An Embedded Probabilistic Test Instrument for Built-In Self-Test Methods

Steven Bielby

steven.bielby@mail.mcgill.ca

260328492

March 2015

Advisor: Gordon W. Roberts

Electrical and Computer Engineering Department

3480 University Street Room 642

Montreal, Quebec

Canada H3A 2A7

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements for the degree of Master of Engineering

McGill University

Abstract

Faculty of Engineering

Department of Electrical Engineering

Master of Engineering

An Embedded Probabilistic Test Instrument for Built-In-Self-Test Methods

by Steven Bielby

As circuits become increasingly complex and testing time continues to increase, it is becoming more and more important to use built-in self-test techniques to ensure that the circuit is working according to its data sheet specifications. This thesis presents an embedded probabilistic test instrument in an IBM 130 nm CMOS technology, which allows for quick and easy probabilistic test evaluation of the bit-error ratio of a device-under-test. The proposed probabilistic test instrument can be used to inject a multitude of test signals into a device under test including clock signals, AC signals, and signals with jitter modulated onto the clock edges.

The device under test's performance is evaluated by extracting the output signal's edge locations. Using statistical methods, the probability distributions associated with the edge locations can be used to calculate the bit error-ratio that can be expected for the device under test. At the core of this work is the use of Sigma-Delta modulation techniques in combination with delay-locked loops to create a timing strobe that is finely and accurately controllable in order to be able to sample the eye-diagram of the device under test in various locations to extract the edge-location data. All of the timing strobes and stimulation signals as programmable in software in order to simplify the hardware design, as well as the work of the test engineer. The probabilistic test instrument was fabricated using IBM CMOS 130 nm technology and, in conjunction with a custom-designed PCB, the design was tested to ensure proper operation. The performance of the test instrument was evaluated by comparing the simulations to the measured results, as well as using external measurement devices to confirm on-chip measurements.

Résumé

De nos jours, les circuits deviennent de plus en plus complexes et le temps pour les tester continu d'augmenter. De ce fait, il devient de plus en plus primordial d'utiliser un mécanisme d'autotest qui sera intégré dans les circuits, afin que ceux-ci soient capables de s'assurer que leur fonctionnement est conforme à leurs fiches de données. Cette thèse introduit un instrument de probabilité qui utilise la technologie IBM CMOS 130 nm. Cet équipement permet une évaluation probabiliste en utilisant la fréquence d'erreurs binaires de l'appareil qui est sous test et ce, rapidement et facilement. L'instrument proposé ci-dessus peut être utilisée afin d'injecter une multitude de signaux tests dans l'appareil sous test. Ces signaux comprennent des signaux d'horloge, des signaux alternatifs, et des signaux gigues qui sont modulés avec les bords d'horloge.

L'appareil testé pour sa performance, est ensuite évalué par extraction de l'emplacement du bord du signal de sortie. En utilisant des méthodes statistiques, les distributions de probabilités associées aux emplacements des bords du signal peuvent être utilisées afin de prédire la fréquence d'erreurs. Le cœur de ce travail est l'utilisation des techniques de modulation Sigma-Delta en combinaison avec une "delay-locked loop" pour créer un signal de synchronisation qui est précisément contrôlable afin d'être capable de prendre des échantillons dans différents endroits. L'instrument de test probabiliste a été fabriqué en utilisant la technologie IBM CMOS 130 nm en conjonction avec un circuit PCB fait sur mesure. La performance de l'instrument a été évaluée en comparant les résultats de mesure avec les simulations initiales ainsi qu'avec l'utilisation de dispositifs de mesure externes.

Acknowledgements

First of all, I would like to express my gratitude towards Professor Gordon Roberts for his guidance, patience, encouragement, advice, and the freedoms that he granted me throughout my time as his student. I have been extremely lucky to have a professor who cares so much about his students and their well-being, as well as being very involved and engaged in every aspect of my research. Secondly, I would also like to also thank all of my colleagues in the Integrated Microsystems Lab for all their wisdom and support. Specifically, I would like to thank Omar Abdelfattah and Dong An for their technical knowledge, Steven Ding for his soldering expertise, Adam Gordon and Ming Yang for their support throughout the research process, and finally Pascale de R-P for her aid in the translation of parts of this thesis.

Next, I would like to thanks McGill University and Natural Sciences and Engineering Research Council of Canada for their financial support throughout my studies and my research.

Finally, I would like to thank my mother Karen, father Greg, and brother Chris for their unwavering support throughout my academic career. Without their love and support, this research would not have been possible.

Contents

Abstract **i**

Résumé **ii**

Acknowledgements **iii**

Contents **iv**

List of Figures **vii**

List of Tables **xi**

Abbreviations **xii**

1 Introduction **1**

1.1 Designing to Reduce Test Time 2
1.2 Scope of Thesis 3
1.3 Background on Probabilistic Test Methods 5
1.3.1 Amplitude-Based Scan Test 7
1.3.2 Time-Based Scan Test 9
1.3.3 Issues with Present-Day Test Methods 10
1.4 Thesis Outline 11

2 Background of the Building Blocks **12**

2.1 D-Flip-Flop 12
2.2 Delay-Locked Loop 13
2.2.1 Phase Detector 18
2.2.2 Charge Pump 20
2.2.3 Loop Filter 22
2.2.4 Voltage-Controlled Delay Line 22
2.3 Sigma-Delta Modulator 23
2.4 Circular Shift Register 26
2.5 Digital-to-Time Converter 27
2.6 Frequency Divider 29
2.7 Voltage-Mode Filter 30

2.8 Time-Mode Filter 31
2.9 Counters 33
2.10 PLL 33
2.11 Summary 35

3 Sigma-Delta Phase Generation Technique 36
3.1 Overview 36
3.2 Phase Encoding 38
3.3 MATLAB/Simulink Simulations 39
3.4 Summary 41

4 Delay-Locked Loop as a Time-Mode Filter 44
4.1 Method for Designing Higher-Order Delay-Locked Loops 45
4.2 Start-Up Circuit 48
4.3 Sub-Gate-Delay Edge-Control of a Clock Signal Using DLLs and Sigma-Delta Modulation Techniques 50
4.3.1 Background 50
4.3.2 Simulink Ideal Simulations 53
4.3.3 Transistor-Level Simulations 56
4.4 Summary 57

5 Architecture of the Probabilistic Test Instrument 58
5.1 Probability Test Instrument System Architecture 58
5.2 Multi-Channel Programmable Bitstream Generator 59
5.3 Probability Extraction Unit 64
5.4 Voltage-mode Filter Design 66
5.5 DTC into Software 68
5.6 Counters 70
5.7 Coarse-Locking and Fine-Locking 73
5.8 Set-up Signals 76
5.9 Design of the Comparator 77
5.10 Practical Design Challenges and Considerations 80
5.10.1 Inverter Sizing 81
5.10.2 Multiplexer Design 81
5.10.3 Real-World Considerations 82
5.11 Printed Circuit Board Design 84
5.12 Simulation Results 86
5.12.1 Delay-Locked Loop 87
5.12.2 Counter 89
5.12.3 Phase-Locked Loop 90
5.12.4 Circular Shift Register 92
5.12.5 Coarse and Fine Locking 92
5.12.6 Summary 94

6 Experimental Results 95
6.1 Experimental Setup 95
6.2 DLL Time-Mode Filter 96
6.3 Probabilistic Test Instrument 104

6.3.1 Circular Shift Register . . . 106
6.3.2 Input Channel 1 and 2 . . . 106
6.3.3 Input Channel 3 . . . 107
6.3.4 Input Channels 4 and 5 . . . 109
6.3.5 Probability Extraction Unit . . . 110
6.3.6 Evaluation of the Performance of the PTI . . . 111
6.3.7 Summary . . . 115

7 Conclusion 117
7.1 Thesis Contribution . . . 117
7.2 Future Work . . . 119

A PTI Pin Names and Descriptions 120

Bibliography 126

List of Figures

1.1 Possible layout of die containing BIST circuitry 2
1.2 Principle operation of a parallel CDF-extraction circuit. 4
1.3 Principle operation of a serial CDF-extraction circuit. Multiple measurements must be taken with varying ΔT to construct the CDF. 5
1.4 BER test setup 6
1.5 An eye-diagram of a received signal generated from a random bit sequence. Superimposed on this figure are the voltage and timing decision levels. . . 8
1.6 The relationship between threshold voltage and BER 9

2.1 D-flip-flop design 13
2.2 Type-I DLL 14
2.3 Type-II DLL 14
2.4 Type-I DLL linear continuous-time model 14
2.5 Type-I DLL with added delay 14
2.6 Type-II DLL architecture 17
2.7 Type-II DLL linear continuous-time model 17
2.8 XNOR logic implementation as a phase-detector 19
2.9 Phase-detector example signals 19
2.10 PDF design 20
2.11 PFD example signals 20
2.12 Charge-pump architecture 21
2.13 1^{st}-order loop filter for a DLL 22
2.14 Voltage-control delay line architecture 23
2.15 General block diagram of sigma-delta modulator 24
2.16 Spreading of quantization noise in an oversampled sigma-delta system . . 25
2.17 Noise shaping of the sigma-delta ADC 26
2.18 SNR vs. OSR for different orders of sigma-delta modulators 26
2.19 Circular shift register implementation using D-flip-flops 27
2.20 Architecture for a basic digital-to-time converter 28
2.21 Example DTC waveform output for a select sequence of 0,1,2,3 29
2.22 Frequency dividing by 4 circuit 30
2.23 Comparison between 4 different orders of butterworth voltage-mode low-pass filters 31
2.24 Comparison between time- and voltage-mode signal representations 32
2.25 Asynchronous counter architecture 33
2.26 Block diagram of a phase-locked loop 34

3.1 Software and hardware breakup of the signal generation module 37

3.2 Simulink block diagram of sigma-delta modulator 40
3.3 Power spectral density plot for 27.9 kHz signal modulated with an OSR of 2048 41
3.4 Sigma-delta encoded 447.625 kHz sine wave with OSR of 128 42
3.5 Time-domain representation of a sigma-delta encoded DC code of 0.2 with an over-sampling ratio of 500 43

4.1 Example of an incorrectly starting DLL 48
4.2 DLL start-up circuit to ensure proper locking conditions 49
4.3 System diagram with a DLL reconstruction filter 50
4.4 Illustrating the phase extraction process when the DLL extracts the average phase between the densities of two coarse delays 51
4.5 Implementation of phase signal generation scheme. Circular memory contains the sigma-delta encoded signal in the phase domain and the DLL is used to extract it. 51
4.6 Ideal and measured responses of the designed DLL 53
4.7 Sigma-delta encoding of the DC code level is done in software and it is filtered by the DLL implemented in hardware 53
4.8 DC code-controlled phase offset. This curve is a locus of points with a resolution of 2.44 ps due to the maximum use of 1024 bits. 54
4.9 RMS jitter of the edges with a full sigma-delta bit sequence. A DC Code delta of 0.05 was used. 54
4.10 RMS jitter of the edges with a truncated bit sequence to 1024 bits. A DC Code delta of 0.05 was used. 55
4.11 Effect of the loop bandwidth of the DLL on the peak-to-peak jitter of the output signal. 55
4.12 Input code level-controlled edge location obtain from the transistor level implementation compared to the ideal truncated output. A DC Code delta of 0.05 was used. 56
4.13 Histogram of rising edge locations for a settled DLL for an input DC code of 0.6. 57

5.1 Overall system design of the PTI 60
5.2 Multi-channel programmable bitstream generator 61
5.3 Multiplying PLL used to properly clock the circular shift register 62
5.4 Threshold-voltage and timing-strobe used to control the sampling location in the eye diagram 63
5.5 Architecture of the probability extraction unit 64
5.6 Timing diagram for the PEU for two different threshold values 65
5.7 Timing strobe sampling phase and the extracted CDF and PDF waveforms 67
5.8 Design of the voltage-mode filter block for input channel 3 68
5.9 Single-bit DTC 69
5.10 Separation of the software and hardware components of the DTC 70
5.11 Parallel-loading shift register implemented to rotate right the counted bits 71
5.12 Controlling circuit to manage the timing between counting and shifting . 71
5.13 Waveforms of the controlling circuit to transition between counting and shifting 72
5.14 Auto-resetting circuit to ensure proper counting 72

5.15 Architecture of the automatic coarse-locking circuit 73
5.16 Waveforms of the automatic coarse-locking circuit demonstrating the phase range of operation 74
5.17 Block diagram for the voltage comparator 78
5.18 Transistor design of general-purpose comparator 78
5.19 Bias circuit for the comparator 78
5.20 The voltage transfer curve for the voltage comparator with the threshold voltage held at 0.6 V and the input swept from 0 to 1.2 V 80
5.21 The gain of the voltage comparator 80
5.22 The operation of the comparator across a range of threshold voltages from 0.1 V to 1.1 V in 100 mV increments while sweeping each input from 0 V to 1.2 V. 81
5.23 Design of the multiplexer. 82
5.24 Design of the electrostatic discharge protection circuit. 83
5.25 PCB "Load Bits X" signal input circuit 84
5.26 Clock debouncing circuit design 85
5.27 Design of the custom PCB to test the PTI 86
5.28 Voltage-controlled delay line transfer characteristic for a 28.57 MHz clock 88
5.29 Loop filter voltage during the settling of a DLL 88
5.30 Locking of the input and output signals of the DLL 89
5.31 The loop filter voltage and waveforms during the settling of the DLL with a 0^o phase shift 90
5.32 The loop filter voltage and waveforms during the settling of the DLL with a 90^o phase shift 90
5.33 The waveforms for the outputs of the multiplying by 2 PLL 91
5.34 VCO voltage-frequency relationship used in the PLL 91
5.35 The waveforms of the circular bitstream after a bitstream of "11000" is loaded and connected circularly 92
5.36 The waveforms of the circular bitstream after coarse-locking and a 45^o phase-shift 93
5.37 The control voltages in the coarse-locking circuit illustrating proper operation 94

6.1 Experimental setup for the testing of the PTI 96
6.2 The fabricated 0.13 μm CMOS die 97
6.3 Packaging of the 0.13 μm CMOS die 98
6.4 DLL loop filter voltage when changing the input DC code from 1 to 0. . . 99
6.5 DC-code input and the corresponding edge locations at the output of the DLL time-mode filter 100
6.6 Edge location histogram for a DC input code of 0.75 100
6.7 The peak-to-peak deviation of the edge locations for a first-order DLL . . 101
6.8 The standard deviation of the edge locations for a first-order DLL 101
6.9 Relationship between the input DC code and the phase of the output when a 45^o phase-encoding is used. 103
6.10 The peak-to-peak deviation of the edge locations for a first-order DLL with a 45^o phase-encoding 103
6.11 The standard deviation of the edge locations for a first-order DLL with a 45^o phase-encoding 104

6.12 The effects of different loop bandwidths on the standard deviation of jitter on the output signal with an input phase-encoding of 90^o. 104
6.13 The effects of different loop bandwidths on the standard deviation of jitter on the output signal with an input phase-encoding of 45^o. 105
6.14 Bench test setup for measuring the fully integrated PEU in a 130 nm CMOS process technology. 105
6.15 Testing of the circular shift register 106
6.16 Output waveform from the circular shift register following the loading of a "1100" repeating bit sequence and clocking with the multiplying PLL . 107
6.17 Sigma-delta encoded 447.265 kHz waveform at the output of the 256-bit circular shift register for input channel 3 108
6.18 Sigma-delta encoded 447.265 kHz waveform at the output of the 256-bit circular shift register for input channel 3 following a digital buffer 109
6.19 Output waveforms from the counter that are being used to extract the CDF corresponding to the DUT for a control voltage of 0.82 V (-20.05^o relative to VC_{OUT} edge) 112
6.20 Probability that $T_{DUT} < T_{STRB}$ where T_{STRB} is varied across 90 degrees 112
6.21 PDF that $T_{DUT} < T_{STRB}$ where T_{STRB} is varied across 90 degrees. This is the derivative of Figure 6.20 which represents the CDF. 114
6.22 DUT edge locations measured with an external oscilloscope compared to those measured with the PTI. External measurements are buffered through a digital buffer prior to measurement. 114
6.23 CDF for the measurements taken both on- and off-chip. 115

A.1 Pin locations on the CQFP80A package 125

List of Tables

3.1 Bit Encoding for Different Phases of DTC 39
3.2 Possible Relationships Between Encoded Signals 39
3.3 Input AC signal frequencies for sigma-delta encoding 39
3.4 Sigma-Delta Filter Functions 40
3.5 The Sigma-Delta Bit-Encoding for Various DC Codes 41

5.1 Clocking Frequencies for the Circular Bitstreams for each Input Channel . 63
5.2 Resistor Values for Voltage-Mode Filter Bank 68
5.3 Select Signals for Corresponding Voltage-Mode Filter Bank Cutoff Frequency 68
5.4 Input Code and Corresponding Phase Shift with Respect to DUT Signal . 75
5.5 Transistor sizing for the voltage comparator 79
5.6 Transistor sizing for the voltage comparator bias circuit 79
5.7 Transistor sizing for the multiplexer in Figure 5.23 82
5.8 Voltage-Control Delay Line Transfer Characteristic for a 28.57 MHz Clock 87

6.1 DC Code Sigma-Delta Representation 99
6.2 Probability that $T_{DUT} < T_{STRB}$ where T_{STRB} is varied across 90 degrees 111

Abbreviations

ATE	**A**utomated **T**est **E**quipment
BER	**B**it **E**rror **R**atio
BIST	**B**uilt-**I**n **S**elf-**T**est
BSG	**B**it**S**tream **G**enerator
CDF	**C**umulative-**D**istribution **F**unction
DAC	**D**igital to **A**nalog **C**onverter
PEU	**P**robability **E**xtraction **U**nit
PTI	**P**robabilistic **T**est **I**nstrument
DLL	**D**elay-**L**ocked **L**oop
DTC	**D**igital-to-**T**ime **C**onverter
ESD	**E**lectro**s**tatic **D**ischarge
OSR	**O**ver-**S**ampling **R**atio
PCB	**P**rinted **C**ircuit **B**oard
PDF	**P**robibility **D**ensity **F**unction
PLL	**P**hase-**L**ocked **L**oop
PRBS	**P**seudo-**R**andom **B**it **S**equence
SNR	**S**ignal-to-**N**oise **R**atio
VC	**V**oltage **C**omparator
VCDL	**V**oltage-**C**ontrolled **D**elay **L**ine
VCO	**V**oltage-**C**ontrolled **O**scillator

Dedicated to my loving parents, Karen and Greg.

Chapter 1

Introduction

Complimentary metal-oxide-semiconductor (CMOS) is a technology that is used to build integrated circuits. These circuits are what make up the majority of today's microprocessors, RAM, image sensors, and other analog and digital circuits. The CMOS fabrication process is a very mature process that involves a series of photo-lithographic printing, etching, and doping steps. Like any real-world process, the CMOS fabrication process is not perfect and it is subject to blemishes and imperfections that can cause failures or undesired variations in performance of the design which, when shipped to a customer, will not perform as expected. For these reasons, it is important to test each and every device after manufacturing to verify the device operates as specified in its device data sheet. Testing the devices is a very expensive process, but it is a necessary part of the development cycle of a CMOS chip. Without a proper test, the product's value would be significantly decreased due to the unknown operating characteristics of the device.

Today, the cost of testing represents a large percentage of the total production cost of a design [1]. Moreover, as transistor sizes decrease, chips are being made with greater number of transistors and, on top of that, designs are becoming more complex with greater functionality and performance. This means that with the maturation of the fabrication process, manufacturing costs will decline, however relative test costs will continue to increase [1]. The cost of testing is measured by test cost per second, multiplied by the test time

$$\text{cost of test} = \text{test cost per second} \times \text{test time (seconds)} \tag{1.1}$$

The cost per second that it costs to test a design is more-or-less a fixed cost that involves that cost of an automated test equipment (ATE) machine. The simplest way to reduce the cost of testing a design is to reduce the amount of time that it takes to test it. The is the job of the test engineer and their role is to guarantee the performance of each device after it is fabricated, in the shortest amount of time. It is in the context of reducing the overall test time, as well as achieving correct test results, that this thesis has been written.

1.1 Designing to Reduce Test Time

With the knowledge that a product will be tested after fabrication, there are different design techniques that can greatly reduce the amount of testing time required to properly test the device. The two techniques are called design for test (DFT) and built-in self-test (BIST) [1]. DFT is a major topic of interest in automated testing environments. The idea with a DFT system is to make the design simpler to test, and also have a more thoroughly testable product. During the design phase, extra built-in circuits are added to the design in order to reduce the testing complexity, or make a test signal available outside of the device.

BIST testing is different from DFT whereby the BIST circuits allow for the device to evaluate its own performance without the need for an expensive additional ATE. A BIST circuit will usually return a pass/fail signal that allows the testing process to quickly and easily determine of the device is working properly or not. Since the testing is being done by the device itself, it is then possible to utilize a much less expensive ATE tester [1]. Figure 1.1 illustrates a possible die layout of a BIST circuit.

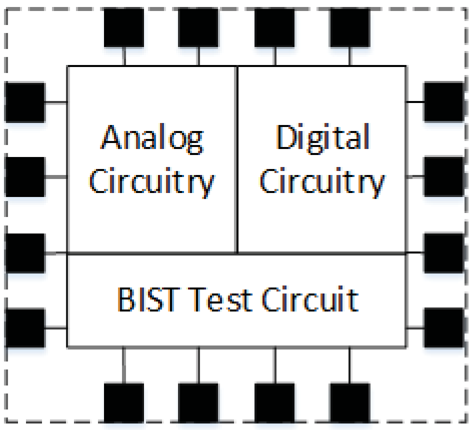


FIGURE 1.1: Possible layout of die containing BIST circuitry

As the market moves towards more system-on chip (SoC) designs, the introduction of embedded test circuits in the design becomes more and more important. As there are limited I/O pins available on a die, it is impossible to route input and outputs access points to each and every block in the design. In this situation, the motivation for using embedded test techniques (DFT and BIST) surpasses simply being able to save testing time but is essential to ensure a proper functioning of the chip as a whole. While the introduction of a test bus [2] can alleviate some of these access issues, it does so at the expense of additional signal integrity issues such as inter-symbol interference (ISI) and noise effects. Whereas, including embedded instruments directly on-chip eliminates many of these signal integrity issues, enables validation of many of the critical elements of the design and, most importantly, does so at low cost. In addition, testing probes have limited bandwidth capabilities, so it is not always possible to test a chip at the full operating frequency without introducing bandwidth-limiting effects in the measurements. In this situation, testing on the chip itself and asserting a pass/fail signal becomes very advantageous.

There are, however, concerns with the introduction BIST test circuits as the measure of performance is being measured by a circuit that itself has not been tested to ensure proper functionality. Currently, the ATE testers that are used for such testing are precisely calibrated to ensure proper measurements, but with BIST circuits, this is not possible. For this reason, BIST circuits are designed to have very low complexity, and very high reliability.

1.2 Scope of Thesis

Transceiver circuits are found in a multitude of communication applications that are used on a daily basis such as mobile phones, computers, and telecommunication equipment [1]. Testing a transceiver circuit and the corresponding communication channel can be an expensive process for an external ATE tester to undertake given the lengthy tests that must be performed to ensure proper operation of the device. To keep the test costs down, a low-cost test alternative must be used, and one method is the implementation of an embedded BIST circuit. The implementation of a BIST circuit into the design comes at the cost of silicon overhead, but reduced test time, as well as being able to test the circuits at their full operating frequencies is gained. This is because test equipment channels are often bandwidth limited so the real full-speed performance is often difficult to measure [1]. In this thesis, a fully-programmable embedded probabilistic test instrument is proposed to reduce test time and cost.

A probabilistic test instrument greatly reduces the time required to test a device under test (DUT) and its corresponding communication channel. The performance metric that is used to test a communication channel is the bit error ratio (BER) and this signifies the average number of transmission errors,μ_{N_E}, versus the number of total bits transmitted, N_T [1], i.e.

$$\text{BER} = \frac{\mu_{N_E}}{N_T} \tag{1.2}$$

Measuring the BER can be done by running through a large number of bits and comparing the received with the transmitted bits and determining the correctness of each one. This method, however is cost-prohibitive, so a probabilistic model is used which greatly reduces the number of bits that have to be transmitted. A more detailed explanation of the probabilistic test is described in Section 1.3. By extracting statistical information regarding the transition locations of the edges of the individual bits, it is possible to estimate the BER of of the communication channel. A simple hardware circuit implementation will generate a cumulative distribution function (CDF) of the transition edges which can be translated into a projected BER for the channel [1].

There are two ways of extracting the CDF of the transition edges: parallel or serial. The parallel implementation uses one N-element delay line, and N counters. A counter timing strobe is tapped out of the delay line after each delay element and attached to the timing strobe input of each counter. The result is that each counter will sample a different time-instant of the eye diagram, all one delay-element (ΔT) apart, as illustrated in Figure 1.2. Here, the D-flip-flop is used to sample the DUT output signal, with each flip-flop connected to a different phase of the delay line. A counter is then used for each flip-flop to count the number of edge transitions. This implementation saves time, but at the expense of a greater silicon area [3]. In contrast, the serial implementation is much smaller, but requires more time to extract the CDF [1]. Figures 1.2 and 1.3 illustrate the design of the parallel and serial CDF measuring circuits.

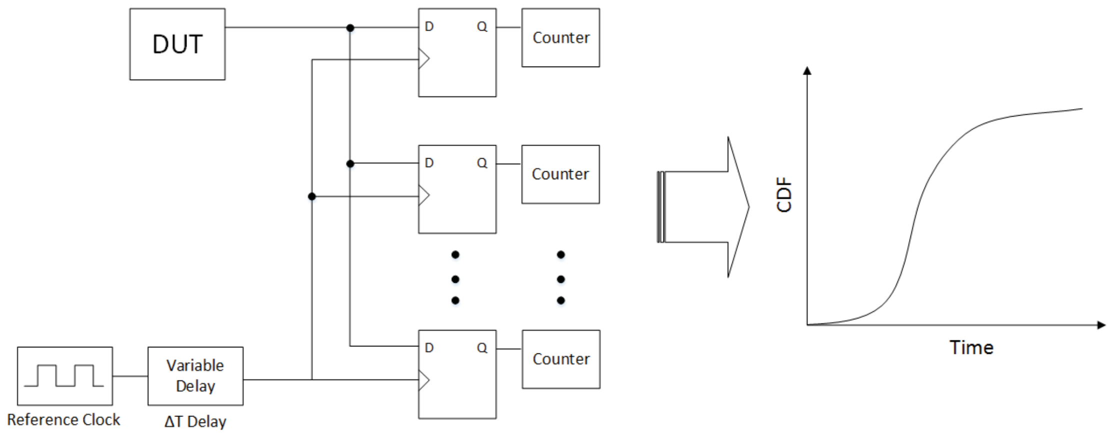


FIGURE 1.2: Principle operation of a parallel CDF-extraction circuit.

The principle of the serial implementation is similar to the parallel realization but uses a single transition detector and counter, and a variable delay element. As illustrated in Figure 1.3, using a variable delay element, the timing strobe is set to different time-instants in the eye-diagram, and the total number of signal transitions is counted for each

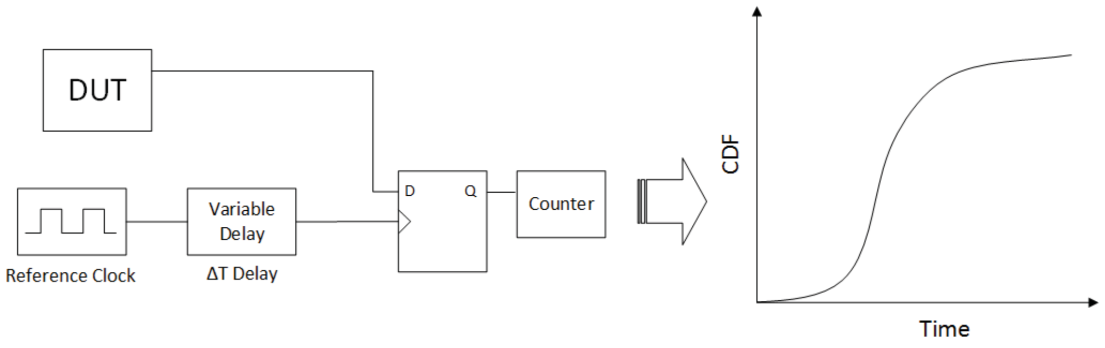


FIGURE 1.3: Principle operation of a serial CDF-extraction circuit. Multiple measurements must be taken with varying ΔT to construct the CDF.

time instant. The test is repeated multiple times until each time-instant is extracted and the CDF can be created.

In this thesis, the hardware design and implementation of the probabilistic test instrument will be covered. The design will be implemented in IBM CMOS 0.13 μm technology using Cadence and Virtuoso design tools. Simulations for the ideal cases will be conducted, and following the fabrication of the design, the experimental results will be collected and analysed.

1.3 Background on Probabilistic Test Methods

The performance measurement for a communication system is how many transmission errors occur in a transmission of a given length. This is called the bit error ratio (BER) and is defined in Equation 1.2 where μ_{N_E} is the number of transmission errors and N_T this the total number of bits transmitted. Following Equation 1.2, given the BER and number of bits transmitted, the number of expected transmission errors over this channel can be computed. Thus, the BER can be understood as the probability that an error will occur, where

$$\text{BER} = P_e \tag{1.3}$$

The BER can therefore be modelled as a random variable with mean μ_{BER} and standard deviation σ_{BER}. The most basic way of testing for a BER is by simply comparing the received signal with the transmitted signal and comparing the two and measuring the number of transmission errors that are detected over a certain amount of time. Figure 1.4 illustrates this setup. However, as the BER is defined as an average based on the number of transmitted bits, a multitude of similar tests must be conducted to compute an average value [1]. Additionally, extracting the average from a random variable with a finite number of measurements will always have a level of uncertainty with it [1]. The greater the number of transmitted bits, the greater the level of certainty of the measurement. As a general rule, to achieve a high level of confidence in the measurements, the number of bits transmitted is approximately 10 times the reciprocal of the BER. For example, to determine if a channel is able to achieve a BER of 1×10^{-14}, it is necessary to transmit approximately 10×10^{14} bits.

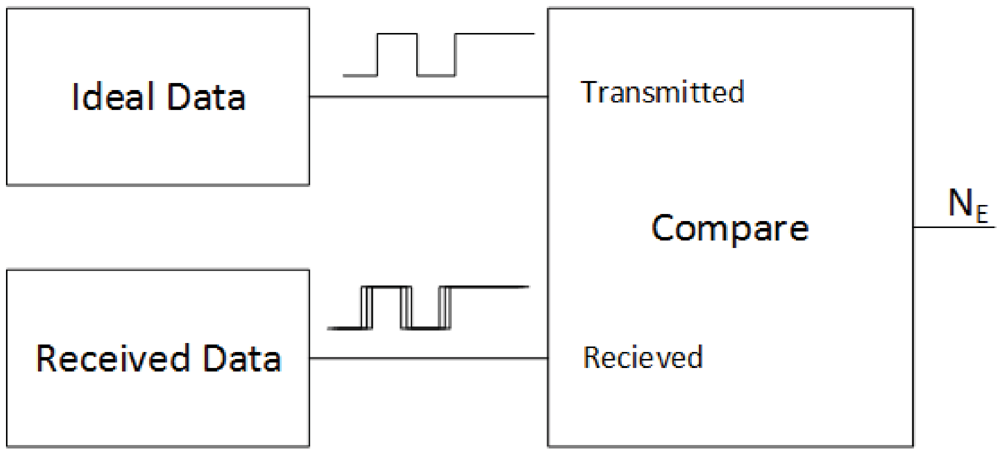


FIGURE 1.4: BER test setup

$$\text{test time} = \frac{\text{number of bits}}{\text{clock speed}} \tag{1.4}$$

If a system operates at a clock speed of 1 GHz, according to Equation 1.4 the test would take

$$\frac{10 \times 10^{14}}{1 \times 10^{9}} = 1 \times 10^{6} s \tag{1.5}$$

The time outlined in Equation 1.5 is prohibitively long and can not be used as a production test. For this reason, a number of different approaches are taken to shorten the amount of test time in the production environment. In this thesis, two of these techniques that are used to estimate the BER that are based on the edge locations will be covered: amplitude-based scan test and time-based scan test. There are two other BER test methods that can be used: the Dual-Dirac jitter decomposition method, and the Gaussian mixture jitter decomposition method [1]. These test methods follow a similar implementation to the time- and amplitude-based tests, but are only used when the input stimulus signals introduce a data-dependent jitter component; for example when a PRBS is used as a stimulus signal. The Dual-Dirac and Gaussian mixture jitter decomposition tests are used to separate the deterministic jitter from the random jitter to evaluate the performance of the DUT. In this thesis, an simple '1010' stimulus signal is being used in the place of a PRBS, so only the amplitude- and time-based scan tests are necessary to evaluate the performance of the DUT. In a future work, if a PRBS test pattern is used, the utilization of such test methods would be appropriate.

For lack of a better statistical model at this time, the distribution of the edge distribution will be assumed to be Gaussian with mean μ_{robust} and standard deviation σ_{robust}. It is often assumed that the variation in the edge transition is caused by uncorrelated unbounded random noise alone. As such, it is common in the industry to refer to σ_{robust} as the random jitter (RJ) component, denoted as σ_{RJ}.

In this work, we make use of the robust mean and sigma method whereby they are calculated by utilizing the first and third quartile values from a Gaussian distribution [4] (denoted as Q1 and Q3) which are gathered from the CDF result. More specifically, from the standard Gaussian distribution, the robust mean is easily calculated as

$$\mu_{robust} = Q1 + \frac{(Q3 - Q1)}{2} \tag{1.6}$$

and the robust sigma is found from

$$\sigma_{robust} = \frac{(Q3 - Q1)}{1.35} \tag{1.7}$$

1.3.1 Amplitude-Based Scan Test

The amplitude-based scan test uses an alternating "1010" pattern to ensure that no periodic pattern or pattern dependent jitter is present [1]. Assuming that each bit has the same probability of error and given the standard deviation of the edges, σ_N, it is possible to define the BER as a function of the threshold voltage, V_{TH}, that is used to sample the eye diagram as described as [1]

$$\text{BER}(V_{TH}) = \frac{1}{2} - \frac{1}{2} \times \Phi\left(\frac{V_{TH} - V_{Loic0}}{\sigma_N}\right) + \frac{1}{2} \times \Phi\left(\frac{V_{TH} - V_{Loic1}}{\sigma_N}\right) \tag{1.8}$$

Solving Equation 1.8, the minimum BER occurs when the threshold voltage is set to the middle of the two logic levels. This makes sense as this would offer the widest point of the eye opening that would be sampled.

The eye diagram that is being referred to is a time-denormalized interleaving of the transmitted bit periods. Adjusting the sampling time and sampling threshold voltages, it is possible to move the decision point into the most favourable position. The eye diagram in shown in Figure 1.5.

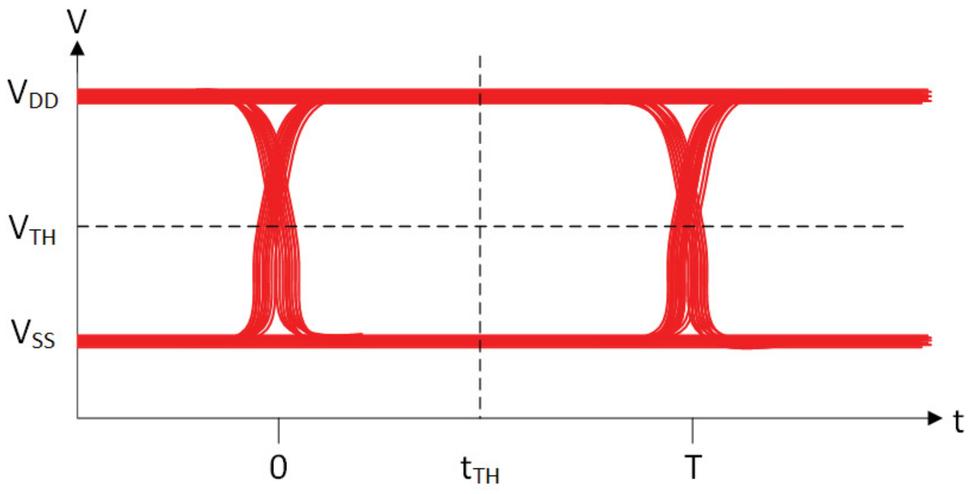


FIGURE 1.5: An eye-diagram of a received signal generated from a random bit sequence. Superimposed on this figure are the voltage and timing decision levels.

In the amplitude-based scan test, a first measurement is taken with the threshold voltage moved closer to the logic 1 voltage level so as to worsen the eye-diagram sampling and hence performance of the system. The reduction in the overall system performance will greatly reduce test time as the average number of bits transmitted to get a single error will greatly decrease. The logic level 0 is insignificant in this measurement so Equation 1.8 can be written as [1]

$$\text{BER}(V_{TH}) \approx \frac{1}{2} - \Phi\left(\frac{V_{TH} - V_{Loic1}}{\sigma_N}\right) \tag{1.9}$$

Next, this process in repeated a second time, but with the threshold voltage moved even closer to the logic 1 voltage level resulting in a degradation of performance and an even shorter test time. Two different performance measurements allow to solve for the two unknowns in Equation 1.9, V_{Logic1} and σ_{N_1}.

The two tests are repeated but with the V_{TH} moved closer to the logic 0 voltage level. Solving again for the two unknowns, V_{Logic0} and σ_{N_0} in this case, and averaging the two σ_N values, it is possible to substitute them into Equation 1.8 to calculate the BER. Additionally, the optimum threshold level for the best performance can be found to occur at [1]

$$V_{TH_{ideal}} = \frac{\sigma_{N_0} V_{Logic1} + \sigma_{N_1} V_{Logic0}}{\sigma_{N_0} + \sigma_{N_1}} \tag{1.10}$$

The idea of using differing threshold voltage levels is illustrated in Figure 1.6. As shown, the idea is to solve for the unknowns when the system is performing sub-optimally where a significantly fewer number of bits need to be transmitted. Using the results from these measurements, it is possible to estimate the BER when the sampling threshold voltage is in the ideal location.

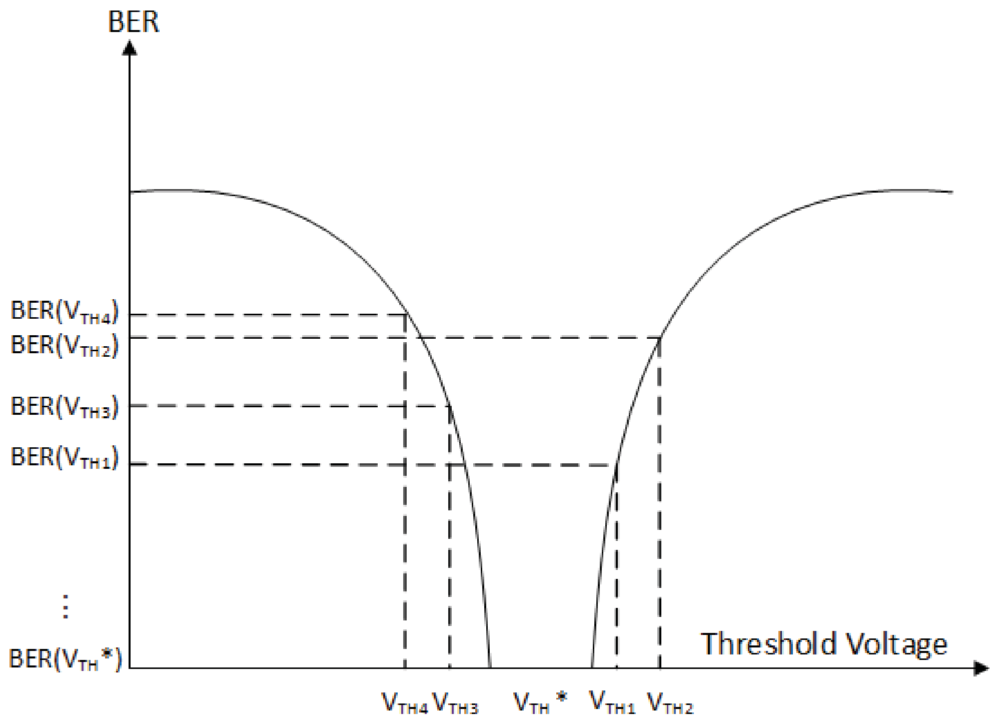


FIGURE 1.6: The relationship between threshold voltage and BER

1.3.2 Time-Based Scan Test

The time-based scan test is similar to the amplitude-based scan test where it uses statistical methods to shorten test time. Here, the BER depends on the voltage noise as well as the jitter of the edge of the sampling signal [1]. In this test, instead of varying voltage threshold levels, it is the sampling time that is altered. Referring to Figure 1.5, this corresponds to the adjustment of the location of the sampling time on the horizontal axis.

It has been shown in [1] that the BER of a channel can be approximated by extracting the RJ from the distribution of edge transitions at the receiver by varying the strobe time, t_{TH}. In doing so, the BER at any sampling instant (referred to as the BER bathtub curve) can be derived according to

$$\text{BER}(t_{TH}) = \frac{1}{2} - \frac{1}{2} \times \Phi\left(\frac{t_{TH} - 0}{\sigma_{RJ}}\right) + \frac{1}{2} \times \Phi\left(\frac{t_{TH} - T}{\sigma_{RJ}}\right) \tag{1.11}$$

Here T is assumed to be the bit duration of the receiver signal (see Figure 1.5). It follows that from Equation 1.11, that if a time sampling instant is chosen to be much less than $T/2$, then the BER expression can be written as

$$\text{BER}(t_{TH}) = \frac{1}{2} - \frac{1}{2} \times \Phi\left(\frac{t_{TH} - 0}{\sigma_{RJ}}\right) \tag{1.12}$$

Solving for σ_{RJ} and substituting back into Equation 1.11, it is then possible to estimate the minimum BER by making only one measurement [1].

1.3.3 Issues with Present-Day Test Methods

The tests outlined in Sections 1.3.1 and 1.3.2 are excellent methods for reducing test time in production, however there are also some issues with the tests. One major issue is the fact that the present-day instruments that are used to capture the transition edges has a limited bandwidth. Often times, the bandwidth is limited such that the measurements can not be taken at the full clock speed of the device, and a sub-sampling process must be used. Another drawback of a bandwidth-limited system is that any means to extract the high-speed signal will be impacted by the bandwidth of the extraction instrument [1]. At high-speeds, testing a communication channel can not be done properly with external equipment. The reason is that the signal integrity is compromised when probes are attached to the system and introduce additional loading effects [5].

Modelling the system to disregard the added loading effects of the probes is extremely difficult as it would need to account for the zero-crossing times as well as the introduced inter-symbol interference introduced by the probes. Thus, this thesis proposes an on-chip BIST test method that will use a probability extraction unit to make all the measurements on chip such that no external probe is needed for testing. This will allow the measurements to be taken at full-speed, and no additional loading effects will be introduced into the system when the measurements are taken so an accurate performance can be measured.

1.4 Thesis Outline

This thesis presents the overall design principles and hardware implementation of an embedded probabilistic test instrument. The following chapters will cover the implementation in greater detail. Chapter 2 provides a background of the basic building blocks that have been implemented in the probabilistic test instrument. Chapter 3 walks the reader through the sigma-delta modulator phase generation techniques. Chapter 4 describes the delay-locked loop (DLL) as a time mode filter and also goes into great detail about a new design method for high-order DLLs, and Chapter 5 describes the architecture of the probabilistic test instrument. Finally, Chapter 6 covers the experimental results, and Chapter 7 is the conclusion.

Chapter 2

Background of the Building Blocks

This chapter will describe the basic building blocks that will be used to construct the probabilistic test instrument (PTI). Each input will be generated by a circular shift register that will generate a variety of signals that will be used to stimulate the device under test (DUT). The signals are all individually programmable such that it is possible for the test engineer to adjust the test signal based on the application. Each circular shift register will either be followed by a voltage-more filter, or a time-mode filter depending on the nature of the input signal. By adjusting the filters, as well as the input bitstreams loaded into the registers, it is possible to achieve different input signals to stimulate and measure the performance of the DUT. At the core of the DUT test circuit is a DLL and circular shift register used to control the timing strobe to sub-gate delay resolution [6].

2.1 D-Flip-Flop

The D-flip flop is a very crucial component in this thesis as it is included in many of the basic building blocks of the PTI. The D-flip-flop that is used is a positive-edge triggered design with set and reset pins. A D-flip-flop will latch the value at the D input on the rising edge of the clock signal and it will be propagated to the output until the next rising clock edge. The design is made up of 6 3-input NAND gates for a total of 36 transistors in each flip-flop. Figure 2.1 shows the design of the D-flip-flop.

