3 FPGA

The modules discussed in this section are all implemented on a Virtex 6 FPGA on an ML605 evaluation board. The following tools were used for the design and verification of the FPGA components: Icarus Verilog for behavioural simulation, Xilinx ISE for synthesis and post-route simulations, and Xilinx Chipscope for real-time observation of signals on the FPGA.

The FPGA implementation allows for the flexibility to experiment with various settings within each module, which would not be possible with an ASIC implementation. However, with the FPGA, the maximum signal speed is significantly lower than the speeds that are possible on the IC – the FPGA on the ML605 is rated for clock speeds of around 300MHz in most applications [25], [26]. Almost all of the components on the FPGA operate at 100MHz. The exceptions are the configuration modules which operate at lower frequencies.

3.3.1 Integer Count Decoder

A decoding module on the FPGA receives the twelve counter bits and decodes them to a binary count format. The eight bits from the synchronous counter require no additional processing; however, the four bits from the Johnson counter are decoded and, as a result, become a three bit binary count. The two counter parts combine to make an eleven bit count.

During the time between two transitions of the fastest synchronous counter bit, the Johnson counter will cycle through all eight of its patterns. Because of this, the Johnson counter will always have the same pattern when the synchronous counter undergoes a data transition. The Johnson counter pattern that immediately follows a

synchronous counter bit transition is the zero code – the pattern that will be decoded to a zero by this module. Figure 3.30 shows an annotated graph of the Johnson counter bits being compared against the lowest bit of the synchronous counter. With each annotation, it can be seen that the Johnson counter has the same pattern immediately following every edge – rising and falling – of the synchronous counter bit. In this case, the zero code is 1100.

Table 3.4: Johnson counter decoding patterns

Johnson counter code	Decoded binary weighted value
1100	000
1000	001
0000	010
0001	011
0011	100
0111	101
1111	110
1110	111

Figure 3.30: Finding the zero pattern based on the Johnson bits and a synchronous bit

In addition, a twelfth bit is generated by toggling a bit on every falling edge of the eleventh bit. This twelfth bit is appended to the high end of the eleven bit count to make a new twelve bit count. It was noted during certain system simulations that with an eleven bit integer count, the loop would fail to lock because of how often the integer count and the phase reference would overflow. Appending a twelfth bit to the integer count – and extending the phase reference to match – solves the issue by doubling the time between overflow events.

Behavioural simulation results with the frequency command word set to 6.05GHz are shown in Figure 3.31 and Figure 3.32. Figure 3.31 shows a comparison between the eleven bit count and the extended twelve bit count. It can be seen from the figure that the twelve bit counter overflows half as often as the eleven bit counter. Figure 3.32 illustrates the phase error output with the eleven bit integer counter and the twelve bit integer counter. As will be discussed in Section 3.3.3, the loop can be considered as locked when the phase error, which is the difference between the reference phase value and the combined integer and fractional counts, converges to a value of zero [2].

Figure 3.31: (A) 11 bit integer counter output and (B) 12 bit integer counter output

Figure 3.32: Phase error output with (A) 11 bit integer counter and (B) 12 bit integer counter

3.3.2 Fractional Lookup Table

On the FPGA, the sampled and bubble corrected 64 bit DCO output signal is used to generate the six bit fractional count. At any given time, half of the outputs of the oscillator will be ones and half will be zeroes. The fractional count represents how much of a rotation has been completed by the leading edge of the oscillator waveform since the last increment of the integer count. Since the DCO can rotate in one of two directions which can change from one startup to the next [14], a switch is included in the fractional lookup table Verilog code to be able to count in either rotation direction.

Figure 3.33 shows a simplified illustration of the DCO as a ring. A selection of output bits have been placed at each quarter of the ring; there are too many output bits to be able to show them all. A rotation direction has been provided for the purpose of discussing a selection of example output codes. With this rotation direction, for example, bit 15 will consistently experience a transition immediately before bit 16. The opposite would be true with the other rotation direction with bit 16 transitioning before bit 15.

Due to the physical layout of the design, the location on the oscillator ring that acts as the clock for the integer counter is very close to bit 32. Therefore, when bit 32 experiences a transition from zero to one, the fractional lookup table should output a binary code of 000000 to indicate that a new oscillation period is beginning. This is the case regardless of the rotation direction of the oscillator. The lookup table outputs codes that correspond to how much of an oscillation has occurred since bit 32 last experienced a zero to one transition. As an arbitrarily chosen example, a binary output code of 100101 – or 37 in decimal notation – would indicate that 37/64ths of a

As previously stated, when bit zero transitions to a high state, the oscillator has just begun a new oscillation. Therefore, the output code is binary 000000. When the leading edge of the oscillation is one quarter of the way through the ring, bit 16 will transition to high. At this point, the output code will be binary 010000. This continues until the 63rd bit transitions high and the output pattern becomes binary 111111. The pattern repeats after this. Table 3.5 lists a selection of output codes corresponding to the rotation direction shown in Figure 3.33.

Figure 3.33: Simplified DCO with a selection of output bits indicated

Table 3.5: A selection of fractional count output codes

DCO output code (64 bit hexadecimal)	Equivalent fractional count (6 bit binary)	Most recent bit transition from 0 to 1
FFFF FFFE 0000 0001	100000	[0]
FFFF 0000 0000 FFFF	101111	[15]
FFFE 0000 0001 FFFF	110000	[16]
0000 0000 FFFF FFFF	111111	[31]
0000 0001 FFFF FFFE	000000	[32]
0000 FFFF FFFF 0000	001111	[47]
0001 FFFF FFFE 0000	010000	[48]
FFFF FFFF 0000 0000	011111	[63]

3.3.3 Phase Detection

The phase detection block consists of two parts: a phase reference generator block and a phase comparison block. It is implemented using Verilog code on the FPGA. A block diagram for the phase detection system can be found in Figure 3.34. The reference phase is generated by integrating an input frequency word over time. It comprises 11 bits to represent the integer count component and 6 bits to represent the fractional count component for a total of 17 bits.

Figure 3.34: Phase detection block diagram

The frequency word represents the expected change in the DCO combined integer and fractional count; its value is dependent on the desired output frequency and can be determined as:

$$FW = 64 * \frac{f_{DCO}}{f_{REF}} \quad (3.6)$$

where the 64 times multiplier acts as a left-shift operation to account for the 6 fractional bits in the phase data.

