

(VDL)²: A Jitter Measurement Built-In Self-Test Circuit For Phase Locked Loops

by

Brandon Ray Kam

Submitted to the Department of Electrical Engineering and Computer
Science in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer
Science at the Massachusetts Institute of Technology

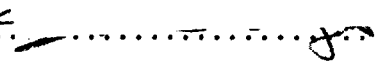
January 27, 2005


[February 2005]

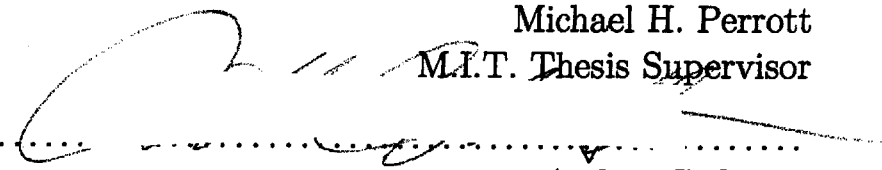
Copyright 2005 Brandon R. Kam. All rights reserved.

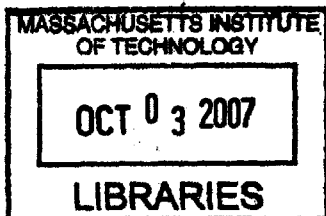
The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author
Department of Electrical Engineering and Computer Science
January 27, 2005

Certified by........
Stephen D. Wyatt
VI-A Company Thesis Supervisor

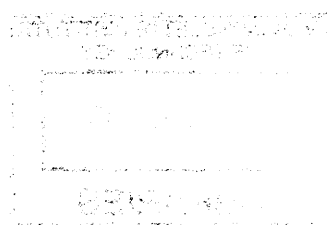
Certified by........
Michael H. Perrott
M.I.T. Thesis Supervisor

Accepted by........
Arthur C. Smith
Chairman, Department Committee on Graduate Theses



ARCHIVES

SEVENORA



(VDL)²: A Jitter Measurement Built-In Self-Test Circuit For Phase Locked Loops

by

Brandon Ray Kam

Submitted to the
Department of Electrical Engineering and Computer Science
January 27, 2005

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This paper discusses the development of a new type of BIST circuit, the (VDL)², with the purpose of measuring jitter in IBM's phase locked loops. The (VDL)², which stands for Variable Vernier Digital Delay Locked Line, implements both cycle-to-cycle and phase jitter measurements, by using a digital delay locked loop and a 60 stage Vernier delay line. This achieves a nominal jitter resolution of 10 ps with a capture range of +/- 150 ps and does so in real time. The proposed application for this circuit is during manufacturing test of the PLL. The circuit is implemented in IBM's 90 nm process and was completed in the PLL and Clocking Development ASIC group at IBM Microelectronics in Essex Junction, Vermont as part of the VI-A program.

VI-A Company Thesis Supervisor: Stephen D. Wyatt
Title: Senior Technical Staff Member, IBM

MIT Thesis Supervisor: Michael H. Perrott
Title: Assistant Professor, MIT

Acknowledgments

I extend my sincerest gratitude to all the people who have helped and supported me during this thesis project.

Thanks to my mentor Steve Wyatt for providing a guiding light during this project. His advice and suggestions in our weekly meetings were crucial to its success. To my campus thesis advisor, Professor Michael Perrott, for his prompt and helpful responses to my calls for assistance. To Ray Arnst, for his layout expertise which enabled the project to be completed on time. To all my IBM managers, mentors and co-workers for making my experience with IBM an enjoyable one.

To Andy Anderson, Professor Duane Boning, Professor Markus Zahn and Kathy Sullivan for their support of the VI-A program, specifically with IBM Microelectronics.

To my family and friends for their love and support, whether we're 5000 miles apart or right next door. To my uncle Steve, who has a greater influence on my life than he or I would probably admit. To my dad, who secretly wishes I was better at football than math, but supports me wholeheartedly with his love and encouragement. To my mom, who above all else is the reason I have been able to come so far.

Last but not least, thanks to Kristi Hamada, whose proofreading skills compensated for my diminishing command of the English language during my time at MIT. More importantly, her companionship and unconditional love throughout these past years have made them the best years of my life.

Contents

1	Introduction and Background	15
1.1	Motivation	15
1.2	Phase Locked Loops	16
1.3	Jitter	17
1.3.1	Phase Jitter	18
1.3.2	Period (RMS or Peak-to-Peak) Jitter	19
1.3.3	Cycle-to-Cycle Jitter	20
1.3.4	Long Term Jitter	21
1.4	Measuring Jitter	21
1.5	Jitter Measurement Techniques	22
1.5.1	Time to Amplitude, Analog to Digital Conversion	23
1.5.2	Dual-Slope Method	24
1.5.3	Delay Line Method	25
1.5.4	Vernier Delay Line	26
1.5.5	Dual Vernier Oscillator	26
1.5.6	Other Methods	27
1.6	Outline	28
2	Design Goals	29
3	Design	33
3.1	Modes of Operation	36
3.1.1	Phase Jitter	37