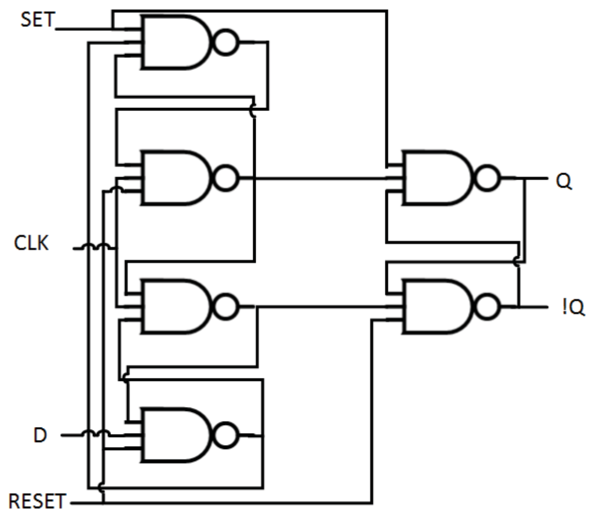


FIGURE 2.1: D-flip-flop design

2.2 Delay-Locked Loop

A delay locked loop (DLL) is a system whose output signal phase is related to the phase of the input signal. A DLL is very similar in design to a phase-locked loop (PLL), except that the voltage-controlled oscillator (VCO) is replaced with a voltage-controlled delay line (VCDL). Typically, PLLs have been more widely used in system architectures due to their ability to be used under many different conditions; from frequency multiplication, to clock-recovery, to signal demodulation [6]. DLLs have emerged as an alternative to PLLs for specific functions where only a single frequency is required. The advantages of a DLL over a PLL is that they are easier to stabilize, simpler to design, and they offer better jitter performance [7]. The jitter performance is the direct result of not having a VCO in the design which means that phase errors induced by supply noise, or component non-idealities, do not accumulate over time which results in a signal with a less jitter. Much of the work in this section follows work that was completed and published in [6].

There are two different types of DLLs that can be implemented. They differ from one another based on their inputs, and their jitter transfer characteristics. These will be referred to as Type-I and Type-II DLLs and are displayed in Figure 2.2 and Figure 2.3 respectively [8].

A DLL is an element where the amount of delay in the delay line is adjusted such that a desired phase relationship is achieved between the input and the output signals. By controlling this relationship, it is possible to accurately adjust the output phase of the DLL. For a Type-I DLL, the reference feedback is compared with a delayed version of itself. This architecture is widely used in DLL-based frequency synthesizers [9] and

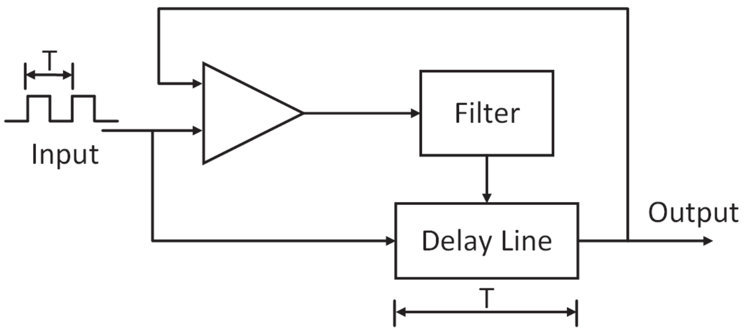


Figure 2.2: Type-I DLL

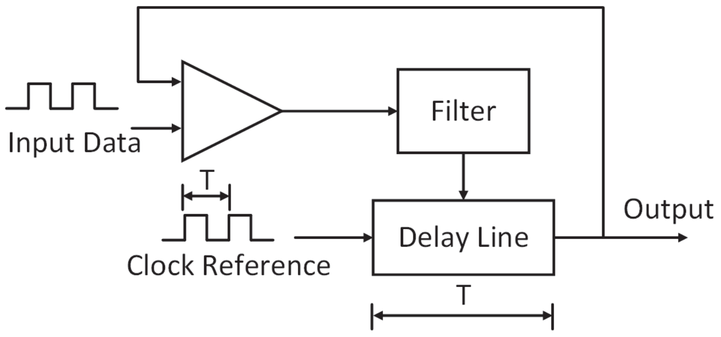


Figure 2.3: Type-II DLL

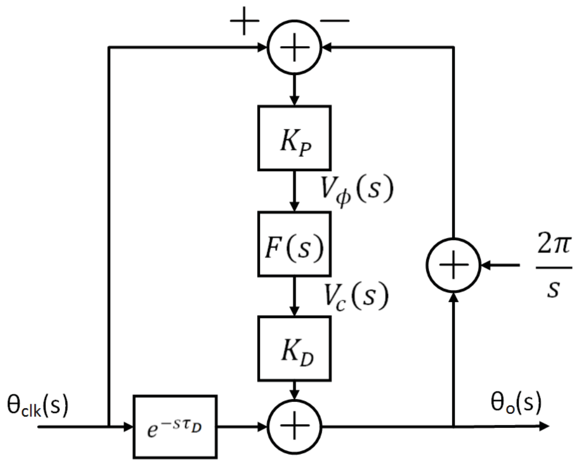


Figure 2.4: Type-I DLL linear continuous-time model

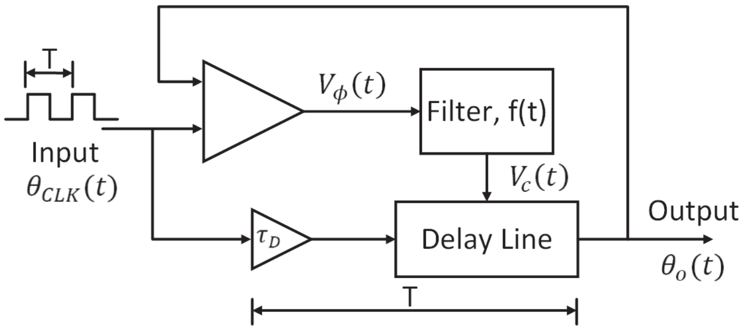


Figure 2.5: Type-I DLL with added delay

multi-phase clock generators [10]. In the Type-II architecture, there are two different inputs that are being compared: a delayed clock reference, and an input data signal. Looking at the excess-phase transfer function of both DLL architectures, the differences between the two designs can be highlighted.

In the Type-I case, the DLL can be represented in the s-domain by the diagram shown in Figure 2.4. This continuous-time model representation includes a small delay block represented by the $e^{-s\tau_D}$ which accounts for the phase offset between the input signal at the phase detector and the VCDL inputs. With the small added delay added to the input to the VCDL, the Type-I DLL can be represented by Figure 2.5 where the extra delay to account for any offset is present. From this design, it is possible to generate the system equations such that the transfer function of this design can be derived. The output phase is represented as

$$\theta_o(t) = \theta_{clk}(t + \tau_D) + K_D V_c(t) \tag{2.1}$$

while $v_\phi(t)$ and $v_c(t)$ are derived to be

$$v_\phi(t) = K_P \left\{ \theta_{CLK}(t) - \theta_o(t) - 2\pi \right\} \tag{2.2}$$

$$v_c(t) = f(t) * v_\phi(t) \tag{2.3}$$

Substituting Equations 2.2 and 2.3 into 2.1, and simplifying in the time-domain leads to

$$\theta_o(t) = \theta_{CLK}(t - \tau_D) + K_D K_P f(t) * \left\{ \theta_{CLK}(t) - \theta_o(t) - 2\pi \right\} \tag{2.4}$$

Similarly, the analysis can be done in the frequency-domain which is shown as

$$\theta_o(s) = e^{-s\tau_D}\theta_{CLK}(s) + K_D K_P F(s) \left\{ \theta_{CLK}(s) - \theta_o(s) - \frac{2\pi}{s} \right\} \tag{2.5}$$

Simplifying and solving for the input-output behaviour of this design leads to

$$\theta_o(s) = \frac{e^{-s\tau_D} + K_D K_P F(s)}{1 + K_D K_P F(s)}\theta_{CLK}(s) - \frac{K_D K_P F(s)}{1 + K_D K_P F(s)}\frac{2\pi}{s} \tag{2.6}$$

The first term in Equation 2.6 accounts for the phase offset between the two input clock points. The extra delay at the input of the VCDL is usually zero, which makes this term simplify to simply θ_{CLK}. The second terms is the dynamic behaviour of the system that will account for the step change in input phase.

The input θ_{CLK} can be represented by a constant clock input as well as an excess phase input. This is represented in the Laplace Domain as follows:

$$\theta_{in}(s) = \theta_{CC}(s) + \phi_{in}(s) = \frac{\omega_{in}}{s^2} + \phi_{in}(s) \tag{2.7}$$

The input-output behaviour can be simplified from 2.6 to be

$$\theta_o(s) = \frac{e^{-s\tau_D} + K_D K_P F(s)}{1 + K_D K_P F(s)} \left[\frac{\omega_{in}}{s^2} + \phi_{in}(s)\right] - \frac{K_D K_P F(s)}{1 + K_D K_P F(s)} \frac{2\pi}{s} \tag{2.8}$$

Solving simply for the input-output excess phase behaviour,

$$\phi_o(s) = \frac{e^{-s\tau_D} + K_D K_P F(s)}{1 + K_D K_P F(s)} \phi_{in}(s) \tag{2.9}$$

Under the condition of no excess delay at the input of the VCDL, $\tau_D = 0$, the output excess phase can be show to track the input phase, i.e.,

$$\phi_o(s) = \left.\frac{e^{-s\tau_D} + K_D K_P F(s)}{1 + K_D K_P F(s)}\right|_{\tau_D = 0} \phi_{in}(s) = \phi_{in}(s) \tag{2.10}$$

As one would expect, the output phase of the DLL is directly matched to that of the input phase. For this reason, the type-I DLL is ideally suited for clock synchronization applications.

A similar analysis can be done with the Type-II DLL. In this case, as mentioned previously, the two inputs to the phase-detector and the VCDL are not connected together. This allows for the Type-II DLL to serve a different purpose to that of the Type-I DLL.

Following a similar analysis procedure to that of the Type-I DLL, the architecture of the Type-II DLL in Figure 2.6 can be transformed to obtain the linear continuous-time model, shown in Figure 2.7.

The system equations for the Type-II DLL are as follows:

$$\theta_o(t) = \theta_{clk}(t + \tau_D) + K_D V_c(t) \tag{2.11}$$

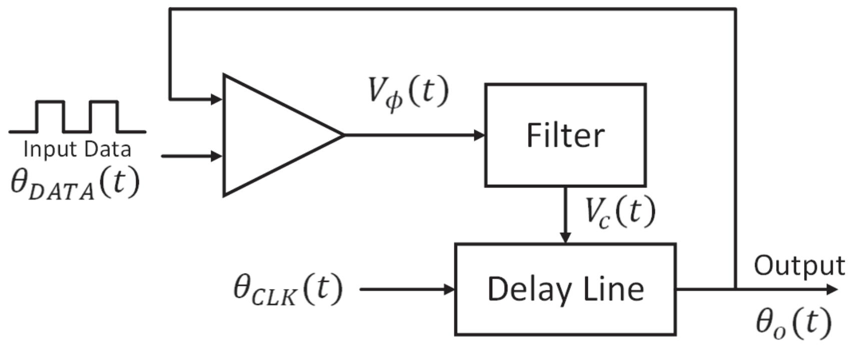


FIGURE 2.6: Type-II DLL architecture

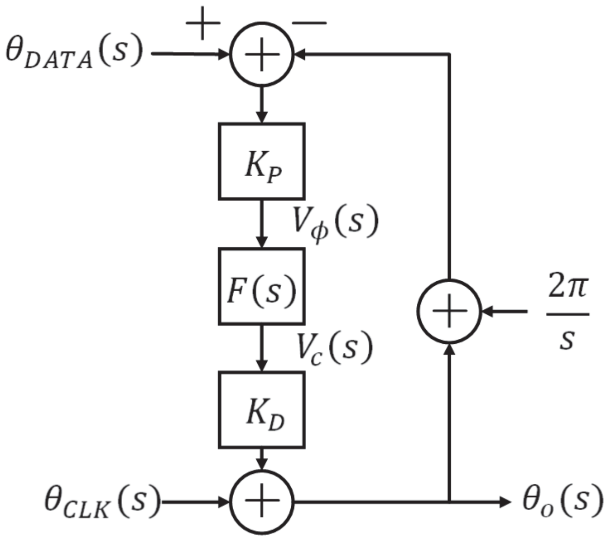


FIGURE 2.7: Type-II DLL linear continuous-time model

$$v_\phi(t) = K_P \left\{\theta_{DATA}(t) - \theta_o(t) - 2\pi\right\} \tag{2.12}$$

$$v_c(t) = f(t) * v_\phi(t) \tag{2.13}$$

The time-domain (Eqn. 2.14) and frequency-domain (Eqn. 2.15) representations for the input-output relationship of this model are derived from Equations 2.11, 2.12, and 2.13, and are as follows:

$$\theta_o(t) = \theta_{CLK}(t) + K_D K_P f(t) * \left\{\theta_{DATA}(t) - \theta_o(t) - 2\pi\right\} \tag{2.14}$$

$$\theta_o(s) = \theta_{CLK}(s) + K_D K_P F(s) \left\{\theta_{DATA}(s) - \theta_o(s) - \frac{2\pi}{s}\right\} \tag{2.15}$$

Solving for the output phase and simplifying leads to

$$\theta_o(s) = \frac{1}{1+K_DK_PF(s)}\theta_{CLK}(s) + \frac{K_DK_PF(s)}{1+K_DK_PF(s)}\theta_{DATA}(s) - \frac{K_DK_PF(s)}{1+K_DK_PF(s)}\frac{2\pi}{s} \tag{2.16}$$

Similar to the substitution that was performed for the Type-I DLL, it is possible to decompose the data signal into both a constant-clock input as well as an excess-phase input. This is shown below where the constant clock is the same frequency as the reference clock at the input to the VCDL:

$$\theta_{DATA}(s) = \theta_{CC}(s) + \phi_{in}(s) = \frac{\omega_{in}}{s^2} + \phi_{in}(s) \tag{2.17}$$

Solving for the excess-phase transfer function, it is possible to extract the behaviour of the Type-II DLL as

$$\phi_o(s) = \frac{K_DK_PF(s)}{1+K_DK_PF(s)}\phi_{in}(s) \tag{2.18}$$

Equation 2.16 has three terms. The first term is representative of the reference clock transfer function, the second terms is representative of the data transfer function, and the third is the term accounts for the change in output phase. With the data input separate from the clock input, it is possible for the Type-II DLL to be utilized for more that simply clock synchronization. If the filter function, $F(s)$, is an integrator, then it can easily be shown that the data transfer function would be low-pass, while the clock transfer function would be high-pass.

With the clock transfer function being high-pass in nature for the Type-II DLL and all-pass in Type-I, it is important for designers to keep in mind that the DLL is subject to jitter propagation from the input reference frequency. For this reason, it is important that the reference clock that is used in these designs has a very low phase noise in order to have a clean output signal.

2.2.1 Phase Detector

The phase detector is a block that compares the phase of two signals and determines if the first is ahead, or behind the second. The simplest phase-detector can be implemented using an XOR or XNOR gate as shown in Figure 2.8. The disadvantage of this simplistic phase-detector is that it only has a linear input range of π radians. This limits the detector's ability to compare signals across a full period range and in turn limits the capture range of the DLL. The phase-detector produces an average voltage output proportional to the phase difference between the two compared signals.

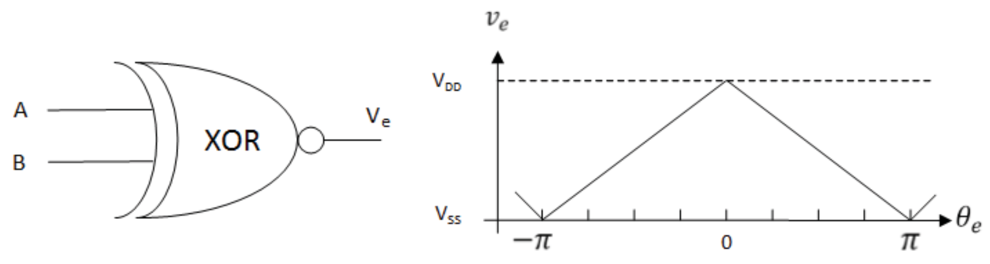


FIGURE 2.8: XNOR logic implementation as a phase-detector

Figure 2.9 shows the working principle of the phase-detector from 2.8. When both signals are the same, the output is equal to V_{DD} and when they are different, the output is equal to V_{SS}. The overall average voltage level of the signal will be proportional to the amount of phase that the two signals differ by.

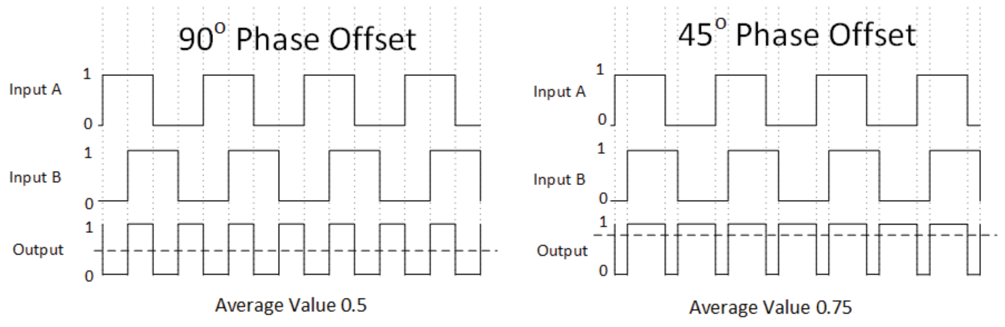


FIGURE 2.9: Phase-detector example signals

For the implementation in a DLL, a phase-frequency detector (PFD) was used in this thesis. The PFD has the added advantage that it can produce an output when the two signals being compared differ not only in phase, but in frequency as well. The PFD is largely used in PLL applications, but was also used in this instance as the phase-detector for a DLL. The PFD implementation also has the added advantage that the linear region of operation spans 4π radians. This ensures that the PFD output is always working in the linear range when comparing two signals of differing phase. The implementation of the PFD that was implemented in this work is shown in Figure 2.10.

The PFD is constructed with two D-flip-flops and a feedback AND gate. Each flip flop is attached to one of the input signals that will be compared. The output of each flip flop is connected to both a signalling line, as well as the feedback AND gate. The signalling line represents either an up or a down pulse, which signals to the charge pump (Subsection 2.2.2) what action to take to align the two signals. The D-flip flops are rising-edge sensitive, so only the rising edges will be detected. When a signal transitions, the

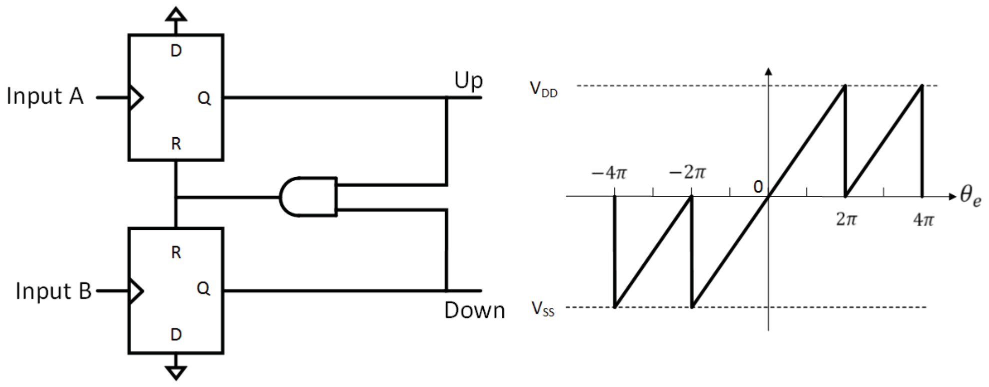


FIGURE 2.10: PDF design

D-flip flop it is attached to asserts a logical high value at the output. The output signals are compared with the AND gate, and when they are both a logical high value, this triggers the reset signal for both D-flip-flops so that the detecting process can start over. The difference in the time that one signal is asserted will correspondingly translate to the width of the pulse that the charge pump block will generate to control the delay line. As an example, if signal A were to transition high, then the "up" signal would transition to high. When the B input transitions, the "down" signal will transition to high and once both the "up" and "down" signals are high, the reset signal will be triggered, which will reset both the "up" and "down" signals to low, such that the process can be repeated for each clock period.

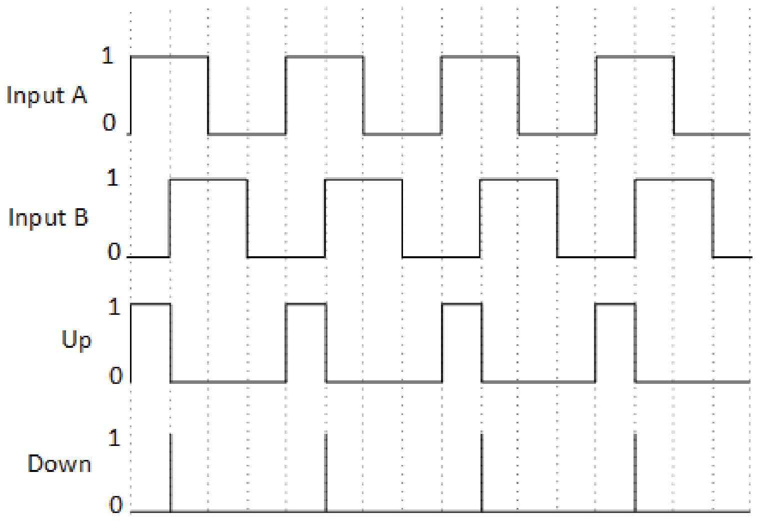


FIGURE 2.11: PFD example signals

2.2.2 Charge Pump

The charge pump is a simple circuit which translates an incoming digital voltage signal into an output current. The charge pump block is a dual-input, single-output design where the inputs are attached to the "up" and "down" line of the phase-detector, and the output is attached to the filter of the DLL. The output is a single-ended output where the "up" and "down" signals are essentially added, and their result is converted into a current. The overall result of this block is that the corresponding "up" and "down" signals will either provide a positive or negative current from the DLL loop filter. The schematic of the charge pump is shown in Figure 2.12.

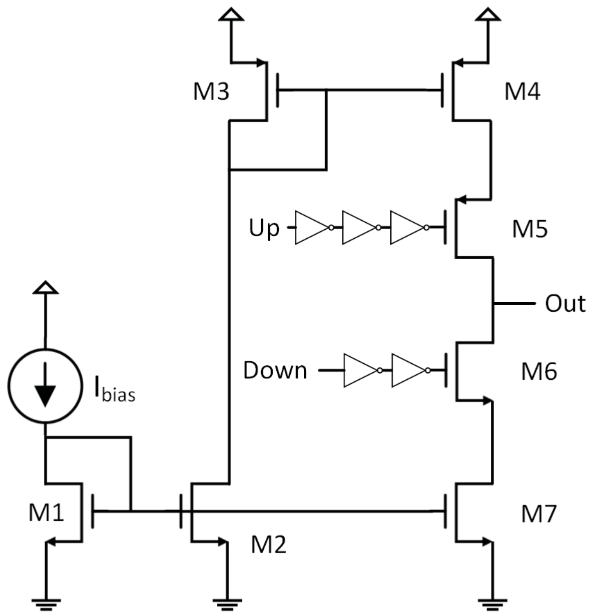


FIGURE 2.12: Charge-pump architecture

The charge pump architecture described in Figure 2.12 shows how the incoming up and down pulses from the phase-detector are converted into a corresponding current. The reference current I_{bias} is set using a simple resistor. Since the transistors are biased to a certain DC operating point, the resistor regulates the current that is seen at the incoming reference branch. In the design, the current was limited to 1mA. Transistors M1, M2, and M7 are used as current mirrors where the current flowing through I_{bias} will be mirrored to the current flowing through the sources of M1, M2, and M7. Similarly, the same is done at the top of the schematic with transistors M3 and M4. The result of this architecture is that transistors M5 and M6 will be used as switches in order to let the current from the branches flow at the desired times towards the output. When "up" is high and "down" is low, a current of 1mA will be sourced to the loop filter via the "out" node which will increase the delay of the VCDL. Alternatively, when "down" is high and "up" is low, a current of 1mA will be sinked from the loop filter and decrease the delay of the VCDL.

2.2.3 Loop Filter

The loop filter of the DLL is the main component which determines the overall filtering characteristics of the DLL. The loop filter function can be calculated by determining the overall desired DLL response. The loop filter serves to extract the DC signal from the output of the charge pump, and pass this along to the VCDL such that the delay can be adjusted so that it is possible to achieve lock for the DLL. The loop filter can be chosen to have either an active, or a passive implementation based on the requirements of the transfer function that is desired. More information about the design and the implementation of the loop filter functions can be found in Chapter 4. The most simplistic loop filter that can be implemented is for a first-order DLL where a simple capacitor is used. The capacitor will be used as a current-in, voltage out filter and the implementation is shown in Figure 2.13.

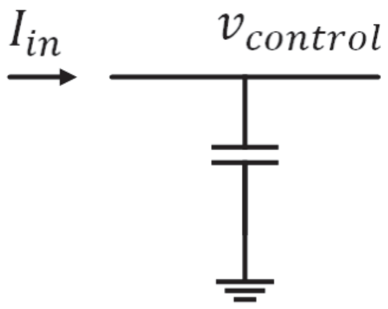


FIGURE 2.13: 1^{st}-order loop filter for a DLL

2.2.4 Voltage-Controlled Delay Line

The voltage-controlled delay line (VCDL) is the component of the DLL which differentiates it from the PLL. The VCDL serves to adjust the delay of the clock reference signal such that once it is compared to the incoming reference signal at the inputs to the phase-frequency detector, they are both exactly in phase. It is possible to adjust the phase of the output signal by simply adjusting the voltage at the control line of the VCDL. In the design that was used in this dissertation, an increase in voltage on the control line will increase the delay of the delay line, and a reduction of voltage will decrease the delay of the delay line. The architecture of the VCDL that was used is simply a series of inverters connected in series with an adjustable amount of capacitance at each node. By increasing the control voltage, more capacitance will be introduced at each node - essentially increasing the RC time constant of the buffer. The increase of the RC time constant of each buffer in the line will introduce more delay from the beginning to the end of the delay line. This architecture is displayed in Figure 2.14.

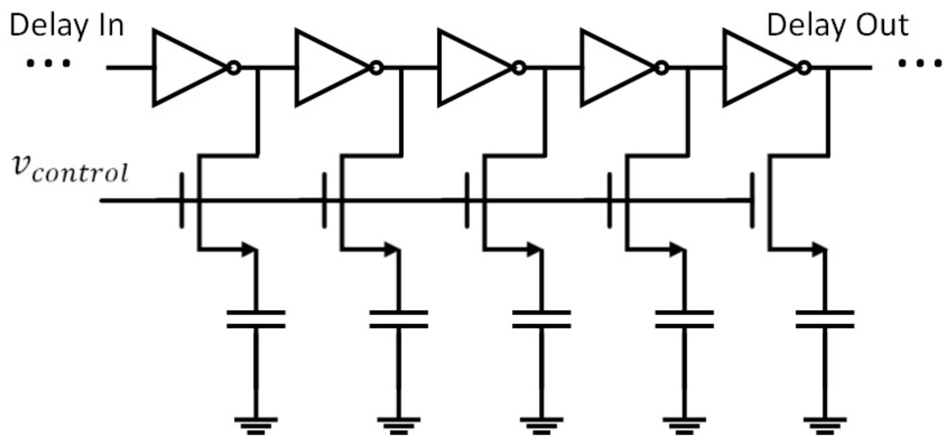


FIGURE 2.14: Voltage-control delay line architecture

It is desired to have a linear voltage-to-delay transfer characteristic such that the signal's phase can be controlled in a predictable manner. The sensitivity by which the delay line can be controlled is determined by running simulations to extract the exact behaviour of the circuit. The majority of the DLLs designed in this dissertation used an architecture of 16 inverters with the attached capacitance implemented as the gate of a large transistor. A transistor was substituted in the place of a capacitor in order to make the layout of the VCDL much smaller without any noticeable degradation in performance.

2.3 Sigma-Delta Modulator

Sigma-delta modulation is a digital signal processing method that is used to encode analog signals into digital signals. The result of sigma delta modulation is that the input analog signal will be represented at the output by a density of digital bits - a density of 1s and 0s. The transformation from an analog to a digital signal is done by sampling the signal at a specific rate, called the sampling rate. The sampling can be done at any frequency above the Nyquist rate in order to fully represent the signal in the digital domain. The metric with which the sampling rate is often referred to is called the over-sampling ratio (OSR). The OSR is defined as

$$OSR = \frac{f_s}{2f_B} \tag{2.19}$$

In Equation 2.19, f_s represents the sampling frequency, and f_B represents the signal bandwidth that is being converted. Figure 2.15 shows the basic implementation of a sigma-delta modulator.

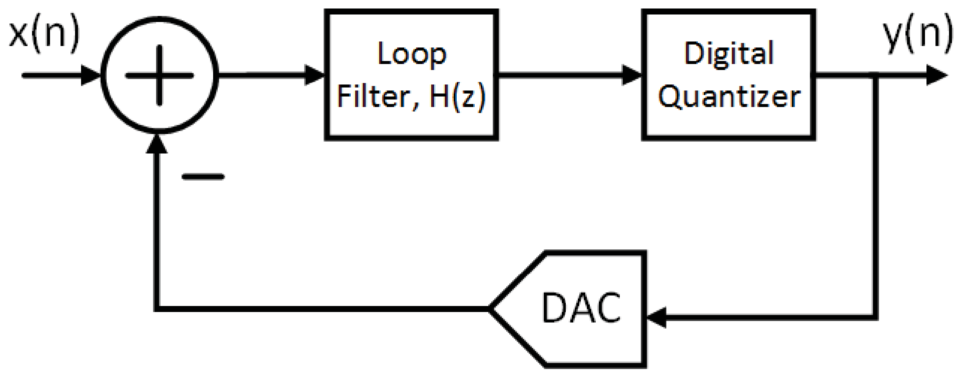


FIGURE 2.15: General block diagram of sigma-delta modulator

Sigma-Delta modulators are a very popular choice as an analog-to-digital converter (ADC), due to the fact that the implementation has a high degree of insensitivity to analog circuit imperfections so they are an excellent choice when being integrated into CMOS designs [11]. The sigma-delta modulator relies on two signal processing techniques - oversampling and quantization noise feedback. Quantization noise is the error that accumulates when the input signal with infinite different levels is quantized into a set of discrete levels. This transformation generates a small error and is referred to as the quantization error [11].

The advantage of oversampling the input signal is identified by Figure 2.16. The quantization noise power is evenly distributed in the range of $[-f_s/2, f_s/2]$ and its PSD magnitude is represented by

$$S_E(f) = \frac{\Delta^2}{12 f_s} \tag{2.20}$$

and the in-band noise power is given by

$$P_E = \frac{\Delta^2}{12 \times OSR} \tag{2.21}$$

where Δ represents the smallest quantization level difference.

When oversampled, the quantization noise power stays the same, but is simply spread over a larger bandwidth. This is represented by the two shaded rectangles in Figure 2.16. When a low-pass filter is used to recover the signal, the resulting oversampled signal will have less quantization noise.

The advantage of oversampling of a signal is represented by the overall signal-to-noise ratio (SNR). SNR is defined as

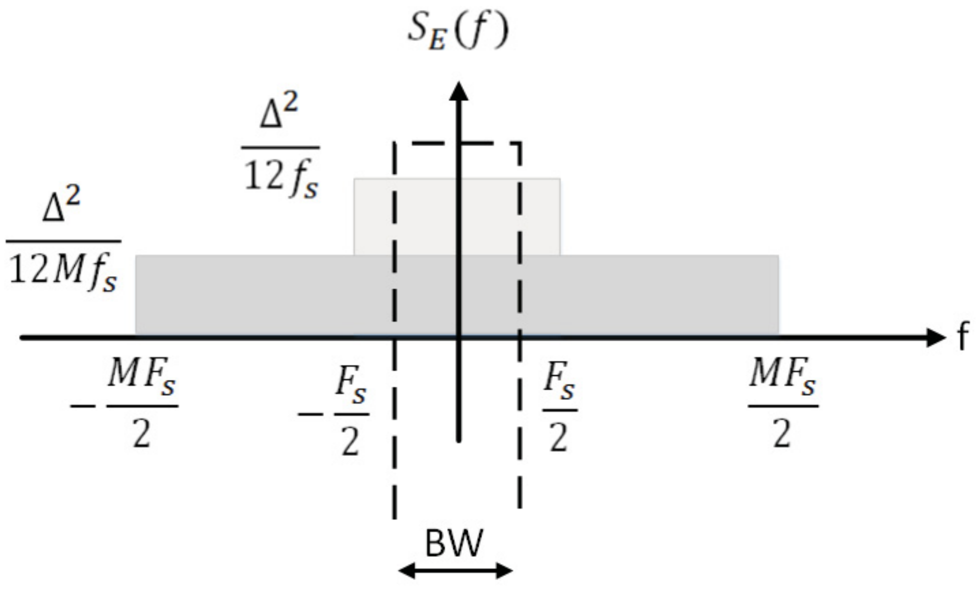


FIGURE 2.16: Spreading of quantization noise in an oversampled sigma-delta system

$$SNR|_{dB} = 10log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \tag{2.22}$$

within the bandwidth of the signal. As the sampling rate increases and the in-band noise is reduced,

$$SNR'|_{dB} = 10log_{10}\left(\frac{MP_{signal}}{P_{noise}}\right) \tag{2.23}$$

and the in-band SNR changes according to

$$\Delta SNR|_{dB} = SNR'|_{dB} - SNR|_{dB} = 10log_{10}(M) \tag{2.24}$$

The change in the effective number of bits of resolution of the signal is represented by

$$\Delta n = \frac{\Delta SNR|_{dB}}{6.02} = \frac{10log_{10}(M)}{6.02} = \frac{1}{2}log_2(M) \tag{2.25}$$

This shows that an extra 0.5 bits of resolution is gained for every time the oversampling ratio is doubled.

Another advantage of the sigma-delta modulator that further increases the performance is that it is possible to filter the quantization noise in such a way that most of its power lies outside of the signal band [11]. This is called noise-shaping and it is represented in Figure 2.17. The higher the sigma-delta modulator order, the more pronounced the noise shaping is, however the increase in order of the modulator greatly increases complexity as well. The benefits of higher order modulators and increased OSR on the SNR are shown in Figure 2.18. Here, the reader can see that a higher-order modulator results in a higher SNR for the same OSR.

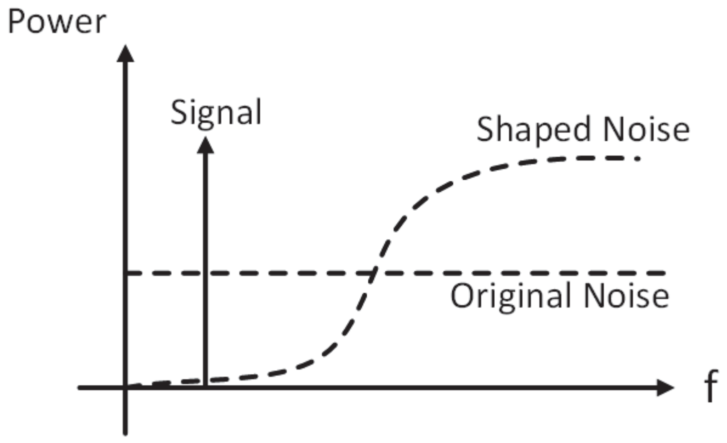


FIGURE 2.17: Noise shaping of the sigma-delta ADC

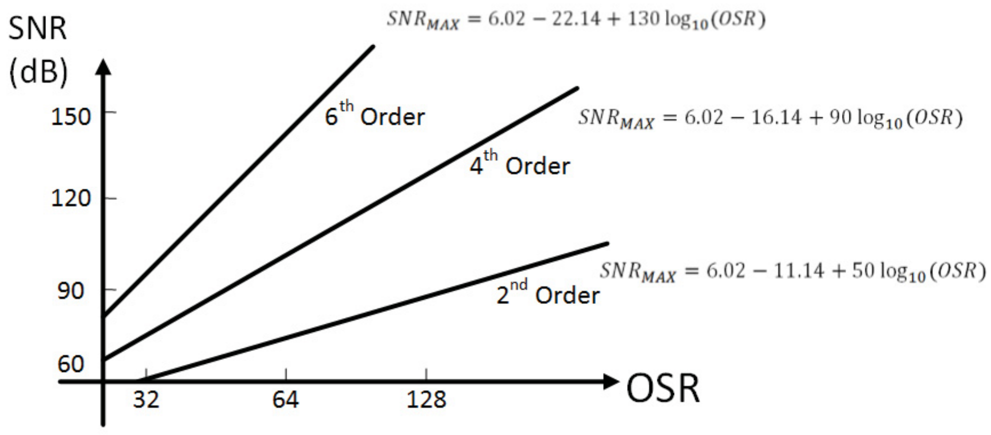


FIGURE 2.18: SNR vs. OSR for different orders of sigma-delta modulators

2.4 Circular Shift Register

The circular shift register is a series of serially connected memory cells that will right rotate a bit sequence using a clock signal to create a circular bitstream. The memory cells that are used in this particular implementation are D-flip-flops and they are arranged in different lengths depending on the application. The use for a circular shift register is that it can easily be loaded using a source and a clock signal, and it is fully programmable even after the chip has been fabricated. This allows for any design containing a circular shift register to be programmed with any desired bit sequence in order to deliver the desired input signal.

The advantage of using a circular shift register over an external source is that it is ideally suited for built-in-self-test (BIST) methods. Once programmed, no external source is required resulting in a self-contained test environment. In addition, the circular shift register could be pre-programmed to contain a desired bitstream such that no initial loading or initialization of the bitstream is required. In order to have a continuously rotating stream that will repeat over and over again, the output of the last D-flip-flop is fed back into the input of the first D-flip-flop to make a continuous stream. At each rising edge of the clock signal, the bits are rotated right one bit. Figure 2.19 shows a representation of the circular shift register implementation.

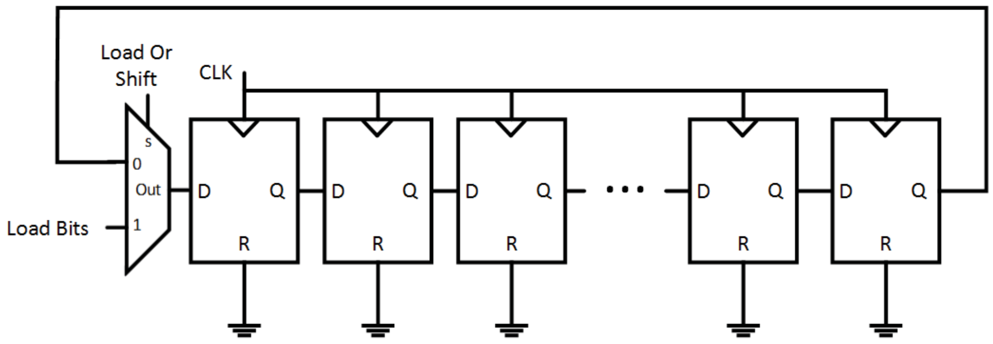


FIGURE 2.19: Circular shift register implementation using D-flip-flops

The implementation of a circular shift register can be used to generate signals with high precision and very low hardware complexity [12]. It can be loaded with any representation of bits that the designer wants and it is fully programmable even after the chip has been fabricated.

2.5 Digital-to-Time Converter

The digital to time converter is a device which translates a digital-representation of a signal into a time-representation of a signal. The converter takes an input of digital bits, and produces an output waveform with the edge transitions located at specific a point in time relative to a reference clock signal. The relationship between the input code selection and the output edge transition locations [13] can be represented as

$$t_{out} = a_t(b_0 + b_1 2^1 + b_2 2^2 + ... + b_{N-1} 2^{N-1}) + t_{os} \tag{2.26}$$

In this equation, t_{out} represents the phase shift of the output signal, a_t represents the conversion relating the digital select bits into time, $b_0...b_{N-1}$ represents the individual input digital select bits for the DTC, and t_{os} is a fixed time offset.

The most simplistic way to construct a digital-to-time converter is by using a DLL in conjunction with a multiplexer. The DLL's delay line will generate an array of clock signals that differ by a fixed phase difference. The multiplexer is then used to select the clock signal at the desired phase. The number of different phases that are available will be limited by the number of delay elements in the DLL delay line, as well as the number of control lines to select the signal. The architecture of the DTC is displayed in Figure 2.20.

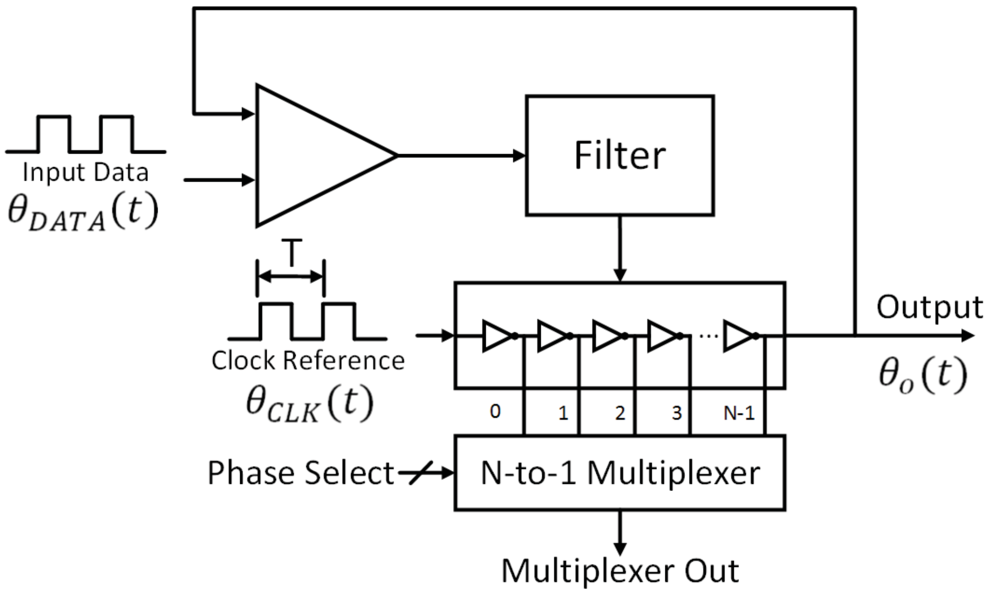


Figure 2.20: Architecture for a basic digital-to-time converter

The input select lines are used to determine which output phase signal is selected from the DTC. An example waveform output is shown in Figure 2.21 and depicts the output waveform from an input of the code sequence 0,1,2,3. In this particular example, there are only 4 different clock phases available to select from, meaning that only 4 equally-spaced phases are fed into the multiplexer.

Figure 2.21 represents a multi-bit DTC implementation. The figure is used to convey the idea of the DTC to the reader. For the implementation in this thesis a single-bit implementation is used that takes the output of a sigma-delta encoded signal to control the select line of the DTC, and hence the output time encoding of the signal. The single bit implementation selects between only two differing phases, as opposed to the multiple-phases of Figure 2.21. The attractiveness of this implementation is the simplicity of being able to control the multiplexer with a single bit. Depending on the desired implementation, any two phase signals can be chosen as the input to the DTC, but for simplicity, phases 90^o apart were chosen.

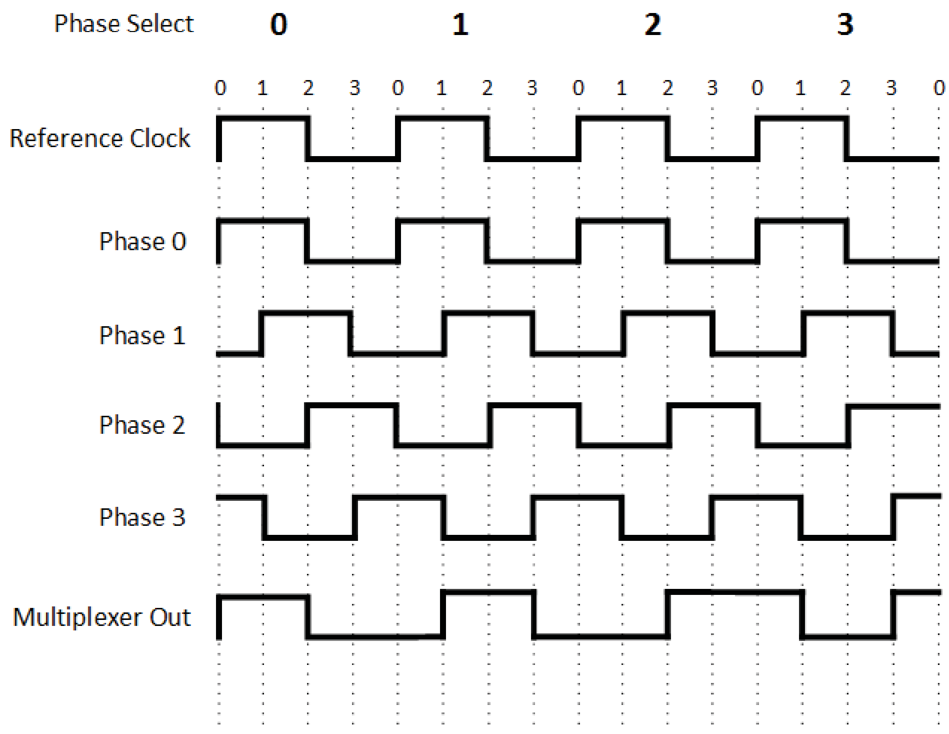


FIGURE 2.21: Example DTC waveform output for a select sequence of 0,1,2,3

The output phase of the single bit DTC with respect to the clock reference can be represented as

$$\phi(x) = \begin{cases} 0^o\colon x = 1 \\ 90^o\colon x = 0 \end{cases} \tag{2.27}$$

where x represents the single bit select input.