The phase comparison block first determines the phase error signal value first by subtracting the combined integer and fractional DCO counts from the reference phase signal as follows:

$$\phi_{ERROR} = \phi_{REF} - ((Int \ll 6) + Frac) \quad (3.7)$$

If the reference phase and the DCO combined count are slightly misaligned, large glitches in the phase error signal will occur. To fix this, a deglitching stage is inserted after the subtraction stage. The deglitching stage works by compensating for phase errors larger than a pre-set threshold. In the case of this implementation, the phase error output signal is an 18 bit signed value and the two input signals are 17 bit unsigned values. A glitch is declared when the magnitude of the phase difference exceeds 2^{16} , and it is corrected by adding or subtracting 2^{17} if the glitch has a negative or a positive sign, respectively.

In Figure 3.35, a reference phase and DCO phase, which are slightly and consistently misaligned from one another, are passed through the phase comparison block to demonstrate the effect of the deglitching process. As can be seen, the non-deglitched result features a large spike after one phase signal rolls over to zero but before the second phase signal rolls over to zero. Since this glitch is a result of having a finite number of bits available, rather than being a true representation of the phase difference between the two signals, it is removed from the phase error signal by the deglitching process.

Figure 3.35: Phase error pre- and post-deglitching

The deglitched output phase error signal can be observed to determine if the loop has settled [2]. Since this is a Type-II loop, the phase error will approach zero as the loop stabilizes. The results of a closed-loop Verilog simulation are shown in Figure 3.36 which demonstrates the locking capability of the full PLL system.

Figure 3.36: (A) Reference and DCO phase inputs and (B) phase error

3.3.4 Digital Loop Filter

The digital loop filter consists of an infinite impulse response filter and a PI controller. Together, they serve to suppress quantization noise from the digital circuits preceding it and to set the control bits that control the DCO operating frequency. The digital loop filter is located on the FPGA.

3.3.4.1 IIR Filter

For this filter, a design decision was made to implement it as an Infinite Impulse Response (IIR) filter rather than as a Finite Impulse Response (FIR) filter. The reason for this decision is that for the same performance, an IIR requires less hardware and fewer stages than an FIR [27]. So for a filter of a fixed order, an IIR will have a sharper transition between the pass and stop bands than an FIR. The tradeoff, however, is that

while FIR filters are inherently stable, IIR filters have the potential to be unstable if not designed correctly [28]. The IIR filter can be checked for BIBO (Bounded-Input, Bounded-Output) stability by verifying that all of its poles reside within the unit circle in the complex z -plane [29].

The IIR is implemented as an elliptic low-pass filter. Compared to Butterworth, Chebyshev I, and Chebyshev II filters, the elliptic filter offers the fastest transition between the pass and stop bands at the expense of having ripples in both the pass and stop bands [30]. The cutoff frequency of the filter is selected to be 10MHz so as to limit the effects of quantization noise on the phase error signal above that point [14]. In order to limit the sampling latency of the filter, the filter is designed to be of third order. The stop band attenuation is designed to be 25dB. As the attenuation increases, the transition region of the magnitude response widens; a design compromise is reached between these two properties by selecting a 25dB attenuation. The pass band ripple is designed to be 1dB.

The filter for this system was built using the Matlab Filter Design & Analysis Tool. The tool provides options to select all of the properties listed above, as well as all quantization options for making a digital filter. The magnitude and phase responses of the filter are shown in Figure 3.37. In the graphs, there are lines for both the idealized filter and for the quantized version of the same filter. The results are close enough that the two lines are overlaid and appear as a single line.

As mentioned previously, the stability of the IIR needs to be verified. Figure 3.38 shows the pole/zero plot of the IIR provided by Matlab. In this graph, there are icons for the reference and for the quantized versions of the same filter. Similarly to the magnitude

and phase graphs, the icons are overlaid on one another in the pole/zero graph. Since all of the poles are within the marked unit circle, the IIR filter is stable.

Figure 3.37: (A) Magnitude and (B) phase response of the IIR

Figure 3.38: IIR pole/zero plot

The IIR is implemented as two cascaded sections with transfer functions $H_1(z)$ and $H_2(z)$ shown in Equations (3.8) and (3.9). The two transfer functions are combined as indicated in Equation (3.10). In order for the filter to have a unity gain, scaling factors are applied to each section. These scaling factors are approximately 0.2725 for H_1 and 0.2298 for H_2 .

$$H_1(z) = \frac{1 + z^{-1}}{1 - 0.67716z^{-1}} \quad (3.8)$$

$$H_2(z) = \frac{1 - 1.0934z^{-1} + z^{-2}}{1 - 1.45071z^{-1} + 0.80240z^{-2}} \quad (3.9)$$

$$H(z) = (0.2725 \cdot H_1(z)) \cdot (0.2298 \cdot H_2(z)) \quad (3.10)$$

These transfer functions can be factored to determine the pole and zero locations to manually verify the stability of the filter as shown below. The zeros and poles are also indicated alongside their respective magnitudes in Table 3.6. In each case, the poles have a magnitude less than 1, which indicates that the poles reside within the unit circle and the filter is stable.

$$H_1(z) = \frac{1 + z^{-1}}{1 - 0.67716z^{-1}}$$

$$H_1(z) = \frac{z + 1}{z - 0.67716} \quad (3.11)$$

$$H_2(z) = \frac{1 - 1.0934z^{-1} + z^{-2}}{1 - 1.45071z^{-1} + 0.80240z^{-2}}$$

$$H_2(z) = \frac{z^2 - 1.0934z + 1}{z^2 - 1.45071z + 0.80240}$$

$$H_2(z) = \frac{(z - (0.5467 + 0.8373j))(z - (0.5467 - 0.8373j))}{(z - (0.7254 + 0.5256j))(z - (0.7254 - 0.5256j))} \quad (3.12)$$

Table 3.6: Poles and zeros of the IIR filter

Pole/Zero	Complex value	Magnitude
Zero 1	-1+0j	1
Zero 2	0.5467+0.8373j	1
Zero 3	0.5467-0.8373j	1
Pole 1	0.67716+0j	0.67716
Pole 2	0.7254+0.5256j	0.8958
Pole 3	0.7254-0.5256j	0.8958

3.3.4.2 PI Controller

The PI controller, or proportional-integral controller, is implemented by splitting the filtered phase error signal into two paths: the proportional and integral paths. Along the integral path, the signal is scaled by a constant factor K_I and accumulated by an accumulator. On the proportional path, the filtered signal is scaled by a second constant K_P . Both of these parameters are implemented as bit shift operations, thus limiting the choices of values to powers of two. With the bit shift operations, a parameter of 2^{-N} corresponds to a right bit shift of N bits.

The scaled result on the proportional path is added to the accumulated result from the integral path and from this new value, the binary weighted bits for the PVT, acquisition, tracking, and fractional control modules are generated. A block diagram of the controller is shown in Figure 3.39.