3.1.2	Cycle-to-Cycle Jitter	37
3.2	Delay Elements	38
3.3	Delay Locked Loop	39
3.3.1	Phase Detector	40
3.3.2	The Counter	41
3.3.3	The Delay Line	41
3.3.4	Theory of Operation	42
3.4	Vernier Delay Line	43
3.4.1	Data Storage	45
3.5	Data Acquisition	46
3.6	Control Block	46
4	Other Design Considerations	49
4.1	Variable Resolution and Post-Processed Calibration	49
4.2	Edge Cases and Centering	50
4.3	Metastability	52
4.4	Corrupt OutCodes	55
4.5	Implementing Internal Reset	56
4.6	Unwanted Jitter Generation	57
4.7	Layout	57
5	Results	59
5.1	Overall Results	59
5.1.1	DLL Performance	59
5.1.2	VDL Performance	61
5.2	Calibration	64
5.3	Power Dissipation	66
5.4	Chip Level Issues	67
5.5	Power Ring	68
5.6	Final Layout	68

6 Future Work and Conclusion	71
6.1 Future Work	71
6.1.1 Improve Delay Elements and Signal Matching	71
6.1.2 Improve Robustness to High Frequency Voltage Variation . . .	72
6.1.3 Offer More Types of Jitter Measurements	72
6.1.4 Wider Frequency Range	73
6.2 Conclusion	74
A Tables	75

List of Figures

1-1	Block diagram of a PLL	17
1-2	Example of phase jitter	18
1-3	Example of period jitter	19
1-4	Example of cycle-to-cycle jitter	20
1-5	Example of long term jitter	21
1-6	Time to amplitude and A/D conversion	24
1-7	Dual slope method	24
1-8	Standard delay line	25
1-9	Vernier delay line	26
1-10	Timing diagram for Vernier oscillator method	27
3-1	Block diagram of proposed (VDL) ² design	35
3-2	PLL and BIST signal routing	36
3-3	DLL waveform example	38
3-4	Block diagram of DLL	38
3-5	Schematic capture of phase detector	40
3-6	Timing diagram of phase detector	40
3-7	Schematic capture of counter	41
3-8	Diagram of delay line	42
3-9	Timing diagram of DLL	43
3-10	Schematic of Vernier cell	44
3-11	Example of Vernier cell	45
3-12	State transition diagram of control logic	47

4-1	Schematic capture of edge detection circuitry	52
4-2	Top: Current implementation, Bottom: Implementation using dividers	54
4-3	Flip flop module with embedded reset	56
4-4	Modified flip flop behavior	57
5-1	Entire DLL sweep	60
5-2	Left: DNL (in LSB) of the DLL, Right: INL (in LSB) of the DLL . .	61
5-3	Full sweep under three process corners	62
5-4	Three VDL sweep variations highlighting the resolution difference . .	63
5-5	Edge detection causing a delay change	64
5-6	Left: Delay of an element over circuit variations, Right: Delay differ- ence of two elements over circuit variations	65
5-7	Correlation between the two delays	66
5-8	Current draw for the $(VDL)^2$ over a complete test run	67
5-9	Final layout of $(VDL)^2$	69
5-10	Final layout of Sturgeon2, quadrants 3 and 4	70
6-1	Constant cycle-to-cycle jitter but increasing long term jitter	73

List of Tables

3.1	Possible TEST configurations	36
5.1	Relation between process variation number and environmental condition	65
A.1	Truth table for thermometer flip flop	75
A.2	Truth table for <i>OutCode</i> flip flop	75
A.3	Control logic truth table	76

Chapter 1

Introduction and Background

This Chapter provides the motivations behind the development of a new type of BIST circuit, as described in this thesis. It will discuss the forces that are driving jitter testing to on-chip solutions and will give an overview of the developed technologies.

1.1 Motivation

Phase locked loops (commonly referred to as PLLs) are found in a large number of computer, wireless, and communication systems with many different applications such as clock recovery, frequency synthesis and clock noise filtration. In the last several decades, a rapid increase in the speeds of digital circuitry have put strenuous demands upon PLL designers. One result of the fast increase in performance is that harmful noise parameters are becoming a relatively larger design problem. One such parameter, jitter, has quickly become one of the most critical parameters in specifying the operation of a PLL. As a result, extremely accurate analysis of PLL jitter characteristics is becoming required during production testing to ensure the PLL performs within its design specifications. With clock signals getting faster each year, the ability to measure jitter in the realm of picoseconds (ps) is becoming essential. However, in high volume production of PLLs there are very few mechanisms implemented to monitor jitter. Thus, PLLs could be shipped to clients who assume the circuit is within jitter specifications, when actually no concrete information has been developed

to either prove or disprove the fact.