2.6 Frequency Divider

The frequency divider is a component which is used to divide the frequency of an incoming periodic clock signal. The frequency divider is generally used to divide frequencies around the chip, and is often used in the feedback loop of a multiplying PLL. The frequency divider has a very simple architecture and is simply a cascade of a specific number of D-flip-flops. In the applications of this thesis, the frequencies will only be divided by 2^N, where N is an integer. The equation with which we can determine how many flip-flops are required to make the desired division is shown in Equation 2.28. Figure 2.22 shows the architecture of a divide-by-4 circuit. In this case, only 2 D-flip-flops are necessary to carry out the division process.

$$Quantity = log_2(N) \tag{2.28}$$

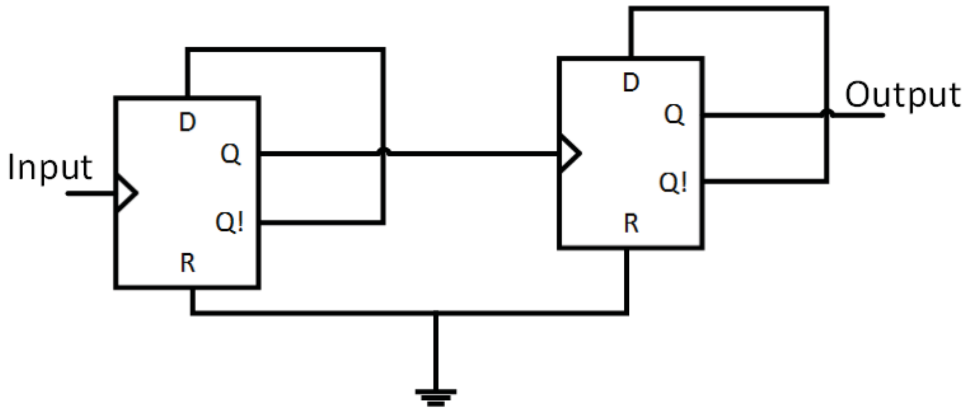


Figure 2.22: Frequency dividing by 4 circuit

2.7 Voltage-Mode Filter

A voltage-mode filter is a component which will block the passage of certain signals based on their frequency. In a voltage-mode signal, information is carried by the voltage level of the signal. A voltage-mode filter can be either low-pass, band-pass, or high-pass. A low-pass filter will pass all the voltage signals whose frequency is below the cut-off frequency. A band-pass filter will pass all the voltage signals whose frequency in between the lower and upper frequency cut-offs. A high-pass filter will pass all the voltage signals whose frequency is above the cut-off frequency.

A voltage-mode filter can posses both active and passive components, can be of many different orders, and can have many different frequency responses. Typically, high-frequency applications utilize passive components, while low-speed applications utilize active components. The reason for this is that the parasitic capacitances associated with active filters can have a very detrimental effect on the circuits function at high speeds. The order of the filter plays a very important role in the roll-off of the filter at the cut-off frequencies. A higher-order filter will have a steeper gain roll-off than a lower-order filter and will result in less unwanted signals being passed through the filter. The higher order the filter is, the closer the frequency response gets to the ideal "brick-wall" cut-off, where all frequencies outside the band are suppressed. With every increase of order, a performance increase of 20dB/decade can be observed in the frequency-response plot. The resulting filters will have an $NX20dB/decade$ performance for an N^{th}-order filter. An example of the gain-attenuation advantages is shown in Figure 2.23 where four Butterworth filters are compared.

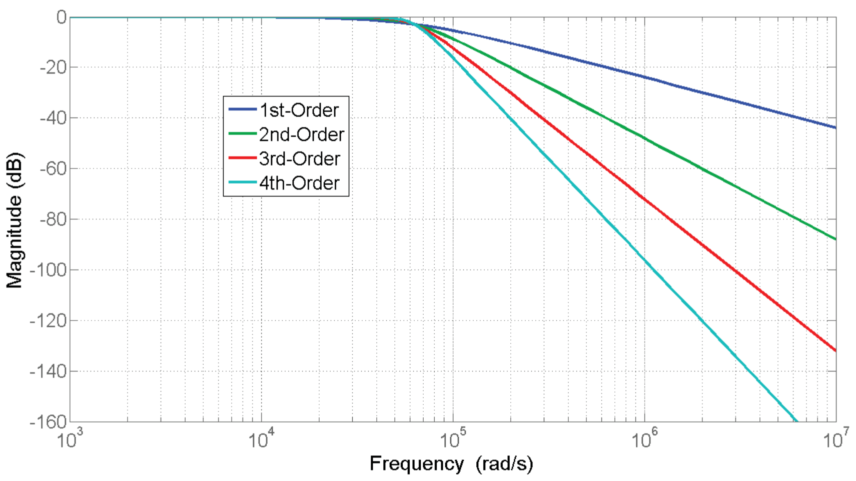


FIGURE 2.23: Comparison between 4 different orders of butterworth voltage-mode low-pass filters

In addition to differing orders of filters, there are also differing filter classes. Each filter type has its own advantages and disadvantages. For example, Butterworth filters are best for flat passband gain, Chebychev and Elliptic have a very steep roll-off at the expense of ripple in the passband region, and Bessel has a very linear phase-response.

2.8 Time-Mode Filter

Time-mode signal processing (TMSP) has become a popular field of study where voltage-domain sampled data is encoded into a time signal where the information is stored with respect to a reference clock edge [14]. This encoding makes the data much less susceptible to amplitude noise which is becoming a limiting factor in the performance of sub-micron analog technologies [14]. A time-mode filter works as a part of TMSP and it is similar to a voltage-mode filter, but instead of filtering out the voltage levels that are not within the passband, it is the time-difference with respect to a reference that will be filtered out. In a voltage-mode circuit, the levels of the voltage are specified with respect to a common level - usually ground. The voltages are measured with respect to this and the rate at which they change is called the frequency of operation. Similarly, with time-mode circuits, the signal is compared to a reference signal but instead of comparing voltage levels, the phase of transition edges are compared. The rate at which the edges change location and complete a full period is determined to be the frequency at which the signal is operating. An comparison between voltage- and time-mode circuits is illustrated in Figure 2.24.

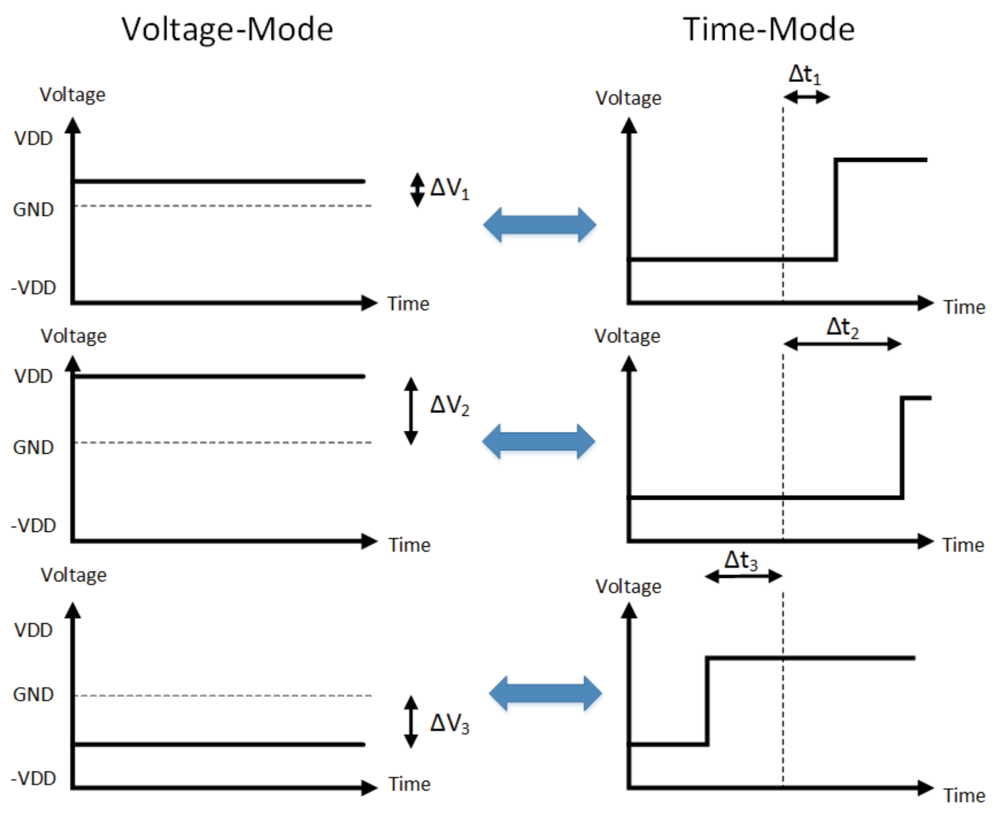


FIGURE 2.24: Comparison between time- and voltage-mode signal representations

The advantage of using time-domain signals is that it an be easily implemented in any technology because a propagation delay is inherent to all materials. Additionally, the resolution at which a signal can be controlled is much greater than the voltage-circuit counterparts. It is possible to achieve picoseconds resolutions in time, while often times microvolt resolutions are the smallest achievable resolutions in voltage-mode circuits.

The time-mode filter is very similar to the voltage-mode filter discussed in Section 2.7, where a phase difference will be passed if the frequency is within the passband of the filter, and it will be attenuated if it is not. Time-mode filters can also be low-pass, band-pass, or high-pass in nature and they can be of many different orders, as well as from many different families of filters.

There are a couple of different ways of implementing time-mode filters: using a PLL, a DLL, or time-mode reconstruction IIR filters [15] [14]. In this thesis, a DLL is used as a time-mode filter. A DLL was selected to be used as the time-mode filter over a PLL for various reasons. PLLs have disadvantages that make their use in high-speed designs very problematic, especially when both excellent performance and reliability are desired [16]. Additionally, The voltage-controlled oscillator (VCO) of the PLL is a major source of problems. If not designed correctly, then there can be a significant amount of phase-noise that will be present on the output edges of the clock signal. Variations in temperature, supply voltage and process variations will greatly affect the operating performances of a PLL.

DLLs, however, are not as susceptible to the problems outlined above. Instead of using a VCO, the DLL incorporates a voltage controlled delay-line (VCDL) (Section 2.2.4). The output of the DLL will be adjusted such that it will align with the input edges to achieve lock, such that the input buffer delay, and the clock skew are reduced to zero. Since the design operates at a single frequency, the DLL was chosen to act as the time-mode filter for this thesis. The implementation of the DLL as a time-mode filter is covered in Chapter 4.

2.9 Counters

A counter is a digital logic block which stores the number of times a positive rising edge has occurred. There are various types of counters, but the one that will be implemented in this thesis is an asynchronous counter. The counter architecture is a serially-connected line of D-flip-flop implemented dividers (Section 2.6). The output of one counter is fed into the input of the next. Thus, when the counting process is finished, a binary-representation of the number is left in the counting block. A representation of the architecture is presented in Figure 2.25.

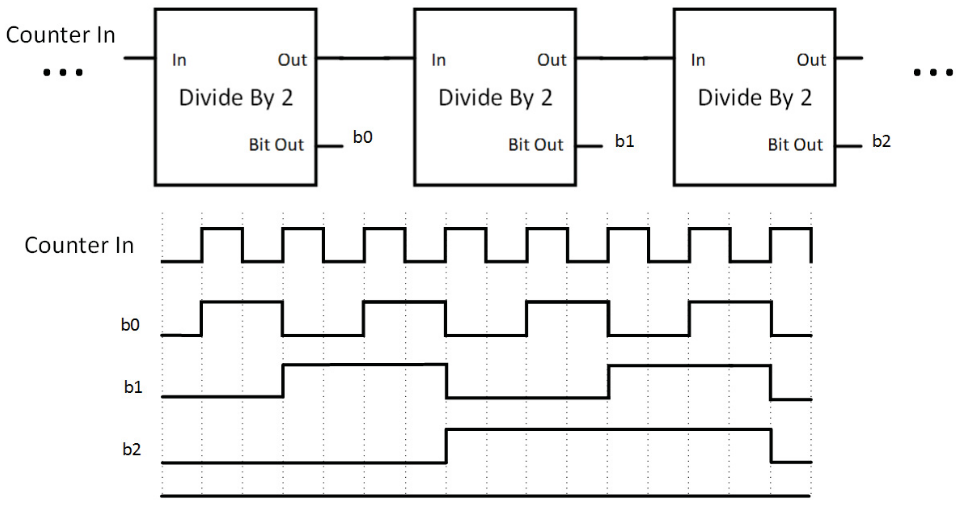


Figure 2.25: Asynchronous counter architecture

2.10 PLL

A phase-locked loop is a similar structure to a DLL, where the output phase is related to the input phase. The difference with the two structures is mainly the fact that the

DLL uses a VCDL, and a PLL uses a voltage-controlled-oscillator (VCO). All the other components in the architecture are the same and were described in Section 2.2. The VCO allows the PLL to work with a number of different frequencies unlike the DLL that can only work with a single frequency. For this reason, the PLL is often used as a clock multiplier on chip. The negative feedback loop is divided down with a series of dividers which results in having the output frequency multiplied by that same ratio. Figure 2.26 shows the basic design of a PLL.

In this work, the PLL is used as a frequency multiplier. The reason that the PLL is used, is that it can reliably multiply the frequency of a clock signal while still keeping the same overall relative phase shift in the signal. This property makes synchronization of signals very simple.

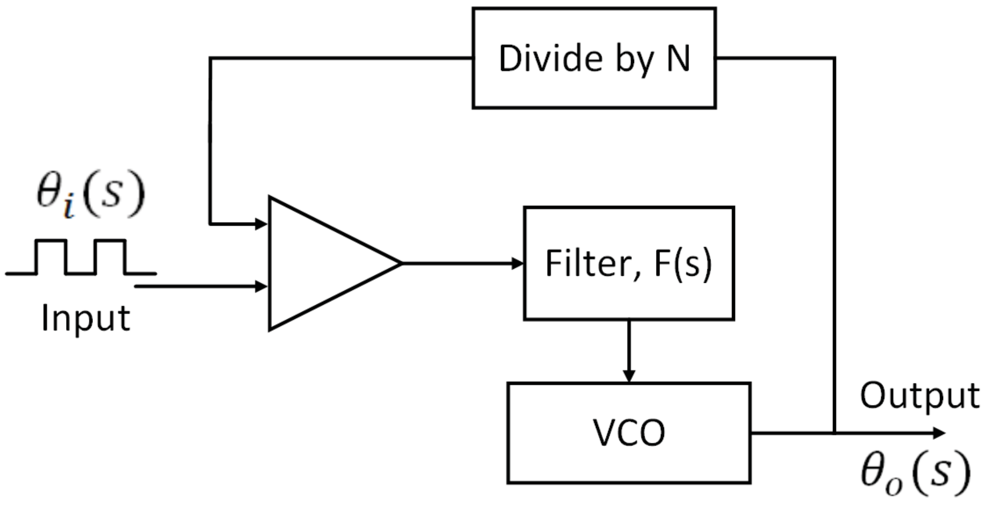


FIGURE 2.26: Block diagram of a phase-locked loop

From Figure 2.26, it can be shown that the overall transfer function can be written as

$$\theta_o(s) = \frac{N_{PLL}\omega_{FR}}{s(N_{PLL}s + K_{VCO}K_PF(s))} + \frac{sN_{PLL}K_{VCO}F(s)}{s(N_{PLL}s + K_{VCO}K_PF(s)}\theta_{in}(s) \qquad (2.29)$$

Where N_{PLL} represents the total division factor in the feedback loop, K_P represents the phase-detector coefficient, K_{VCO} represents the VCO constant, and ω_{FR} represents the free-running frequency of the PLL.

The divider in the feedback loop will serve to multiply the output frequency by the N-dividing ratio. From a purely logical standpoint, the divider works because the VCO is matching the two signals at the input to the phase detector. The frequency of the signal in the feedback loop is N-times lower in frequency due to the divider, hence the output is

N times the frequency of the input frequency. This can be shown mathematically where a ramp input phase change is a constant input frequency described by

$$\theta_{in}(t) = \omega_{in} t \cdot \mu(t) \iff \theta_{in}(s) = \frac{\omega_{in}}{s} \tag{2.30}$$

Taking the derivative we find

$$\omega in(s) = s\theta_{in}(s) = \frac{\omega_{in}}{s} \tag{2.31}$$

Applying the final value theorem to Equation 2.29, the output frequency can be shown to be

$$\omega_o(\infty) = \lim_{s\to 0} s\omega o(s) = \lim_{s\to 0} \frac{N_{PLL}\left[\omega_{FR}s + K_{VCO}K_P F(S)\omega_{in}\right]}{sN_{PLL} + K_{VCO}K_P F(S)} = N_{PLL}\omega_{in} \tag{2.32}$$

This shows that the output frequency converges to a multiple of the input frequency which is determined by the divider in the feedback loop of the PLL. The loop filter for a PLL is designed very similarly to the design of the DLL loop filter. This process is described in great detail in Section 4.1.

2.11 Summary

In this section, the building blocks and concepts were introduced. This section covered the major components including the DLL, PLL, sigma-delta modulator, DTC, counters, as well as a variety of filters. These building blocks will be used to construct the PTI in the following sections.

Chapter 3

Sigma-Delta Phase Generation Technique

This chapter will cover the design and implementation of the sigma-delta encoding technique that is used to encode the signals that are used in the probabilistic test instrument (PTI). The PTI uses various signals to stimulate the DUT so that it is possible to perform a variety of tests. The PTI uses a signal generator that consists of a 1XN circular shift register that circulates N encoded bits in a repeating fashion. The bitstream is subsequently filtered by a filter whose type depends on the encoding of the signal (voltage or time) [6]. A theory of the design and operation of the bitstream generator is covered in this chapter, as well as the simulations that are run to verify the operation of the implementation of the design using MATLAB and Simulink.

3.1 Overview

The stimulation signals for the PTI are all generated using a circular shift register that continuously circulates N bits. In this chapter, the design and implementation of this signal generator will be covered in detail. The signal generator consists of three distinct parts: the sigma-delta modulator which first encodes the input signals into a bitstream, the digital-to-time/digital-to-voltage converter which encodes the bits into a time or voltage signal, and the circular shift register which stores and continuously circulates the bit sequence. In this design, the sigma delta modulator and the digital-to-time/digital-to-voltage converters are implemented in software, and the circular shift register is implemented in hardware. It is the software portion of the signal generator that will be the major focus of this chapter.

Implementing part of the design in software has some major advantages. First, implementing part of the design in software significantly reduces the hardware complexity of the design and second, it allows for a great deal of flexibility. Figure 3.1 illustrates the breakdown between hardware and software of the components of the phase signal generator.

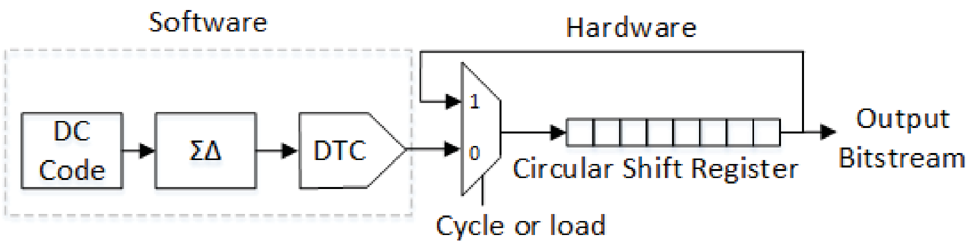


FIGURE 3.1: Software and hardware breakup of the signal generation module

As Figure 3.1 illustrates, the sigma-delta phase generation of the signals is done in software. A DC code is presented to a sigma-delta modulator where it is encoded and the output bits are stored in the circular shift register. The oversampling ratio, defined in Section 2.3, is adjusted along with the frequency of the signal to ensure that the output bits are a multiple of the total number of pits that can be stored in the circular shift register. Without a multiple of the period, the output signal from the circular shift register would not be a smooth signal following the filter, and would lead to measurement errors.

The sigma-delta modulator works by encoding the DC code into a digital bitstream whose values take on either V_{DD} or V_{SS}. The average value is of the sigma-delta bitstream is proportional to the input DC code and is extracted from the bitstream using a low-pass filter as described in Sections 2.7 and 2.8. The sampling frequency of the sigma-delta modulator is the same as the reference clock frequency which here is 114.56 MHz. In addition to DC codes, the phase signal generator can encode AC signals as well. The principle is the same for the AC signal where the output density of bits is proportional to the instantaneous input signal.

The output of the 1XN circular shift register is fed into a DTC (described in Section 2.5) where the signal is time-encoded whereby the DC code level is proportional to the phase of the signal. The relationship between the DC code and the output phase of the signal in relation to the reference clock signal can be described as

$$\phi_o = (\phi_{max} - \phi_{min}) \times |(DC\_Code - 1)| \tag{3.1}$$

The output phase from the DTC is related to the overall phase difference between the encoding signals, as well as the input DC code. The implementation for this thesis uses a DC code of 1 to represent the minimum phase shift, and a DC code of 0 to represent the maximum phase shift. By changing the DC code input, it is possible to accurately control the phase of the output signal from the phase signal generator. The application of these signals will be covered in more detail in Chapter 5.

3.2 Phase Encoding

The phase encoding used in the DTC is fully-programmable and can be adjusted to meet the test engineer's timing needs. Since the DTC is also located in software as illustrated in Figure 4.7 it is possible to easily change the encoding scheme to change the phase-range over which the time-encoded signal spans. This is done by changing the bit pattern associated with each sigma-delta level. Table 3.1 illustrates the different possible phase-encoding that can be applied to the signals, and Table 3.2 illustrates some examples of the relationship between to signals that can be used to span a desired phase range. Both encoding schemes are implemented using a duty cycle of 50%. It is important to note that each period must be the same length. That is, for example, a signal encoded with 8 bits must be clocked at twice the speed as a signal encoded with 4 bits, as is shown in Table 3.1.

In this thesis, sigma-delta encoded AC signals will be used to stimulate the DUT. The circular shift register that the sigma-delta encoded AC signals will be loaded into can store up to 256 bits. Having only 256 bits limits the frequency of the input signal given the fact that the sampling time is a set 114.5 MHz. Since the output length, and sampling time are set, the only thing that can be changed is the number of periods of the input signal that occur in the N sample points - 256 in this case. In order to encode a signal with a high SNR, one must ensure that the input signal frequency, F_T, and the sampling frequency, F_S, are mathematically related according to the coherency equation

$$F_T = \frac{M}{N} F_S \tag{3.2}$$

Where M and N are positive integers and $M < N/2$ to satisfy the Nyquist Sampling theorem. In this case, F_T, F_S, and N are set, so all that can be changed is M. Additionally, to get the best results, both M and N should be relatively prime - meaning that they should have no common factors. Thus, M can be set to be 1,3,5,7, etc. For the purposes of this thesis, four different reconstruction filters are available to extract the original AC signals from their sigma-delta encoded bitstreams as mentioned in Section

TABLE 3.1: Bit Encoding for Different Phases of DTC

Phase Shift	Bits Encoding	Phase Bit Encoding	Clock Speed
0	4	1100	$4 \times CLK_{ref}$
	8	11110000	$8 \times CLK_{ref}$
45	4	-	
	8	01111000	$8 \times CLK_{ref}$
90	4	0110	$4 \times CLK_{ref}$
	8	00111100	$8 \times CLK_{ref}$
135	4	-	
	8	00011110	$8 \times CLK_{ref}$
180	4	0011	$4 \times CLK_{ref}$
	8	00001111	$8 \times CLK_{ref}$

TABLE 3.2: Possible Relationships Between Encoded Signals

Phase 0	Phase 0 Encoding	Phase 1	Phase 1 Encoding	Phase Range (degrees)
90	0110	0	1100	0-90
180	0011	90	0110	90-180
135	00011110	45	01111000	45-135

TABLE 3.3: Input AC signal frequencies for sigma-delta encoding

M	AC Signal Frequency
1	447.265 kHz
3	1.341 MHz
5	2.236 MHz
7	3.130 MHz

2.7. This results in M values of 1,3,5, and 7 being selected. The resulting frequencies of the input AC signals are thus set, and are enumerated in Table 3.3.

3.3 MATLAB/Simulink Simulations

The software portion of the phase generator was designed and tested in MATLAB and Simulink. Figure 3.2 illustrates the sigma-delta implementation of the software portion of the generator.

The implementation in Figure 3.2 is a simple first-order modulator with a signal transfer function (STF) of 1. The noise transfer function of this modulator is calculated as

$$\text{NTF}(z) = \frac{1}{1 + H(z)} \tag{3.3}$$

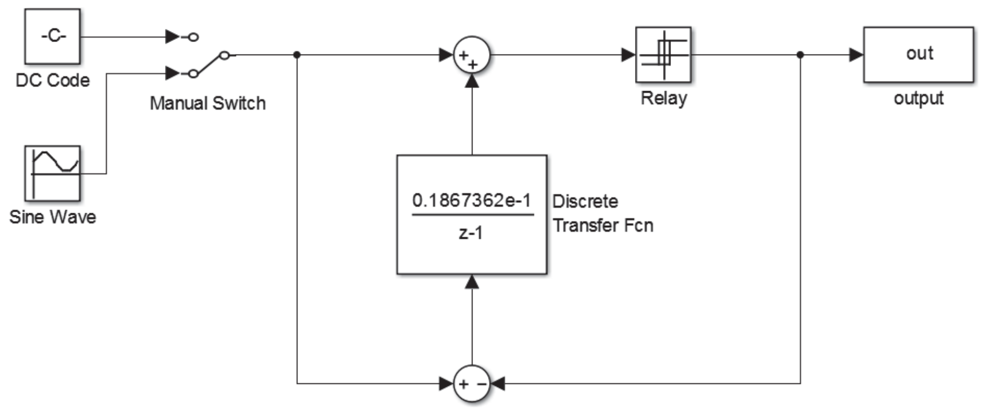


FIGURE 3.2: Simulink block diagram of sigma-delta modulator

A higher-order implementation of the modulator can easily be implemented by changing the filter in the modulator. The filter coefficients were calculated by using DSMOD software [17]. Table 3.4 describes normalized filter coefficients for higher-order implementations. All modulators are designed with $F_p = 1$ rad/s, $F_{Stop} = 1.5$ rad/s and $F_s = 1000$ rad/s for an OSR of 500 using an Inverse Chebychev pole-zero placement algorithm. The resulting bitstream is down-sampled in order to fit into the circular shift register, but without any loss in resolution.

TABLE 3.4: Sigma-Delta Filter Functions

Order	Passband Edge Attenuation	Loop Filter Transfer Function, H(z)
1	10	$\mathrm{H(z)} = \frac{0.1867e-1}{z-1}$
2	20	$\mathrm{H(z)} = \frac{(0.18848e-1)z-0.18672}{z^2-1.999z+1}$
3	30	$\mathrm{H(z)} = \frac{(0.2345e-1)z^2-(0.4663e-1)z+0.2318130e-1}{z^3-2.999z^2+2.999*z-1}$
4	40	$\mathrm{H(z)} = \frac{(0.2868e-1)z^3-(0.8565e-1)z^2+(0.8524e-1)z-0.2828e-1}{z^4-3.999z^3+5.999z^2-3.999z+1}$
5	50	$\mathrm{H(z)} = \frac{(0.3413e-1)z^4-(0.1359)z^3+(0.203)z^2-(0.1347)z+0.3355e-1}{z^5-4.999z^4+9.999z^3-9.999z^2+4.999z-1}$
6	60	$\mathrm{H(z)} = \frac{(0.3966e-1)z^5-0.1975z^4+0.3935z^3-0.3919z^2+0.1952z-0.3888e-1)}{z^6-5.999z^5+14.999z^4-19.999z^3+14.999z^2-5.999z+1}$

The power-spectral density for the bitstream out of the first-order sigma-delta modulator is illustrated in Figure 3.3. As expected, the sigma-delta modulator shows noise-shaping of the signal and pushed the noise out of band where it will be filtered out with the time- or voltage-mode filter.

The architecture in Figure 3.3 is also used to encode the DC code signals into a sigma-delta representation. With the noise-shaping performance of the sigma-delta it is possible to encode very high precision signals with a density of 1s and 0s.

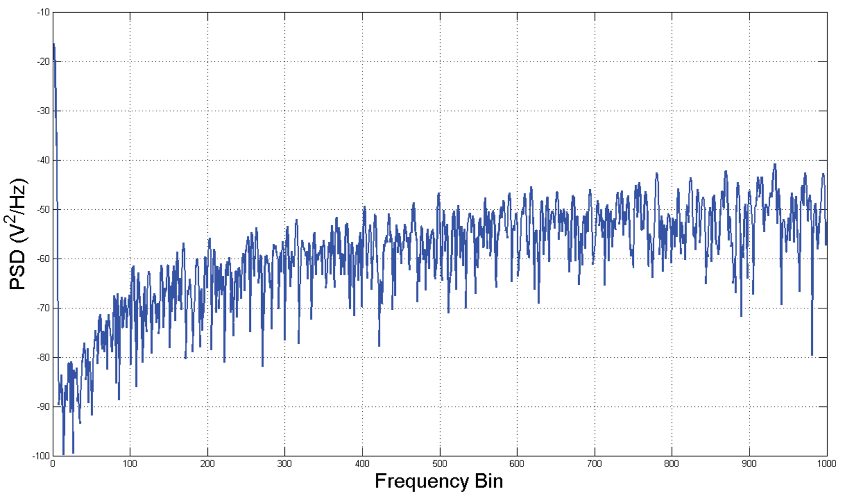


FIGURE 3.3: Power spectral density plot for 27.9 kHz signal modulated with an OSR of 2048

Table 3.5 represents the encoding schemes for various DC codes spaced by DC codes of 0.2. Intermediate DC code encoding is possible but at the expense of a greater number of circulating bits.

TABLE 3.5: The Sigma-Delta Bit-Encoding for Various DC Codes

DC Code	Sigma-Delta Encoding (repeating)
0	0
0.2	00001
0.4	00101
0.5	01
0.6	11010
0.8	11110
1	1

Figure 3.4 illustrates a time-domain representation of a 447.625 kHz sigma-delta encoded sine wave with an over-sampling ratio of 128. The original signal is shown in red, and the sigma-delta encoded representation is shown in blue. Similarly, Figure 3.5 illustrates a time-domain representation of a sigma-delta encoded DC code of 0.2 with an over-sampling ratio of 500.

3.4 Summary

This chapter covered the design and implementation of the sigma-delta encoding technique that is used to encode the stimulation signals that are used in the PTI. The advantages of implementing the bitstream in software was introduced, as well as the

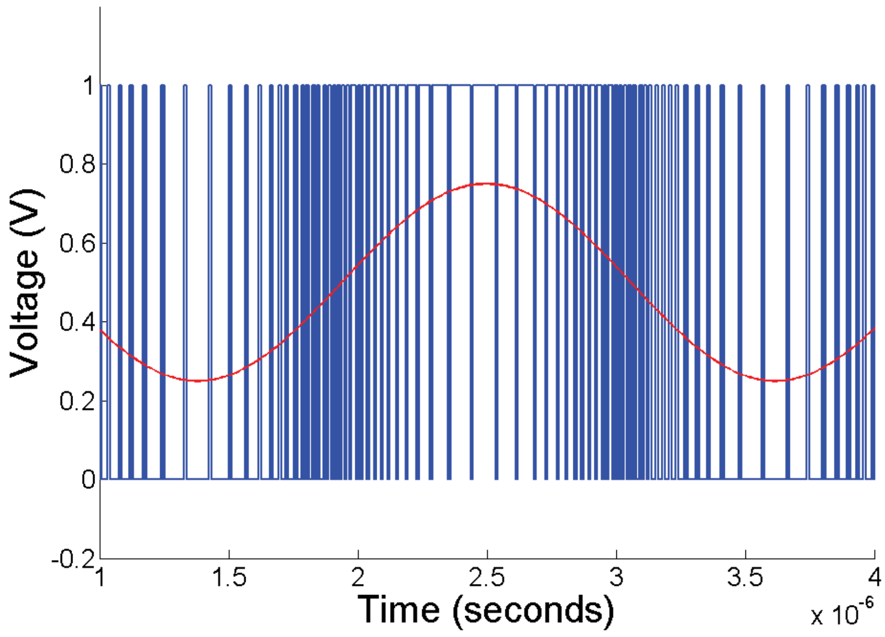


Figure 3.4: Sigma-delta encoded 447.625 kHz sine wave with OSR of 128

methods for generating both AC and DC input signals. The bitstreams are stored in a 1XN register that circulates N encoded bits in a repeating fashion. The bitstream is subsequently filtered by a filter whose type depends on the encoding of the signal (voltage or time). A theory of the design and operation of the bitstream generator was covered, as well as the possible phase-encodings used. Finally, time-domain simulations were using MATLAB/Simulink run to verify the correctness of the implementations.

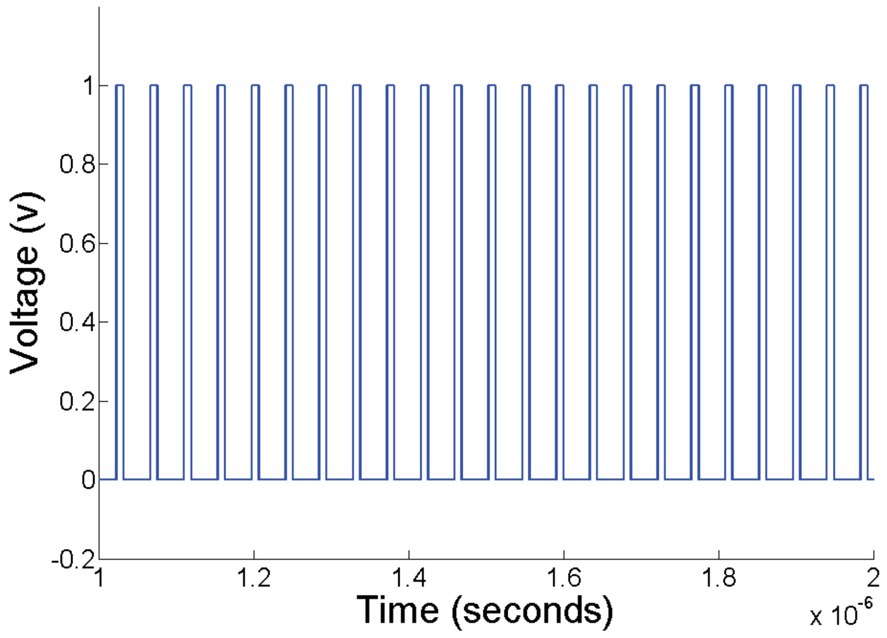


FIGURE 3.5: Time-domain representation of a sigma-delta encoded DC code of 0.2 with an over-sampling ratio of 500

Chapter 4

Delay-Locked Loop as a Time-Mode Filter

This chapter will discuss using a delay-locked loop (DLL) as a time-mode filter, as well as give an introduction to a new method for designing higher-order DLLs. It will cover the design and simulation results for the implementation of the DLLs used in the design of the probabilistic test instrument. Simulations were first done with MATLAB/Simulink to verify the proposed theory of operation and finally transistor-level simulations were completed with the use of Cadence/Virtuoso.

There are many systems today that require the control of the edge of a clock signal. That is, being able to accurately vary the phase of the clock signal to a very high resolution. Although it is possible to simply delay the phase of the signal using a delay element, this delay will not be robust to variations in process, voltage, or temperature when then device in physically manufactured in silicon. Instead, the use of a DLL where the output is fed back to the input by the use of a feedback loop, allows a designer to very accurately control the phase of the signal when the DLL is used as a phase-reconstruction filter. As is true when designing a PLL, the goals when designing a DLL are very similar: the main issues are to reduce the static phase offset at the output, as well as keeping a very low amount of jitter on the output clock edge.

As supply voltages continue to drop, it is becoming more and more difficult to achieve acceptable power-supply noise induced jitter when using a PLL [18]. While it is possible to design for these difficulties, it is important to note that a system does not fundamentally require the implementation of a voltage-controlled oscillator when the system already supplies a clock at the correct frequency. In this respect, the implementation of a DLL is a smarter choice over a PLL because it is possible to eliminate some of the shortcomings and complexity of the implementation of a VCO.

A DLL can only work with one frequency so, when compared with a PLL, there are fewer applications where it could be implemented. However, in those applications it is possible to see better results due to the lack of phase accumulation at the VCO. The application that will be discussed in this thesis is the use of a DLL as a phase reconstruction filter such that it can be used to accurately control the edge of a clock signal.

4.1 Method for Designing Higher-Order Delay-Locked Loops

One of the contributions of this work is that a different method of designing higher-order DLLs is presented. The general idea of this method was first suggested and researched by Sadok Anouini, but with the principles focused on designing PLLs [19]. The method presented here extends this work to facilitate the design of higher order design of DLLs as well [6].

Most higher order DLL design methods take on a dominant pole approach where the designed will set the bandwidth of the DLL and then, once connected in feedback, the overall frequency response and stability will be determined. The problem with this approach is that the overall frequency response of the DLL is not explicitly specified – it is just the result of the feedback-connected design and is not necessarily the frequency response that the designer is striving for. An alternative design approach that is presented here is one where the overall desired frequency response of the DLL is set, and then the filter coefficients are determined based on both the delay-line and the charge-pump constants. This will result in a design that is exactly what the designer is looking for. This method allows for a general pole-zero placement synthesis method for designing DLLs of arbitrary order with an emphasis on Type-II DLLs.

The data input the to the DLL can be broken down into two components - the constant clock component (which is the same as the reference frequency), as well as the excess phase component. If an analysis is performed on the block diagram, it can seen that the excess-phase transfer function from input to output – when using the filter as an integrator – is a low-pass response (see Section 2.2).

The DLL design process begins by approximating both the excess phase transfer function,

$$\Phi_A(s) = \frac{c_N s^N + c_{N-1}s^{N-1} + ... + c_1 s + c_0}{s^N + d_{N-1}s^{N-1} + ... + d_1 s + d_0} \tag{4.1}$$

and the filter function,

$$F(s) = \frac{a_N s^N + a_{N-1}s^{N-1} + ... + a_1 s + a_0}{s^N + b_{N-1}s^{N-1} + ... + b_1 s + b_0} \tag{4.2}$$

as a ratio of two polynomials of N^{th} order. Following from Equation 2.18, it is possible to represent the excess-phase of the Type-II transfer function as

$$\Phi_A(s) = \frac{K_D K_P F(s)}{1 + K_D K_P F(s)} \tag{4.3}$$

And substituting in Equation 4.2,

$$\Phi_A(s) = \frac{K_D K_P \frac{a_N s^N + a_{N-1}s^{N-1} + ... + a_1 s + a_0}{s^N + b_{N-1}s^{N-1} + ... + b_1 s + b_0}}{1 + K_D K_P \frac{a_N s^N + a_{N-1}s^{N-1} + ... + a_1 s + a_0}{s^N + b_{N-1}s^{N-1} + ... + b_1 s + b_0}} \tag{4.4}$$

After simplifying, this leads to

$$\begin{aligned}\Phi_A(s) = & \frac{K_D K_P a_N s^N + K_D K_P a_{N-1} s^{N-1} + ...}{(1 + K_D K_P a_N)s^N + (b_{N-1} + K_D K_P a_{N-1})s^{N-1} + ...} \\ & \frac{... + K_D K_P a_1 s + K_D K_P a_0}{... + (b_1 + K_D K_P a_1)s + (b_0 + K_D K_P a_0)}\end{aligned} \tag{4.5}$$

Since the DLL is being designed is for a low-pass functionality as discussed in Section 2.2, it is desired to have the signals passed without any degradation. For this reason, the DLL is designed to have a DC gain of 1 so that the signals below the cut-off frequency will not be suppressed. To realize this constraint, a pole is needed at DC in Equation 4.5. By observation, it can be seen that this constraint sets the requirement for $b_0 = 0$. With this set, the equation for the filter function can be simplified from Equation 4.2 to

$$F(s) = \frac{a_N s^N + a_{N-1}s^{N-1} + ... + a_1 s + a_0}{(s^{N-1} + b_{N-1}s^{N-2} + ... + b_1)s} \tag{4.6}$$

Now, working backwards from the loop filter transfer function, it is possible to solve for the filter function from Equation 4.3 to get

$$F(s) = \frac{1}{K_D K_P} \frac{\Phi_A(s)}{1 - \Phi_A(s)} \tag{4.7}$$

Substituting Equation 4.1 into Equation 4.7 and simplifying leads to

$$\Phi_A(s) = \frac{1}{K_D K_P} \frac{c_N s^N + c_{N-1} s^{N-1} + ... + c_1 s + c_0}{(1-c_N)s^N + (d_{N-1}-c_{N-1})s^{N-1} + ... + (d_1 - c_1)s + (d_0 - c_0)} \tag{4.8}$$

Given the realization conditions that were set previously for the DC gain of 1, the two lowest coefficients must be equal. This sets the constraint of $c_0 = d_0$ and it follows that the overall phase-filter function general form should be changed from Equation 4.1 to take the form

$$\Phi_A(s) = \frac{c_N s^N + c_{N-1} s^{N-1} + ... + c_1 s + d_0}{s^N + d_{N-1} s^{N-1} + ... + d_1 s + d_0} \tag{4.9}$$

This shows that to have a DC gain of 1, a pole is needed at DC in the filter function. Working from a perspective of realizing a desired excess phase input-to-output transfer function, denoted by $\Phi_A(s)$, according to the analysis above, as long as the desired transfer function takes on the general form in Equation 4.9 where the lowest two coefficients are equal, the resulting loop filter will contain a pole at DC.

Given a desired overall phase-transfer function, the filter coefficients can now be determined from this transfer function. Combining Equations4.6, 4.7, and 4.9, the filter function can be written as

$$F(s) = \frac{\frac{1}{K_P K_D}\left[\frac{c_N}{1-c_N}s^N + ... + \frac{c_2}{1-c_N}s^2 + \frac{c_1}{1-c_N}s + \frac{d_0}{1-c_N}\right]}{s^N + \frac{d_{N-1}-c_{N-1}}{1-c_N}s^{N-1} + ... + \frac{d_1 - c_1}{1-c_N}s} \tag{4.10}$$

It follows that the loop filter, $F(s)$, realization conditions are summarized below:

$$\begin{aligned} a_N &= \tfrac{1}{K_D K_P}\tfrac{c_N}{1-c_N} \\ a_{N-1} &= \tfrac{1}{K_D K_P}\tfrac{c_{N-1}}{1-c_N} \\ &... \\ a_1 &= \tfrac{1}{K_D K_P}\tfrac{c_1}{1-c_N} \\ a_0 &= \tfrac{1}{K_D K_P}\tfrac{d_0}{1-c_N} \end{aligned} \tag{4.11}$$

$$\begin{aligned} b_{N-1} &= \tfrac{d_{N-1}-c_{N-1}}{1-c_N} \\ &... \\ b_1 &= \tfrac{d_1-c_1}{1-c_N} \\ b_0 &= 0 \end{aligned}$$

The use of a pole-zero placement method to form high-order DLLs greatly simplifies the design process and ensures that the overall transfer function matches the desired transfer function. Using this approach, designing high-order DLLs is a very simple process.

4.2 Start-Up Circuit

When the DLL is implemented at the transistor-level, it is important for the designer to be aware of the practical design limitations that the DLL circuit is constrained to. In the case of the DLL the VCDL has a limited range over which the signal can be delayed. The range is approximately 0-1.4 periods of delay. The architecture is such that a low control voltage will result in a small delay, and a large control voltage will result in a large delay. Once the VCDL is placed inside the DLL, it is important that the control pulses from the charge-pump are adjusting the control voltage in the appropriate direction to ensure that the control voltage has not reached VDD or VSS. For example, a situation where the control voltage is giving incorrect pulses in shown in Figure 4.1.