Figure 3.39: Block diagram of the PI controller

The specific values of the parameters were decided using a manual tuning technique similar to the technique used in [31]. The results of the tuning technique were also verified by checking the expected noise results as per the analysis in Section 2.4. While observing the phase error signal, the parameter K_I was set to zero and the parameter K_P was swept across a range of values to find the minimum value that allows the system to settle. Then, using the values of K_P near the threshold between the system settling and not settling, the value of K_I was swept across a range of values. The objective of this second sweep is to find a value for K_I that allows the system to lock within an acceptable time frame.

The plots in Figure 3.40 show the results of sweeping K_P from 2^{-4} to 2^{-8} with K_I set to zero. In the cases of $K_P = 2^{-4}$, 2^{-5} , and 2^{-6} , the system is capable of locking. The plots to the right of these three cases show a closer view of the phase error signal in the steady region. However, in the cases of $K_P = 2^{-7}$ and 2^{-8} , the system is unable to lock when K_I is set to zero.

After sweeping K_P in isolation, K_I was swept using a selection of K_P values. Figure 3.41 shows the results of the K_I sweep simulations for $K_P = 2^{-4}$, which is the

chosen value of K_P . The K_I sweep simulation results for other values of K_P can be found in Appendix A. The objective of this second sweep is to identify reasonable options for K_I before verifying those values in the Section 2.4 analysis. Based on the plots in Figure 3.41 and on the expected noise performance, the chosen value for K_I is 2^{-11} .

With the PI controller parameters set and the IIR filter transfer function known, the phase margin of the ADPLL can be derived by the using the open loop transfer function. From the analysis in Section 2.4, the phase margin is 54.8° and the loop bandwidth is 1.1MHz.

Figure 3.40: K_P sweep simulation results showing the phase error with K_I set to zero

Figure 3.41: K_I sweep simulation results showing the phase error with K_P set to 2^{-4}

3.4 Additional Supporting Components

In addition to the components described in previous sections, there are a number of extra modules that are necessary to successfully implement the system.

The first two modules, the configuration and compression modules, serve to reduce the physical pin Input/Output (I/O) requirements of the IC, the FPGA, and the Raspberry Pi. The reduction in pin requirements resulting from these modules is illustrated in Table 3.7. It would not be possible to implement the system without these modules because there are not enough pins available to accommodate the requirements.

The third module discussed in this section is a Verilog model of the DCO. With the real DCO being designed manually, it is necessary to have a model available for the purposes of behavioural simulations. The DCO model was instrumental in testing and refining the digital modules before implementing them on the IC and on the FPGA.

Table 3.7: I/O pin requirement reductions

Module pairing	Non-reduced pin requirements	Reduced pin requirements
Configuration modules between the Raspberry Pi and the FPGA	310	3
Configuration modules between the FPGA and the IC	267	3
Compression modules between the FPGA and the IC	64	6
Total for FPGA	641	12
Total for IC	331	9
Total for RPi	310	3

3.4.1 Configuration Modules

The configuration modules, which are sometimes referred to in the block diagrams and in the Verilog code as serial to parallel converters or parallel to serial converters, are used for setting user-programmable bits within the system. These modules allow users to change a selection of the PLL parameters while the system is running. Two variations of the configuration module pairings exist in the system: one between the Raspberry Pi and the FPGA, and the other between the FPGA and the IC. The specific sets of bits implemented in the configuration module pairings were chosen because those bits rarely, if ever, have to change more than once; thus, they are well suited for a parallel to serial and serial to parallel conversion module.

Due to the fact that the parallel to serial converter provides a clock to the serial to parallel converter and the fact that the signal timing of the configuration blocks relative to the rest of the system is not critical, the configuration blocks may be operated at different clock speeds than the rest of the system. This is useful for two reasons. First, it allows the configuration block between the FPGA and the IC to be run at a relatively slow speed to ensure proper data transfer. Second, it removes any need to synchronize the clocks between the Raspberry Pi and the FPGA; the RPi can be given exclusive control over the clocking of the configuration module pairing that exists between it and the FPGA.

The configuration block pairing between the FPGA and the IC sets the values of the DCO latch settings, the retime clock delay paths, the $\Delta\Sigma$ modulator mode, and the bubble correction module enable bit. A configuration module pairing is used between the FPGA and the IC in order to reduce the number of I/O pins required between them. As illustrated in Figure 3.42, the configuration modules require only three signal lines to

transfer data. When the configuration modules are not in use, the signal lines between them remain at zero. Upon receiving a pulse on the Initiate Transfer signal input, the parallel to serial converter pushes the input data along with a clock to the serial to parallel converter. When all of the data has been transferred, the parallel to serial converter ceases activity on the clock and data lines while also sending a pulse on the Serial Load line. This final pulse causes the data that has been sent to the serial to parallel converter to be placed on the output pins of that module. The behaviour of the configuration module pairing is illustrated in Figure 3.43.

The bit values mentioned for the previous configuration module are provided to the FPGA via the configuration module between it and the RPi. In addition to those bits, this second configuration module also contains the desired frequency control word for closed loop operation, a set of loop bandwidth control bits, and a set of DCO control bits (e.g. PVT, acquisition, tracking, and fractional) that allow for direct control of the DCO in an open loop format for testing and debugging purposes. The parallel to serial module on the RPi is implemented using Python and the signals are output from the General Purpose Input/Output (GPIO) pins. To ensure a reliable data transfer between the RPi and the FPGA, the Python program is set to operate at a speed of 500Hz.

Figure 3.42: Configuration modules between the FPGA and the IC

Figure 3.43: Simulation waveforms of a configuration module pairing

3.4.2 Compression Modules

The compression modules, consisting of a compressor on the IC and a decompressor on the FPGA, reduce the 64 bit output received from the bubble corrector to six bits to reduce the number of connections required between the IC and the FPGA. On the FPGA, a decompressor reverses the compression to restore the original 64 bit data.

When searching from the lowest bit index to the highest, the compressor finds the first transition that switches from low bits to high bits. The index of the first high bit is noted and this value becomes the compressed data value. A selection of DCO data and the corresponding compressed data is shown in Table 3.8. The decompressor receives the index from the compressor and rebuilds a 64 bit data signal with a low to high transition in the corresponding index.

While the DCO ideally will have an equal split of 32 high bits and 32 low bits due to its symmetry, there is a possibility that some bits may transition slightly early or slightly late [17]. This could result in 33 high bits and 31 low bits being captured by the DCO output latches and passed to the compressor, for example. In this sort of situation, and in any other situation that involves the DCO output not being 32 consecutive high bits and 32 consecutive low bits, the compressor will not be able to convey the extra information required to reconstruct such a pattern. In these cases, the compressor will still report on the first transition low to high transition that it finds and the decompressor will still reconstruct a pattern that has 32 consecutive high bits and 32 consecutive low bits starting at the reported index.