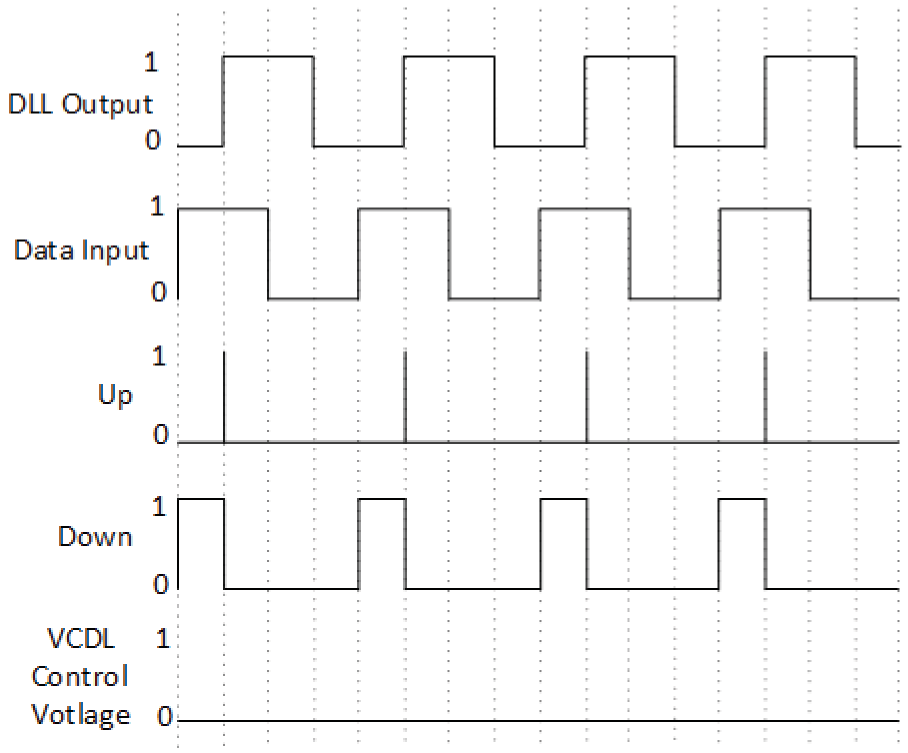


FIGURE 4.1: Example of an incorrectly starting DLL

In Figure 4.1, the VCDL control voltage initially starts at 0, but the controlling signal needs to be moved slightly to the left (back in time, or removing phase). For this to happen, the phase detector will be sending "down" pulses to the phase-detector which will, in turn, attempt to apply less voltage to the control line of the VCDL. However, since the VCDL control voltage is already at zero, the voltage on the control line will remain unchanged. To achieve a proper lock, the DLL should increase the phase of the output signal such that it is delayed by almost one full period. At this point, the VCDL will be able to properly match the phase of the input signal such that a proper lock is achieved.

To solve this problem, a start-up circuit was designed for the DLL. The start-up circuit ensures that when the VCDL control voltage is started from a zero voltage, the first pulse that is seen by the charge pump is an "up" pulse such that the voltage on the delay line will always be increased, regardless of the order of the incoming signals. If this were not the case, then it is possible that a "down" pulse is issued first and the DLL will never achieve locking conditions.

The start-up circuit architecture is a simple modification to the input of the PFD where there are simply two D-flip-flops that will ensure that the "up" signal will be asserted first by making sure that the data input edge is detected before the DLL output feedback edge. It works by placing a D-flip-flop at the D-inputs to the flip-flops in the PFD. Typically, these are always tied to a logical high, but in this case, they will only be high once the data signal edge has been detected. This will discard any previous feedback DLL output edges until a data signal edge has been detected. The architecture of this design is illustrated in Figure 4.2.

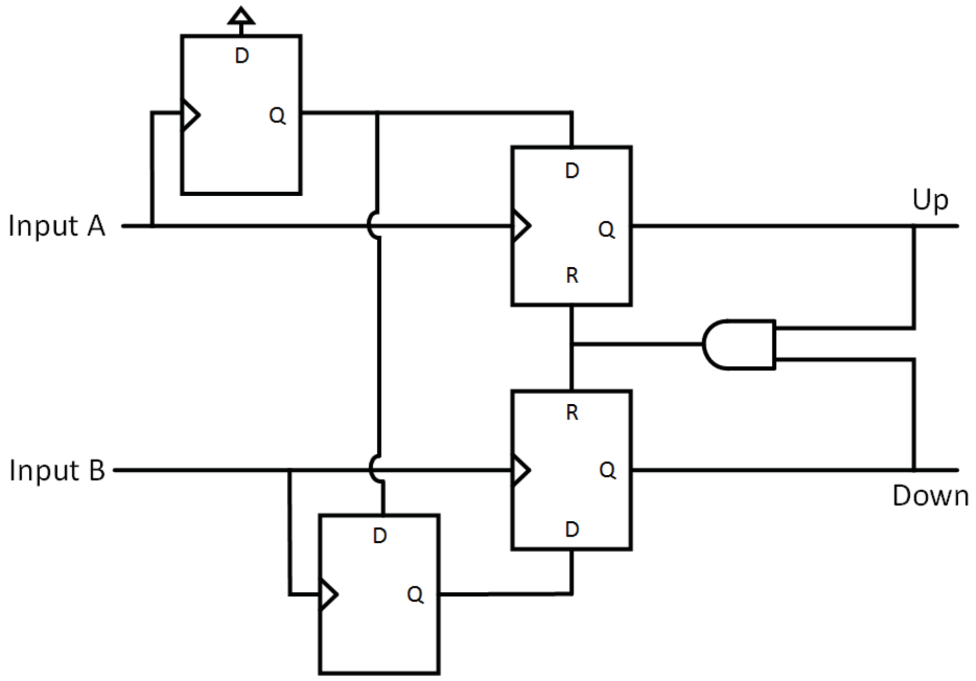


Figure 4.2: DLL start-up circuit to ensure proper locking conditions

With the modified PFD shown in Figure 4.2, with a control voltage of initially zero, it is possible to guarantee that the controlling circuit will align the input and output pulses by changing the phase of the output signal in the correct direction given the delay limitations of the VCDL. The initially zero voltage on the control voltage line is achieved by using a single nMOS transistor and an initial setup control line.

4.3 Sub-Gate-Delay Edge-Control of a Clock Signal Using DLLs and Sigma-Delta Modulation Techniques

4.3.1 Background

The most widely used method for edge control is one where a DLL is implemented and the edge locations are tapped out at each point in the delay line [20]. The resolution achievable with this method, however, is limited by the minimum gate delay available in the delay line. In the approach described in this section, a DLL will be used as a reconstruction filter acting on a sigma-delta modulated bitstream, which will allow for sub-gate-delay edge control [6].

In this work, the type of DLL that is classified as type-II will be considered (Section 2.2). Specifically, this is one in which the input to the voltage-controlled delay line (VCDL) and the phase detector (PD) are separate signals as shown in Figure 2.3. To achieve sub-gate-delay timing resolution, a clock reference will be applied to the input of the VCDL, and a phase-modulated sigma-delta bitstream will be applied to the input of the phase detector. The use of a phase-modulated bitstream will allow for an accurate control of the output signal edge [12][21].

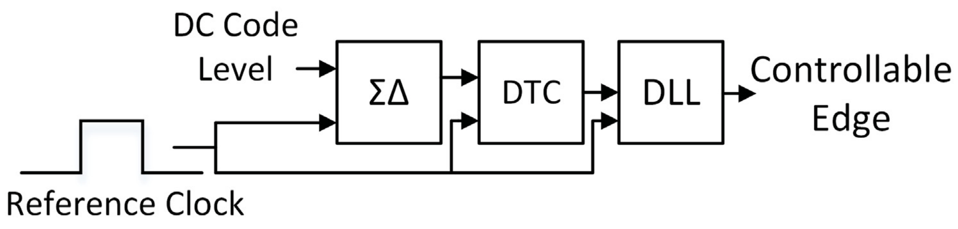


Figure 4.3: System diagram with a DLL reconstruction filter

An overview of the sub-gate-delay edge control circuit is shown in 4.3. Here the system consists of three main blocks: a 1-bit sigma-delta modulator, a digital-to-time converter and a type-II DLL. The input to the system is a DC code level – anywhere between 0 and 1 – that will be encoded with a one-bit sigma-delta modulator. The output of this modulator is converted into a time signal using a digital-to-time converter (DTC). As the modulator output is 1-bit wide, the output of the DTC will select between two distinct time instants, essentially creating a phase modulated output signal with two distinct phases. This phase modulated signal is then applied to the input to the DLL, which, in essence, acts as a time-domain or phase reconstruction filter. Hence, the output of the DLL is a lowpass-filtered version of its input, resulting in a precisely positioned edge relative to the DLL clock reference as illustrated in Figure 4.4. The control of the edges is attained by simply adjusting the input DC code level at the input of the sigma-delta modulator. The use of a sigma-delta modulator allows for a very high-precision encoding where the quantization noise can subsequently be filtered out using a low-pass DLL as a reconstruction filter.

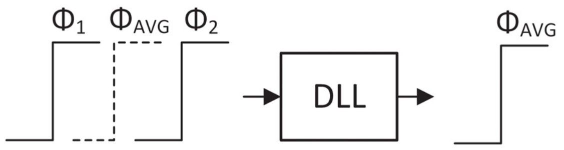


FIGURE 4.4: Illustrating the phase extraction process when the DLL extracts the average phase between the densities of two coarse delays

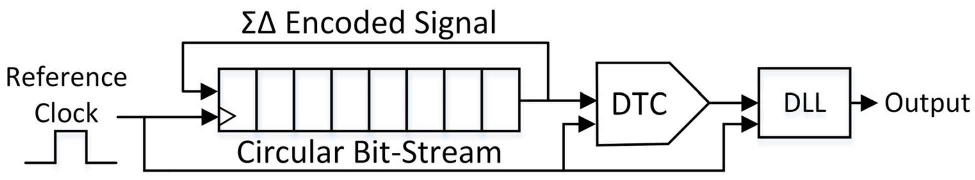


FIGURE 4.5: Implementation of phase signal generation scheme. Circular memory contains the sigma-delta encoded signal in the phase domain and the DLL is used to extract it.

Figure 4.5 illustrates the implementation that will be used for the sigma-delta encoding of the DC code level. Rather than use a physical sigma-delta block, a circular memory block is used to output the encoded bitstream [22]. This approach will reduce the hardware complexity required for this design when compared to that in [21] as it will only require a bank of circular memory to excite the DLL. Additionally, since the circular memory is programmable, it is also possible to excite the DLL with a wide variety of encoded signals. In this application, the bit sequence corresponding to a specific DC code is captured from a sigma delta modulator and stored in the circular memory where it will be repeatedly shifted to the right at a frequency of f_s. This repeated shifting mimics the output of the sigma-delta modulator. A DLL is then used to reconstruct the encoded signal.

The DLL that was designed is a type-II DLL as it provides two possible time-domain filtering functions. The transfer function from the PD input to the DLL output can be shown to be low-pass in nature, whereas the transfer function from the clock reference input to DLL output is high-pass in nature. Thus, by injecting the sigma-delta encoded signal into the lowpass terminal, it is possible to filter out the high-frequency quantization noise associated with the DTC output edges and be left with a very clean signal whose average phase is directly related to the input DC code level.

The lowpass filter characteristics of the DLL will be determined by the transfer functions of the individual blocks that make up the DLL. Through the appropriate selection of components, the DLL can be made to behave in a prescribed manner. The design process for this DLL is outlined in Section 4.1. As a design example, a third-order DLL was designed and simulated to showcase the advantage of the higher-order designs, as well as to show the effectiveness of the design technique described earlier in this section. The sigma-delta modulator was designed with a passband of 100 kHz. In order to filter out the quantization noise, the closed-loop response of the DLL was set to have a 3rd-order Bessel response with a 3-dB bandwidth of 40 kHz, leading to

$$\Phi_{II}^{D}(s) = \frac{1.588 \times 10^{16}}{s^3 + 6.11 \times 10^5 s^2 + 1.558 \times 10^{11} s + 1.588 \times 10^{16}} \tag{4.12}$$

The phase detector is to be implemented as a sequential phase detector whose gain coefficient KP is given by

$$K_P = \frac{I_P}{2\pi} \tag{4.13}$$

For a bias level 1 mA, the phase detector coefficient K_P is 0.16 mV/rad. The VCDL is implemented as a cascade of 16 inverters each with a capacitive load and is characterized to have a voltage-to-phase gain coefficient K_D of 7.35 rad/V. The loop filter transfer function $F(s)$ can then be derived to find

$$F(s) = \frac{1.357 \times 10^{19}}{s^3 + 6.114 \times 10^5 s^2 + 1.558 \times 10^{11} s} \tag{4.14}$$

The loop filter in Equation 4.14 is implemented using an active filter between the phase detector output and the delay line. A Sallen-Key filter topology was used in addition to a single capacitor and voltage buffer. Figure 4.6 shows the correctness of the previous design method by comparing the desired transfer function to the measured transfer function of the DLL.

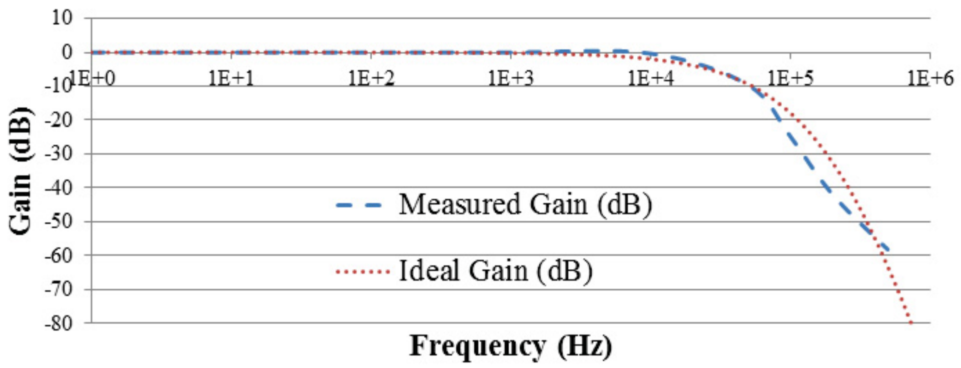


FIGURE 4.6: Ideal and measured responses of the designed DLL

4.3.2 Simulink Ideal Simulations

In order to verify the ideas of this chapter, two separate MATLAB/Simulink simulations were performed. The first involved the setup shown in Figure 4.3, where the output of the sigma-delta modulator was sent directly to the DTC input. The second involved the setup shown in Figure 4.7 where the output of the sigma-delta modulator was truncated to 1024 bits and the resulting bitstream was loaded into a circular memory bank. Next, the input DC code level was adjusted and the edge of the output signal from the DLL was recorded. The results for the two separate situations are shown in Figure 4.8. As is evident from this plot, over an input DC level range of 0 to 1, the output edge followed the input DC level in a linear manner. It is important to note that in this graph, the input DC code delta that was simulated is 0.05, however the minimum code difference that can be simulated with 1024 bits is 1/1024.

In analyzing the results in Figure 4.8, there was no noticeable difference between the two approaches involving the sigma-delta modulated output or the circulated bitstream – the largest error was 0.07^o. With this very linear DC-code-to-phase-transfer characteristic for the DLL, it is then possible to adjust the edges in a very predictable fashion, and it will be possible to modulate any type of waveform onto this edge with very little distortion.

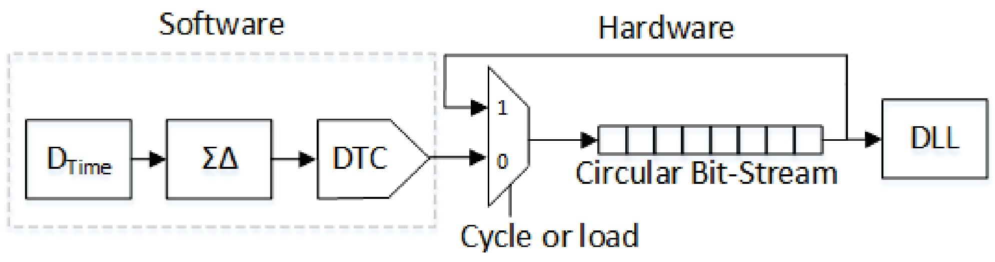


FIGURE 4.7: Sigma-delta encoding of the DC code level is done in software and it is filtered by the DLL implemented in hardware

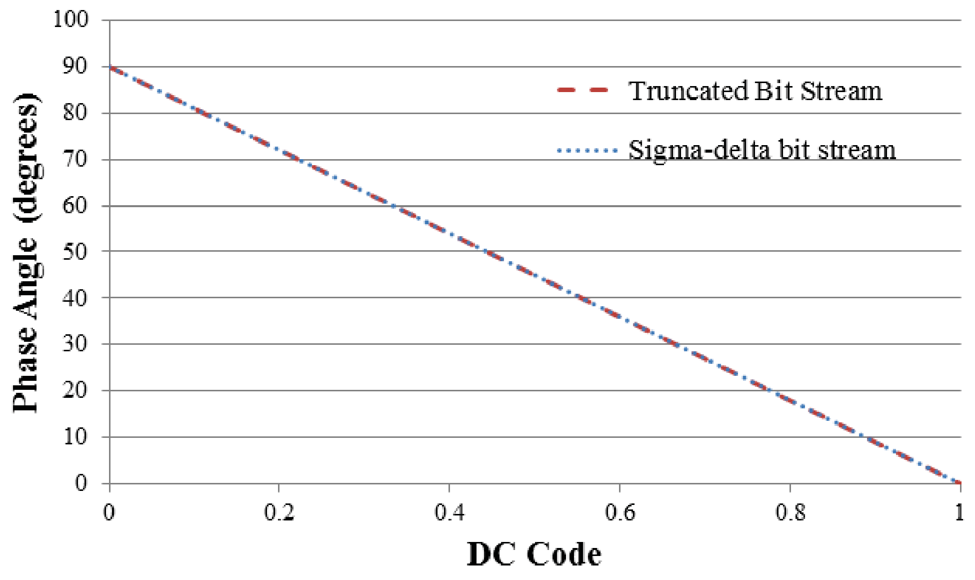


FIGURE 4.8: DC code-controlled phase offset. This curve is a locus of points with a resolution of 2.44 ps due to the maximum use of 1024 bits.

Figures 4.9 and 4.10 illustrate the phase error of the edge when the response has settled to a steady-state value for a full sigma-delta input and a truncated input, respectively. It can be noted that the RMS jitter of the edge is much greater in Figure 4.10 with the truncated bit sequence. The reason for this is due to the fact that the circular bitstream does not contain an exact multiple of sigma-delta bitstream periods, so there is some extra error that is introduced.

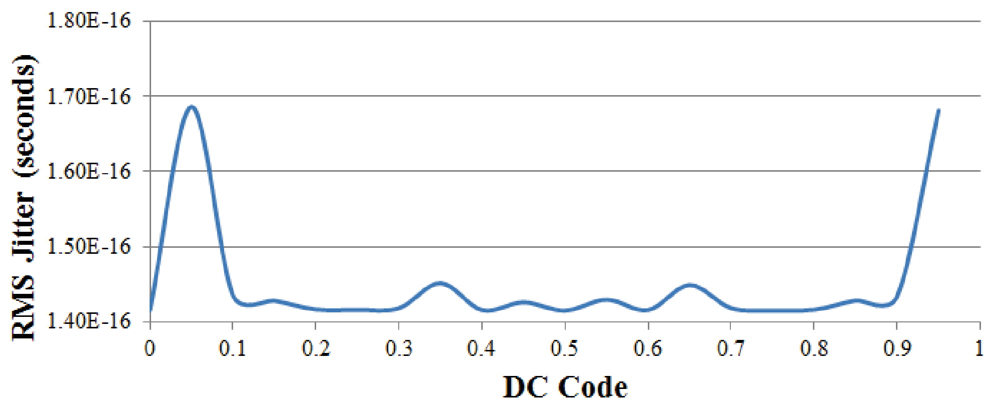


FIGURE 4.9: RMS jitter of the edges with a full sigma-delta bit sequence. A DC Code delta of 0.05 was used.

The DLL as a time-mode filter works with the same principles as a voltage-mode filter. The lower the loop bandwidth of the DLL, the more the higher frequencies will be suppressed. Therefore, to have a DLL with the least amount of output jitter, the designer should use the smallest loop bandwidth possible. The drawback of using a

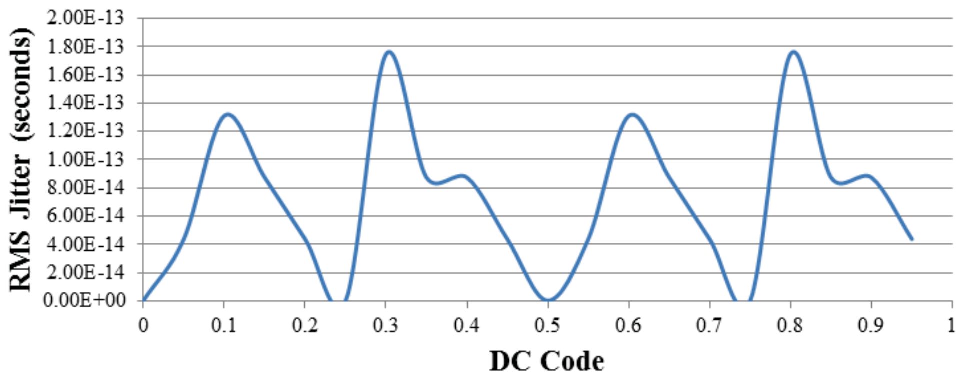


FIGURE 4.10: RMS jitter of the edges with a truncated bit sequence to 1024 bits. A DC Code delta of 0.05 was used.

very small loop bandwidth is that the settling time for the DLL will be increased as the loop bandwidth is decreased. Figure 4.11 illustrates the difference in the peak-to-peak jitter of the output signal with two different bandwidths. The DLL with the 825 kHz bandwidth has a maximum peak-to-peak jitter of 2.49×10^{-9} seconds, while the DLL with a bandwidth 1000 times smaller at 825 Hz has a maximum peak-to-peak jitter of 2.38×10^{-11}.

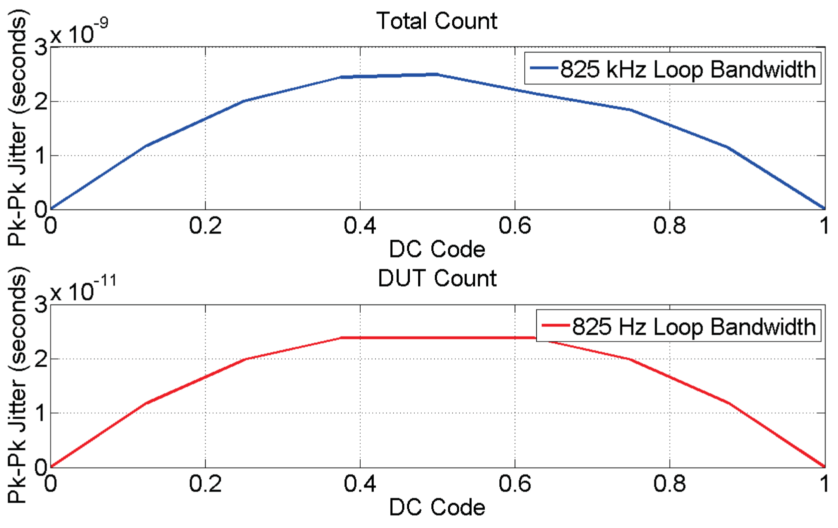


FIGURE 4.11: Effect of the loop bandwidth of the DLL on the peak-to-peak jitter of the output signal.

If one were to want to reduce the jitter of the output of the DLL, the circular bitstream size could be adjusted to match a full period of the sigma-delta modulator, the system's overall bandwidth could be reduced, or a higher-order DLL could be implemented.

4.3.3 Transistor-Level Simulations

Following the proof-of-concept of the design with Simulink, the system was prepared for fabrication in a 130 nm CMOS process from IBM. The design is based on the implementation of Figure 4.7. As the models used at this level are no longer ideal, it is expected to see some differences from the MATLAB/Simulink simulations.

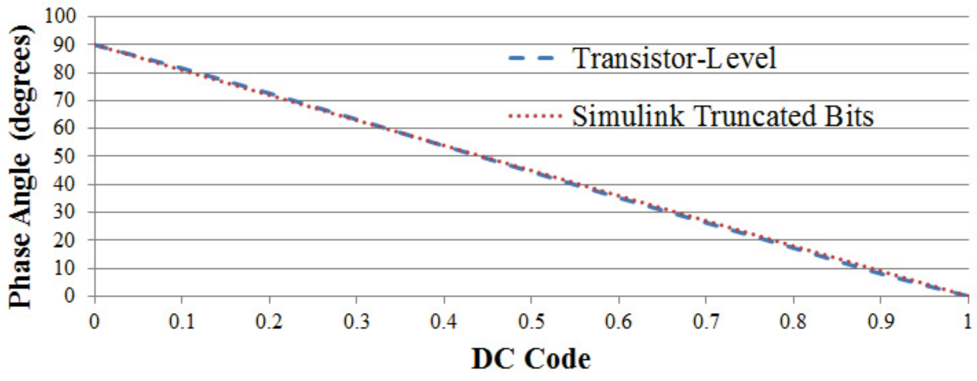


FIGURE 4.12: Input code level-controlled edge location obtain from the transistor level implementation compared to the ideal truncated output. A DC Code delta of 0.05 was used.

Figure 4.12 illustrates the DC code-controlled phase offset of the circuit. Like before, Figure 4.12 depicts a linear relationship between the input DC code level and the average edge location, with a maximum error for the transistor-level simulations of 0.7^o. The RMS jitter could not be properly collected due to the accuracy limits of the simulator. RMS jitter metrics will be collected once a physical design has been fabricated.

As shown in Figures 4.8 and 4.12, the phase angle is tunable from 0 to 90 degrees. The minimum input code level difference that can be implemented with 1024 bits is 1/1024, which results in a phase step of 0.088 degrees (2.44 ps at 100 MHz), however the accuracy of the edge is largely dependent on the amount of jitter on the edge. When comparing the proposed design of adjusting the phase of the edge with the use of a sigma-delta modulated bitstream with previous methods using a tapped out delay line, it is obvious that this proposed method is superior in terms of edge placement resolution. As described in Section 2.2.4, the DLL that was implemented utilizes a VCDL built with 16 separate inverters. If one were to use this DLL to extract different clock signals, the minimum phase-resolution would be $360/16 = 22.5$ degrees. With the new proposed implementation, it is shown that it is possible to tune the edge of the DLL output to a significantly greater resolution.

Figure 4.13 illustrates a rough approximation of the edge locations at the output of the DLL, with an input DC code of 0.6. The spread of the edges is representative of the

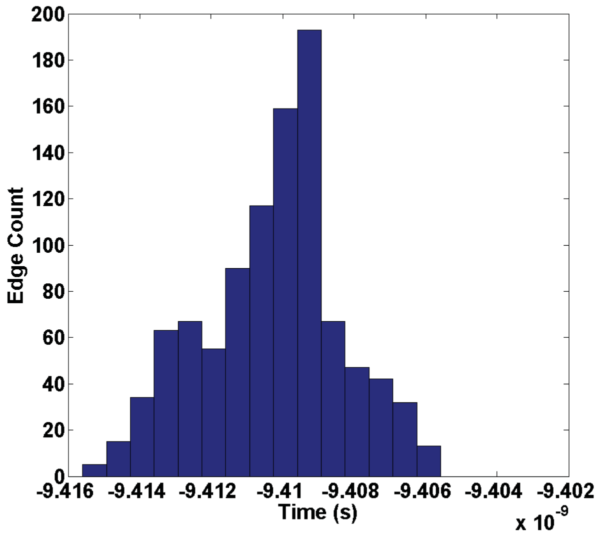


FIGURE 4.13: Histogram of rising edge locations for a settled DLL for an input DC code of 0.6.

jitter of the signal at the output. As mentioned previously, it was not possible to extract the exact edge locations for the calculation of a jitter metric due to the limitations of the simulator. The results of the approximate edge locations from Figure 4.13 are what were expected with a Gaussian-like distribution of the edge locations.

4.4 Summary

This chapter discussed using a delay-locked loop (DLL) as a time-mode filter, as well as gave an introduction to a new method for designing higher-order DLLs. It covered the design and simulation results for the implementation of the DLLs used in the design of the PTI. Simulations were first done with MATLAB/Simulink to verify the proposed theory of operation and finally transistor-level simulations were completed with the use of Cadence/Virtuoso. It was shown that it was possible to use a DLL in combination with an input bitstream to control the phase of the output signal.

Chapter 5

Architecture of the Probabilistic Test Instrument

This chapter presents the design of the probabilistic test instrument (PTI) that will be used to extract the edge-location timing information for a high-speed circuit. The extracted edge-locations will be used to construct a PDF that will be used to extract the bit-error-rate (BER) associated with the communication channel and receiver. The circuit is designed for embedded built-in self-test (BIST) methods that will reduce the error that is currently present with probing methods, and is applicable to high-speed transceiver circuits for data communications. This chapter, along with Chapter 2, discusses the working principles of the basic building blocks that make up the design of the PTI. The ideal simulation results using MATLAB and Simulink are presented in this chapter, and the transistor-level simulations will follow in Chapter 6. Any signal references are described in between quotation marks (“”) and these signals are outlined in Appendix A.

5.1 Probability Test Instrument System Architecture

The probabilistic test instrument (PTI) is designed to run with a clock speed of 114.5 MHz and consists of two main modules: the multi-channel programmable bitstream generator (BSG), and the probability extraction unit (PEU). The system architecture follows from the design first described in [13], but many design modification and improvements have been made. The BSG serves to generate the stimulating test signals for the device under test (DUT) as well as the clock signals for the timing and sampling inside the PEU. The BSG has many different functions and can generate many different AC, DC, phase-modulated, and pseudo-random bit sequences through the use of an array of circular bitstreams and filters. Multiplexers and control lines are used to load the bi streams and to select the desired functions of the BSG.

The PEU is the unit that does the extraction of the timing information relating to the DUT and compiling it into a histogram for easy readout and calculation of the BER. The architecture of the PEU is fairly simplistic and is composed of mostly digital logic which makes the design very portable to different technologies. Additionally, as discussed in Chapter 1, the PEU loading of the DUT circuit is insignificant as the added capacitance is simply the input capacitance to the comparator. Further, the performance of the DUT will always be the same as the test circuit will always remain on-chip and no connections are being added or removed for the testing. Figure 5.1 illustrates the overall system architecture of the design.

5.2 Multi-Channel Programmable Bitstream Generator

The multi-channel programmable bitstream generator (BSG) is composed of both software and hardware components. The function of the BSG is to generate the input test signals fed into the PEU such that the performance of the DUT can be measured. As illustrated in Figure 5.2, there are two distinct portions of the BSG: software and hardware. The software portion contains the components to generate the input test stimulants, while the hardware side contains the circular memory to circulate the generated bitstream, the filters to extract the desired frequencies of the signals, and multiplexers and D-flip-flops to control the routing of the signals to the correct outputs.

The BSG has 5 different input channels labelled 1-5 in Figure 5.2. The different channels are used to stimulate the DUT, as well as regulate the timing and voltage threshold levels for the PEU. Having a multitude of test channels enables the test engineer to extract a wide array of information from the DUT by using simple controls and bit-loading techniques.

Channel 1 is used as a pseudo-random bit sequence test stimulant. This channel is used to load the circular memory with any bit sequence that the test engineer would like to input into the DUT. Throughout the tests completed in this thesis, a repeating "1010" pattern in the PRBS input due to the fact that this is the input sequence that is required in both the amplitude- and time-based scan tests described in Sections 1.3.1 and 1.3.2. The PRBS can be programmed to have a repeating bit length of 20 bits.

Channel 2 is used to control the timing of the bits from channel 1 to the DUT. First, the software portion of this channel uses a sigma-delta modulator to modulate with either a DC code or an AC signal depending on what kind of timing the test engineer

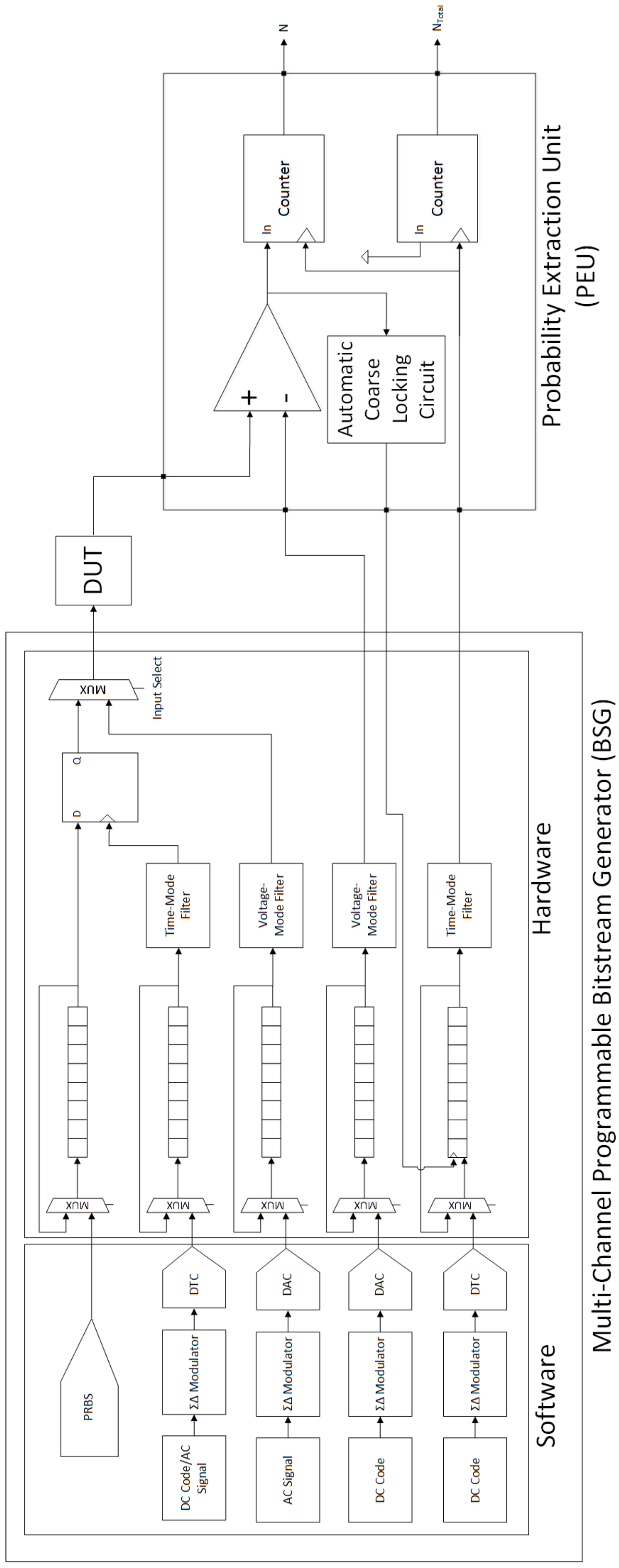


Figure 5.1: Overall system design of the PTI

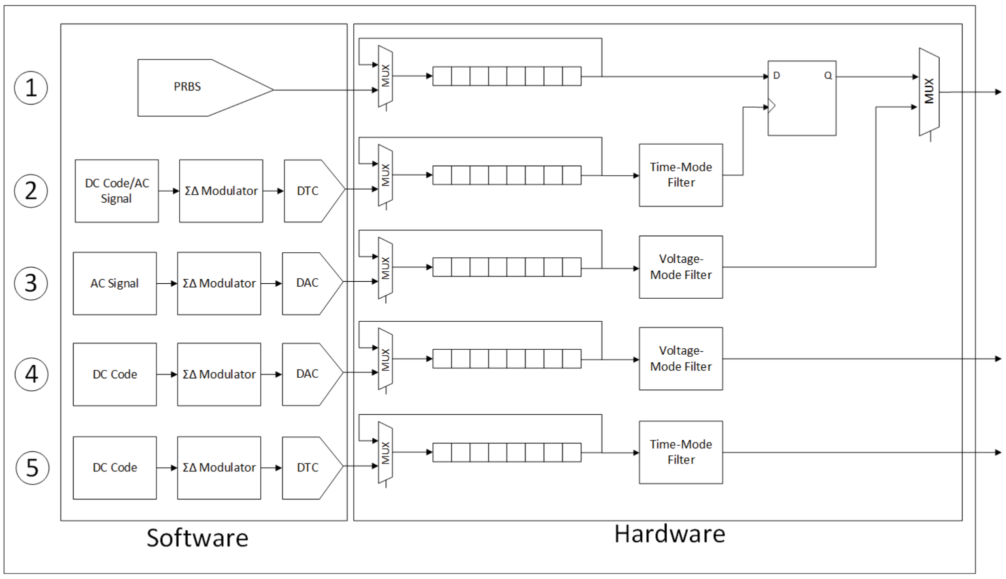


FIGURE 5.2: Multi-channel programmable bitstream generator

would like to generate. Next, the bit sequence is fed through a DTC where it is converted into a time-modulated signal as described in Section 2.5. The time-modulated signal is stored in a 256-bit circular memory where it is circularly shifted to generate a test signal. A time-mode filter (covered in Section 2.8) extracts the DC component of the encoded signal such that the appropriate timing for channel 1 is achieved. The ability to encode both a DC code and an AC signal allows the test engineer to control the amounts of input jitter that is seen on the edges of the PRBS. An encoded DC signal will provide a clean jitter-free edge, while an AC signal will provide an edge that varies in relation to the encoded AC frequency and amplitude. This timing option allows a great deal of flexibility for the test engineer to test the DUT and perform jitter-transfer and jitter-tolerance tests. Given the fact that the output of the circular bitstream is time encoded with 4 bits per period, and additionally used to clock a D flip-flop, the shifting of the bits in the register needs to be done at 8 times the frequency of the desired signal at the output of the D flip-flop. With the DUT working at a frequency of 28.625 MHz, the overall shifting speed must be done at 229 MHz (twice the frequency of the reference clock). To achieve this, a multiplying PLL must be used as illustrated in Figure 5.3.

Channel 3 is used to stimulate the DUT with a simple AC signal. The AC signal is encoded with sigma-delta modulation techniques, fed through a digital to analog converter (DAC), and stored in a 256-bit circular memory. Using different voltage-mode filters, the encoded AC signal is extracted and presented at the input of the DUT by correctly selecting the analog multiplexer control signal of channel 3. The voltage-mode filter (covered in Section 2.7) can be tuned to 4 different cutoff frequencies: 830 kHz, 2.48 MHZ, 4.15 MHz, and 5.80 MHz. The design of these filters and the associated control signals is described in Section 5.4.

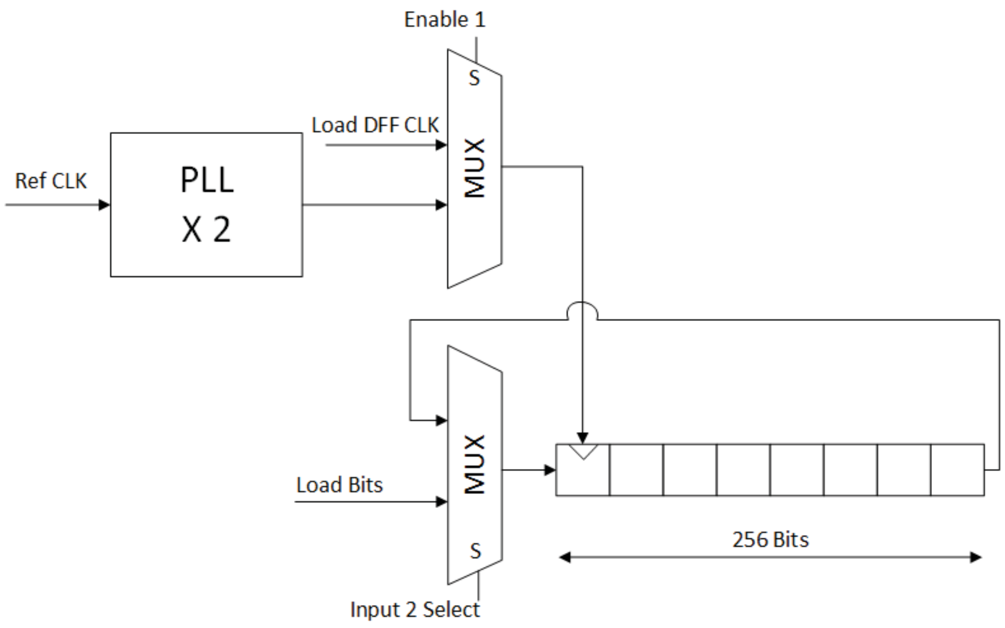


FIGURE 5.3: Multiplying PLL used to properly clock the circular shift register

Channels 4 and 5 are used to control the threshold voltage, v_{TH}, and the timing-strobe, t_{STRB}, of the PEU. Channel 4 uses a similar method as channel 3 to control the threshold voltage of the PEU's comparator. However, instead of encoding an AC signal, channel 4 encodes a DC signal where it is stored in a circular memory of 20 bits. A voltage-mode filter is used to extract the DC component of this signal which is then fed into the PEU's comparator. The threshold voltage set at this node will be used to control the sampling voltage in the PEU - that is, the sampling location in the eye diagram. Channel 5 is used to control the timing strobe of the PEU. The timing strobe is generated by encoding a DC code with a sigma-delta modulator and feeding it through a DTC. The time-encoded signal is then stored in a 64-bit circular memory after which the DC component of the time-signal is extracted using a time-mode filter. Through the use of Channels 4 and 5, the threshold-voltage and time-strobe are controlled and used to sample the eye-diagram as illustrated with the dashed lines in Figure 5.4. Channels 4 and 5 can be controlled independently to establish the desired sampling location of the eye diagram.

Each of the 5 channels is independently clocked from the others based on the encoding that has been done on the signal. The base clock frequency that the PTI uses is 114.5 MHz. Input channel 1 is clocked out at 57.25 MHz because the DUT runs at a clock rate of 28.625 MHz: one bit for a 1 and one for a 0 makes a "10" sequence have an output period of the desired 28.625 MHz. Input Channel 2 is a time-encoded signal where there are four bits that represent a full period at 28.625 MHz. The result is that the bitstream needs to be clocked at four times the desired output rate in order for the output to have

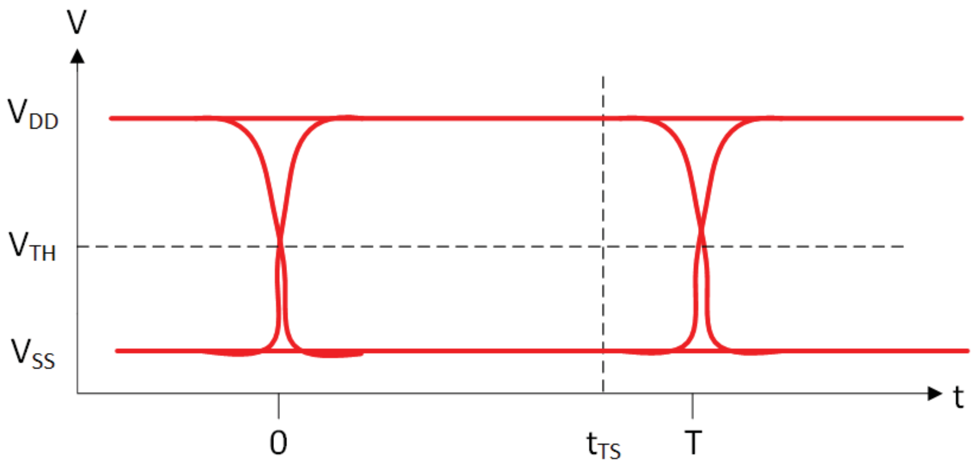


Figure 5.4: Threshold-voltage and timing-strobe used to control the sampling location in the eye diagram

the correct frequency, and then doubled again because it is being fed into a D-flip-flop where only the rising edges are used to sample. The result is that the circular bitstream for input channel 2 is clocked at 229 MHz. Input channel 3 is simply a sigma-delta encoded AC signal, so the circular bitstream is clocked at the same frequency as was used to sample - 114.5 MHz. Similarly, input channel 4 is clocked at the reference clock signal rate because it is also a sigma-delta encoded signal. Finally, input channel 5 is clocked by a PLL (see Section 5.7) and, since it is time-encoded with 4 bits per period with an output clock frequency of 28.625 MHz, its operating frequency is 114.5 MHz. The clocking frequencies are summarized in Table 5.1.

Table 5.1: Clocking Frequencies for the Circular Bitstreams for each Input Channel

Input Channel	Clocking Speed
1	57.25 MHz
2	229 MHz
3	114.5 MHz
4	114.5 MHz
5	114.5 MHz

Each circular shift register was designed out of serially-connected D-flip-flops. This design choice was made to ensure that the memory cells would be robust and properly latch and circulate the data that was being loaded into each of them. The disadvantage of the implementation of serially-connected D-flip-flops is that the memory cells are quite large and take up a proportionally large amount of area. For the implementation of this thesis, however, reliability was paramount so the large D flip-flop implementation was utilized.

5.3 Probability Extraction Unit

The probability extraction unit (PEU) is a hardware component of the design that detects the edge locations of the DUT that will, in turn, be used to calculate the bit error ratio of the DUT. The overall architecture of the PEU is illustrated in Figure 5.5.

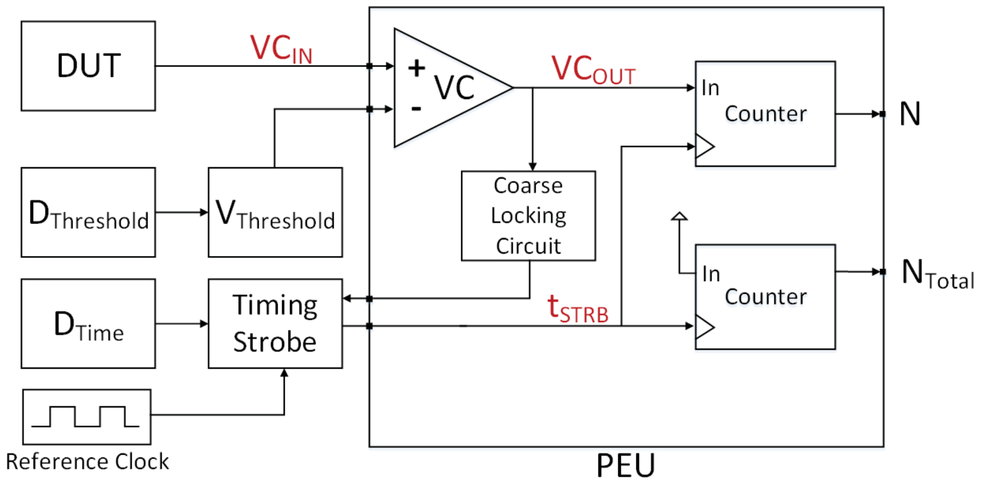


FIGURE 5.5: Architecture of the probability extraction unit

The PEU consists of a voltage comparator (VC), an automatic coarse-locking circuit, and two counters. The $V_{Threshold}$ and timing strobe blocks are simplified inputs coming from the BSG. The VC is used to determine when the incoming signal is greater than the set voltage threshold from input channel 4. The resulting decision signal coming out of the voltage comparator is then fed into two separate modules - one counter and the automatic course-locking timing circuit. The automatic coarse-locking timing circuit is used to lock onto the signal at the output of the voltage comparator so that when fine adjustments are made to the timing strobe, t_{STRB} from input channel 5, it is within its 90^o linear phase-range such that is it possible to adjust it around the VC_{OUT} edge. The design of the coarse-locking circuit is described in Section 5.7.

Input channel 5 controls the timing strobe location for the counters. By varying the phase of the strobe, it is possible to sample the eye-diagram from Figure 5.4 in a predictable and controlled manner.

Figure 5.6 illustrates the working principle of the decision circuit of the PEU. With an input sinusoidal signal into the VC and a constant threshold voltage, the VC_{OUT} signal represents when the VC_{IN} signal is above or below the threshold value, v_{TH}. When above, the value is a 1, and when below, it is a 0. The working principle is very similar to that of a very high-gain differential amplifier. The VC_{OUT} signal is held at the input to the counter and a sample is taken at the timing strobe's, t_{STRB}, rising edge. If

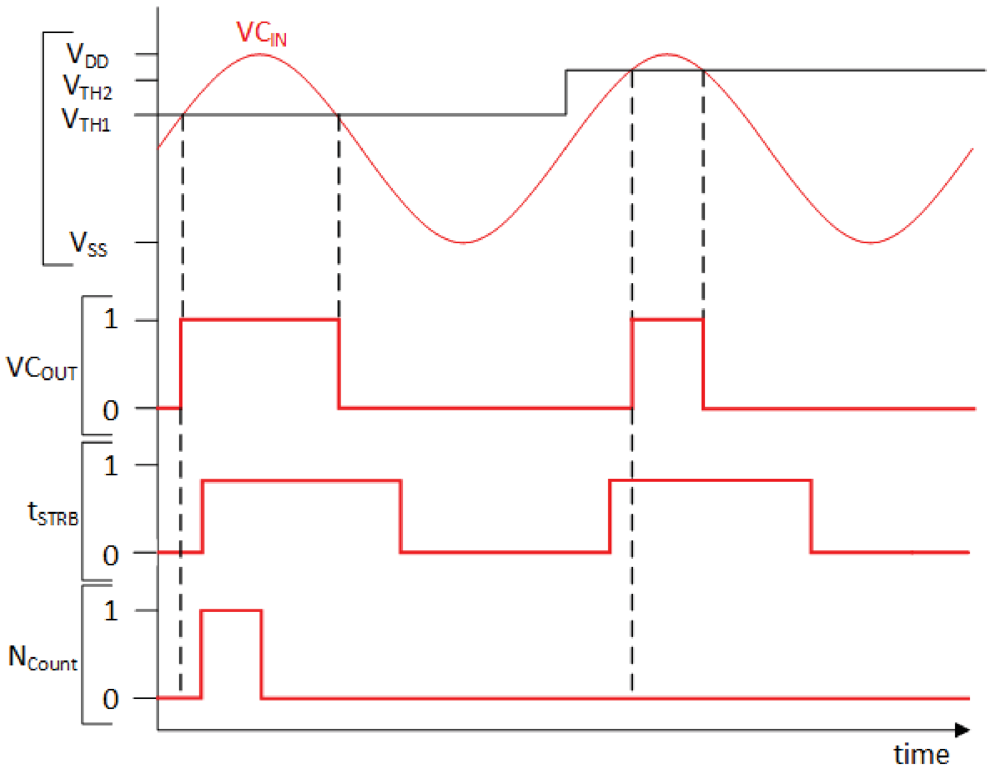


FIGURE 5.6: Timing diagram for the PEU for two different threshold values

the VC_{OUT} signal is a 1 when the sample is taken, the count will be incremented. If the VC_{OUT} signal is a 0 when the sample is taken, then the count will not be incremented. There are two counters in the PEU in Figure 5.5. The top counter is attached to the output of the comparator, and it counts the number of rising edge transitions from the DUT, VC_{OUT}, that are ahead of the timing strobe, t_{STRB} (denoted by the value N). The bottom counter counts the total number of timing strobe rising edge transitions (denoted as N_{Total}. Using the output values of these two counters over a range of timing strobe locations, it is possible to extract the CDF corresponding to the DUT and, as described in previously, use these results to compute the BER of the received signal.

In Figure 5.6, the first period of the VC_{IN} sinusoidal signal represents the situation where the count will be incremented because the VC_{OUT} signal is high when at rising edge of the t_{STRB} timing signal. Similarly, the second period of the sinusoidal input signal represents the situation where the count is not incremented because the VC_{OUT} signal rises only after the t_{STRB} signal. It is by using the final counts of the number of counted high signals versus the number of t_{STRB} edges, that the performance of the system will be measured. In both the amplitude- and time-based scan tests, the threshold voltage will be kept to be a constant voltage and the phase of the t_{STRB} signal will be adjusted around the rising edge of the VC_{OUT} signal. The adjustment of the phase of the t_{STRB} signal around the rising edge will allow for the extraction of the performance of the DUT.

Figure 5.7 illustrates the insight that will be gained from the PEU and the location of the edge locations coming out of the DUT. Figure 5.7(A) shows the relationship between then input DC code and the t_{STRB} phase relative to VC_{OUT}. Therefore, by varying the input DC code, the t_{STRB} phase is controllable over 90 degrees from -45 to 45. Using the PEU and extracting a measurement for each sampling location, the CDF and PDF of the DUT can be extracted as shown in Figure 5.5(B) and Figure 5.5(C), respectively. The CDF is obtained by simply plotting the fraction of times the VC_{OUT} signal is ahead of the timing strobe. Finally, the PDF in Figure 5.5(C) is obtained by taking the derivative of the CDF in Figure 5.5(B).

Using the extracted PFD from the PEU, it is possible to approximate the BER that is achievable with this device. As previously stated, the CDF is calculated by computing the fraction of the number of edges whereby the rising edge of VC_{OUT} is ahead of t_{STRB} as detected by the PEU. The experiment is repeated k-times across all the available DC codes and the resulting PDF is illustrated in Figure 5.7(C). Each normalized point can be calculated with the following formula

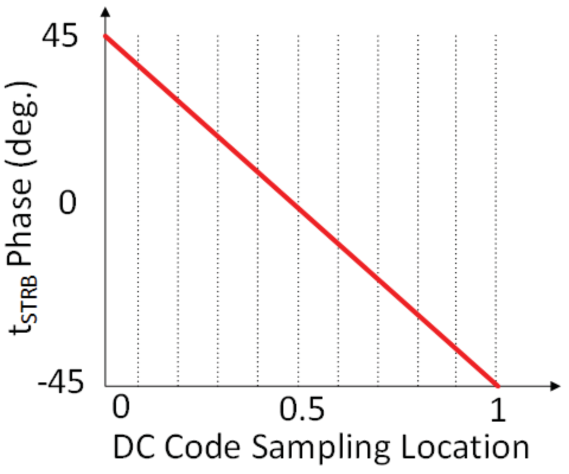
$$\mathrm{CDF}(k) = \frac{N(k)}{N_{Total}} \tag{5.1}$$

where k represents the k_{th} DC code in the range from 0 to 1. With the PDF, it is possible to calculate the mean and standard deviations that will be used to estimate the BER as described in Sections 1.3.1 and 1.3.2.

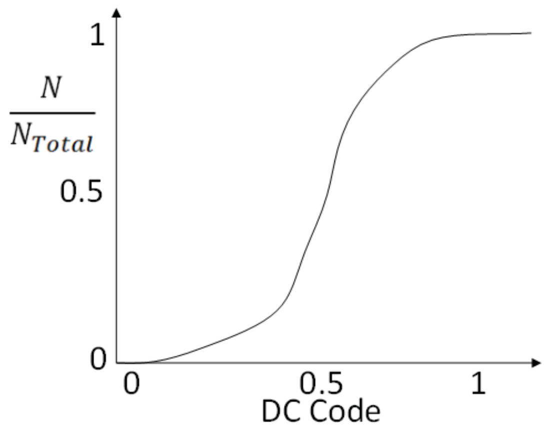
5.4 Voltage-mode Filter Design

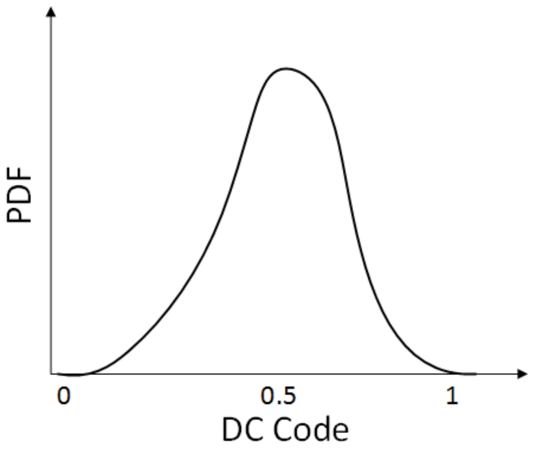
The voltage-mode filter design relates to the block in the input channel 3 path. The voltage-mode filter consists of four different cutoff frequencies that can be selected based on the encoded signal. The design of the voltage-mode filter block is illustrated in Figure 5.8.

The voltage-mode filter has one input and one output, but four different frequency-responses based on the values of the select signals, S1 and S2. The path from the input to the output is determined by the select signals to the multiplexers. The analog multiplexers were designed to have a minimum transmission resistance to reduce the loading effects that they have on the frequency-response. The transmission resistance of the multiplexer was measured to be 700 Ω. At the output of the final multiplexer



(A) Sampling locations of signal with varying DC code inputs to t_{STRB}, relative to VC_{OUT}





(B) Sample CDF of edge locations from the input signal to the PEU (C) Sample PDF of edge locations from the input signal to the PEU

FIGURE 5.7: Timing strobe sampling phase and the extracted CDF and PDF waveforms

is a single capacitor that will be part of any selected resistor combination. With each resistor being a different value, four different RC filters are realized and each one can be selected with the appropriate S1 and S2 values.

Table 5.2 illustrates the different RC filters that are implemented and Table 5.3 illustrates the appropriate signals to select them. The transmission resistance for the two multiplexers of 700 Ω each was taken into account when determining the values for the resistors. The 1400 Ω difference was subtracted from the resistors as can be seen in the differences in the second and third columns from Table 5.2. The voltage-mode filter bank was designed to have multiple resistors and a single capacitor due to the large layout size of a capacitor and the relatively small layout size of the resistors.

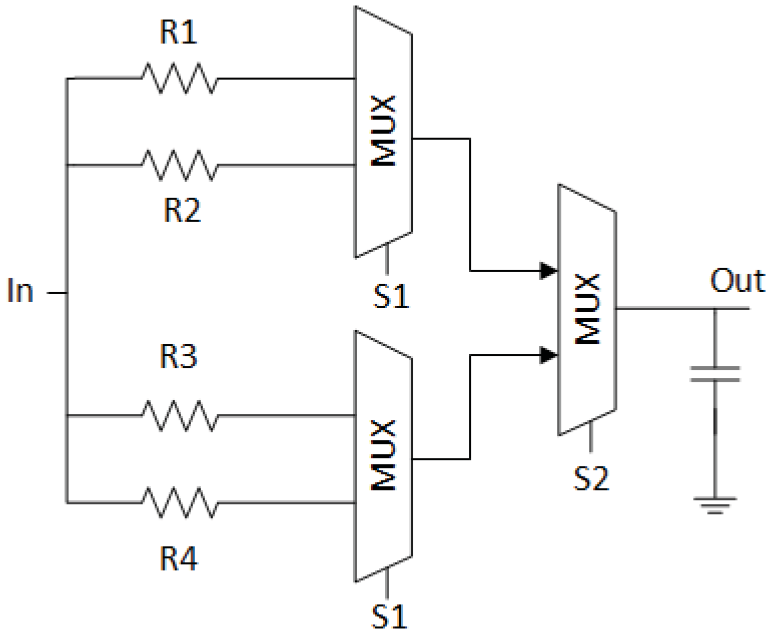


Figure 5.8: Design of the voltage-mode filter block for input channel 3

Table 5.2: Resistor Values for Voltage-Mode Filter Bank

Resistor	Resistor Value (Ω)	Resistance from Input to Output (Ω)	Capacitor Value	Cutoff Frequency (Hz)
R1	17.796k	19.196k	10pF	829.075k
R2	5k	6.402k	10pF	2.486M
R3	2.439k	3.839k	10pF	4.145M
R4	1.344k	2.744k	10pF	5.80M

Table 5.3: Select Signals for Corresponding Voltage-Mode Filter Bank Cutoff Frequency

S1	S2	Cutoff Frequency (Hz)
0	0	829.075k
1	0	2.486M
0	1	4.145M
1	1	5.80M

5.5 DTC into Software

The digital-to-time converter (DTC) illustrated in Figure 2.20 is quite complex and must be properly designed to ensure minimal phase noise, as well as minimizing power and layout area. In order to circumvent these issues, as well as reduce the number of components that need to be tested on-chip, the DTC in this dissertation has been moved off-chip and into software. Having the signals generated off-chip allows for a great deal of flexibility as to which signal is encoded and the desired phase separation between the high and low signals.

Implementing the DTC completely in software significantly decreases the hardware complexity of the design at the sacrifice of silicon area overhead. Having the DTC in software means that each bit and its associated timing information must be stored in memory instead of being generated on-chip. As covered in Section 2.5, the digital-to-time converter takes an input bit-sequence and translates it into a time-encoded output bit-sequence. The hardware complexity and optimization of all the components in order to minimize jitter is a very time-consuming task so the process was completely eliminated and moved into software. The output bits of the DTC that are modelled in software will be stored in a circular shift register as was described in Section 2.4. The time-encoded representation will be a greater length than the input bitstream due to the fact that the time information for the transitions must be stored as well. The software implementation is illustrated in Figure 5.9

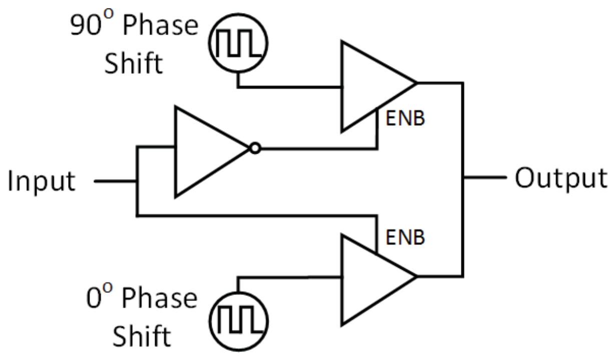


FIGURE 5.9: Single-bit DTC

An input 0 translates to a 90 degree phase-shift, and a 1 translates to a 0 degree phase-shift indicating

$$\phi(x) = \begin{cases} 90^o\colon x = 0 \\ 0^o\colon x = 1 \end{cases} \tag{5.2}$$

where $\phi(x)$ represents the output phase given the input, x.

The 90-degree phase-difference between the two signals means that that the output period is divided into four distinct parts to form a full 360 degree period. This can be represented by substituting the time-encoded signals by a four-bit representation. The 0 input is represented by a “0110” sequence corresponding to a 90^o phase shift, and the 1 input represented by a “1100” sequence corresponding to a 0^o phase shift. The output of the DTC is loaded into the circular shift register where it is right-shifted at 4 times the reference frequency such that the output corresponds to the appropriate time-encoded signal at the desired frequency. Figure 5.10 illustrates the overall design of the DTC and the separation between the software and hardware components.

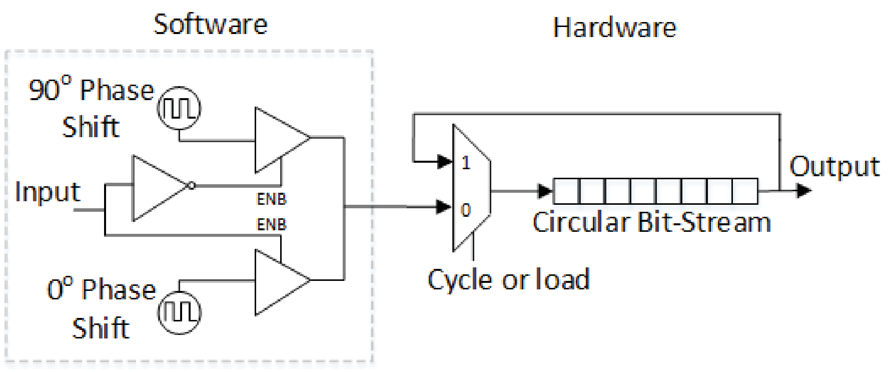


FIGURE 5.10: Separation of the software and hardware components of the DTC

The advantage of off-chip encoding is that the phase-encoding is completely programmable, and only the shifting frequency would need to be adjusted if a different phase-encoding were utilized. It is possible to encode, anywhere from 0 to 360 degrees in 90 degree increments with a 4 bit implementation, as well as smaller increments at the expense of a greater number of bits. This would not be possible to do with the traditional implementation. Additionally, the lack of on-chip hardware for the DTC removes possible sources of jitter and noise on the output signals.

5.6 Counters

The counters are an integral part of the PEU and are implemented to count the number of times the VC_{OUT} signal is ahead of t_{STRB}, as well as the total number of t_{STRB} pulses. An asynchronous counter is implemented in this thesis because it offers a simple, yet robust architecture. The counting line is initialized to all 1's and then the incoming VC_{OUT} rising edges will toggle the bits serially. The reason that the counter is designed this way is that if all the counters were initialized to zero, the first count would toggle every counter in the line and the output would have to be inverted. Instead, the counter initializes to high, and the output is taken at the inverted output of each divider illustrated in Figure 5.11.

Once the counter has finished counting, it is important for the circuit to have the ability to provide the counted number to the outside-world such that it is possible for the test engineer to read and decipher the results. As shown in Figure 2.25, the counter is a serial line of dividers. Once the counting stops, the corresponding counted number sits on the lines between adjacent dividers. To extract these numbers, a parallel-loading shift register was constructed using a series of D-flip-flops. The values at each node are shifted into the counter, and finally the counter rotates to the right to serially shift out the divider's bits. The architecture for the shifting register can be seen in Figure 5.11.

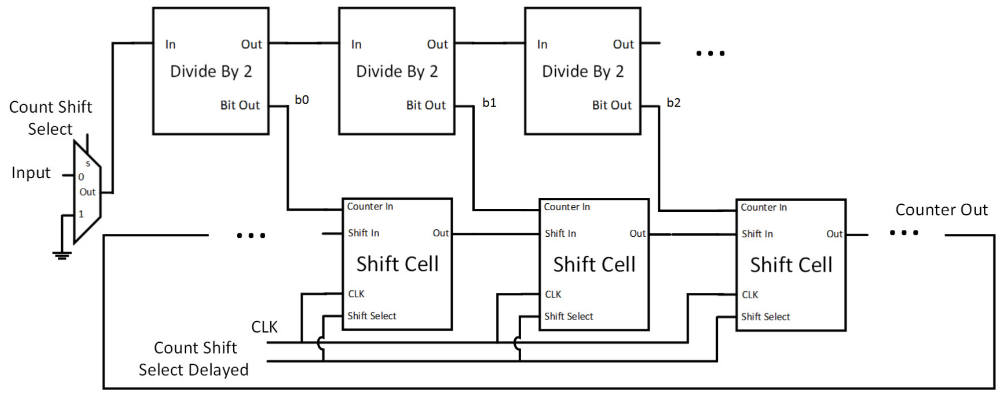


FIGURE 5.11: Parallel-loading shift register implemented to rotate right the counted bits

The timing of the counter and the shift register is of great importance to guarantee that the counting has stopped when the right-shifting occurs. If there is a timing error between these two actions, then it is possible that the same digit could be erroneously shifted out twice in adjacent bits and lead to an incorrect result. In order to guarantee that the timing of the circuits is correct, a small control circuit was developed. This circuit can be seen in Figure 5.12.

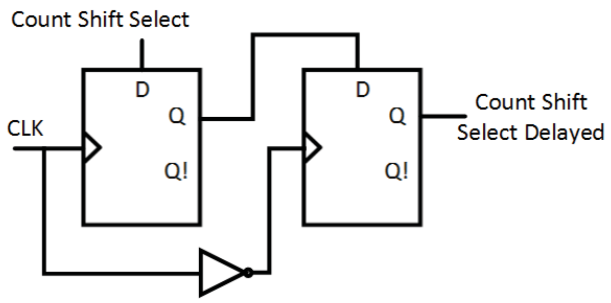


FIGURE 5.12: Controlling circuit to manage the timing between counting and shifting

The controlling circuit of Figure 5.12 is designed with two D-flip-flops, an inverter, and various multiplexers. The counting block has two states - counting and shifting. The counting is done asynchronously and the counter is incremented each time an incoming rising edge is detected. The shifting is done synchronously with the use of the clock signal. The transition from counting to shifting is done with the control of the "count-shift-select" control line. For a low voltage level, the counter is in the counting state, and a high voltage level will transition the counter into a shifting state. The transition between the two states is crucial such that no single bit is counted twice. The two D-flip-flop implementation creates a control signal for the shift register on the falling edge of the shifting clock signal. The control signal will transition the shift register from a loading state to a shifting state. The clock signal is asserted on the falling edge of the shifting clock such that an adequate amount of time is given for the transient signals to stabilize. Once the shift register has been transitioned into the shifting state, the shifting clock signal rotates the register to the right by one bit with every rising edge. The register is connected in a feedback loop such that the same bit-sequence will be continuously circulated to facilitate reading of the output value. In order to determine where the bit-sequence begins and ends, two shift registers are initially loaded with a high value such that it is easy to spot when the sequence repeats. A depiction of the timing of the counter's signals is shown in Figure 5.13.

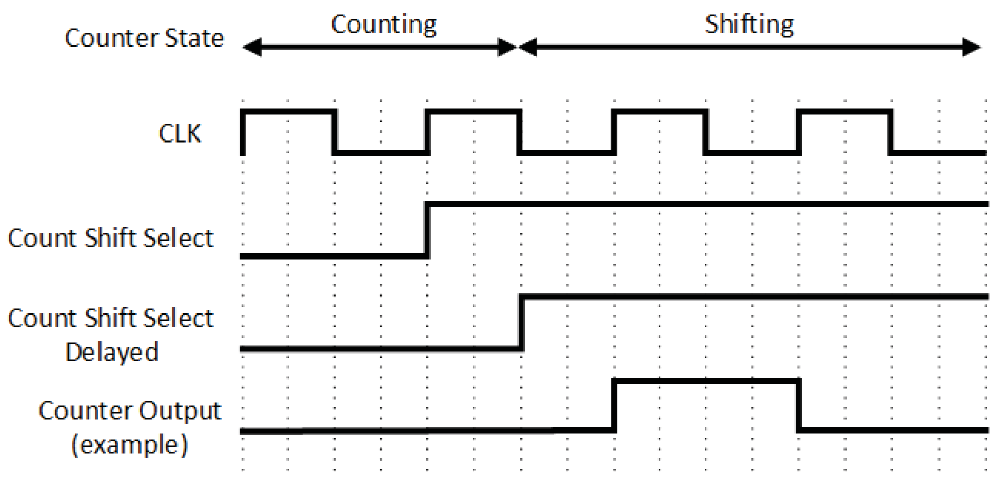


FIGURE 5.13: Waveforms of the controlling circuit to transition between counting and shifting

A counting issue will arise when two subsequent similar values are presented at the input to the counter. The reason is that with two subsequent 1s, there are no edges that will cause the D-flip-flops to toggle. To remedy this situation, a simple reset circuit was designed such that on the falling edge of the input clock the input into to the counter would be reset to zero. This small circuit was implemented between the voltage comparator and the counter block and is shown in Figure 5.14.

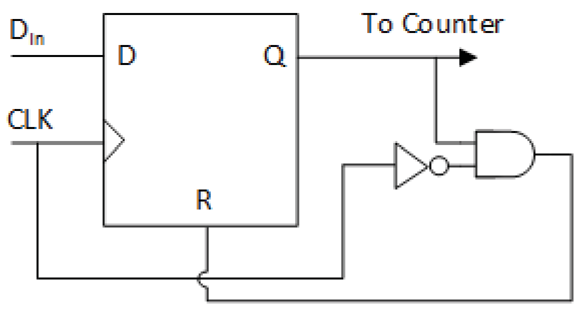


FIGURE 5.14: Auto-resetting circuit to ensure proper counting

5.7 Coarse-Locking and Fine-Locking

The PTI uses both a coarse- and fine-locking mechanism to correctly place the phase of the sampling edge, t_{STRB}, of the circuit in the desired location. The fine-locking mechanism only has a phase range of 90 degrees, so if the signal phase of the output of the voltage comparator is not within the range of operation, it will be impossible for the fine-locking mechanism to finely adjust the phase of the timing strobe to extract the performance of the DUT. To remedy this issue, a coarse-locking mechanism is also used where the timing strobe signal is automatically adjusted to be within the range of the fine-locking mechanism. The coarse-locking mechanism is made up of a DLL, PLL, D-flip-flop, and multiplexers. The architecture of this block is illustrated in Figure 5.15.

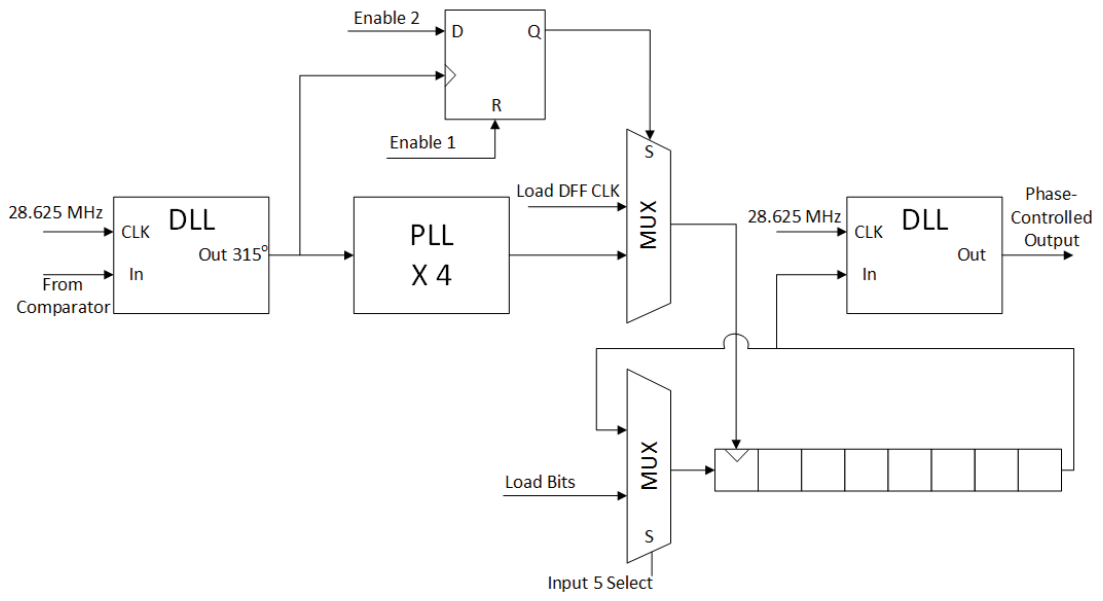


FIGURE 5.15: Architecture of the automatic coarse-locking circuit

The automatic coarse-locking mechanism begins with the output of the comparator feeding into a DLL that is clocked at 28.625 MHz. The reason that the clock speed is 28.625 MHz for this DLL is because that is the speed at which the DUT is working. The function of the first DLL in this circuit is to delay the signal 315 degrees such that over the operating range of the fine-locking circuit, there is a possibility to chose any locations from -45^o to 45^o with respect to the rising edge of the voltage comparator output to allow a large sampling range.

Following the DLL, the signal is fed into a PLL where the frequency is multiplied by 4. The multiplied signal will be used as the clocking signal for the fine-tuning circuit that uses a time-encoded sigma-delta DC code in a circular-bitstream and a time-mode filter as shown in Figure 5.1 for input channel 5. The clock that is being used is 4 times the speed of the DUT due to the fact that the sigma-delta encoding is also time-encoded using 4 bits per period such that a 90 degree phase range can be used, as covered in Section 5.5. Since the clocking signal is locked to -45^o (315^o from the DLL), a 0^o encoding in the fine-tuning circuit will correspond to an overall phase difference of -45^o. Similarly, if the fine-tuning circuit is loaded to have a 90^o phase shift, then the overall corresponding phase difference will be 45^o with respect to the voltage comparator output signal. This situation is illustrated in Figure 5.16.

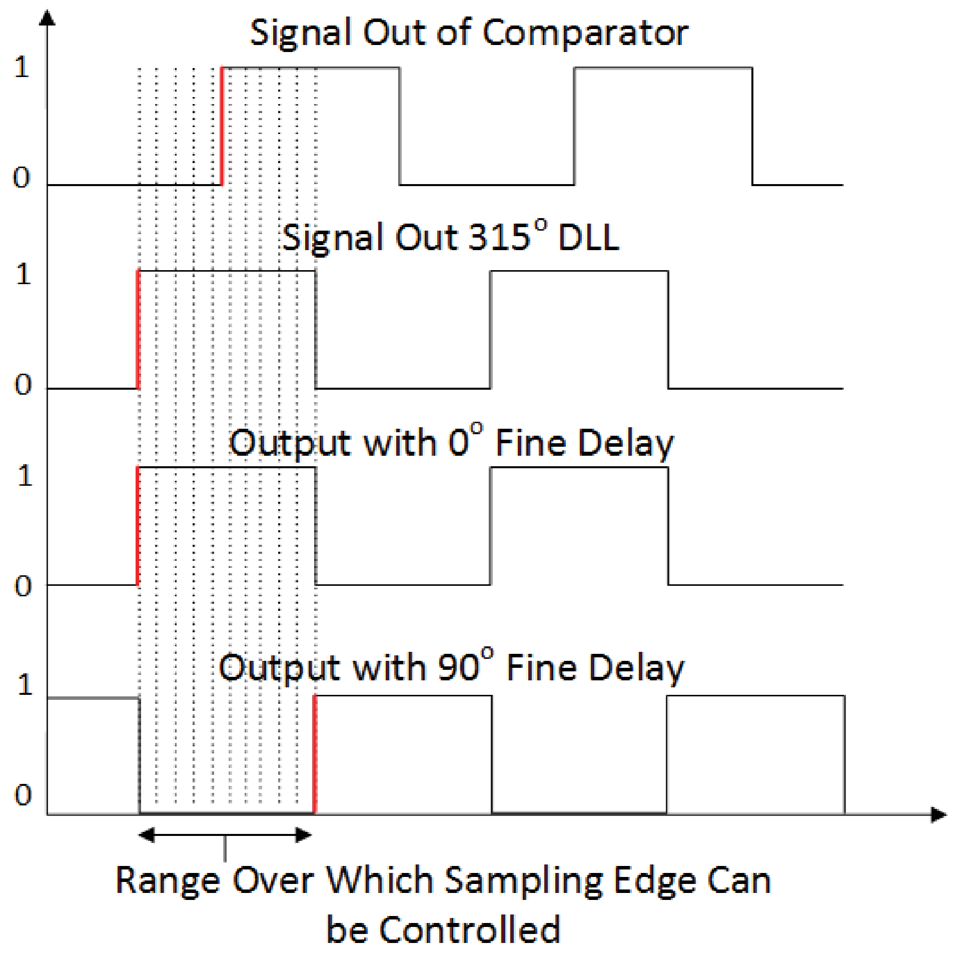

Signal Out of Comparator
1
0
Signal Out 315° DLL
1
0
Output with 0° Fine Delay
1
0
Output with 90° Fine Delay
1
0
Range Over Which Sampling Edge Can be Controlled

FIGURE 5.16: Waveforms of the automatic coarse-locking circuit demonstrating the phase range of operation

As mentioned in Section 5.2, the circular bitstream for input channel 5 can contain 64 bits. Due to the fact that each bit is time encoded with four bits, this means that the sigma-delta encoding can be a maximum of 16 bits long. 16 bits represents a total phase resolution of $\frac{1}{16} \times 90^o = 5.625^o$, or 545 ps. It is possible for the test engineer to place timing strobe signal edge accurately to a resolution of 5.625^o. Table 5.4 describes the corresponding input codes and the timing strobe's phase delay difference with respect to the DUT signal.

Table 5.4: Input Code and Corresponding Phase Shift with Respect to DUT Signal

Input Code	Phase Relation (degrees)
0000 0000 0000 0000	-45
0000 0000 0000 0001	-39.375
0000 0000 0000 0011	-33.75
0000 0000 0000 0111	-28.125
0000 0000 0000 1111	-22.5
0000 0000 0001 1111	-16.875
0000 0000 0011 1111	-11.25
0000 0000 0111 1111	-5.625
0000 0000 1111 1111	0
0000 0001 1111 1111	5.625
0000 0011 1111 1111	11.25
0000 0111 1111 1111	16.875
0000 1111 1111 1111	22.5
0001 1111 1111 1111	28.125
0011 1111 1111 1111	33.75
0111 1111 1111 1111	39.375
1111 1111 1111 1111	45

The D-Flip-Flop in Figure 5.15 is used to control the loading of the bits into the circular bitstream. The enable signals, “Enable 1” and “Enable 2” are used to control this process and are described in detail in Section 5.8. Additionally, Chapter 4 covers the fine clock control architecture and design.

Input channel 5 is a very important channel due to the fact that it contains a time-encoded signal and, since it is used for sampling purposes, it is very important that the phase of the signal with respect to other signals in the design is correct. For this reason a small start-up circuit is designed such that the circular bitstream only begins shifting the signal on the rising clock edge of the output of the DLL with the 315^o phase-shift. The reason for this is the encoding that is used in the DTC. A 0^o phase-shift corresponds with a “1100” bit sequence and a 90^o phase-shift corresponds with a “0110” bit sequence. It is important that the output signal from the circular bitstream and the DUT signal match up appropriately, else the time encoding will not be what is desired. For example, if the output of the circular bitstream is one-quarter period offset in phase, then the output bit sequences will resemble “0110” for a 0^o phase shift and “0011” for a 90^o phase-shift. This will result in the sampling edge not being centered around the DUT edge and the measurements results will not be correct. The start-up circuit is illustrated as the D-flip-flop in Figure 5.15. The “Enable 1” and “Enable 2” signals are used to correctly operate the start-up circuit and these are covered in Section 5.8.

5.8 Set-up Signals

In order to get the chip working as desired, there is a specific sequence of asserting control signals that needs to be followed. The first order of business when the chip is powered up is to load the circular memories with the desired bitstream that was generated in software. There are two options of loading the circular bitstreams with the bit sequences: manually, or automatically with the help of an ATE. The loading of the bit sequences are covered in Section 5.11.

The signals that must be controlled are “Enable 1”, “Enable 2”, “Load DFF CLK”, and the “Input X Select” signals where “X” represents the input channel number of the bitstream that is being loaded.

First, to load the circular shift register, the appropriate “Input X Select” signal must be set to 1. A 1 in this situation will allow the loading of new bits, while 0 will feed back the output of the shift register to the input such that is it connected in a circular manner. Next, “Enable 1” and “Enable 2” must be set to 0. Setting “Enable 1” to zero will allow the shifting of the bits into the circular shift register to be controlled manually, or from an external source as described in Section 5.11. When set to 1, the shifting clock is attached to the reference clock so the bits will be shifted at 114.5 MHz. By controlling the rate at which the bits are shifted, it makes the process of loading the shift register easier on the test engineer. Additionally “Enable 1” set to 0 serves to initialize all the DLLs and PLLs to ensure that they are in the correct start up state.

The bits that are loaded into the circular shift register are loaded through the “Load Bits X” input, where “X” represents the appropriate circular shift register input channel. For instance, to load bits in the circular shift register associated with input channel 5, the input to use would be “Load Bits 5”. Once the bits have been loaded into the appropriate circular shift register, the “Input X Select” signal is set to zero such that the circular shift register output is fed back to the input and that any change on the “Load Bits X” input signal does not have any effect on the loaded bits. The “Enable 1” signal should then be asserted to 1 to connect the shift registers to the correct clock so that they are rotated at the correct frequency.

Prior to any measurements, the “Reset Counter” input should be toggled to reset the counters, and the “Count Shift Select” signal should be set to 0 to select the count function. Setting it to 1 will enable the shifting function to shift out the count number after a test has been completed.

Following the loading of input channel 5’s circular bitstream and the toggling of “Enable 1” to 1, a sufficient amount of time should be allowed for the DLL and PLL circuits to settle to their constant values until "Enable 2" is toggled to 1. The reason is that this input contains a time-sensitive bit sequence, so it is important that the shifting of the bits corresponds with the rising edge of the DUT clock. If this is not done, then the fine-control's phase could possibly not line up with the DUT and voltage-comparator edge locations. See Section 5.7 for a more detailed explanation. Once "Enable 2" has been set to 1, it is possible to begin taking measurements. The measurements are started by first setting the "Count Shift Select" signal to 0, and then setting the "Reset Counter" to 1 (this is an active low signal). To stop the measurements, the "Count Shift Select" signal should be set to 1 so that the final count numbers are serially torated at the output so that the values can be recorded.

The counter contains 16 different counting blocks, so a number up to $2^{16} - 1$ can be counted. With a clock speed of 28.625 MHz, this corresponds to a maximum test time of $\frac{65535}{28.625 \times 10^6 s^{-1}} = 2.28ms$. If a longer time is taken, each overflow bit should be counted and added to the final count accordingly.

5.9 Design of the Comparator

The comparator is a very important and central part of the PEU design. The comparator compares the output voltage of the DUT against that of the threshold voltage (input $V_{Comparator}$) and outputs a 1 if the DUT voltage is above that of the threshold voltage, and a 0 otherwise. The output of the comparator is described as

$$V_o = \begin{cases} 1 : V_{DUT} > V_{TH} \\ 0 : V_{DUT} > V_{TH} \end{cases} \tag{5.3}$$

The comparator design specifications need to be able to satisfy both the required speed performance as well as the wide-range operating voltage range such that the threshold voltage can be manipulated over the entire available range from ground to V_{DD}. An operational amplifier could be implemented as a comparator, but in this dissertation it would not be appropriate due to the high-frequency operation of the design. Instead, a comparator design where sensitivity and propagation delay are optimized, is implemented. The block diagram of the voltage comparator is shown in Figure 5.17.

The pre-amplification stage is used to amplify the input signal to improve the comparator's sensitivity such that the minimum input difference on which the comparator can make a decision is increased [23]. Additionally, the pre-amplification stage isolates the input from switching noise from the decision circuit stage [23]. The decision circuit will make the decision whether the input is above or below the threshold voltage, and

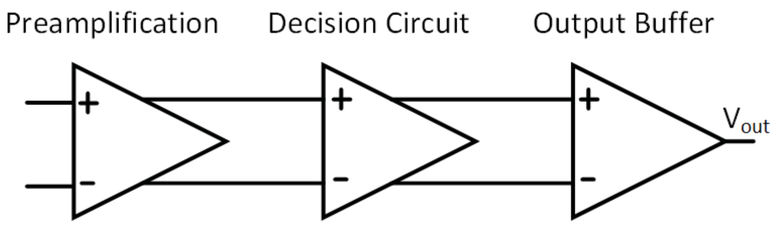


FIGURE 5.17: Block diagram for the voltage comparator

the output buffer ensures that the comparator can properly drive the following circuits and ensure that the decision circuit can work under the proper biasing conditions.

The design of the comparator stage is shown in Figure 5.18 [23]. This architecture in particular was chosen because it is a very good general-purpose comparator. It offers a very large rail-to-rail operation range for input signals, and it also has a sufficiently large bandwidth so that the the comparator can easily operate a the desired speeds in this PEU design. The sizing of each transistor in the design of the voltage comparator is summarized in Table 5.5.

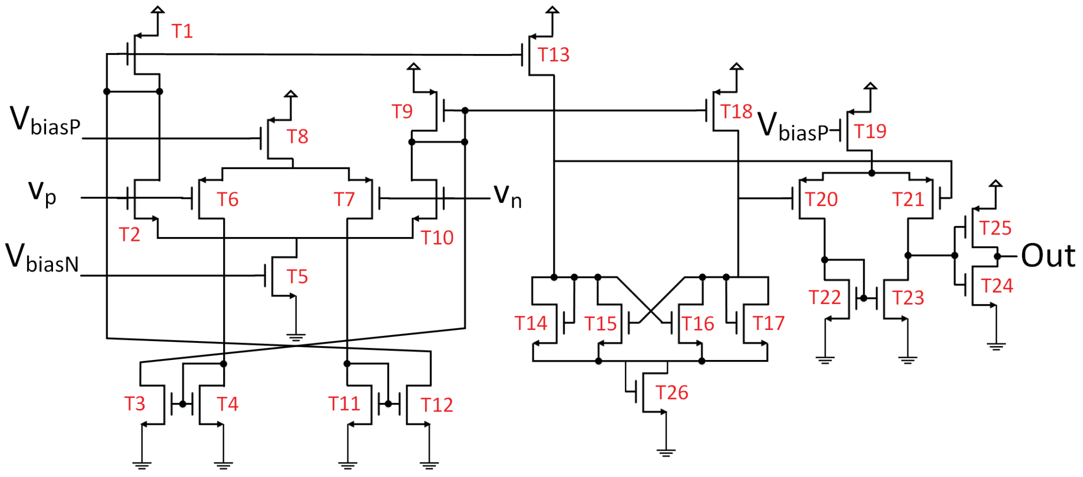


FIGURE 5.18: Transistor design of general-purpose comparator

The bias circuit for the nMOS and pMOS devices is shown in Figure 5.19, and the sizing for each of the transistors is listed in Table 5.6.