Table 3.8: A sample of DCO data and the corresponding compressed representation

64 bit DCO data (hexadecimal)	6 bit compressed data (decimal)
0000 0000 FFFF FFFF	0
0000 00FF FFFF FF00	8
0000 FFFF FFFF 0000	16
00FF FFFF FF00 0000	24
FFFF FFFF 0000 0000	32
FFFF FF00 0000 00FF	40
FFFF 0000 0000 FFFF	48
FF00 0000 00FF FFFF	56

3.4.3 Verilog DCO Simulation Model

In order to test the digital components in behavioural system simulations, it is necessary to have a Verilog model of the DCO. Due to the manual design of the real DCO, any simulation that uses the real DCO is prohibitively slow. The DCO model is designed to have 64 output tapping points to match the real DCO and it also incorporates optional jitter noise in the outputs. Unlike all of the other digital modules in the system which only require the use of Verilog code, this module makes use of SystemVerilog for the fork/join_none construct. The behaviour of the Verilog DCO model is illustrated in Figure 3.44 and is described below.

The model receives the digital control signals from the control system and uses them to calculate the expected output frequency and period. Since there are 64 tapping points on the DCO, it is expected that after every 1/64th of a period, one bit will transition high while the bit on the opposite side of the ring – the bit that is 32 index positions away – will transition low. With that taken into consideration, the DCO model calculates the expected amount of time that will pass between the start of the cycle to each transition pair. This produces an array of 64 increasingly long time delays.

In addition to the nominal time delays, the model also optionally generates jitter in the rising and falling edges of each bit individually for a total of 128 separate random values. The jitter is generated using a normal distribution which treats the RMS value of the jitter as the standard deviation and using a mean value of zero. The jitter values have a resolution of 1 fs.

To use the delays and jitter values, all 128 transitions are executed in their own forks within a fork/join_none construct. Each fork starts with a delay equal to the appropriate nominal delay plus one of the randomized jitter amounts. Since the jitter can be positive or negative, the total delay can be greater or less than the nominal delay. As a result of the model using the join_none version of the fork command, which does not wait for any of the forks to complete before continuing to the next command in the code, it is also necessary to have a delay equal to the period of the DCO outside the fork command to delay the start of the following oscillation cycle.

Figure 3.44: Annotated illustration of the Verilog DCO model fork/join_none operation

Chapter 4: Measurement Results

4.1 Overview and Test Bench Setup

This section will describe the test environment for the ADPLL. A schematic illustrating the test bench can be found in Figure 4.1. The test bench consists of the Device Under Test (DUT) mounted on a custom Printed Circuit Board (PCB), a Virtex 6 FPGA mounted on the ML605 Evaluation Board, a Raspberry Pi 2 Model B computer, a Rohde & Schwarz FSWP phase noise analyzer and spectrum analyzer, an Agilent DSAX91604A oscilloscope, a Crystek CCHD-575 100MHz square wave oscillator, a Windows computer, an Agilent E3646A Dual Output DC Power Supply, and two Keithley 2400 Sourcemeters. The ML605 board is also connected to the FMC XM105 Debug Card [32] which provides access to many general purpose I/O headers via the high pin count FMC header on the evaluation board.

Figure 4.1: Test bench setup

The test PCB holds the IC portion of the ADPLL as well as the required support circuits. Because the FPGA operates at 2.5V and the IC operates at 1.2V, voltage level shifting circuits are required on all of the I/O pins to convert between these two voltages as illustrated in Figure 4.2. This design uses four SN74AVCH16T245 voltage level converters from Texas Instruments [33].

Figure 4.2: Voltage level shifters between the FPGA and the IC

The test PCB and the FPGA board are connected via two ribbon cables and two SMA cables. Originally, the two boards were connected by only the two ribbon cables. However, early testing indicated that the ribbon cables were unable to reliably carry the 100MHz clock signals. Therefore, the clock signals were removed from the ribbon cables and redirected through two SMA cables which are more suitable for transmitting high frequency signals.

The Raspberry Pi (RPI) computer is connected to the FPGA and is used for supplying configuration data to the FPGA. As was discussed in Section 3.4.1, the RPI provides the programmable latch bits, the programmable delay bits, the $\Delta\Sigma$ modulation mode control bits, the bubble correction enable bit, the frequency word for closed loop operation, and the DCO control bits for open loop operation.

The phase noise/spectrum analyzer and oscilloscope are used to measure the 6GHz DCO output signal from the IC which is probed on chip using a Picoprobe Dual Model 40A four pin probe. The measurements from these devices are 50 Ω terminated.

The Windows computer is connected to the FPGA and is used for reading signals on the FPGA using Xilinx Chipscope Analyzer software. This software works in a manner similar to a digital logic analyzer, except that it is capable of viewing any signal on the FPGA. This includes signals that are never sent to any I/O pins.

The voltages used by the different parts of the IC are as shown in Table 4.1. Because of the varying voltage requirements, and also to provide power supply isolation between different parts of the system, a bank of low dropout voltage regulators is placed on the test PCB. An option is included on the test board to bypass the regulators and

supply each voltage from a different power supply. This option is useful for determining the current consumption of the different parts of the system.

Table 4.1: Voltages and currents of components on the IC

Component	Voltage
Digital core	1.15V
Counter core	1.15V
Oscillator	1.2V
DCO latches	1.2V

4.2 IC Physical Measurements

The IC was manufactured using the TSMC 65nm CMOS General Power (GP) design kit and the TSMC 65nm GP Standard Cell Libraries. It is packaged in an 80 pin Ceramic Quad Flat Pack (CQFP80). A screenshot of the IC layout can be found in Figure 4.3 and a micrograph of the IC can be found in Figure 4.4. The screenshot has annotations showing the locations of some of the key parts of the system. A list of the dimensions of the annotated parts and of the IC itself can be found in Table 4.2. The listed area for the RTWO does not include the empty spaces in the corners, nor does it include the area inside the ring.

Table 4.2: Key dimensions of the IC

	Length	Width	Area
Digital core	188.48 μm	137.28 μm	25,874 μm^2
Integer counter	22.48 μm	20.15 μm	453.0 μm^2
RTWO	418.6 μm	360.1 μm	94,490 μm^2

Figure 4.3: Annotated screenshot of the IC layout

Figure 4.4: Micrograph of the IC

4.3 Open Loop Measurements

Before testing the complete ADPLL, tests were performed on specific parts of the system in an open loop configuration. In the open loop configuration, the PI controller on the FPGA does not control the binary DCO control bits. Instead, these bits are controlled by the configuration module on the RPi. This allows the DCO to be tested with a fixed control word. In order to obtain the best possible measurement results for the DCO phase noise, it was necessary to turn off as many of the connected devices as practical. For these tests, after loading in the control bits from the RPi and passing those bits through the control module on the IC, the RPi was disconnected from the FPGA and the reference clock was turned off. Not taking these steps resulted in poorer phase noise performance.