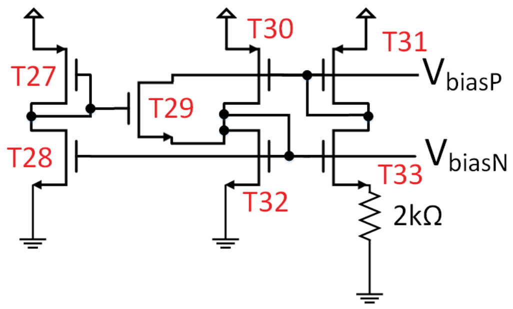


FIGURE 5.19: Bias circuit for the comparator

Table 5.5: Transistor sizing for the voltage comparator

Transistor	Width	Length
T1	6 um	300 nm
T2	3 um	300 nm
T3, T4, T11, T12	3 um	300 nm
T5	3 um	600 nm
T6, T7	6 um	300 nm
T8	6 um	600 nm
T9	6 um	300 nm
T10	3 um	300 nm
T13	6 um	300 nm
T14, T15, T16, T17	3 um	300 nm
T18, T19	6 um	300 nm
T20, T21, T25	6 um	300 nm
T22, T23, T24	3 um	300 nm
T26	3 um	3 um

Table 5.6: Transistor sizing for the voltage comparator bias circuit

Transistor	Width	Length
T27	3 um	6 um
T28	15 um	600 nm
T29	3 um	300 nm
T30, T31	30 um	600 nm
T32	15 um	600 nm
T33	60 um	600 nm

The voltage comparator was characterized and tested to ensure proper operation across all the desired voltage ranges. Figure 5.20 illustrates the voltage transfer curve of the comparator with a threshold voltage of 0.6 V. The input, V_{in} was then swept from VSS to VDD. The offset seen in the figure is a systematic offset that is negligibly small for the uses in this design. Taking the derivative of the voltage transfer curve allows for the calculation of the gain of the comparator which equals 5782 V/V, as illustrated in Figure 5.21.

In order to demonstrate the wide-range of common-mode operation, the voltage comparator's threshold voltage was varied from 0.1 V to 1.1 V in 100 mV increments, and the input voltage was swept from 0 to 1.2 V in each test. The results are illustrated in Figure 5.22 and demonstrates the wide voltage-range operation of the comparator. This allows the voltage threshold to be varied such that the eye-diagram can be sampled across a wide voltage-range.

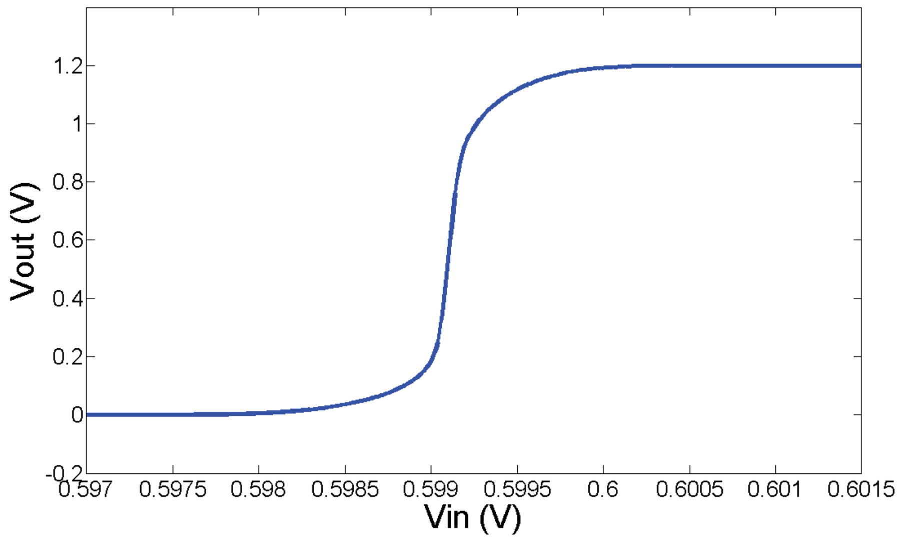


FIGURE 5.20: The voltage transfer curve for the voltage comparator with the threshold voltage held at 0.6 V and the input swept from 0 to 1.2 V

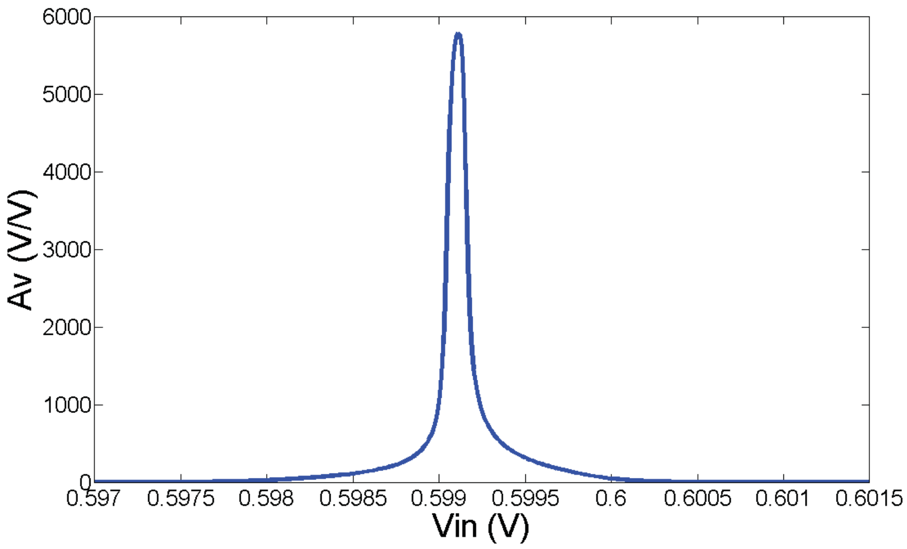


FIGURE 5.21: The gain of the voltage comparator

5.10 Practical Design Challenges and Considerations

In this section, a brief description of the practical design challenges and considerations will be covered. There are many different design considerations that must be taken into account in a design that will be used in the real-world. Topics that will be covered in this section include: inverter sizing, multiplexer design, and real-world considerations.

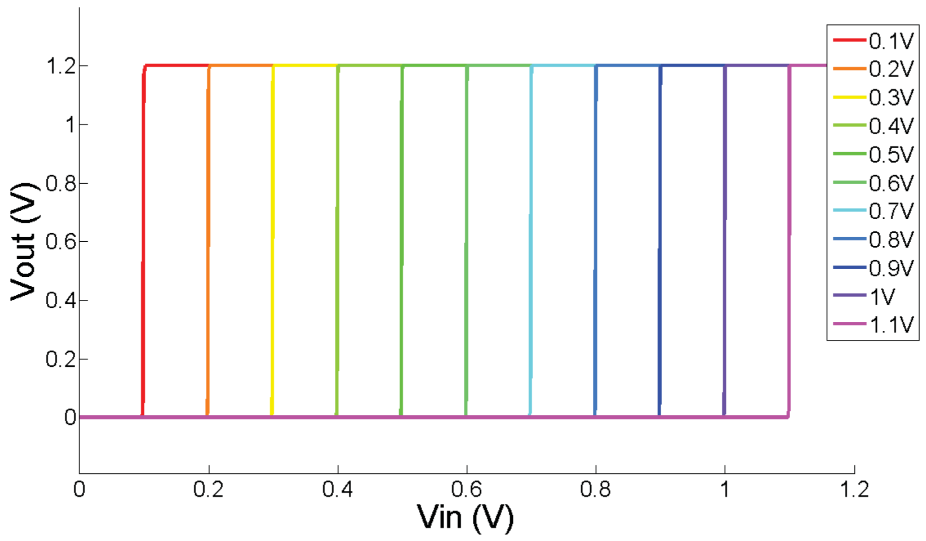


FIGURE 5.22: The operation of the comparator across a range of threshold voltages from 0.1 V to 1.1 V in 100 mV increments while sweeping each input from 0 V to 1.2 V.

5.10.1 Inverter Sizing

Sizing the inverter is a very important design step because the inverter is used in many different applications throughout the design of the PTI. The inverter is one of the most simplistic components, but it must be sized in a way that the threshold voltage is approximately in the middle of the voltage range between VDD and VSS, and that it additionally must have a large enough driving power to be able to drive a large amount of capacitance throughout the chip.

The inverter uses tow gates, a pMOS gate at the top, and an nMOS gate at the bottom. Following many simulations, the ideal sizing for the inverter was determined to have sizings of 300 nm and 1.27 μm for the pMOS and nMOS transistor widths, respectively.

5.10.2 Multiplexer Design

The design of the multiplexer is important because two different factors have to be balanced to achieve a desirable result. First, the transmission resistance must be minimized such that a signal attenuation is minimized, and second, the input capacitance must be minimized so that the signals being multiplexed are capable of driving the signal through the multiplexer. There is a trade-off with the resistance and capacitance. To reduce the resistance, the sizes of the transistors must be increased, but the increase of size also increases the input capacitance of the multiplexer. The final design of the multiplexer has a transmission resistance of 700 Ω and a capacitance that is easily driven with an inverter. Figure 5.23 illustrates the design of the multiplexer and Table 5.7 outlines the sizes of the transistors used in its construction.

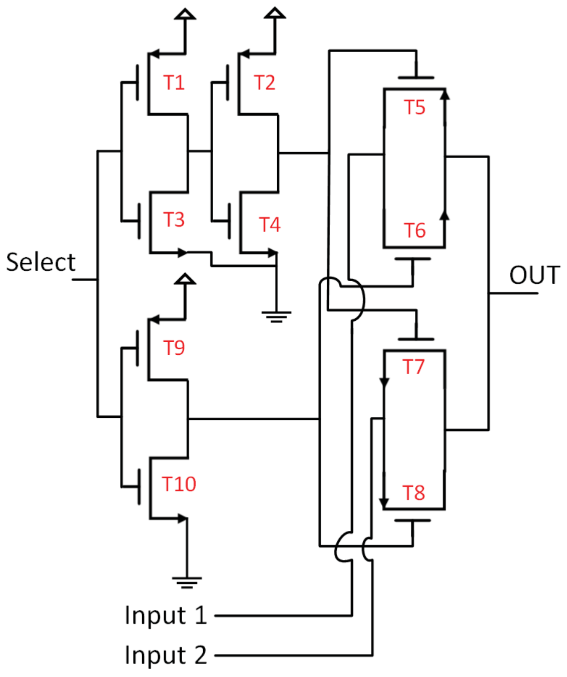


Figure 5.23: Design of the multiplexer.

Table 5.7: Transistor sizing for the multiplexer in Figure 5.23

Transistor	Width	Length
T1, T2	4.2 um	300 nm
T3. T4	1 um	300 nm
T5, T8	1.4 um	300 nm
T6, T7	6.3 um	300 nm
T9	8.4 um	300 nm
T10	2 um	300 nm

5.10.3 Real-World Considerations

In addition to the designing of the circuit so that the on-chip signals performed correctly, there were also many factors that had to be taken into account when the CMOS chip would be connected to the outside world, outside of the chip. These include the added capacitance of the pads and bond wires, electrostatic discharge, as well as loads into the measurement equipment.

Each pad that was used on the chip adds a large amount of capacitance to the design. Each pad is connected to a bond wire, and then to an external pin in the package. This is a large amount of extra capacitance, so it was necessary to ensure that the on-chip components had the capability to drive this capacitance at a relatively-high speed. To ensure that this was possible, each digital output was fitted with a digital buffer that was sized to have four inverter stages that increase in transistor size. The final inverter is 40 times the size of a standard inverter in order to be able to properly drive a large output load. Digital buffers were additionally added inside the circuit where a large amount of capacitance needed to be driven.

The output buffers are also included to drive the extra capacitance that is added by the presence of the electrostatic discharge protection (ESD) circuits. ESD is the sudden flow of electricity between two objects when they are contacted together. The sudden flow of electricity is usually the result of a build-up of static charge on an instrument or person and, when this comes into contact with a portion of the circuit, can damage the electronic components due to the exposure to a very high voltage and hence creation of a large current. The ESD protection circuits are placed at each pad contacting the outside world and are designed to be a set of two diodes that will discharge the damaging current to VSS. Figure 5.24 illustrated the way that the diodes are connected to provide ESD protection. One diode protects against damaging high voltages, and the other protects against damaging low voltages. Additionally, the ESD protection will protect against and voltage surges from the power supplies that are not properly filtered out by the shunt capacitors. Each ESD diode was designed to be very large ($47\mu m \times 68\mu m$) to be able to handle a large amount of current.

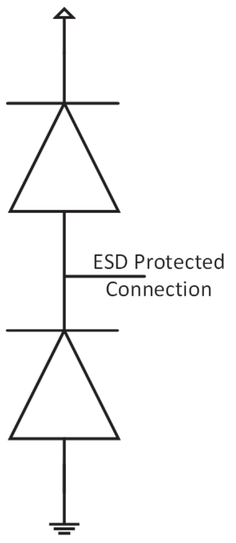


FIGURE 5.24: Design of the electrostatic discharge protection circuit.

Finally, it is important to take into account the signals that are driven off-chip that will be used as test-points for the external measurement equipment. The initial design called for an oscilloscope probe to be used to probe different locations of the circuit. An oscilloscope probe has a very large resistance, so the output stages were designed to be able to drive this resistance. As the PCB design was created, the measurement methods changed slightly to using SMA cables to attach to the oscilloscope to be able to gather information more easily. SMA cables are designed to have 50Ω of resistance, so the final design turned out to have an output signal swing of only 200 mV, instead of the planned full 1.2 V. If designing another chip, this would be important to take into account in the initial design phase.

5.11 Printed Circuit Board Design

The printed circuit board (PCB) was fabricated once the CMOS design was completed and packaging process was finalized. It is with the PCB that the design of the PTI is tested using a series of mechanical switches, SMA connectors, power supplies, and signal sources. The PCB board that was fabricated was designed to have 4 layers: a top layer, a power plane, a ground plane, and a bottom layer. The design of the PCB was done in such a way to allow the testing to be done with the greatest flexibility, but still have a very simple design. All select signals for the system are implemented using mechanical switches that simply select either 1.2 V or 0 V as the input voltage. This minimizes the number of voltage sources that are required to control the operation of the PTI. The switches that were used are single-pole double throw (SPDT). Single pole means that only one set of switches is being switched when the switch is moved, and double throw means that there are two different positions that the switch can be set to.

Loading a signal into the circular bitstream using the "Load Bits X" input can be done either manually using the switches, or automatically using an ATE. Having the option to load the registers two different ways was implemented in order to facilitate the testing process for those who have access to an ATE, but still allow the testing to be done by those who do not. Figure 5.25 illustrates the architecture of this design.

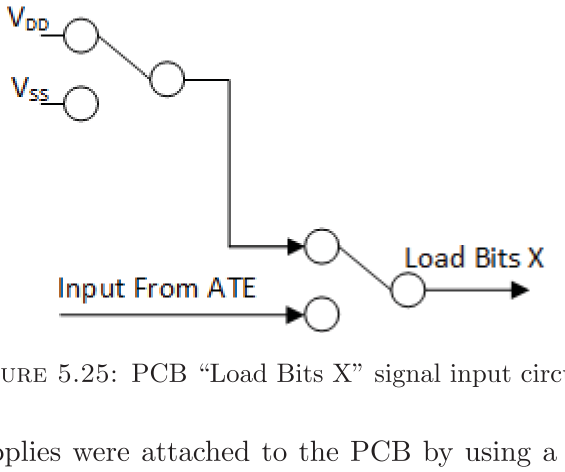


FIGURE 5.25: PCB "Load Bits X" signal input circuit

The DC power supplies were attached to the PCB by using a single wire and once on chip, capacitors were used to reduce the amount of high-frequency noise. Two 22

μF capacitors were placed in series from power to ground in parallel with two 1 μF capacitors. The values of the capacitors were used such that they could cover a wide range of frequencies. Additionally, during operation it is seen that, due to the way the capacitors are constructed, large capacitors have a large parasitic inductance at higher frequencies which increases the impedance of the capacitor to ground. The smaller 1 μF capacitors are placed closer to the chip so that the resonance is filtered out as well.

The input to the "DFF Clock" input is a very important input due to the fact that it is used as a clocking signal. The clocking signal needs to be very clean and without any false edges such that the circular bitstreams can be loaded properly without any false-shifting. A known problem with switches, however, is switch bouncing where false glitches can be seen in the output waveform as the new connection is made and contacts physically move from one position to another. As the the switch settles to its new position, the contact tends to mechanically bounce which leads to the circuit being opened and closed many times and creating false edges that would be interpreted as false clock signals.

The bouncing issue issue was solved with the implementation of the clock-debouncing circuit illustrated in Figure 5.26. The debouncing circuit has two different parts, the low-pass RC filter, and the Schmitt Trigger. The low-pass RC circuit was designed to have a time constant of 20 ms in order to smooth out the bouncing signals using a resistor of 20 $k\Omega$ and a capacitor of 1 μF. Finally, a Schmitt Trigger is implemented which implements hysteresis to shift the switching threshold levels to reduce the likelihood of false edges. The overall design of the PCB is illustrated in Figure 5.27

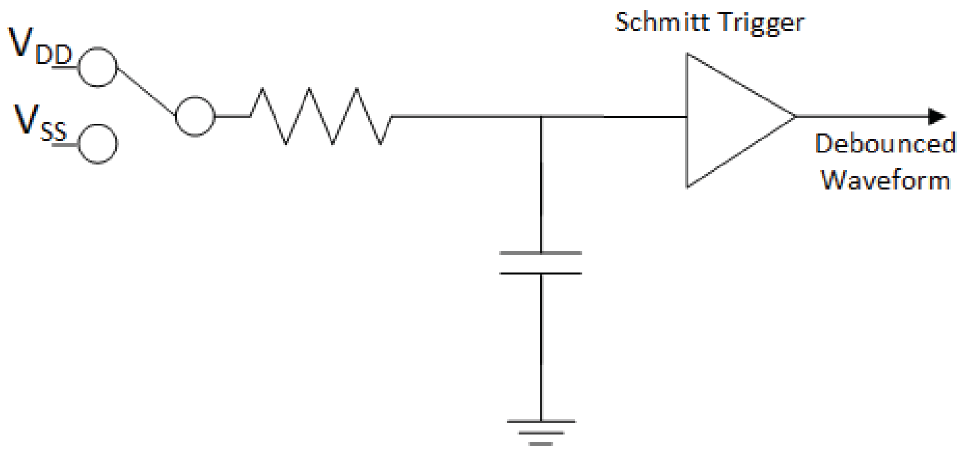


FIGURE 5.26: Clock debouncing circuit design

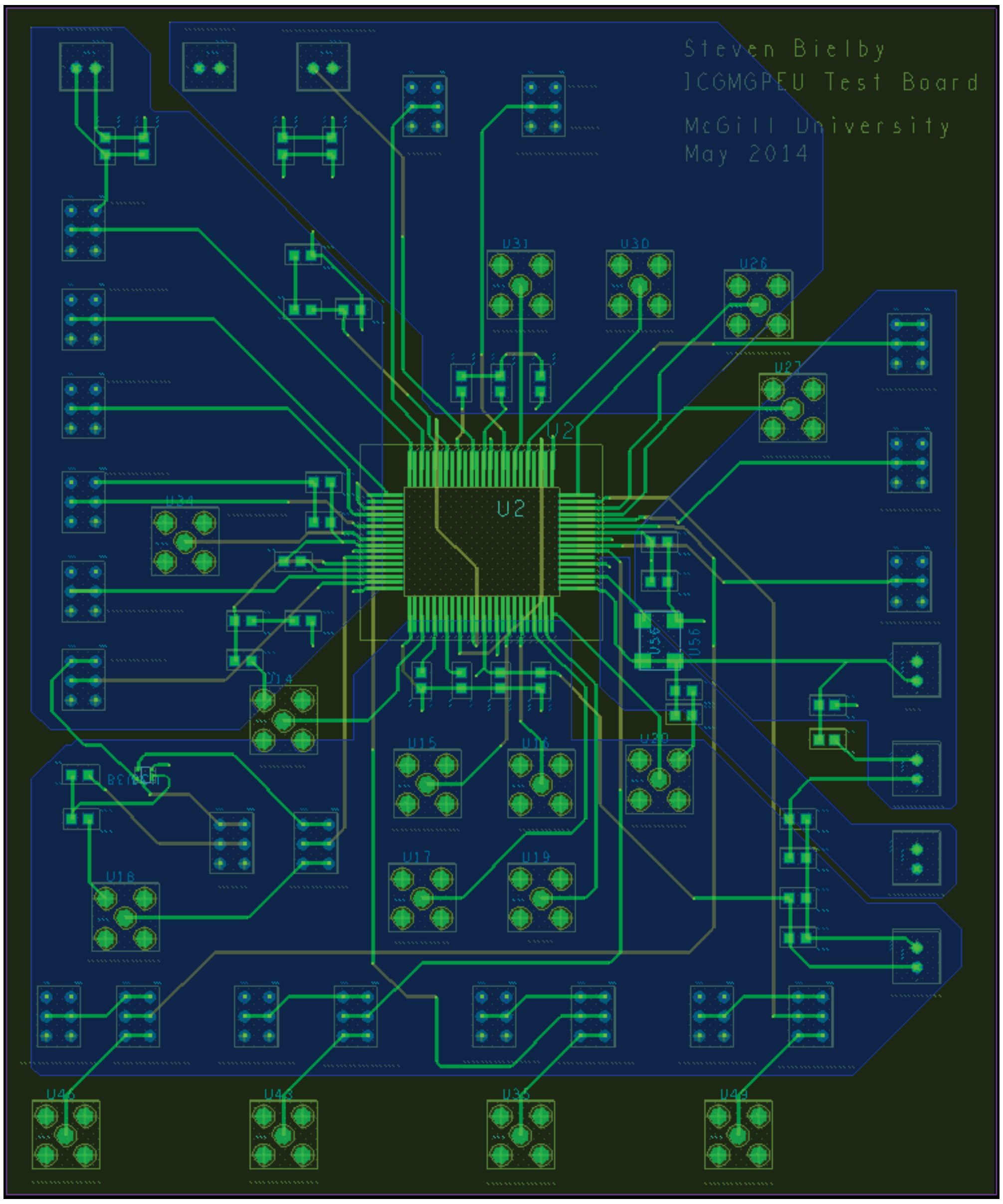


FIGURE 5.27: Design of the custom PCB to test the PTI

5.12 Simulation Results

Each of the individual components were tested to ensure proper operation, and were subsequently combined into a larger design once each component performed as designed. It is essential to ensure that each component is working as designed to guarantee a problem-free design. Additionally, testing the components as they are designed reduces the likelihood that an error will be found in the larger design where it will be significantly more difficult to find the issue. It should be noted that some of the tests were done with a wider bandwidth than what is being used in the PCB design in order to reduce the overall simulation time of each component. Without increasing the bandwidth, some simulations would take on the order of months to complete. The bandwidth change will not have a significant impact on the basic working principle of the component.

5.12.1 Delay-Locked Loop

The delay-locked loop was tested to ensure that it would properly lock to the input signal. The control-voltage-to-delay transfer characteristic of the VCDL was determined by building the delay line and measuring the differing delays that were associated with each voltage level. The results of these measurements can be seen in Table 5.8 and Figure 5.28. When designing the VCDL, the operating frequency played a large part in the design. The VCDL was designed to be capable of delaying the incoming reference signal anywhere from 0 to 1.4 periods. This gives the DLL more than a full period of operation, and it allows the output signal to have a full period of delay from the incoming reference signal. The region around the full period of delay is designed to be as linear as possible as this is where the DLL's operating region will be.

Table 5.8: Voltage-Control Delay Line Transfer Characteristic for a 28.57 MHz Clock

Control Voltage (V)	Delay (ns)	Fraction of Period
0	2.2	0.06
0.1	2.3	0.07
0.2	2.6	0.07
0.3	3.3	0.09
0.4	4.9	0.14
0.5	7.8	0.22
0.6	12.7	0.36
0.7	19.3	0.55
0.8	25.7	0.73
0.9	31.7	0.91
1	37.7	1.08
1.1	43.7	1.25
1.2	49.3	1.41

Tests on the full DLL were conducted with a loop filter with a value of 33 nF which translates to a DLL bandwidth of approximately 8.3 kHz. This small bandwidth is excellent for filtering out the high-frequency components of the input bitstream. Figure 5.29 illustrates the settling of the DLL. It should be noted that the initial voltage for the DLL starts around 500 mV, and not at zero. This was done simply to reduce the total simulation time that is required. It is known that that the settling voltage is much above the starting voltage point, so there is no detrimental effect to have a different starting point closer to the final settling voltage. Figure 5.30 shows the locking of the input and output waveforms which shows that the DLL is functioning as designed.

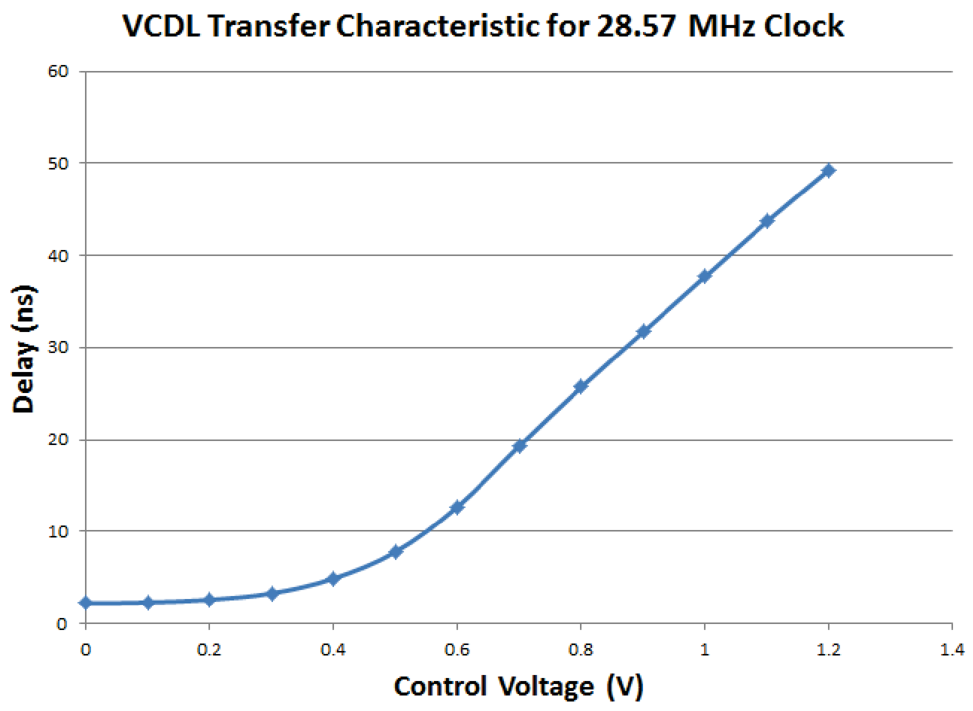


FIGURE 5.28: Voltage-controlled delay line transfer characteristic for a 28.57 MHz clock

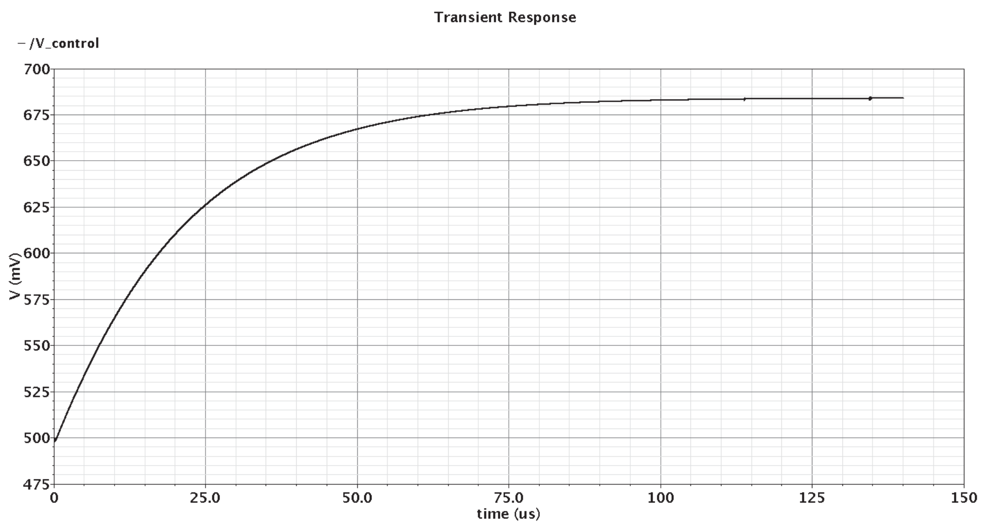


FIGURE 5.29: Loop filter voltage during the settling of a DLL

Traditionally, the best way to analyze the settling characteristic is to plot the output phase against time and monitor the settling of this phase to a constant value. As can be seen in Figure 5.29, this was not done. Instead, the voltage from the loop filter was plotted against time. The reason for this is that, as is shown in Figure 5.28, the phase-voltage relationship is linear in the region that the DLL is operating in. Therefore, it is reasonable to use the voltage characteristic to measure the performance of the DLL in the same way that the phase is traditionally used. Figure 5.29 uses a simple clock of

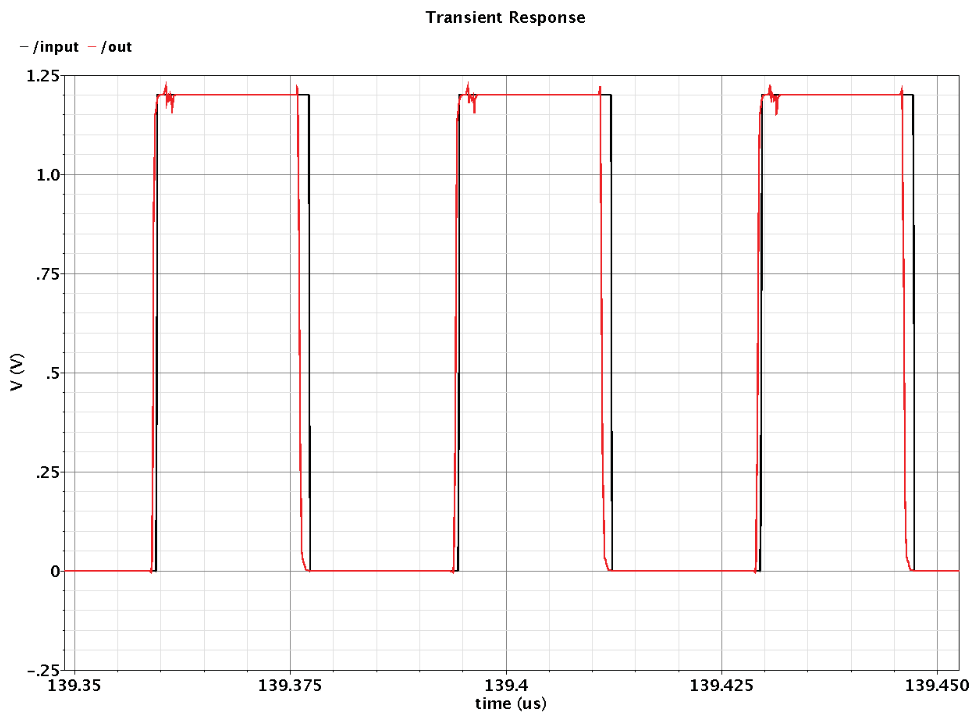


FIGURE 5.30: Locking of the input and output signals of the DLL

the same frequency as the operating frequency of the DLL (28.625 MHz) as the input. This input will test whether or not the DLL can lock to a signal without any issues. As seen in the figure, the loop filter voltage settles and the output and input phases are locked as expected. The test was repeated once again to test with a sigma-delta input as will be done with the chip. Figures 5.31 and 5.32 shows the same results, but this time using a time-encoded sigma-delta input rather than a constant clock signal. The two different sigma-delta inputs represent the maximum and minimum phase that can be controlled with the DC code inputs.

5.12.2 Counter

The counter was tested to ensure that the count outputs correctly correspond to the number of output edges that are detected in the PTI. The test for this component was done by simply introducing a number of edges and making sure that the output corresponds to the number of edges that were introduced at the input. The number is serially shifted out of the counter as described in Section 2.9.

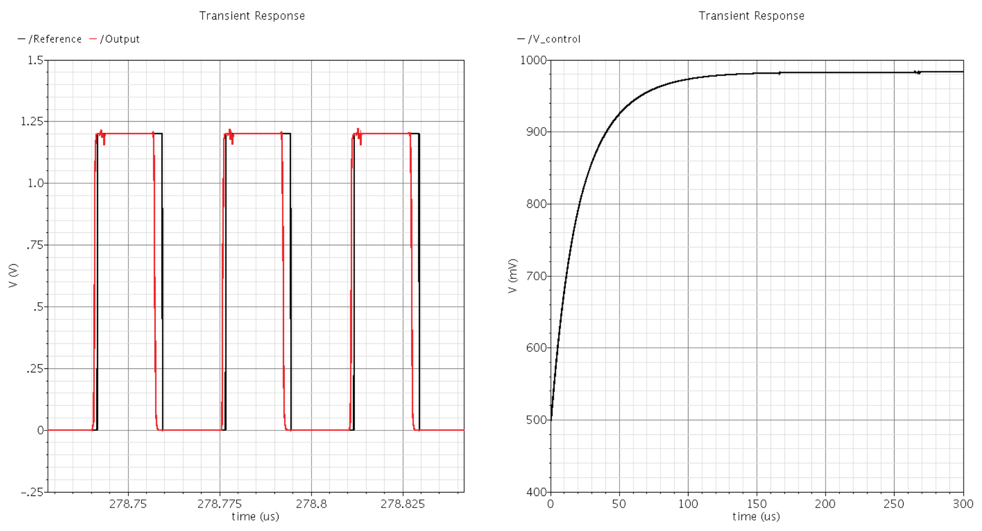


FIGURE 5.31: The loop filter voltage and waveforms during the settling of the DLL with a 0^o phase shift

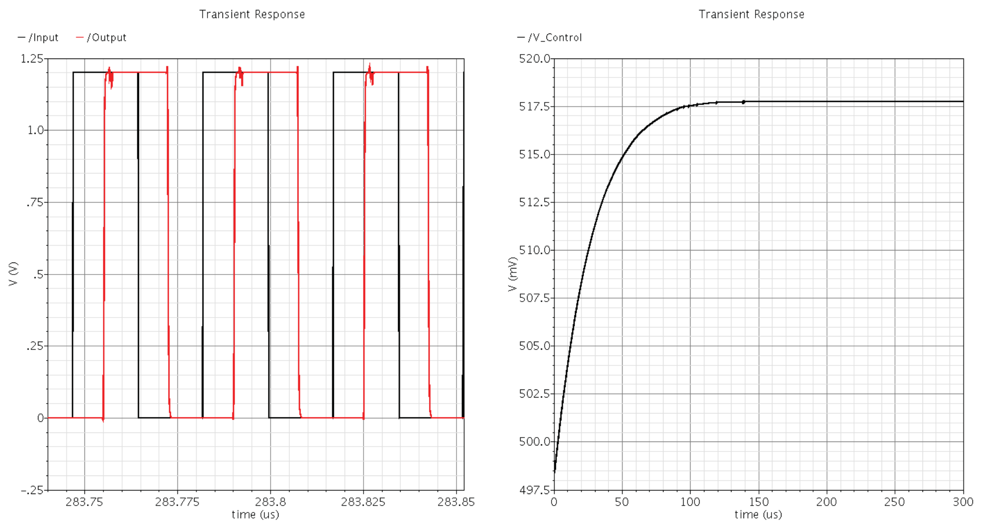


FIGURE 5.32: The loop filter voltage and waveforms during the settling of the DLL with a 90^o phase shift

5.12.3 Phase-Locked Loop

The PLL was tested in a similar way to the DLL. As described previously in this chapter, the PLLs in this design are used solely for multiplying frequencies to meet timing requirements. The input to the PLL is a clock signal, and the output is the multiplied signal. The multiplication factor is directly dependent on the feedback divider as described in Section 2.10. Figure 5.33 illustrates the settling of the loop filter voltage, as well as the output waveforms and their corresponding frequencies for the multiplying by 2 PLL. The argument for using the filter voltage instead of the phase of the signal in the settling characteristic is similar to that of the DLL: the VCO range over which the

PLL is operating is linear so there is a direct linear correspondence between the loop filter voltage and the output frequency.

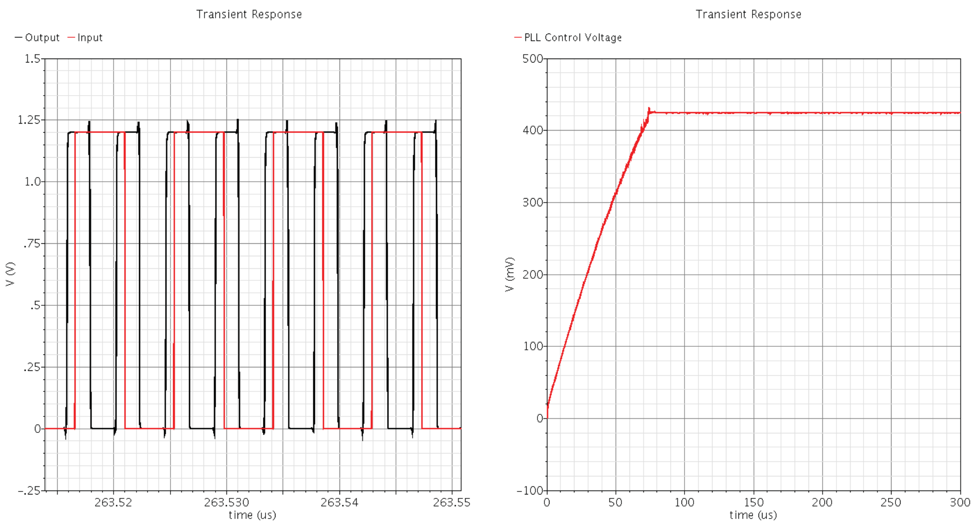


FIGURE 5.33: The waveforms for the outputs of the multiplying by 2 PLL

From Figure 5.33, it is possible to see that the output frequency is twice that of the input frequency as desired. Additionally, the control voltage of the PLL is shown to reach a steady-state value as anticipated. The settling of the control voltage's final value is exactly what is expected when comparing to Figure 5.34 where the relationship between the PLL output frequency and VCO control voltage is shown.

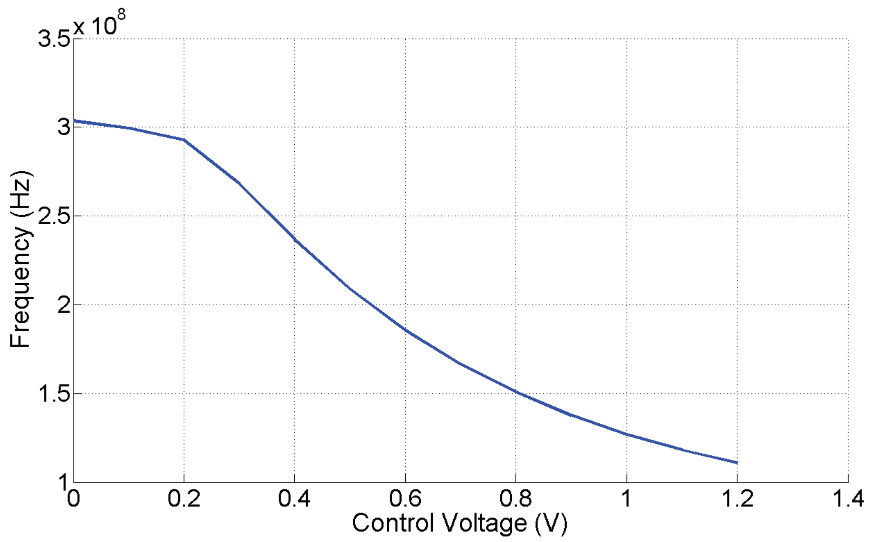


FIGURE 5.34: VCO voltage-frequency relationship used in the PLL

5.12.4 Circular Shift Register

The circular shift register is one of the most important pieces in the design of the PTI, so it is of the utmost importance that it works properly. To test the circular shift register, a sample bitstream was loaded into the registers, it is connected circularly in feedback, and the output bitstream is compared to the loaded bitstream to make sure that they match. Additionally, it is made sure that the bitstream repeats as it is designed to do. Figure 5.35 illustrates the waveforms of the circular shift register. The red line in the input bit sequence and the black line is the sequence that is shifted out of the circular register. It can be seen that both waveforms are the same which concludes that the circular shift register works as intended.

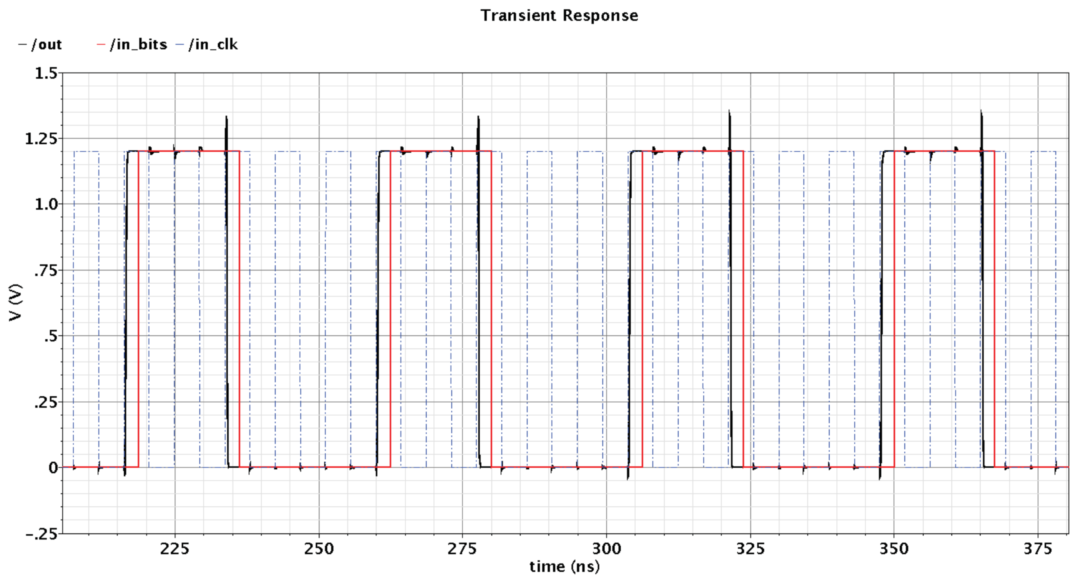


FIGURE 5.35: The waveforms of the circular bitstream after a bitstream of “11000” is loaded and connected circularly

5.12.5 Coarse and Fine Locking

The automatic locking circuit was described in detail in Section 5.7 and is used to synchronize the output bitstream from the circular shift register with the output of the comparator so that it is possible to extract very fine edge-transition information. Without this circuit, the fine adjustment of the edge would not necessarily be done around the rising edge of the output comparator waveform as desired. To ensure that the circuit was working as it was designed, test simulations were done with input waveforms of arbitrary phase to guarantee that the circuit worked under all input conditions. Figure 5.36 shows the output waveform from the circular shift register that has been clocked by the multiplying PLL such that the output waveform is locked to the correct phase so that measurements can be taken. The waveform that is loaded into the circular shift register is one that alternates a “1010” time-encoded pattern such that the output is located at a phase-shifted 0^o. This is achieved because it is alternating back and forth between a -45^o and a 45^o phase-shift, and the lowpass time-mode filter extracts the average value of 0^o. The black curve shows the time-encoded output from the circular shift register, and the red curve shows the locked filtered output from the DLL which will be used as the timing input to the counters.

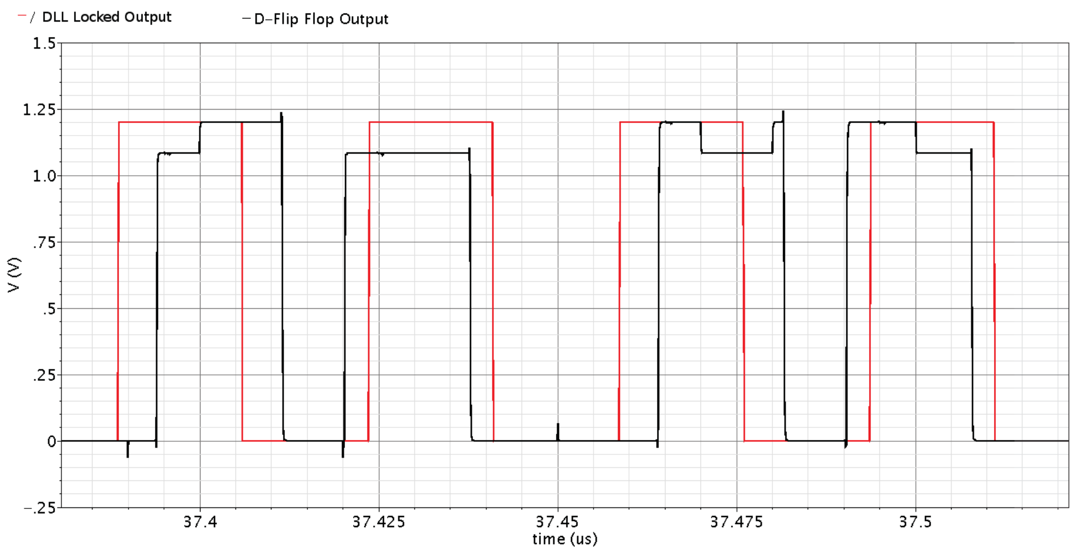


FIGURE 5.36: The waveforms of the circular bitstream after coarse-locking and a 45^o phase-shift

There are 2 DLLs and 1 PLL in the circuit as outlined in Section 5.7. Figure 5.15 outlines the general architecture for the coarse-locking circuit and Figure 5.37 shows the different control voltages for the controlling DLLs and PLLs in the automatic coarse-locking circuit. The “DLL 1 Voltage” is in reference to the first DLL which takes the input from the comparator to delay the signal 315^o so that is is possible to accurately control the edge anywhere from -45^o to 45^o. The voltage can be seen to settle to a constant value. It was started with an initial condition to reduce the simulation times. As mentioned in Section 4.2, each DLL contains a start-up circuit to guarantee proper operation of the DLL. The second voltage, “PLL Voltage”, is the control voltage for the multiplying PLL that will multiply the DLL output by 4 to properly clock the circular bitstream so that the output is of the correct frequency. Finally, the third voltage in the figure shows the voltage for the second DLL which acts as the time-mode filter for the circular bitstream. Here, the signal is only enabled once the PLL has properly settled to a constant value to ensure that the clocking happens at a constant rate. Also, it should be noted that the additional logic covered in Section 5.7 ensures that the clocking of the circular bitstream begins on the correct pulse so that the rising edge is located at the correct phase-location. The two Figures 5.36 and 5.37 show the correctness of the design of the coarse-locking circuit.

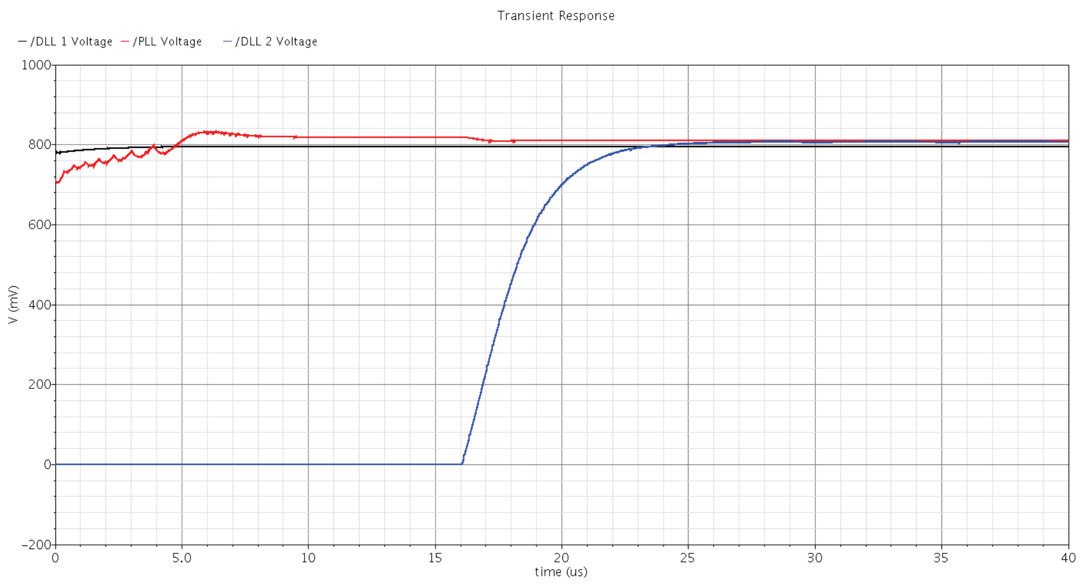


FIGURE 5.37: The control voltages in the coarse-locking circuit illustrating proper operation

5.12.6 Summary

This chapter covered the architecture of the PTI that is used to extract the edge-location timing information for a high-speed circuit. The extracted edge locations will be used to construct a PDF that will be used to extract the bit-error-rate (BER) associated with the communication channel and receiver. This section covered the system architecture, practical design considerations, PCB design, as well as the simulation results for the components using MATLAB and Virtuoso.

Chapter 6

Experimental Results

The chapter presents the results from the measurements of the PTI once it had been returned from fabrication. The PTI CMOS was fabricated in a IBM 0.13 μm process with a die size of 1.2 mm X 1.2 mm. The die was packaged in an CQFP80A package from Mosis that contains 80 pins. The package was mounted on a specifically-designed printed circuit board (PCB) with input and output channels that allow for stimulation and measurement of the PTI. This chapter will investigate the final design of the PTI, as well as its operational characteristics. The results will be compared with the simulated measurements and final quality of design will be discussed. Finally, future design improvements will be suggested for the PTI.

6.1 Experimental Setup

The experimental setup for the testing of the design is shown in Figure 6.1. The experimental setup consists of three major components: the HP 81130A programmable pattern generator with two channel outputs, the PCB board mounted with the PTI and the associated discrete components for testing, and a signal measurement instrument which in this case was the Agilent 80000B Digital Signal Analyser.

The HP 81130A is a programmable pattern generator that was used to load all the circular shift registers with the correct bit sequences. Using its two separate channels, it is possible to easily load the channels by providing both the input bitstream as well as the input D-flip flop loading clock signal "Load DFF CLK" (see Appendix A). Using the control switches mounted as discrete components on the PCB, it is possible to control the "Load X Select" input signals to select which circular shift registers are in the load configuration, and which ones are in the circular-shifting configuration. Each circular

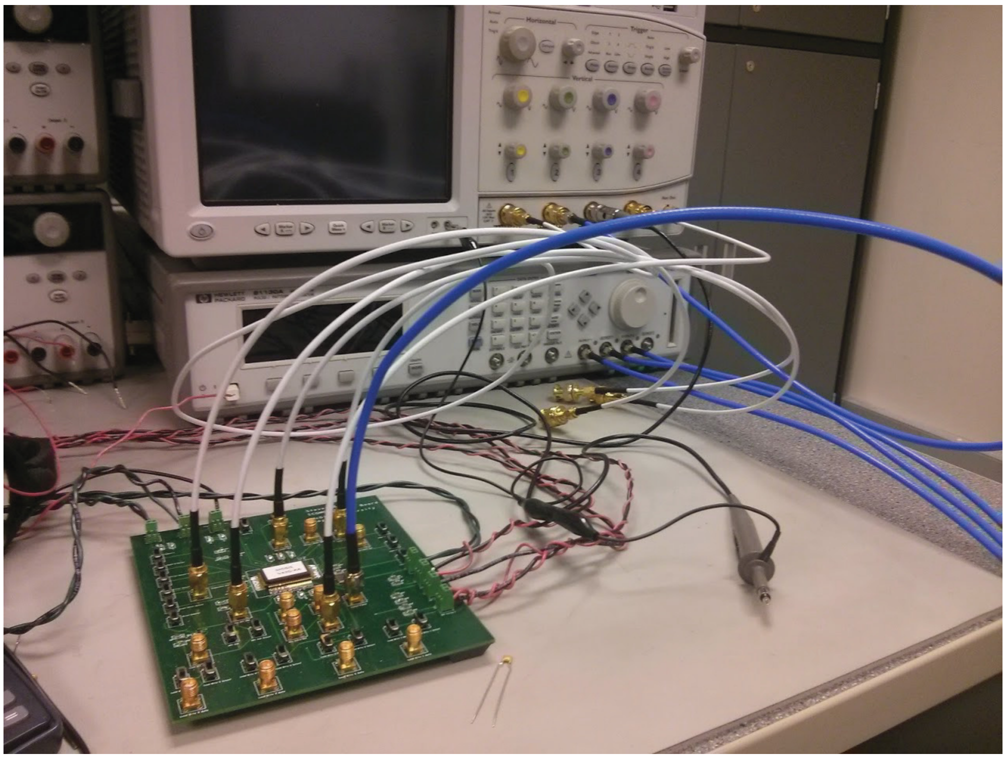


FIGURE 6.1: Experimental setup for the testing of the PTI

shift register is then programmed by using the appropriate "Load Bits X" signal, as well as the "Load DFF CLK" signal which are both generated by the HP 81130A pattern generator. See section 5.8 for the process to load the signals properly.

The Agilent 80000B is used as an oscilloscope to extract the timing and voltage characteristics from the PTI. The 80000B can handle signals up to 13 GHz, and can sample up to a speed of 40 Gbits/sec. The advantage of the 80000B is that it it possible to save each sample in the waveform for later analysis.

The 0.13 μm CMOS design was designed with the use of the Cadence design suite and was fabricated. The result of the fabricated chip can be seen in Figure 6.2. Following the fabrication, it was packaged by Mosis in a CQFP80A package. Inside the package, bond wires are used to connect the CMOS die to the package pins. Figure 6.3 illustrates how the chip is connected to the package.

6.2 DLL Time-Mode Filter

A type-II DLL (Figure 2.3) is to be used as a time-mode filter in the design of the PTI as described in Chapter 4. In the design phase of the thesis, an extra DLL was placed on the CMOS die in order to be able to easily verify its operational characteristics. Without a separate DLL, it would have been impossible to properly characterize the DLL due to

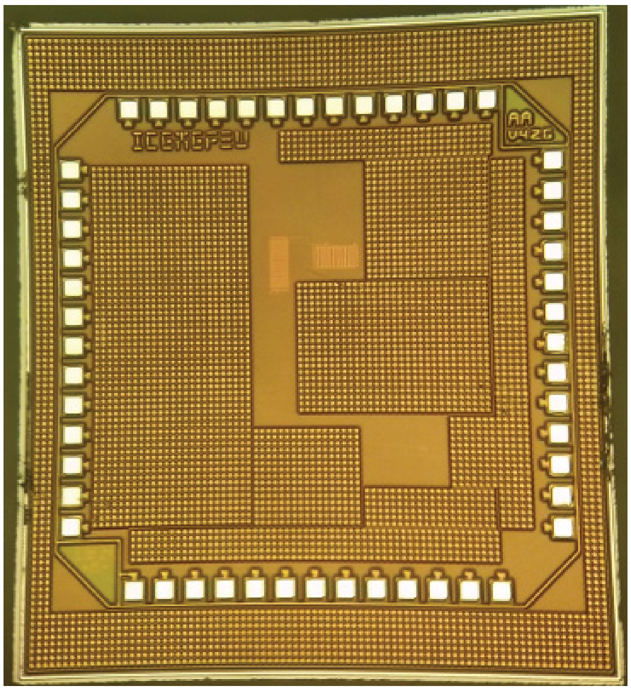


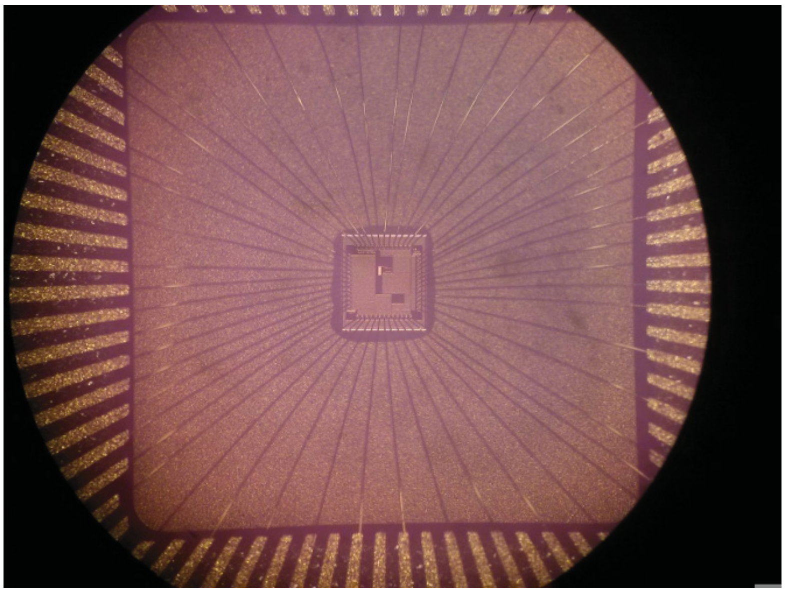
FIGURE 6.2: The fabricated 0.13 μm CMOS die

the very complicated nature of the design of the PTI and the limited output signal pins on the CMOS die. The DLL being tested in this design is the exact same design and layout as the DLLs that are used in the PTI, so all of the results extracted from these tests are possible to infer to the DLLs in the PTI.

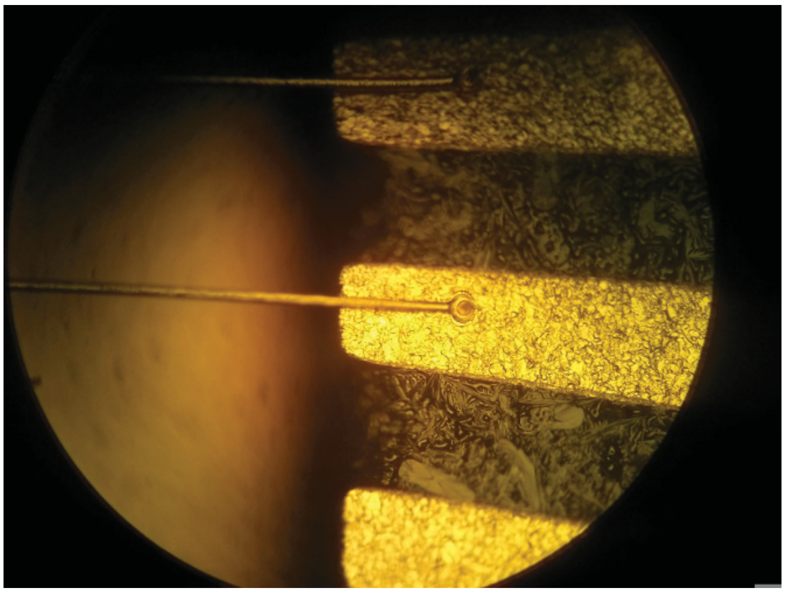
The DLL has two inputs, one loop filter connection, and one output. The two inputs are the clock signal input, and the bitstream input as described in Section 2.2. The loop filter connection is simply a point connection from the charge pump output to the VCDL as illustrated in Figure 2.13, so it is only possible to implement a first-order design. Finally, the output connection is the output of the DLL and will be used to analyze its performance.

The loop filter was chosen following the procedure of DLL designs outlined in Chapter 4. With a delay line constant of $60.04 \times 10^{-9} \frac{sec}{V}$, a charge pump constant of 0.16 mV/rad, a clock frequency of 28.625 MHz, and a chosen bandwidth of 8250 Hz, a capacitor value of 33 nF was obtained. With a lower bandwidth, one can expect better performance due to the fact that the DLL is extracting the DC average value from the bitstream.

A first-order sigma-delta modulator designed and tested in software in Chapter 3, was used to encode a variety of DC codes. The codes and their corresponding sigma-delta representation is shown in Table 6.1.



(A) The connections from the CMOS die to the packaging



(B) Connection of bond wire from CMOS die to package

FIGURE 6.3: Packaging of the 0.13 μm CMOS die

The bitstreams were then time-encoded using the DTC outlined in Section 2.5. This means that each 1 bit is represented by a "1100" sequence corresponding to a 0^o phase-shift, and each 0 bit is represented by a "0110" sequence corresponding to a 90^o phase-shift. Therefore, each repeating bit sequence that is used to stimulate the DLL is $8 \times 4 = 32$ bits in length. The DC code variation allows for the output edge to be controlled anywhere from 0 to 90 degrees and, depending on different experimental conditions, it is possible to use different phase-encodings for the DTC which will result in a smaller range, but finer resolution for the control of the output edge signal. Additionally, the duty cycle

Table 6.1: DC Code Sigma-Delta Representation

DC Code	Sigma-Delta Encoding (repeating)
0	00000000
0.125	10000000
0.25	11000000
0.375	11100000
0.5	11110000
0.625	11111000
0.75	11111100
0.875	11111110
1	11111111

of the input bitstream and clock signals will not have any effect on the performance of the DLL due to the fact that only the rising edges for each signal are registered. Figure 6.4 shows the settling of the loop filter voltage when changing the input DC code from 1 to 0.

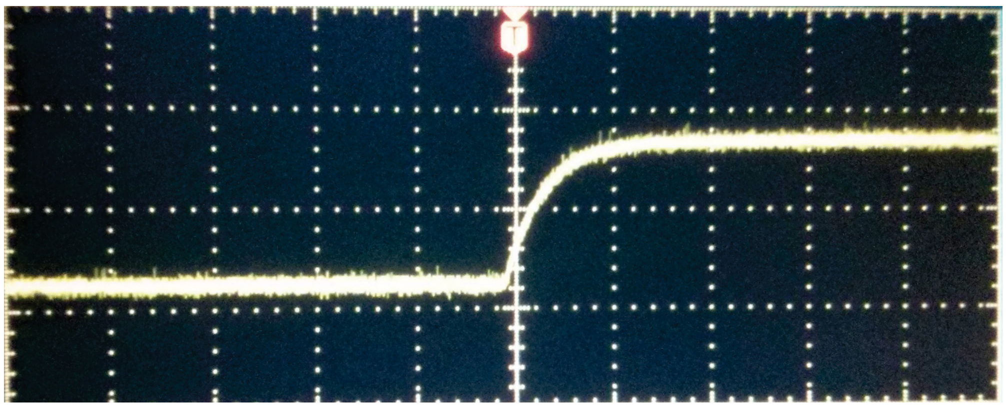


Figure 6.4: DLL loop filter voltage when changing the input DC code from 1 to 0.

The performance of the DLL time-mode filter to control the locations of the output edges was measured, and the results are shown in Figure 6.5. This figure illustrates the relationship between the input DC code, and the phase of the output edge with respect to the input constant clock signal. In the same figure, the ideal phase shift, as well as the transistor level edge locations. As seen in the figure, the output edge locations follow the expected edge locations. The linearity of the DC-code to edge-location transfer function makes the adjustment of the location of the edge locations simple and predictable, as was desired for this design.

In addition to measuring the phase of the edges, the standard deviation and peak-to-peak jitter of the edges was measured. The dominating source of the jitter is the quantization noise that passes through the filtering process. To measure the jitter, the sigma-delta modulated DC code is loaded into the HP pattern generator and applied to

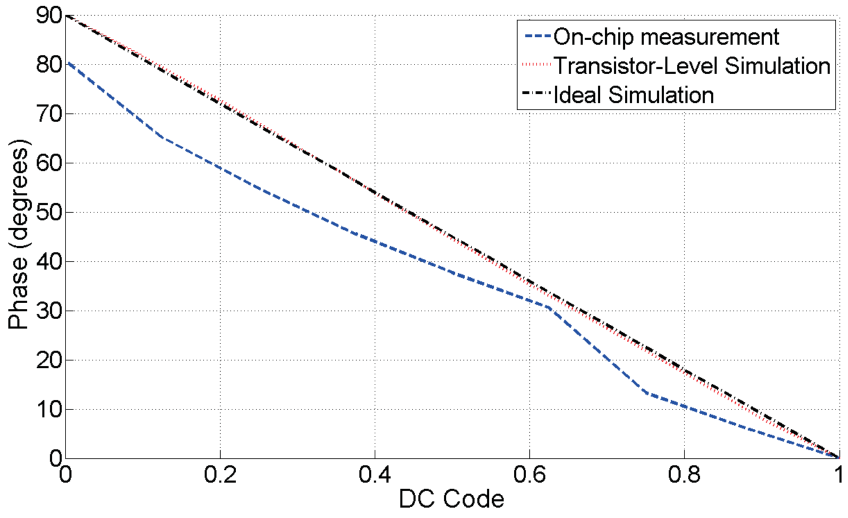


FIGURE 6.5: DC-code input and the corresponding edge locations at the output of the DLL time-mode filter

the input of the DLL. The same DC codes from Table 6.1 were used and the results are illustrated in Figures 6.7 and 6.8 for the peak-to-peak jitter and the standard deviation of the edge jitter, respectively. For each test, a minimum of 20 000 edge transitions were collected. Figure 6.6 shows an example of the extracted edge location histogram. A count in each bin is added when the edge transition is measured at each location.

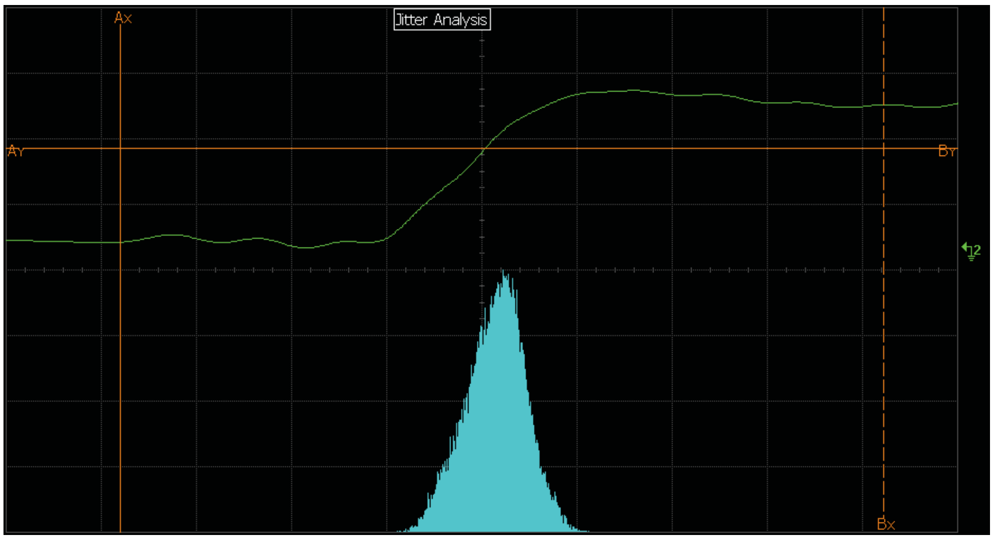


FIGURE 6.6: Edge location histogram for a DC input code of 0.75

The peak-to-peak jitter that is measured with the first-order DLL is found to be at a maximum of 338 ps for a signal of 28.625 MHz (34.9 ns period). The peak-to-peak jitter was measured as the location of the latest edge-transition minus the location of the earliest edge-transition. As this is the peak-to-peak jitter, this is the worst-case scenario that would be encountered, and the majority of the jitter is significantly less

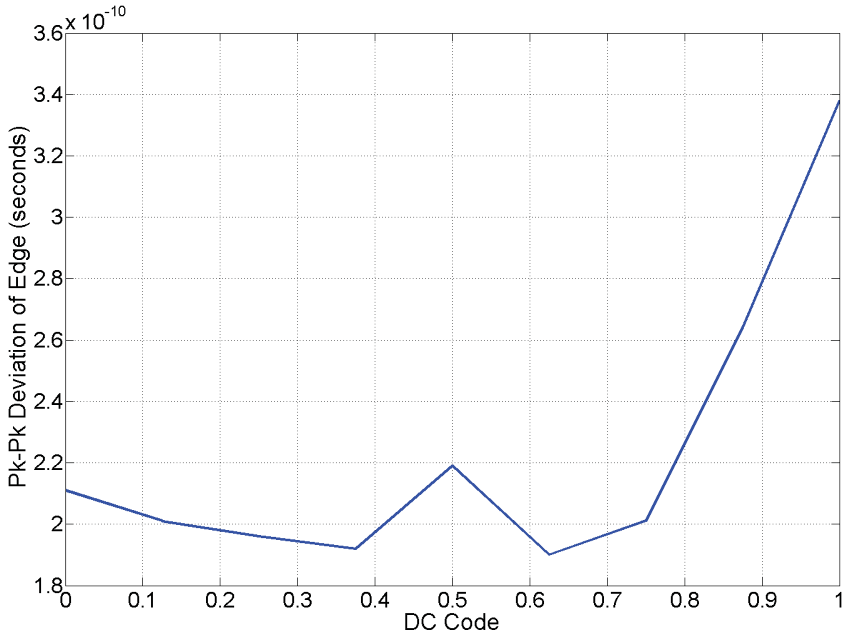


Figure 6.7: The peak-to-peak deviation of the edge locations for a first-order DLL

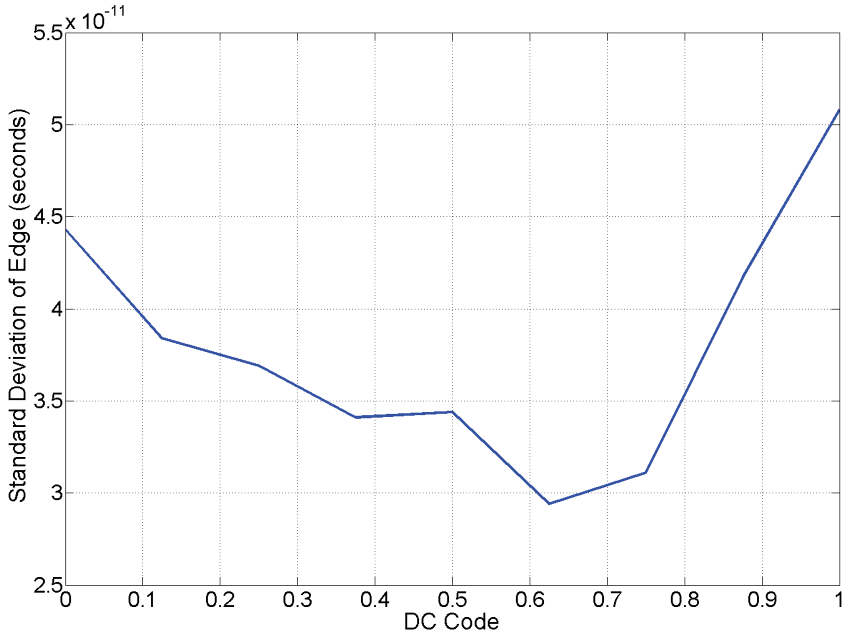


Figure 6.8: The standard deviation of the edge locations for a first-order DLL

than these values. The standard deviation of each edge is reported in Figure 6.8, where the standard deviation of the edge in each location was measured. Here, the standard deviations vary from 29.4 ps to 50.8 ps. It is important to note that in this case, the standard deviation and the root-mean squared (rms) values are the same. The reason is that the edge locations are being measured from the mean location of the edge so, when computing the standard deviation with Equation 6.1 where μ is the mean value

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{6.1}$$

is the same as calculating the rms value with the equation below:

$$x_{rms} = \sqrt{\frac{1}{N}(x_1^2 + x_2^2 + ... + x_N^2)} \tag{6.2}$$

When the measured result is compared to that in [13], the results are a significant improvement in the jitter performance. It is reported in [13] that the best-case jitter performance obtained with a 50 MHz clock is 65 ps, and outside a small DC code range of 0.35 to 0.55 this jitter increases as high as 1000 ps. In this work, the worst-case jitter performance that is measured is 50.8 ps.

From the measurements acquired in this section, it is possible to see that the jitter increases as the DC code approaches 1, and performs best from input codes ranging between 0.3 and 0.7. While it increases only slightly outside this range, in order to have the least amount of jitter, it is best to operate within this range.

The experiments were repeated for a phase encoding of 45^o so that the edge was controllable between 0^o and 45^o in the same fashion as the previous experiments. This would allow for a greater resolution between subsequent edge placements with the same number of bits in the sigma-delta controlling sequence. For a 45^o encoding a greater total number of bits is be necessary to represent the 45^o phase shift over the 90^o in the DTC, and the shifting clock has to be doubled in frequency. With the 45^o phase-shift, 8 bits are required for each sigma-delta bit instead of 4 with the 90^o shift. The result is an edge that can be controlled with twice the resolution, but at the expense of having twice the number of bits circulated in the circular bitstream. The 45^o encoding of the signal is represented by a "01111000" bitstream, while a 0^o is represented by "11110000". Figure 6.9 illustrates the relationship between the input DC code, and the edge location when a phase-encoding of 45^o is used.

Figure 6.10 shows the peak-to-peak jitter of the edge and Figure 6.11 shows the standard deviation of the jitter as the DC code is varied from 0 to 1.

The tests were again repeated, but this time they were done using a smaller loop bandwidth of 825 Hz (loop capacitor of 330 nF) to be able to compare the average jitter at the output. As shown in Chapter 4, it is expected that with a smaller bandwidth, less jitter will be present on each edge. Figures 6.12 and 6.13 compare the effects of

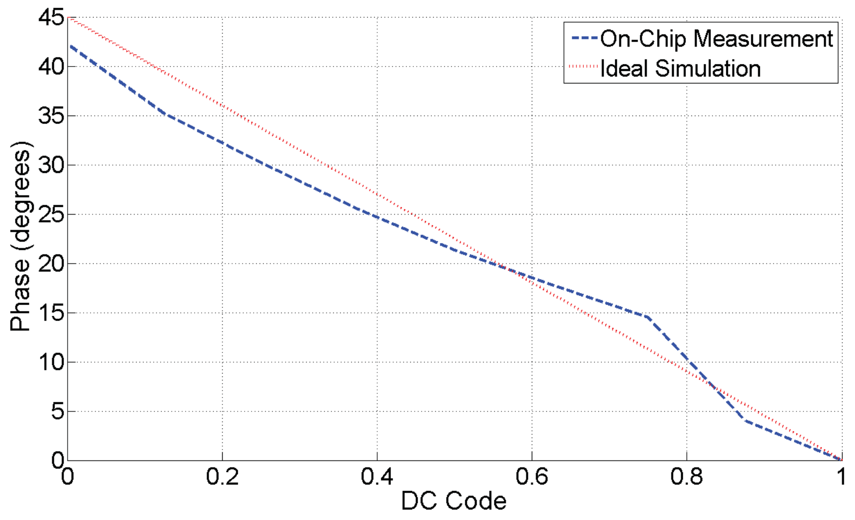


FIGURE 6.9: Relationship between the input DC code and the phase of the output when a 45^o phase-encoding is used.

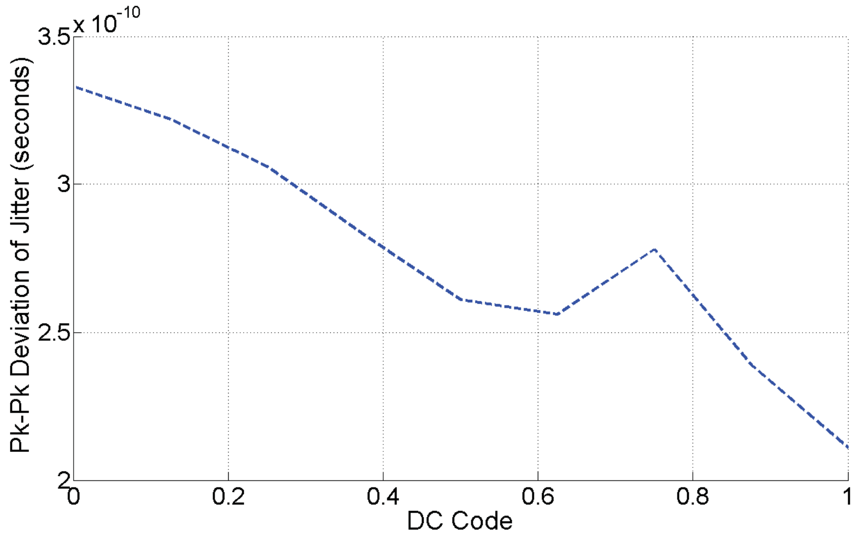


FIGURE 6.10: The peak-to-peak deviation of the edge locations for a first-order DLL with a 45^o phase-encoding

different loop bandwidths on the standard deviation of jitter on the output signal with an input phase-encoding of 90^o and 45^o, respectively.

It is obvious that with a reduced bandwidth of the DLL, the standard deviation of the jitter decreased as expected in the 45^o phase-encoding case. With the 90^o phase-encoding, the two results are very similar and this can be attributed to the fact that the jitter has already been minimized for this configuration and clock input.

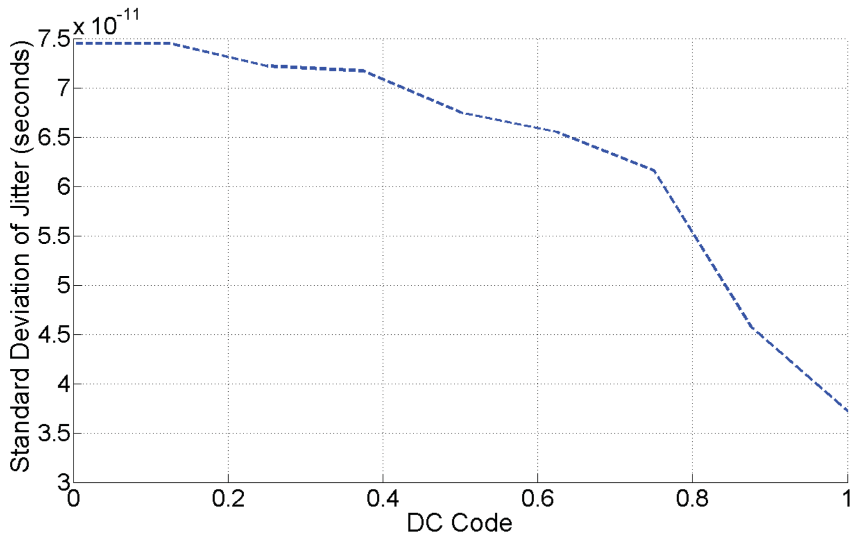


FIGURE 6.11: The standard deviation of the edge locations for a first-order DLL with a 45^o phase-encoding

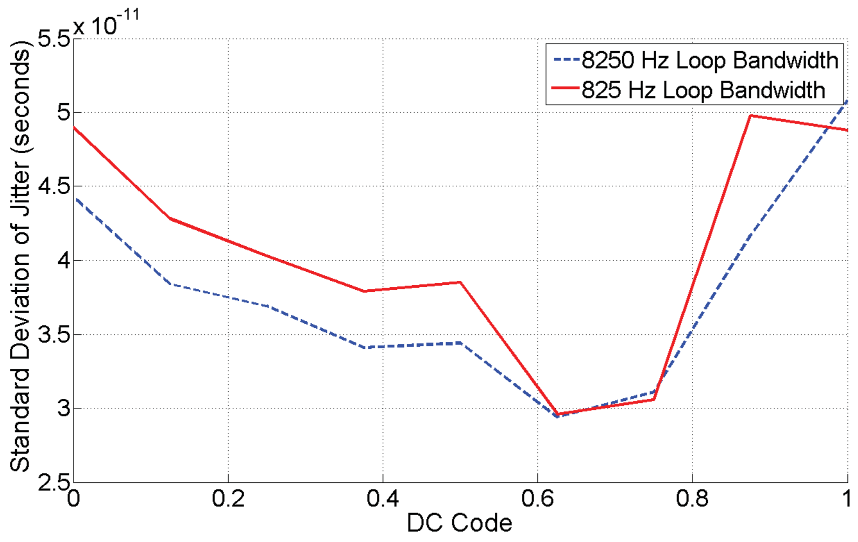


FIGURE 6.12: The effects of different loop bandwidths on the standard deviation of jitter on the output signal with an input phase-encoding of 90^o.

6.3 Probabilistic Test Instrument

In this section the Probabilistic Test Instrument (PTI) that was outlined in Chapter 5 is tested and the results are reported. The PTI takes up the majority of the CMOS die that was fabricated for these tests. The PTI will generate the test stimulus signals, as well as measure the performance of the DUT by means of statistical extraction.

The experimental setup of this design is exactly the same as that for the DLL time-mode filer. All of the PCB power and control voltages are connected to DC power

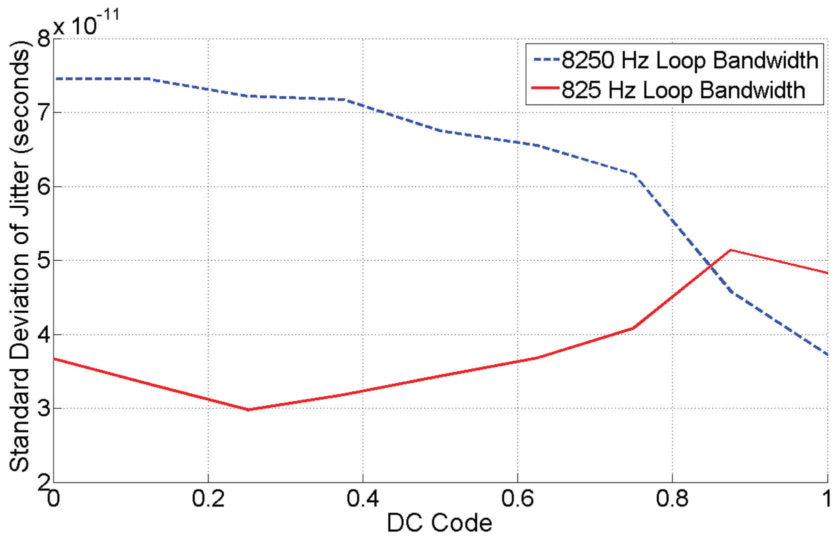


FIGURE 6.13: The effects of different loop bandwidths on the standard deviation of jitter on the output signal with an input phase-encoding of 45^o.

supplies using simple copper wires, input bit channels are connected to a pattern generator using a series of SMA cables, and the outputs are connected to an oscilloscope for measurement gathering. Figure 6.14 shows the setup of the testing environment.

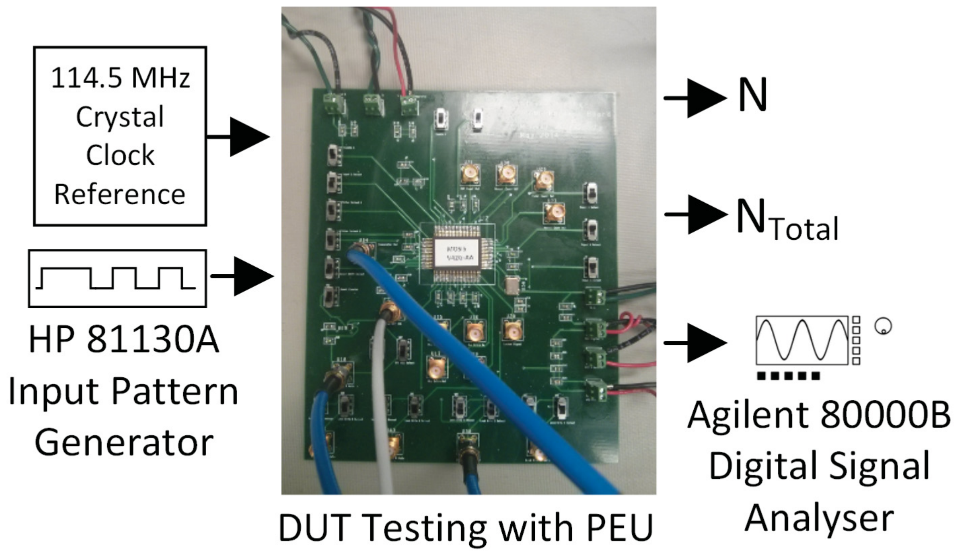

114.5 MHz Crystal Clock Reference
HP 81130A Input Pattern Generator
DUT Testing with PEU
N
N_Total
Agilent 80000B Digital Signal Analyser

FIGURE 6.14: Bench test setup for measuring the fully integrated PEU in a 130 nm CMOS process technology.

The design of the PTI is very complex, so the first steps that were taken in testing were to ensure that all the components were working as designed. First, the BSG was tested to ensure that it was possible to stimulate the DUT with the proper signals, and following that, the PEU was tested. The following subsections outline the testing that was done to evaluate the performance of each component.

6.3.1 Circular Shift Register

The circular shift resister was designed to be possible to test with a dedicated output pin connected to the output of the 256-bit circular shift register. By monitoring the waveforms on this pin, it is possible to determine whether or not the component is working correctly. In this test, a sequence of pulses of increasing length was loaded into the shift register and circularly shifted out. The result proved that the component was working as desired and is illustrated in Figure 6.15 which shows that the output is exactly what is expected and confirms that the circular shift registers are indeed working. As not every shift register has a dedicated output pin, the other registers are tested in conjunction with other components. Additionally, since they are all the same architecture, it can be expected that the other shorter shift registers work as expected.

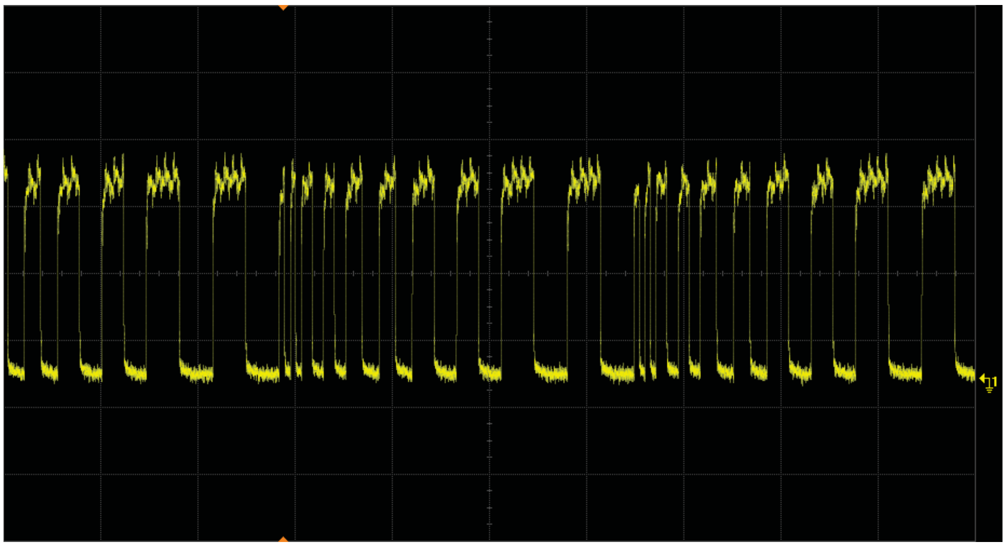
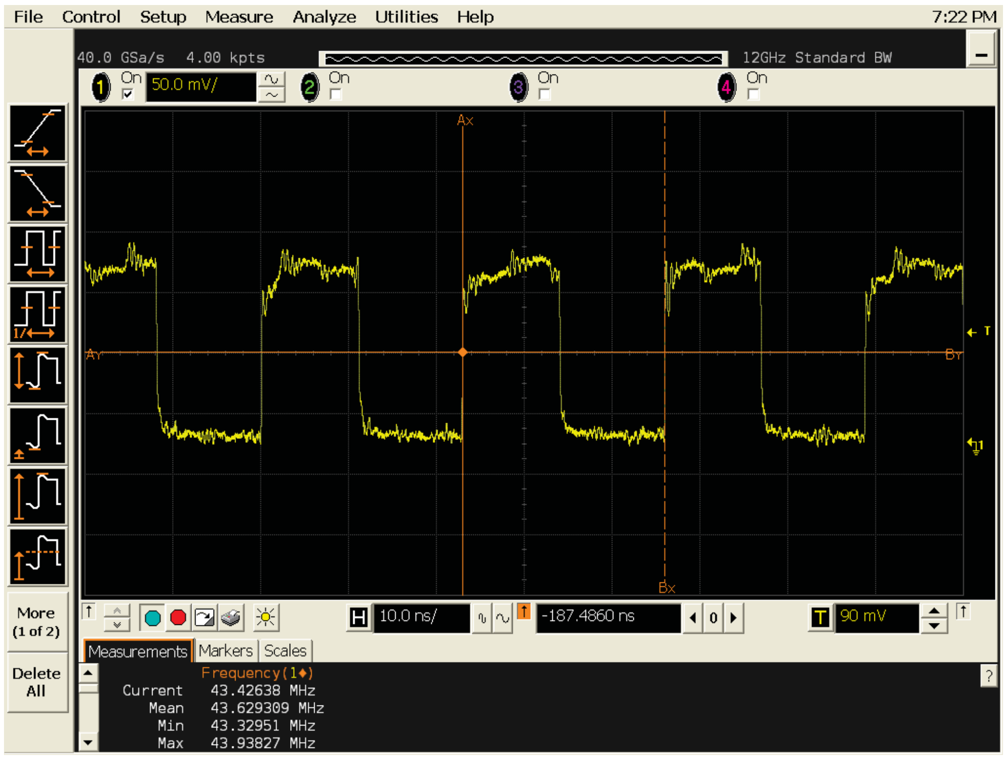


FIGURE 6.15: Testing of the circular shift register

6.3.2 Input Channel 1 and 2

Input channel 1 was tested in conjunction with input channel 2. The two channels together make up the output bit sequence from pin “Out Inputs 13” which will subsequently be fed into the DUT. As shown in Figure 5.3, a multiplying PLL is used to bring the clocking frequency up to the required speed such that the bits are timed correctly to achieve the desired output sequence. In this test, a simple “1100” pattern, signifying a time-encoded 0^o phase-shifted signal, was loaded into the register to ensure that the PLL was able to multiply the clock correctly, and the output of the DLL was monitored. The results of this test are illustrated in Figure 6.16.