The frequency range of the DCO in the open loop mode is 5.43GHz to 6.23GHz. On average, the step size of the PVT bits is 86MHz, the step size of the acquisition bits is 319kHz, and the step size of the tracking bits is 19kHz. Although they cannot be measured directly, the step size of an individual varactor within the bank of fractional bits is expected to be 19kHz as well, since the structure of the fractional varactor bank is the same as one segment of the tracking varactor banks.

A time domain representation of the DCO output, terminated by 50 Ω , is shown in Figure 4.6 to illustrate the shape and fast transition times of the output signal.

The measured phase noise of the DCO during open loop operation at 5.7GHz is shown in Figure 4.5. The simulated phase noise is also displayed in the figure for comparison purposes. From the plot, it can be seen that the phase noise of the measured result follows a different shape from the simulation, resulting in higher phase noise up to a 200kHz offset. The maximum difference between the two results, 17dB, occurs at a

20kHz. After 200kHz, the measured result and the simulated result are very similar, with the measured result being up to 4dB better than the simulated result. The discrepancies in the results may be due to a number of factors. The simulation model is a greatly simplified model of the oscillator which may impact the accuracy of the simulation results. There may also be noise coupling into the oscillator from other components in the test setup, which may influence the measured result while not being accounted for in the simulation environment.

Figure 4.5: Open loop phase noise of the DCO at 5.7GHz

Figure 4.6: DCO output in the time domain

The reference clock was also analyzed for its phase noise performance. The result is shown in Figure 4.7. The reference clock noise is very similar to the expected performance given in [18]. The phase noise plot of the reference clock stops at 30MHz due to equipment limitations. The phase noise analyzer has a maximum frequency of 30% of the carrier frequency.

Figure 4.7: Measured phase noise of the 100MHz reference clock

The counter data on the return path from the chip to the FPGA can be analyzed using Chipscope to ensure that the integer counter is performing as expected. A retime clock delay line can be adjusted to increase or decrease the clock delay to ensure the proper sampling of the counter. Figure 4.8(A) demonstrates the output of the synchronous counter when the delay line is not configured properly. Note that on two occasions in the graph, the counter outputs a value that does not match any of the surrounding values. This indicates that a bit within the counter was not sampled at the correct time. Figure 4.8(B) illustrates the output of the synchronous counter when the delay line is configured correctly. In this second graph, there are no errors in the counter output.

It was also noted during both open and closed loop testing that the performance of the counter is noticeably better when the Johnson counter and synchronous counter clock delays are set to the same value. This is even the case when there are two or more

different settings that work for both counters. For example, it could work well with both counters set to 0100 or both set to 0010, but it would not work well if one counter were set to 0100 and the other to 0010. Since the two counters are in the same physical region of the IC, it is possible that setting the clocks to different delays causes cross-coupling issues within the counter module. Fortunately, the two counters are both able to accept the same delay amount, as expected from the simulation results in Section 3.2.5.

Figure 4.8: Counter data with (A) incorrect delay setting and (B) correct delay setting

4.4 Closed Loop Measurements

After the open loop measurements, the FPGA was configured for closed loop operation. The locking range of the ADPLL is from 5.48GHz to 6.22GHz. The locking time of the system can be determined by observing the phase error signal using Chipscope and noting how much time passes before the signal settles near zero. Figure 4.9 and Figure 4.10 illustrate the settling time from behavioural Verilog simulations and from measured results, respectively. From the two figures, the settling time of the system matches the expected result of 25 μ s.

Figure 4.9: Simulated phase error

Figure 4.10: Measured phase error

The output of the system was measured at a frequency of 5.7GHz. The spectrum of the output is shown in Figure 4.11 for a 20GHz span, in Figure 4.12 for a 10GHz span, and in Figure 4.13 for an 80MHz span. In the 20GHz span image, the first four harmonics of the output signal can be seen at 5.7GHz, 11.4GHz, 17.1GHz and 22.8GHz. From the 10GHz span image, reference spurs can be seen at 100MHz offsets from the carrier frequency. The first of these can be seen at -38dBc at 5.6GHz and 5.8GHz. The reference spurs are believed to be caused by insufficient cancellation of frequency pulling by the dummy cells in the RTWO. In the 80MHz span image, two close-in spurs can be seen at a 780kHz offset above and below the carrier with a power of -29dBc. This spur is believed to be caused by coupling between the RTWO and the reference clock. Of the remaining spurs in the spectrum, it is expected that many of them are caused by noise

coming from the RPi, the FPGA board, and the USB connection between the FPGA and the Windows computer.

Figure 4.11: ADPLL output spectrum from 4GHz to 24GHz with harmonics labeled

Figure 4.12: ADPLL output spectrum from 4GHz to 14GHz with harmonics labeled

Figure 4.13: ADPLL output spectrum from 5.66GHz to 5.74GHz

The phase noise of the ADPLL is shown in Figure 4.14 for an output frequency of 5.7GHz. The loop bandwidth and the DCO output latch settings were adjusted manually to find the best phase noise performance. The best bandwidth setting was just above 1MHz, as can be seen in the phase noise plot. The in-band phase noise is -100dBc/Hz and the integrated RMS jitter of the output is 664.9fs between 1kHz and 40MHz.

Figure 4.14: ADPLL phase noise at 5.7GHz

The measured phase noise of the ADPLL is higher than the expected phase noise by 10dB to 20dB for most of the measured frequency range up to 30MHz. This discrepancy increases starting at a 30MHz offset when the measured result reaches a noise floor that the theoretical model does not account for. It is expected that the cause of the extra noise in the system is due to a combination of cross-coupling of the clocks and other high speed signals, switching noise in the DCO latches and digital circuits, and

power supply noise. In particular, it is believed that the presence of the RPi, the FPGA board, the USB connection on the FPGA board, and the voltage level shifters on the test PCB may have all been contributing noise to the chip through their shared ground connections.

It is also believed that the reference clock, which on its own is a very low phase noise signal, may have had its noise increased by the input buffers on the chip. However, it would not be possible to measure this effect due to a lack of access to the signal on the chip. Following the analysis from Section 2.4, specifically the reference clock transfer function described in Equation (2.21) and Figure 2.15, the in-band reference clock phase noise contribution on the ADPLL output is 35dB higher than the phase noise of the reference clock itself. If the reference clock noise were to be raised by the input buffers, this could have a significant impact on the phase noise performance of the ADPLL, especially at low frequency offsets.