The output frequency of the waveform is expected to be 57.25 MHz but, as shown in the figure, the output frequency of this waveform is only 43.42 MHz. Upon further



```
40.0 GSa/s   4.00 kpts          12GHz Standard BW
1 On 50.0 mV/
H 10.0 ns/   -187.4860 ns   0   T 90 mV

          Frequency(1♦)
Current   43.42638 MHz
   Mean   43.629309 MHz
    Min   43.32951 MHz
    Max   43.93827 MHz
```

FIGURE 6.16: Output waveform from the circular shift register following the loading of a "1100" repeating bit sequence and clocking with the multiplying PLL

investigation, the reason for which the output frequency is not what is expected is due to the fact that there was extra parasitic capacitances that are present at the output of the ring oscillator. At the end of the design phase, an extra test point was added to the PLL to monitor the operation, and the extra capacitance was not taken into account. For this reason, it is not possible for the ring oscillator to oscillate fast enough to bring the output frequency up to the desired 229 MHz. The basic functionality of the PLL is correct, and it has been measured that the control voltage on the VCO line is at its minimum in order to maximize the output frequency, but with the added capacitance, it is not possible for the output frequency to be brought up to what it needs to be for proper operation.

6.3.3 Input Channel 3

Input channel 3 is a channel designed for stimulating the DUT with AC signals, as covered in Section 5.2. Input Channel 3 uses a circular shift register that can hold up to 256 bits that is subsequently fed into a voltage-mode filter. The input waveform that is loaded into the shift register is sigma-delta encoded, and the voltage-mode filter is used as a reconstruction filter to extract the original input waveform. Following from Chapter 3, the first input signal was encoded with an input frequency of 447.265 kHz and was loaded into the circular shift register. The waveform was loaded and the resulting output from the circular shift register is shown in Figure 6.17.

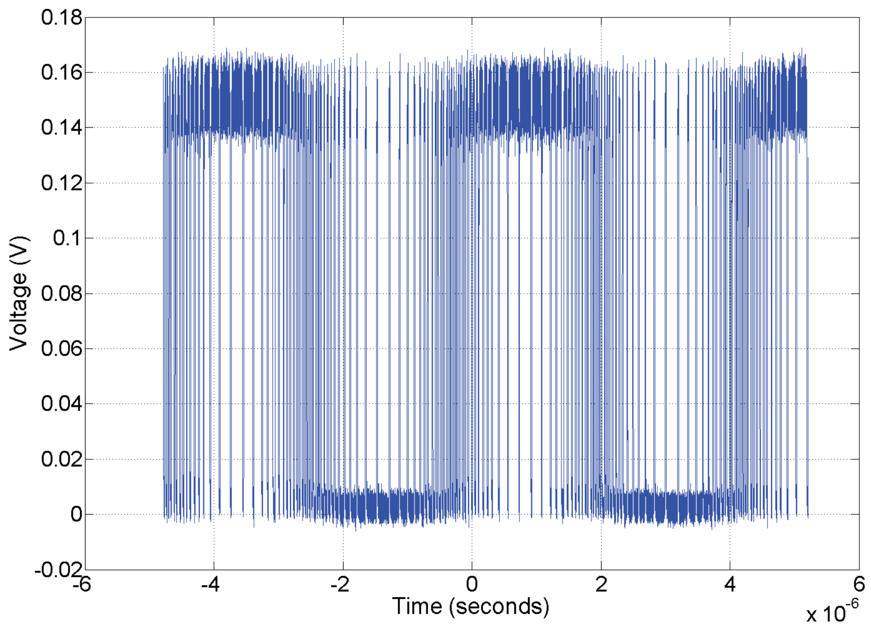


FIGURE 6.17: Sigma-delta encoded 447.265 kHz waveform at the output of the 256-bit circular shift register for input channel 3

The subsequent waveform was then put through a voltage-mode reconstruction filter with a cutoff frequency of 829.075 kHz. Due to a mistake by the author, the output testing node was included to have a digital buffer such that it was possible to drive the output resistance and capacitance that is seen on the package's pads, and the oscilloscope. Unfortunately, this results in a square wave output, and not a sine-wave output as should be expected. However, it is possible to infer from the period of the square wave, that the filter is working properly, and that the digital buffer is acting as an amplifier to saturate the signal past the threshold voltage of the transistors. This result is illustrated in Figure 6.18. Note that in this figure, the output is being shifted at a faster rate in order to test the limits of the registers.

With the input channel 1 not working as desired due to the multiplying PLL, the input channel 3 output can be connected to the DUT in order to provide a stimulating signal. This signal will not be programmable like channel 1 would provide, but it allows measurements to still be taken regarding the DUT and PEU. Since the shift register for this input is clocked at 114.5 MHz, 4 bits must be used to make up the clock signal to meet the timing requirements of the DUT.

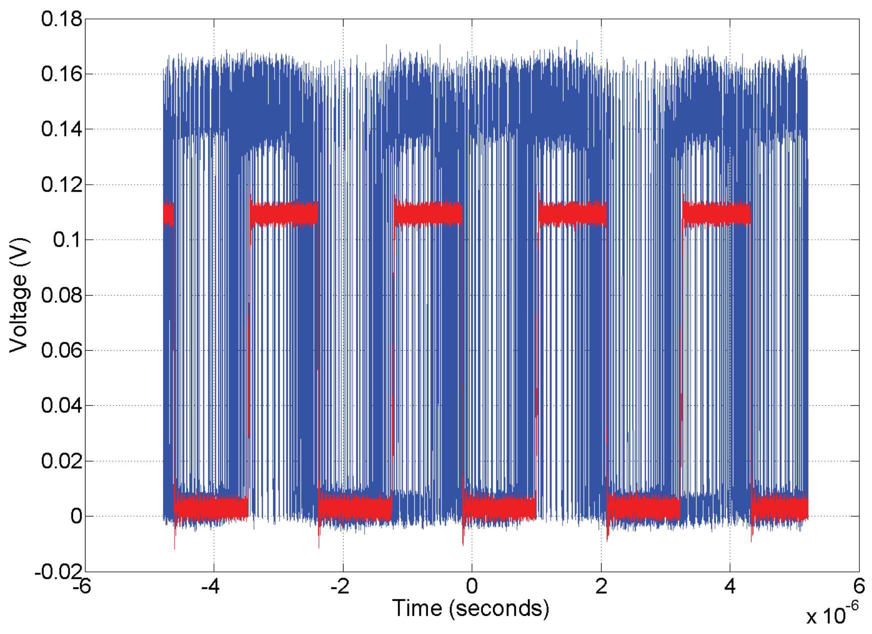


FIGURE 6.18: Sigma-delta encoded 447.265 kHz waveform at the output of the 256-bit circular shift register for input channel 3 following a digital buffer

6.3.4 Input Channels 4 and 5

Input channel 4 is used to set the threshold level at the input to the comparator so that the sampling location can be changed as desired. This was done by loading a variety of sigma-delta encoded DC signals into the 20 bit register and monitoring the output threshold voltage levels. Additionally, it is possible to provide a voltage to the comparator to ensure that the correct voltage level is used.

Input channel 5 was tested to ensure that proper locking of the signal was done, and that it was possible to move the edge of the signal around by controlling the input DC code that is loaded into the 64-bit circular shift register. Input channel 5 includes the coarse-locking of the signal at the output of the DUT, and also the fine-tuning of the edge via the sigma-delta DC code and time-mode filter as covered in Chapter 4. The first DLL to delay the signal to 315^o of the original signal worked correctly. This ensures that the subsequent 90 degree controllable phase range allows for the edge to be controlled anywhere from -45^o to 45^o with respect to the edge of the DUT. As the PLLs in the design are limited in frequency multiplication (due to the small error by the author), it it impossible for the coarse-locking mechanism to work as intended by locking to the output signal of the comparator. To remedy this situation, a wire was soldered to the loop filter of the DLL at the output of the coarse-locking circuit, and the phase was controlled manually by simply adjusting the voltage to the loop filter. This approach allowed the input channel to work correctly, but with a small limitation in the resolution at which the timing strobe can be controlled.

6.3.5 Probability Extraction Unit

Following the testing of the BSG, the PEU was tested to extract the bit error-rate for the DUT. As the input channel 1 was not working, it was impossible to add extra jitter on the edges of the clock. Instead, a simple clock from channel 3 was used to stimulate the DUT. The threshold levels for the input of the comparator was adjusted manually to ensure that the correct levels were being used to sample the DUT output signal. Following the comparator, two counters were used to count the number of edges: one for counting the total number of timing strobes, and another for counting the number of edges of the VC_{OUT} ahead of the timing strobe for the duration of the test. Each counter has two modes, a counting mode and a shifting mode. The counting mode is enabled for the duration of the test, and once completed, the shifting mode is enabled such that the counted numbers can be shifted out serially and recorded. Each counter utilizes 16 separate counting cells to make a full 16-bit counter for a total of 65 536 possible counts. If the output ever passes the number of counts, an overflow signal is raised so it is known to the tester that the counter has overflowed and that this should be taken into consideration for the data gathering.

The DUT was connected to the PEU and BSG and stimulated with a clock signal of 28.625 MHz from input channel 3. The comparator threshold voltage was set at 0.6 V and the t_{STRB} clock signal was varied across the output waveform of the DUT. As discussed in the previous section, the timing strobe location (for in put 5) is adjusted with the help of an external voltage source used to set the voltage on the loop filter of the DLL, essentially controlling the phase of the timing strobe. The voltage resolution with which this signal is controlled is 1 mV which allows for the edge to be controlled with a similar resolution to what was initially designed.

With the manual voltage control, a 1 mV change in loop filter voltage amounts in approximately 600 ps of edge movement. Following from Section 5.7 the original design of 64 bits of time-encoded signals allows for a resolution of 545 ps which are on very similar time scales. Each test was conducted by adjusting the phase of the timing strobe, and counting the number of high clock signals at the point in time. As the timing strobe is moved across the sampling eye, more high signals will be counted. Comparing the number of counted high signals against the total signals counted, it is possible to extract the information about the jitter of the edges of the DUT. Table 6.2 outlines the results from the tests and the corresponding probability that the VC_{OUT} edge, here called T_{DUT} for timing purposes, is before T_{STRB} as T_{STRB} is varied across 90 degrees.

TABLE 6.2: Probability that $T_{DUT} < T_{STRB}$ where T_{STRB} is varied across 90 degrees

Control Voltage (mV)	Relative Phase (degrees)	CDF
0.82	-20.05	0
0.83	-13.18	0
0.84	-3.45	0
0.85	5.14	0.26388
0.86	12.58	0.73302
0.87	18.99	1
0.88	25.41	1
0.89	31.59	1

The results were extracted by using the two counters. Using the shift capability of the counters, the results were then serially shifted out in binary form. It must be noted that the design is such that the bits are shifted out in reverse order, so a simple mirroring of the output bits must be done. Figure 6.19 shows an example of the shifting of the output count values so that it is possible to record the results. Each counter has two “1” bits that signal the beginning of the shifting period. The corresponding locations where both the “Total Count” (blue curve) and the “DUT Count” (red curve) signals line up with the two “1” pulses is the beginning of the shifting of the binary representation of the number of bits counted. In this particular example, there are 0 DUT edges counted, and 33 107 total edges counted.

Table 6.2 confirms the expected results where the later the timing strobe, the more edges will be counted. Additionally, the phase at which the edges are located are approximately in the correct position due to the fact that the first DLL in the design was used to move the control signal to -45^o, meaning that the edge would be located at approximately 0^o in the relative fine-locking tuning range. Figure 6.20 shows a graphical representation of the data from Table 6.2 which corresponds to the CDF of the edges.

6.3.6 Evaluation of the Performance of the PTI

In this section, the overall performance of the PTI will be discussed. The experimental results from the previous subsections will be used to extract the estimated BER and the overall performance of the PTI instrument.

The number of input signals that can be used to stimulate the DUT for a variety of tests is very limited due to the fact that the PLL at input channel 2 is not capable of multiplying the reference clock to twice it’s original frequency. As a result, it is

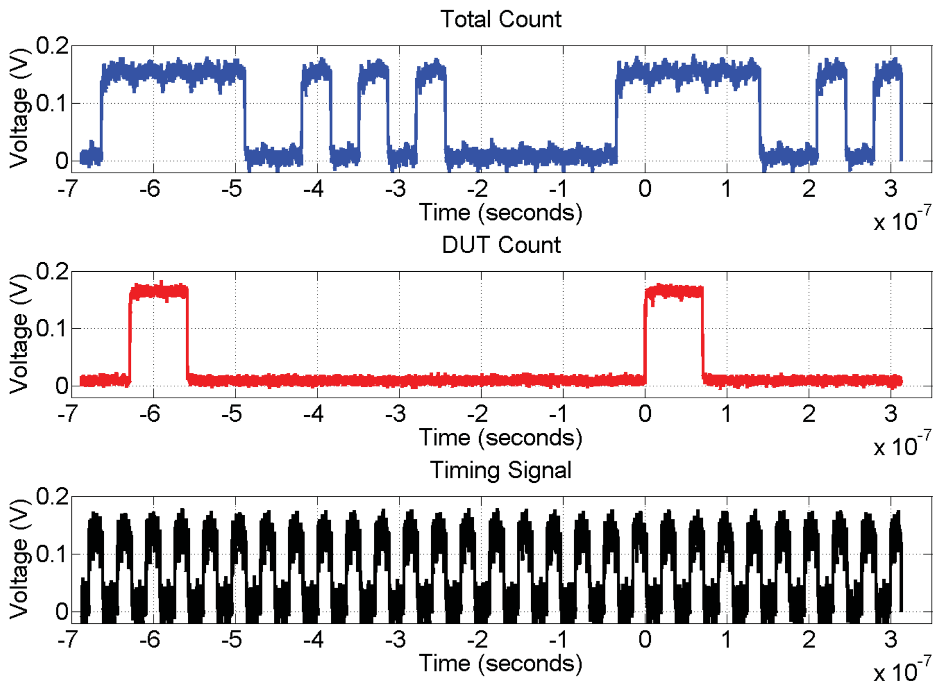


FIGURE 6.19: Output waveforms from the counter that are being used to extract the CDF corresponding to the DUT for a control voltage of 0.82 V (-20.05^o relative to VC_{OUT} edge)

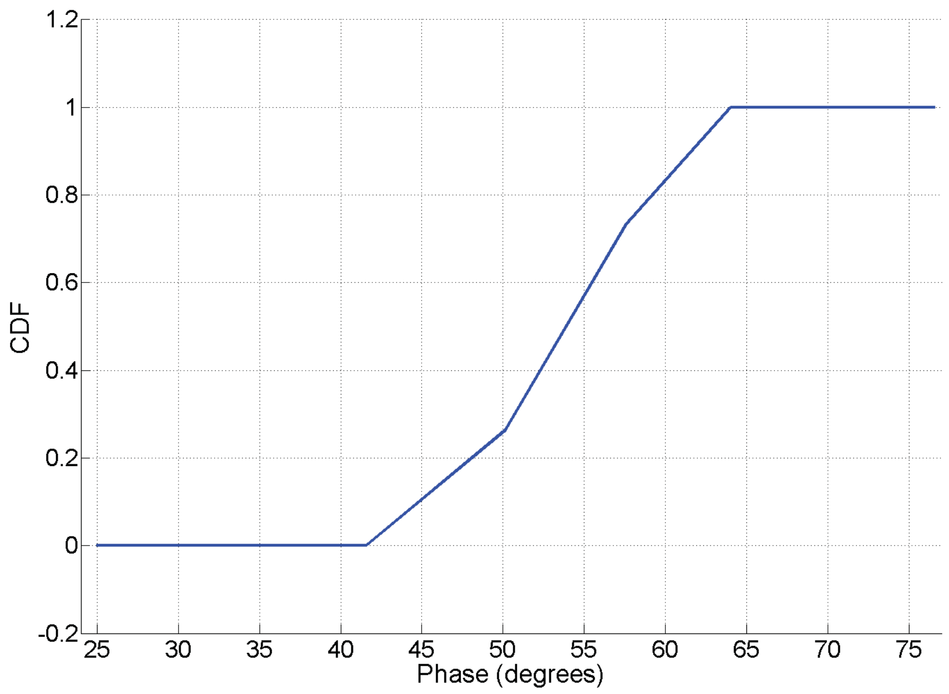


FIGURE 6.20: Probability that $T_{DUT} < T_{STRB}$ where T_{STRB} is varied across 90 degrees

impossible to perform any jitter tolerance tests with this instrument. The test that was performed is the time-based scan test outlined in Section 1.3.2 where the timing strobe, T_{STRB}, is swept across the eye diagram and the corresponding probabilities of $T_{DUT} < T_{STRB}$ are extracted. The cumulative distribution function (CDF) is shown in Figure 6.20, and its derivative, the probability density function (PDF), is shown in Figure 6.21.

Using the PDF, it is possible to extract the edge location's standard deviation. For lack of a better statistical model at this time, the distribution of the edge distribution is assumed to be Gaussian, and the PDF's approximated distribution is also plotted in Figure 6.21. In this particular case, one standard deviation is equal to 5.4 degrees, or 524 ps. Having extracted the standard deviation it is possible to find the BER by using the formula 1.11 which is again repeated below

$$\mathrm{BER}(t_{TH}) = \frac{1}{2} - \frac{1}{2} \times \Phi\left(\frac{t_{TH} - 0}{\sigma_{RJ}}\right) + \frac{1}{2} \times \Phi\left(\frac{t_{TH} - T}{\sigma_{RJ}}\right) \tag{6.3}$$

where t_{TH} will be equal to $\frac{0.5}{28.625M} = 17.5ps$. Substituting these results into 6.3,

$$\mathrm{BER}(17.5ns) = \frac{1}{2} - \frac{1}{2} \times \Phi\left(\frac{17.5ns - 0}{524ps}\right) + \frac{1}{2} \times \Phi\left(\frac{17.5ns - 34.9ns}{524ps}\right) \tag{6.4}$$

and simplifying,

$$\mathrm{BER}(17.5ns) = 3.7954 \times 10^{-245} \tag{6.5}$$

The BER in Equation 6.5 is extremely low due to the exceptionally long bit period and reasonably low jitter for this particular setup. The correctness of the results gained from the PTI can be very loosely checked by using an external oscilloscope to repeat the same process where the edge locations are measured and the BER is calculated. It must be noted that the results measured by the external oscilloscope do not reflect the communication channel exactly as there is a digital buffer between the channel and the oscilloscope. Without this buffer, when probing the communication channel, the characteristics would be changed and it would not reflect the real-world operation of the channel. Figure 6.22 shows the edge locations that were measured with the external oscilloscope in comparison to those extracted using the PTI itself.

As can be observed in Figure 6.22, the standard deviation for the off-chip measurement is slightly smaller than that gathered from the PTI. While the signal paths to each

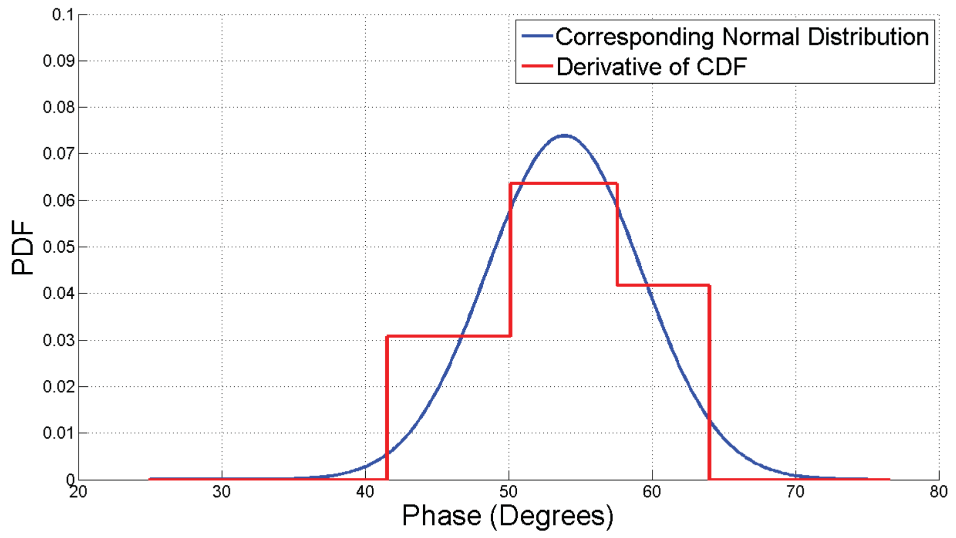


FIGURE 6.21: PDF that $T_{DUT} < T_{STRB}$ where T_{STRB} is varied across 90 degrees. This is the derivative of Figure 6.20 which represents the CDF.

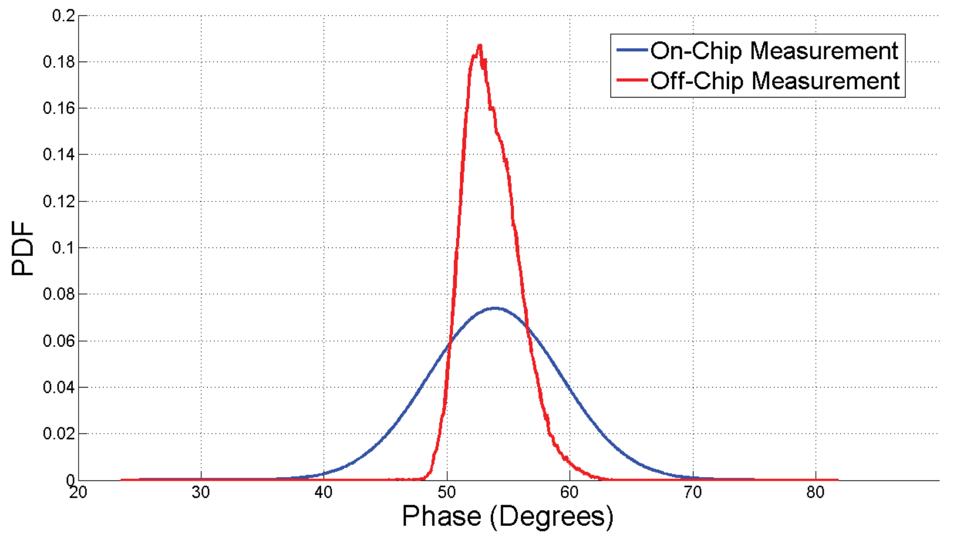


FIGURE 6.22: DUT edge locations measured with an external oscilloscope compared to those measured with the PTI. External measurements are buffered through a digital buffer prior to measurement.

measuring instrument are slightly different, i.e., additional buffers are included to drive off-chip signals, the results show similar behaviour, albeit with the on-chip PEU having about twice the RMS noise variation (specifically, the on-chip noise standard deviation was found to be 5.4 degrees or equivalently 524 ps and the off-chip noise variation was about 226 ps). The extra noise suggests the PEU is noisier than the high-performance oscilloscope. In so far as using the PEU results to compute BER, consider that if the sampling instant were placed 2 ns away from the clock reference edge, the BER would be

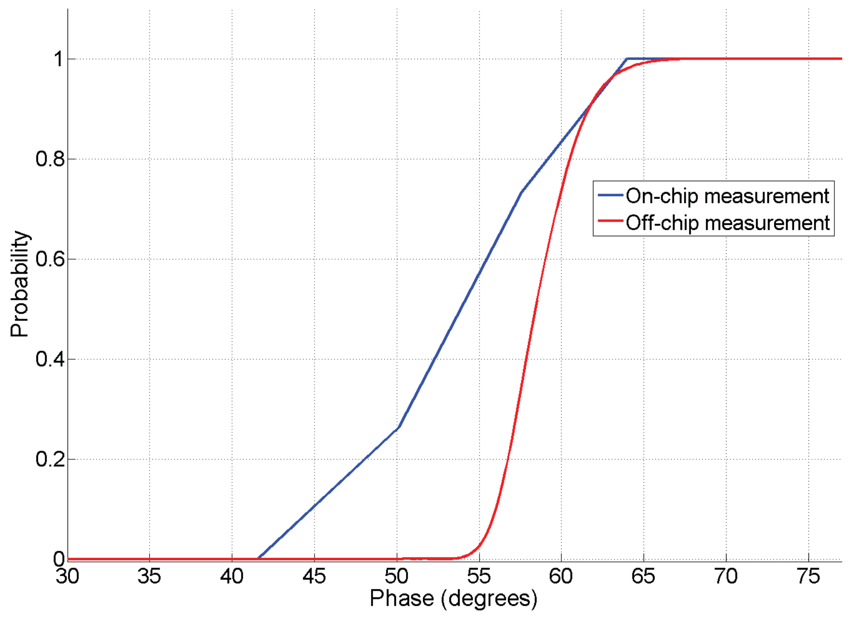


FIGURE 6.23: CDF for the measurements taken both on- and off-chip.

$$\mathrm{BER}(2ns) = \frac{1}{2} - \frac{1}{2} \times \Phi\left(\frac{2ns - 0}{524ps}\right) + \frac{1}{2} \times \Phi\left(\frac{2ns - 34.9ns}{524ps}\right) \tag{6.6}$$

and simplifying leads to

$$\mathrm{BER}(2ns) = 3.45 \times 10^{-5} \tag{6.7}$$

As shown in Equation 6.7, this results in a bit error approximately every 30k bits. As expected, with a very small amount of jitter and a very long period, the BER is extremely low. However, if the clock speeds were to increase, and the sampling locations changed, the BER would drop significantly.

6.3.7 Summary

In this section, the experimental results of the PTI were collected and evaluated. Although there were a few problems with the jitter injection test circuit in the BSG due to the PLL not working as desired, it was possible to extract the standard deviation of the jitter of the DUT, and approximate an error by using the CDF, PDF, and statistical models. The DLL as a time mode filter was tested and confirmed to work correctly, and the advantages of using a smaller filter bandwidth were demonstrated. Using the DLL to control a timing strobe, the performance characteristics of the DUT were extracted and it was determined that a standard deviation of the jitter was measured at 524 ps. This measurement was compared against that of a high performance oscilloscope where a standard deviation was found to be 226 ps. The extra noise suggests the PEU is noisier than the high-performance oscilloscope.

Chapter 7

Conclusion

7.1 Thesis Contribution

This thesis presents a built-in self-test probability test instrument that extracts and predicts the BER for a particular DUT. The instrument generates its own test stimulant signals for the DUT and also extracts the probabilistic data associated with the edge transitions so that it is possible to estimate the BER of the DUT. The instrument is composed of two modules: the multi-channel bitstream generator, and the probability extraction unit. The multichannel bitstream generator is completely programmable, which makes it extremely flexible and adaptable to different test stimulant signals that the test engineer would like to use and is ideal for both RF and mixed-signal devices. This thesis showed that it is possible to implement the PTI on a CMOS die in a BIST testing environment and extract the timing and probabilistic experimental data in order to estimate the BER.

The advantage of the PTI over previous devices is that it offers the test engineer an option to greatly reduce testing time by using probabilistic methods over a lengthy transmit and compare test. Additionally, it is an improvement over current probabilistic designs due to the fact that it is built in to the device that it is testing, so there is no additional loading of the communication channel whilst the test is taking place. This gives the test engineer the best and most accurate information in the performance of the communication channel.

The design that was presented can stimulate the DUT with AC signals, phase-modulated signals, as well as DC signals. Additionally the eye-diagram can be sampled at any voltage and time location in order to easily extract all of the information related to the operation of the DUT. The multi-channel bitstream generator was presented in

Chapter 5 where a bank of circular shift registers is used to generate a variety of signals to stimulate the DUT. The BSG is composed of three blocks: the software generator, the circular shift register, and the filter. The signal would fist be synthesized in software where sigma-delta modulators would encode the signal into a 1XN bitstream. Limitations as to the quality of the encoded signal were limited by the number of bits of memory in the circular shift registers. The bits were loaded into the circular shift register, and then circularly shifted to make a single continuous signal. This signal would then be filtered to produce the desired stimulating or timing signal for the DUT. The time-mode filters used were DLLs due to the fact that they would offer superior jitter performance over a PLL due to the lack of a VCO. A new method for designing DLLs was described which greatly simplifies the design process.

A prototype DLL was implemented separate to the PTI, and its operation was experimentally verified. The experimentation results outlined in Chapter 6 show how the edge of the signal can be accurately controlled with a sigma-delta time-encoded bitstream. It was shown that the modulator can be used to adjust the edge of a clock signal very accurately with a phase resolution of approximately 5^o and 10^o using an encoding of 45^o and 90^o respectively. Additionally, it is shown that a lower loop bandwidth results in less jitter on the output edge.

In Chapter 5, the PEU portion of the PTI was presented. The PEU is the block that extracts the probabilistic quantities from the operation of the DUT that are used to estimate the BER. Using a voltage comparator, a controllable timing strobe, and two counters, the PEU counts the number of times the VC_{OUT} signal is ahead of the timing strobe, as the timing strobe is swept across the eye diagram. Using this information, the jitter of the DUT output signal is measured and the standard deviation of this jitter, as well as the sampling location, is used to estimate the BER of the DUT using the time-based scan method presented in Chapter 1.

Matlab and Simulink simulations of the individual building blocks were presented in Chapter 5, and the experimental results and were presented in Chapter 6. It was shown that the PTI was able to extract the jitter characteristics for the VCDL DUT, and the BER was calculated. Depending on where the output signal is sampled in the eye diagram, it is possible to expect a wide range of performance. Given that the DUT operates at a fairly low speed and with a small amount of jitter, the estimated BER at the ideal sampling location was found to be extremely low at 3.7954×10^{-245}. With an increasing clock speed, the sampling location will be moved closer to the edge location, and the BER will drop significantly.

This thesis showed that it is possible to implement a BIST probability test instrument that will estimate the BER of a DUT. The design is be implemented on a CMOS so that there is no additional loading on the communication channel when the test measurements are being made, so that an accurate estimation of the BER can be made.

7.2 Future Work

There are many improvements that can be made to the implemented design in this thesis that are a good source of future work. The first is that higher-order DLLs should be implemented as filters in order to minimize the phase noise in the timing strobes. Some small issues should be fixed in the design of the circuit, namely the small parasitic issue that was encountered such that it was not possible to have the PLL multiply the frequency to the desired rate. It is possible to reduce the memory size that was used in the circular shift registers. A large memory cell was used for robustness purposes, however this introduced a huge limitation in the number of memory cells that could be incorporated in the design which in turn limits the resolution with which the signals can be controlled. These memory cells can be implemented as sRAM cells, or even a D-flip-flop implementation that uses feedback loops which more than halve the number of transistors used in the design.

In a future implementation, if a PRBS bit-sequence is to be used as an input stimulus signal, then alternate test methods should be used; namely the the Dual-Dirac jitter decomposition method and the Gaussian mixture jitter decomposition method discussed in Section 1.3. Utilization of such test methods will allow for the separation of the deterministic jitter from the random jitter to evaluate the performance of the DUT.

If a non-complex design is required, it is possible to reduce the whole size of the implementation such that it would be much more economical to fit into a production design. Finally, it would be an excellent improvement to have the circuit working at much greater speeds so that it would be possible to test very high speed communication circuits.

Appendix A

PTI Pin Names and Descriptions

Table A.1: Pin Names and Descriptions for the Fabricated CMOS Chip

Pin Number	**Pin Name**	**Pin Type**	**Description**
1	GND	Bidirectional	——
2	——	——	——
3	OUT DFF 256	Output	Output of the 256 bit circular memory so that we can verify that the output is correct.
4	——	——	——
5	DLL 2 Filter	Bidirectional	The filter for the second DLL. Only a first order design is possible.
6	——	——	——
7	CAP EXTRA	Bidirectional	The filter for the extra DLL. Only a first order design is possible.
8	——	——	——
9	DUT IN	Input	The input of the DUT (VCDL) which must be connected to out inputs 13 (pin 43).
10	——	——	——

Continued on next page

Table A.1 – continued from previous page

Pin Number	Pin Name	Pin Type	Description
11	DUT OUT	Output	The output of the DUT (VCDL) which must be connected to the input of the comparator (pin 58).
12	—	—	—
13	VDD ANALOG 1	Bidirectional	—
14	—	—	—
15	VSS	Bidirectional	—
16	—	—	—
17	DLL EXTRA CLK IN	Input	The clock input of the extra DLL. This must operate at 114.285 MHz.
18	—	—	—
19	DLL EXTRA IN	Input	Input to the extra DLL. This will be the sigma delta encoded bitstream.
20	—	—	—
21	DLL EXTRA OUT	Output	Output of the DLL. This will be the placed edge.
22	—	—	—
23	OUT14	Output	Negative 45 degree shifted signal out of the comparator. This is the automatic locking of the coarse signal.
24	LOCKED SIGNAL	Output	The output of the DLL with the controlled edge of the signal.
25	V 3.3	Bidirectional	3.3 V input for the level shifting of the clock input.
26	—	—	—
27	REF CLK IN	Input	The input reference clock input of a 114.285 MHz.
28	—	—	—

Continued on next page

Table A.1 – continued from previous page

Pin Number	**Pin Name**	**Pin Type**	**Description**
29	DUT V CONTROL	Input	The voltage control of the VCDL DUT whose performance will be measured.
30	—	—	—
31	LOAD BITS 4	Input	Loading the bits for the input number 4.
32	INPUT 4 SELECT	Input	Selecting the 4 input.
33	VSS	Bidirectional	—
34	VDD DIGITAL 1	Bidirectional	—
35	LOAD BITS 5	Input	Loading the bits for the input number 5.
36	INPUT 5 SELECT	Input	Selecting the 5 input.
37	MANUAL COUNT TOTAL	Output	Counting the output manually of the total number of bits.
38	TOTAL COUNT OUT	Output	Automatic counting of the signal that will be serially shifted out when count is not selected.
39	LOAD BITS 1	Input	Loading of the bits for the input number 1.
40	INPUT 1 SELECT	Input	Selecting the 1 input.
41	VSS	Bidirectional	—
42	—	—	—
43	OUT INPUTS 13	Output	The output of the inputs 1 and 3 that will be connected to the DUT, or directly to the comparator.
44	—	—	—

Continued on next page

Table A.1 – continued from previous page

Pin Number	**Pin Name**	**Pin Type**	**Description**
45	MANUAL COUNT DUT	Output	Counting the output manually of the DUT.
46	—–	—–	—–
47	DUT COUNT OUT	Output	The automatic count of the DUT that will be shifted out serially.
48	—–	—–	—–
49	INPUT 1 OR 2 SELECT	Input	Control signal for the OUT INPUTS 13 to choose either input 1, 2 or 3.
50	—–	—–	—–
51	VSS	Bidirectional	—–
52	VDD ANALOG 2	Bidirectional	—–
53	—–	—–	—–
54	V REFERENCE	Input	Reference voltage for the comparator.
55	—–	—–	—–
56	DLL 3 FILTER	Bidirectional	Filter for the delay locked loop 3. This can only be a single capacitor which will result in a first order system.
57	—–	—–	—–
58	COMPARATOR IN	Input	The input to the comparator which will be connected either to the output of the DUT, or the output13.
59	—–	—–	—–
60	PLL 1 FILTER	Bidirectional	The filter for the PLL 1.
61	—–	—–	—–
62	ENABLE 2	Input	—–
63	—–	—–	—–
64	ENABLE 1	Input	—–

Continued on next page

Table A.1 – continued from previous page

Pin Number	**Pin Name**	**Pin Type**	**Description**
65	INPUT 2 SELECT	Input	Select the input 2.
66	—	—	—
67	LOAD BITS 2	Input	Load bits input 2.
68	FILTER SELECT 1	Input	Filter select 1.
69	—	—	—
70	FILTER SELECT 2	Input	Filter select 2.
71	—	—	—
72	COMPARATOR OUT	Output	Output of the comparator that is used for probing purposes.
73	VSS	Bidirectional	—
74	VDD DIGITAL 2	Bidirectional	—
75	LOAD DFF CLK	Input	Clock signal that is used to load all the DFFs.
76	PLL 2 FILTER	Bidirectional	The filter for the second PLL that is used to multiply by 4.
77	DLL 1 FILTER	Bidirectional	Filter for DLL 1.
78	COUNT SHIFT SELECT	Input	Select signal for either counting or shifting the signals out.
79	RESET COUNTER	Input	Reset the counters to zero.
80	VSS	Bidirectional	—

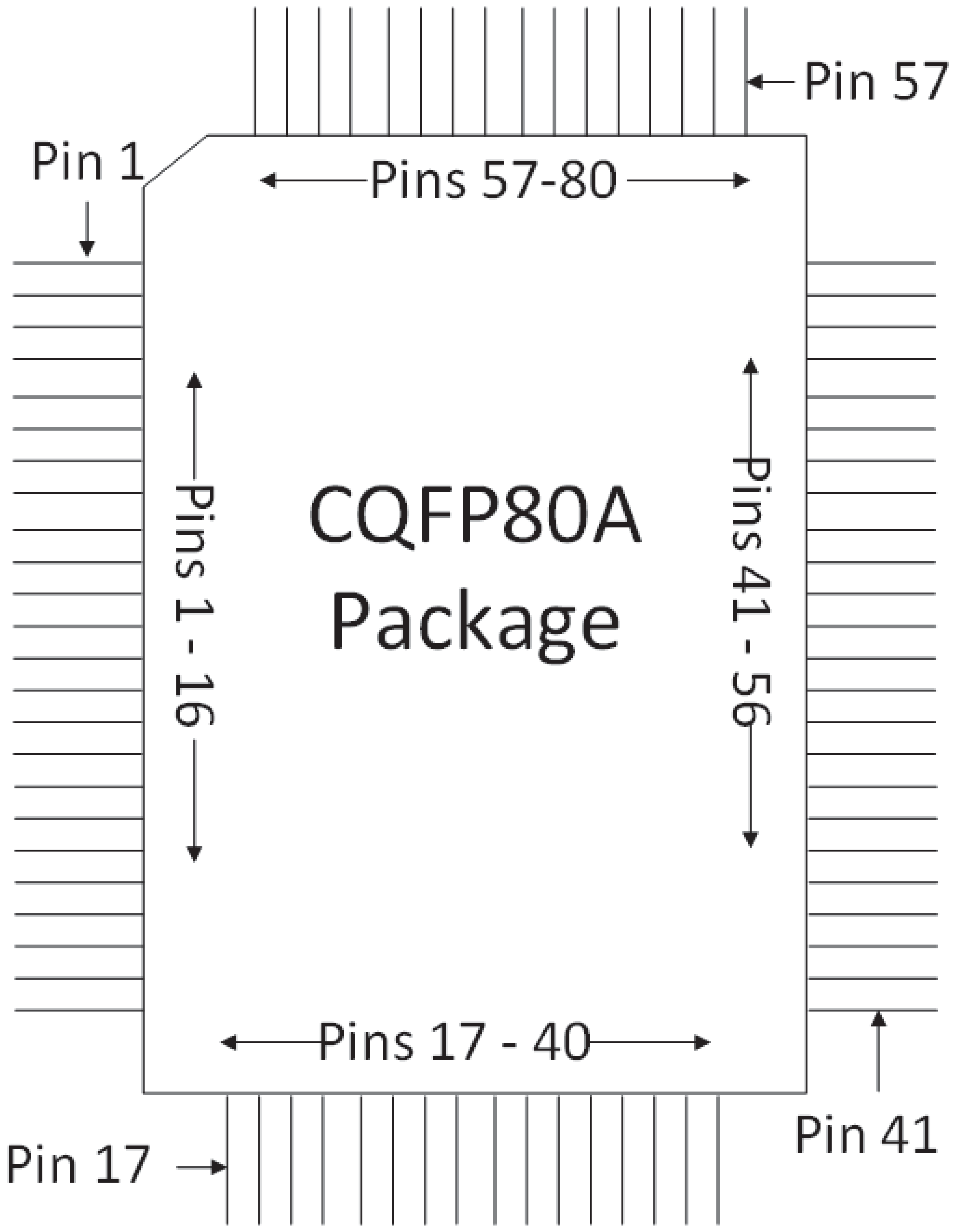


FIGURE A.1: Pin locations on the CQFP80A package

Bibliography

[1] G. W. Roberts, F. Taenzler, and M. Burns. *An Intorduction to Mixed-Signal IC Test and Measurement 2nd. ed.* Oxford University Press, New York, New York, 2012.

[2] S. Sunter and A. Roy. Adaptive parametric BIST of high-speed parallel I/Os via standard boundary scan. *IEEE International Test Conference*, pages 1–9, Oct. 2011. CA., USA.

[3] N. Abaskharoun and G. W. Roberts. Circuits for on-chip sub-nanosecond signal capture and characterization. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 251–254, May. 2001. San Diego, CA.

[4] A. Papoulis. *Probability and Statistics.* Prentuce-Hall, New Jersey, 1990.

[5] W. Ding, M. Pan, W. Wong, D. Chow, M. Peng Li, and S. Shumarayev. On-die instrumentation to solve challenges for 28nm, 28 Gbps timing variability and stressing. *International Test Conference.*

[6] S. Bielby and G. W. Roberts. Sub-gate-delay edge-control of a clock signal using DLLs and $\Sigma\Delta$ modulation techniques. *IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–5, May 2014. Toronto, Canada.

[7] C.-K. Ken Yang. Delay-locked loops - an overview. *Phase-Locking in High-Performance Systems IEEE Press.*

[8] J. R. Burnham, W. Gu-Yeon, C.-K. K. Yang and H. Hindi. A comprehensive phase-transfer model for delay-locked loops. *IEEE Custom Integrated Circuits Conference.*

[9] G. Chien and P. Gray. A 900-MHz local oscillator using a DLL-based frequency multiplier technique for pcs applications. *IEEE Journal of Solid-State Circuits*, 35 (12):1996–1999, Dec. 2000.

[10] M.-J. Lee, W. Dally and P. Chiang. Low-power area-efficient high-speed I/O circuit techniques. *IEEE Journal of Solid-State Circuits*, 35(11):1591–1599, Nov. 2000.

[11] J. M. de la Rosa. Sigma-delta modulators: Tutorial overview, design guide, and state-of-the-art survey. *IEEE Transactions on Circuits and Systems*, 58(1):1–21, Jan. 2012.

[12] B. Dufort and G. W. Roberts. Signal generation using periodic singal and multi-bit sigma-delta modulated streams. *Proceedings of the IEEE International Test Conference.*

[13] A. Chowhury. A probabilistic test instrument using sigma-delat phase signal generation technique for mixed signal embedded test. Master's thesis, McGill University.

[14] A. Ameri and G. W. Roberts. Time-mode reconstruction IIR filters for sigma-delta phase modulation applications. *Proceedings of the 21st Edition of the Great Lakes Symposium on VLSI.*

[15] M. Guttman and G. W. Roberts. Sampled-data IIR filtering using time-mode signal processing circuits. *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium.*

[16] M.-J. E. Lee, W. J. Dally, T. Greer, H.-T. Ng, R. Farjad-Rad, J. Poulton, and R. Senthinathan. Jitter transfer characteristics of delay-locked loops — theories and design techniques. *IEEE Journal of Solid-State Circuits*, 38(4):614–621, April 2003.

[17] X. Haurie and G. W. Roberts. A design, simulation and synthesis tool for delta-sigma-modulator-based signal sources. *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 715–718.

[18] T. H. Lee, K. S. Donnelly, J. T. C. Ho, J. Zerbe, M. G. Johnson, and T. Ishikawa. A 2.5 V CMOS delay-locked loop for an 18 Mbit, 500 megabyte/s DRAM. *IEEE Journal of Solid-State Circuits*, 29(12):1491–1496, Dec. 1994.

[19] S. Aouini, K. Chuai and G. W. Roberts. Anti-imaging time-mode filter design using a PLL structure with transfer function dft. *IEEE Transactions on Circuits and Systems*, 59(1):66–79, Jan. 2012.

[20] T. E. Rahkonen and J. T. Kostamovaara. The use of stabilized CMOS delay lines for the digitization of short time intervals. *IEEE Journal of Solid-State Circuits.*

[21] P. Kumar, V. Kratyuk, G.Y. Wei and U.K. Moon. A sub-picosecond resolution 0.5-1.5 GHz digital-to-phase converter. *IEEE Journal of Solid-State Circuits*, 43 (2):414–424, Feb. 2008.

[22] E. M. Hawrysh and G. W. Roberts. An integration of memory-based analog signal generation into current DFT architectures. *IEEE Proceedings on Circuits and Systems*, 59(1):66–79, Jan. 2012.

[23] R. Jacob Baker. *CMOS Circuit Design, Layout, and Simulation.* IEEE Press, Piscataway, New Jersey, 2005.