At high frequency offsets, it is expected that the overall phase noise should be very similar to the phase noise of the open loop DCO. However, in the case of the measured results, it is clear that the closed loop phase noise of the system exceeds the noise of the open loop DCO by 10dB between the loop bandwidth frequency offset and 30MHz offset. As discussed previously, the discrepancy increases above 30MHz offset. It appears that the DCO in the closed loop setting is being impacted by additional switching noise from nearby circuits, such as the PDC latches.

Figure 4.15 illustrates how the ADPLL may have been able to perform if the DCO weren't subjected to additional switching noise. In the figure, a hypothetical closed loop phase noise performance is constructed using the measured low frequency offset

performance of the closed loop ADPLL and the measured high frequency offset performance of the open loop DCO. The closed loop ADPLL data in the figure was chosen for having the best in-band phase noise performance observed during testing. The open loop DCO data is the same as in Figure 4.5. The point where the two lines intersect is the point where the hybrid noise plot switches its source. The integrated RMS jitter between 1kHz and 40MHz offset of the hybrid noise plot is 306.4fs.

Figure 4.15: Hybrid phase noise profile combining closed and open loop measurements

4.5 Performance Summary

A summary of the performance of the ADPLL is given in Table 4.3. A comparison to a selection of other published ADPLLs is given in Table 4.4. The works included in the comparison table were chosen for having similar architectures to the work presented in this thesis.

Banerjee's Figure of Merit (BFM) [34] can be used to compare the performance of devices with different reference and output frequencies. The equation for the BFM is

$$BFM_{dB} = \mathcal{L}_{In-band} - 10 \log(f_{ref}) - 20 \log(N) \quad (4.1)$$

where N is the ratio between the output and reference frequencies.

A second Figure of Merit (FoM), presented in [35], makes use of the measured jitter and DC power consumption. It is presented in Equation (4.2). It is worth noting that the power consumption of this system was not optimized during testing due to time constraints. However, options exist within the system to reduce power consumption at the expense of higher phase noise.

$$FoM = 10 \log((Jitter[s])^2 \cdot (P_{DC}[mW])) \quad (4.2)$$

When discussing the active area and power consumption in Table 4.3 and Table 4.4, the values given for this work include estimates of how much extra area and power would be required if the FPGA modules were designed on the chip. The estimates were created by synthesizing and layout out the Verilog code using the same tools that were used to implement the on-chip modules.

Table 4.3: Summary of ADPLL performance

Technology	65nm CMOS
Active area	0.15mm ²
Total chip area	1.39 × 1.40 mm ²
Supply voltage	1.15V (Digital), 1.2V (others)
Power consumption	137mW
Reference frequency	100MHz
Output frequency range	5.48GHz to 6.22GHz
In-band phase noise (f _{out} = 5.7GHz)	-100dBc/Hz @ 200kHz offset
Integrated RMS jitter (1kHz to 40MHz)	664.9fs
Banerjee Figure of Merit (BFM) – Eqn. (4.1)	-215dB
Figure of Merit (FoM) – Eqn. (4.2)	-222dB

Table 4.4: Performance comparison against other ADPLLs

	This work	Takinami [14]	Samarah [36]	Wu [37]	Kim [38]
Technology	65nm	65nm	0.13μm	40nm	65nm
Output frequency	5.7GHz	4.13GHz	2GHz	2.65GHz	2GHz
Reference frequency	100MHz	78MHz	20MHz	50MHz	80MHz
In-band phase noise	-100dBc/Hz @ 200kHz offset	-108dBc/Hz @ 200kHz offset	-104dBc/Hz @ 200kHz offset	-108dBc/Hz @ 200kHz offset	-103dBc/Hz @ 4MHz offset
Integrated RMS jitter	0.66ps (1kHz-40MHz)	Not reported	0.70ps (1kHz-100MHz)	0.44ps (1kHz-30MHz)	3.15ps (10kHz-100MHz)
BFM – Eqn. (4.1)	-215dB	-221dB	-217dB	-219dB	-210dB
FoM – Eqn. (4.2)	-222dB	Requires jitter	-231dB	-237dB	-220dB
Power consumption	137mW	45mW	15mW	10.7mW	10.8mW
Active area	0.15mm ²	0.61mm ²	0.43mm ²	0.5mm ²	0.047mm ²

Although the ADPLL performed well when set to output frequencies that were multiples of 100MHz, which were the integer mode frequencies, the ADPLL did not perform well when set to other frequencies, which were the fractional mode frequencies. It appeared during testing that the output signal was being heavily influenced by cross-

coupling with the reference clock. Being limited to using integer frequencies also limited the ability to test the $\Delta\Sigma$ modulator. The purpose of the $\Delta\Sigma$ modulator is to improve the frequency resolution in the fractional mode as well as reduce the output spurs resulting from fractional mode operation [16]. Without meaningful fractional results, the full impact of the modulator could not be tested. When the $\Delta\Sigma$ operating mode was changed, or when the $\Delta\Sigma$ was turned off, the phase noise performance of the system did not change.

It is expected that the system experienced a degradation in performance as a result of power supply noise. In addition to the power supplies connected directly to the test PCB, it is believed that the FPGA board contributed to the power supply noise. The ML605 FPGA board contains many individual components that cannot be deactivated and uses a power supply that was not likely designed to be a low noise supply. Additionally, the FPGA board was connected through a USB connection to a computer. The USB connection likely contributed some noise to the power supplies as well. The Raspberry Pi received its power from a generic USB wall adapter, which also contributed noise to the system. All of these factors together likely resulted in a significant amount of noise on the system's power connections, which impacted the ADPLL phase noise performance.

Chapter 5: Conclusion

This chapter will summarize what was designed in this thesis, issues in the design, and improvements to the design that could be made in the future.

5.1 Accomplishments

This thesis presented the design of an ADPLL using digital standard library components. The design implemented techniques to reduce overall noise such as dynamic element matching in the tracking bits, a balanced activation sequence of the acquisition and tracking bits throughout the physical layout, multi-phase DCO outputs, and delta-sigma dithering on the DCO control word.

The digital components were designed using Verilog code and implemented using automated synthesis and layout tools. A high-speed counter, capable of counting at frequencies greater than 6GHz, was also presented in this thesis. Although this counter uses digital standard cells, its architecture allows it to operate reliably at high speeds.

The implementation of the digital circuits was successfully split between a custom IC and an FPGA, which enabled the observation of digital signals that would be unavailable if the entire ADPLL were implemented on a single IC. The FPGA also enabled a level of debugging that would not otherwise be possible by virtue of being reprogrammable.

Although the overall noise performance is higher than expected from the theoretical analysis, the sources of noise can be explained in such a way that the theoretical models can be made to fit the measured phase noise results.

5.2 Issues in the Design

Although having the system split between the FPGA and the chip was beneficial for the purposes of design flexibility, debugging, and testing, it also introduced certain issues. The connection between the FPGA and the chip was made with a pair of 30cm long ribbon cables. These cables resulted in very long and thin ground connections between the FPGA and the chip. A small amount of shielding was provided on the ribbon cables in the form of multiple ground-signal twisted pairs. However, there were too many signals on the two ribbon cables for every signal to be part of a twisted pair. The cables were very close to reaching their upper frequency limit, as evidenced by the need to move the two 100MHz clock signals off of the ribbon cables and into the SMA cables. Additionally, the signals carried on the ribbon cables were entirely single-ended.

Having multiple clocks in the same layout regions may have also contributed to noise problems. In particular, the $\Delta\Sigma$ modulator used 1/8th of the DCO frequency as its clock, whereas the other logic in the same layout area used the retimed 100MHz clock. This may have caused cross-coupling problems between the two different clock domains. A similar effect was also observed in the integer counter. The Johnson and synchronous counters in the integer counter were placed in the same layout region as one another but were driven by two different retimed clock delay lines. When the delay lines were set to different settings, resulting in two distinct clock timings, the performance of the counter was noticeably degraded.

It is suspected that the ground connection between the FPGA and the test PCB may have allowed for noise from the FPGA ground to affect the system. In addition, the Raspberry Pi connected to the FPGA also contributed noise through the ground. The

system had no means to reduce the noise coming from the grounds of any attached components. Ferrite beads may have helped in this regard, but it would not have been practical to adapt the current test setup to accommodate ferrite beads. This would have required a redesign of the test PCB.

As discussed previously, the ADPLL experienced greatly degraded performance when set to fractional frequency codes. It is believed that the reason for this is cross-coupling of the reference clock into the DCO. In addition to limiting the frequencies available for testing, the lack of meaningful fractional frequency results meant that the $\Delta\Sigma$ modulator could not properly be tested. Therefore, it is unknown whether or not the $\Delta\Sigma$ modulator is functioning properly in the current design.

5.3 Future Work

A number of improvements could be made to the current design in order to get better results. Some of the improvements may be done with the current system, while other improvements would require a new test PCB, a new FPGA board, or a new chip.

The performance of the current system could be optimized in two ways. First, the DCO latch settings should be optimized using an automated script and automated test equipment. A prototype of such a script exists, but time limitations with the availability of the test equipment prevented this level of optimization. Instead, a limited amount of manual optimization was performed during the available time.

The current system could also be optimized in terms of its power consumption. The DCO in particular consumed the majority of the power required by the system. An option exists in the current system to reduce the power consumption of the DCO by disconnecting the power supplies to certain parts of the DCO. The tradeoff is that as the

DCO power is reduced, its noise is increased. It would be worthwhile to investigate a possible balanced between reducing the DCO power while minimizing the impact of the increased noise on the overall system performance.

On-chip, better care should be taken to separate digital logic components that rely on different clock domains. In particular, the $\Delta\Sigma$ modulator should be separated from the main digital core logic and the two parts of the integer counter should be separated from one another.

On the test PCB, all of the grounds are connected directly together. This means that all of the components on the chip, the voltage level shifters, the voltage regulators, the FPGA, and the RPi all share a ground plane. If the grounds were to be isolated through the use of ferrite beads, that would help to filter out some of the high frequency noise passing through the ground plane.

The entire system could be made more compact using a variety of methods. The first method would be to replace the large ML605 FPGA board with a smaller board with fewer on-board components. By using a smaller FPGA board, cable connections between the test PCB and the FPGA could be eliminated entirely by soldering the FPGA board directly to the test PCB. In addition, a smaller board would have fewer components contributing to noise through the ground.

The second method to make the ADPLL more compact would be to use a bare FPGA and attach it directly to the test PCB. This method would allow the FPGA to be placed very close to the IC while maintaining the ability to reprogram the digital logic contained within. In addition, this method would allow for the elimination of the voltage level shifters between the FPGA and the chip if the chosen FPGA uses the same I/O

voltage as the chip. However, unlike the first method, this method would involve routing all of the connections required to make the FPGA operational and reprogrammable rather than having those elements available on a pre-made board. This requirement makes the second method more complex than the first.

The third method to make the system more compact would be to integrate all of the FPGA on the IC. This would remove the ability to reprogram the digital logic, which is the main advantage to using the FPGA, but if designed correctly, would likely result in the best performance.

Appendix A: Additional Figures

The following figures are simulation results for the PI Controller from Section 3.3.4.2. The figures illustrate the K_I sweep results for the values of K_P not shown previously.

Figure A.1: K_I sweep simulation results showing the phase error with K_P set to 2^{-5}

Figure A.2: K_I sweep simulation results showing the phase error with K_P set to 2^{-6}

Figure A.3: K_i sweep simulation results showing the phase error with K_p set to 2^{-7}

Figure A.4: K_i sweep simulation results showing the phase error with K_p set to 2^{-8}

Bibliography

- [1] J. Wood, T. C. Edwards and S. Lipa, "Rotary Traveling-Wave Oscillator Arrays: A New Clock Technology," *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1654-1665, November 2001.
- [2] R. B. Staszewski and P. T. Balsara, All-Digital Frequency Synthesizer in Deep-Submicron CMOS, Hoboken, NJ: John Wiley & Sons, Inc., 2006.
- [3] J. W. M. Rogers, C. Plett and I. Marsland, Radio Frequency System Architecture and Design, Norwood, MA: Artech House, 2013.
- [4] I. L. Syllaos, R. B. Staszewski and P. T. Balsara, "Time-Domain Modeling of an RF All-Digital PLL," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 6, pp. 601-605, June 2008.
- [5] B. Miller and R. J. Conley, "A Multiple Modulator Fractional Divider," *IEEE Trans. Instrum. Meas.*, vol. 40, no. 3, pp. 578-583, June 1991.
- [6] R. E. Best, Phase-Locked Loops: Design, Simulation, and Applications, 5th ed., New York: McGraw-Hill, 2003.
- [7] C. Cao, Y. Ding and K. K. O, "A 50-GHz Phase-Locked Loop in 0.13 μ m CMOS," *IEEE J. Solid-State Circuits*, vol. 42, no. 8, pp. 1649-1656, August 2007.
- [8] J. Craninckx and M. S. J. Steyaert, "A Fully Integrated CMOS DCS-1800 Frequency Synthesizer," *IEEE J. Solid-State Circuits*, vol. 33, no. 12, pp. 2054-2065, December 1998.

- [9] H. R. Rategh, H. Samavati and T. H. Lee, "A CMOS Frequency Synthesizer with an Injection-Locked Frequency Divider for a 5-GHz Wireless LAN Receiver," *IEEE J. Solid-State Circuits*, vol. 35, no. 5, pp. 780-787, May 2000.
- [10] G.-Y. Tak, S.-B. Hyun, T. Y. Kang, B. G. Choi and S. Park, "A 6.3-9-GHz CMOS Fast Settling PLL for MB-OFDM UWB Applications," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1671-1679, August 2005.
- [11] Z. Shu, K. L. Lee and B. H. Leung, "A 2.4-GHz Ring-Oscillator-Based CMOS Frequency Synthesizer With a Fractional Divider Dual-PLL Architecture," *IEEE J. Solid-State Circuits*, vol. 39, no. 3, pp. 452-462, March 2004.
- [12] F. M. Gardner, "Charge-Pump Phase-Lock Loops," *IEEE Trans. Commun.*, vol. 28, no. 11, pp. 1849-1858, November 1980.
- [13] F. M. Gardner, *Phaselock Techniques*, 3rd ed., Hoboken, NJ: John Wiley & Sons, 2005.
- [14] K. Takinami, R. Strandberg, P. C. P. Liang, G. Le Grand de Mercey, T. Wong and M. Hassibi, "A Distributed Oscillator Based All-Digital PLL With a 32-Phase Embedded Phase-to-Digital Converter," *IEEE J. Solid-State Circuits*, vol. 46, no. 11, pp. 2650-2660, November 2011.
- [15] Z. Bai, X. Zhou and R. Mason, "A Novel Injection Locked Rotary Traveling Wave Oscillator," in *2014 IEEE Symposium on Circuits and Systems (ISCAS)*, Melbourne, VIC, 2014.

- [16] R. B. Staszewski, D. Leipold, K. Muhammad and P. T. Balsara, "Digitally Controlled Oscillator (DCO)-Based Architecture for RF Frequency Synthesis in a Deep-Submicrometer CMOS Process," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 50, no. 11, pp. 815-828, November 2003.
- [17] Z. Bai, X. Zhou, R. D. Mason and G. Allan, "A 2-GHz Pulse Injection-Locked Rotary Traveling-Wave Oscillator," *IEEE Trans. Microw. Theory Tech.*, vol. 64, no. 6, pp. 1854-1866, June 2016.
- [18] "CCHD-575 Ultra-Low Phase Noise Oscillator," Crystek Corporation, Fort Myers, FL, April 2017.
- [19] R. B. Staszewski, C.-M. Hung, N. Barton, M.-C. Lee and D. Leipold, "A First RF Digitally-Controlled Oscillator for Mobile Phones," *2005 IEEE Radio Frequency integrated Circuits (RFIC) Symposium - Digest of Papers*, pp. 119-122, 2005.
- [20] W. Kester, "Converting Oscillator Phase Noise to Time Jitter," Analog Devices, Inc., 10 August 2009. [Online]. Available: <http://www.analog.com/media/en/training-seminars/tutorials/MT-008.pdf>. [Accessed 4 March 2017].
- [21] L. Xu, S. Lindfors, K. Stadius and J. Ryyänen, "A 2.4-GHz Low-Power All-Digital Phase-Locked Loop," *IEEE J. Solid-State Circuits*, vol. 45, no. 8, pp. 1513-1521, August 2010.
- [22] I. Varun and T. K. Gupta, "Ultra-Low Power NAND Based Multiplexer And Flip-Flop," *2013 Nirma University International Conference on Engineering (NUiCONE)*, pp. 1-5, 2013.

- [23] Z. Bai, "A Novel Clock Distribution System with Injection Locked Rotary Traveling Wave Oscillator and Built-In Self-Test," Ottawa, Canada, 2014.
- [24] R. Mason, X. Zhou, R. Smith and E. Cathcart, "An RTWO Based All-Digital PLL with Low Phase Noise," unpublished.
- [25] "ML605 Hardware User Guide," Xilinx, Inc., October 2012.
- [26] "Virtex-6 FPGA Data Sheet: DC and Switching Characteristics," Xilinx, Inc., March 2014.
- [27] L. R. Rabiner, J. F. Kaiser, O. Herrmann and M. T. Dolan, "Some Comparisons Between FIR and IIR Digital Filters," *The Bell System Technical Journal*, vol. 53, no. 2, pp. 305-331, February 1974.
- [28] Y. C. Lim, S. Xu and J. W. Lee, "Recursive Filters for Compensating the Mismatch in Time-Interleaved ADCs," in *2015 IEEE International Conference on Digital Signal Processing (DSP)*, Singapore, 2015.
- [29] V. Lesnikov, T. Naumovich and A. Chastikov, "The Formulation of Criteria of BIBO Stability of 3rd-order IIR Digital Filters in Space of Coefficients of a Denominator of Transfer Function," in *2014 East-West Design & Test Symposium (EWDTS)*, Kiev, 2014.
- [30] M. K. Soni, R. Mehra and R. Kumar, "Design, Performance and Cost Analysis of Various Band pass IIR Filters for Myriametre Band Applications," in *2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)*, Amritsar, 2015.

- [31] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," *Transactions of the A.S.M.E.*, vol. 64, no. 11, pp. 759-768, November 1942.
- [32] "FMC XM105 Debug Card User Guide," Xilinx, Inc., June 2011.
- [33] "SN74AVCH16T245 16-Bit Dual-Supply Bus Transceiver with Configurable Level-Shifting / Voltage Translation and Tri-State Outputs," Texas Instruments Incorporated, Dallas, TX, November 2015.
- [34] T. A. D. Riley, N. M. Filiol, Q. Du and J. Kostamovaara, "Techniques for In-Band Phase Noise Reduction in $\Delta\Sigma$ Synthesizers," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 50, no. 11, pp. 794-803, November 2003.
- [35] X. Gao, E. A. M. Klumperink, P. F. J. Geraedts and B. Nauta, "Jitter Analysis and a Benchmarking Figure-of-Merit for Phase-Locked Loops," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 2, pp. 117-121, February 2009.
- [36] A. Samarah and A. C. Carusone, "A Digital Phase-Locked Loop With Calibrated Coarse and Stochastic Fine TDC," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1829-1841, August 2013.
- [37] Y. Wu et al., "A 3.5–6.8-GHz Wide-Bandwidth DTC-Assisted Fractional-N All-Digital PLL With a MASH $\Delta\Sigma$ -TDC for Low In-Band Phase Noise," *IEEE J. Solid-State Circuits*, vol. 52, no. 7, pp. 1885-1903, July 2017.
- [38] S. Kim et al., "A 2 GHz Synthesized Fractional-N ADPLL With Dual-Referenced Interpolating TDC," *IEEE J. Solid-State Circuits*, vol. 51, no. 2, pp. 391-400, February 2016.