As a result, designers have recently been looking towards built-in self-test (BIST) solutions. In a high volume production setting such as at IBM, BIST solutions can offer a low-cost, accurate, and fast solution in determining jitter, if designed correctly. This paper will describe a BIST circuit that can determine PLL jitter with under 15 picosecond accuracy over a wide input frequency range. The project was completed at the IBM Microelectronics site in Essex Junction, Vermont in the ASIC PLL and Clocking Development department, through the MIT VI-A program. The design effort was completed using IBM's 90 nanometer (nm) technology and required the use of Cadence Analog Design Environment, Spectre, PowerSpice, EriE Extraction, and MATLAB.¹

1.2 Phase Locked Loops

A basic phase locked loop contains several elements: a phase detector/comparator, a loop filter, a voltage-controlled oscillator (VCO), and an optional frequency divider [1]. A block diagram of a PLL is shown in Figure 1-1. The description that follows is intended to provide a very basic understanding of the PLL operation. The PLL uses negative feedback based on sensing the phase difference, via the phase detector, of the input frequency from a reference source and the feedback frequency from the VCO. The output of the phase detector generates voltage pulses that depend on which input's phase is leading (or lagging) the other.

For example, f_{in} might be operating at a slightly faster frequency than f_{fbk} , thus the *INC*, or increase, signal would be asserted. This *INC* signal is sent to the charge pump, which can be represented by two voltage controlled current sources, one that sinks current and one that sources current. In this case, the *INC* signal would turn the top (or sourcing) current source on. This current, I_{cp} , is sent into the loop filter and charges a capacitor. The loop filter makes sure that the voltage on the capacitor changes gradually and prevents the spiking attributes that *INC* and *DEC* exhibit

¹PowerSpice and EriE Extraction are IBM proprietary development tools.

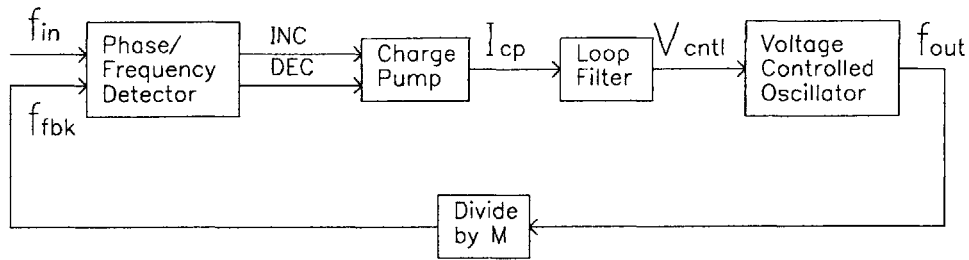


Figure 1-1: Block diagram of a PLL

through its low pass behavior. As the capacitor charges, the value of the control voltage, V_{ctrl} , slowly rises, and this is connected to the input of a voltage controlled oscillator. Since V_{ctrl} is increasing, the oscillation frequency will begin to increase as well, and this process continues until the feedback frequency exactly equals the input frequency.

The *divide by M* circuit shows one application of phase locked loops - the synthesis of a higher frequency than the input frequency. In this case, $f_{out} = M \times f_{fbk}$. When f_{fbk} is exactly the same as f_{in} , the PLL is said to be locked. If the input frequency varies within a certain tolerance, the phase detector will recognize the change and adjust accordingly such that the output frequency of the VCO still matches the input frequency.

1.3 Jitter

An agreement on the definition of jitter has historically been difficult to standardize. In regards to PLLs, it has vaguely been defined as the variation in period, frequency or phase of a signal as compared with its ideal value. However, as clock speeds have become much faster, jitter perturbations which have previously been insignificant are now beginning to affect circuit functionality. As jitter becomes an increasingly important metric in PLL performance specifications, ways of specifying different types of jitter have been created.

Here four different types of jitter will be discussed [2]. Each type of measurement

is pertinent to various types of applications. These jitter measurements are broadly defined in [3]. Jitter measurements like these give designers information such as how much valid cycle time they have to do their necessary computation.

1.3.1 Phase Jitter

Phase jitter refers to the deviation of the rising or falling edge of a data clock as compared to a reference clock and is depicted in Figure 1-2 below.

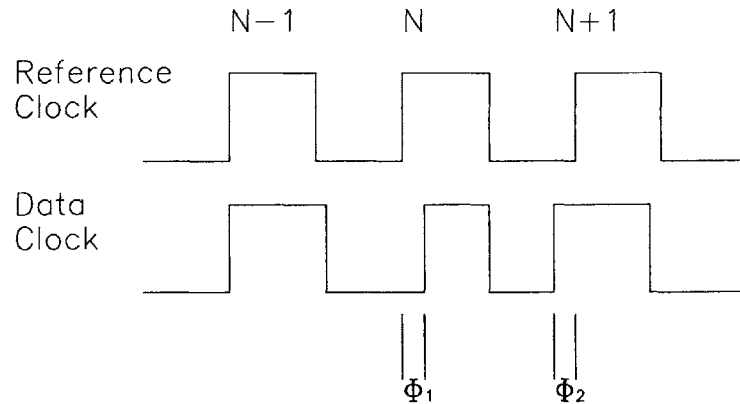


Figure 1-2: Example of phase jitter

Phase jitter is a measure of the static phase offset that occurs between the input and feedback signals of the PLL and is often what the phase/frequency detector is attempting to measure. Phase jitter is defined as

$$J(NT) = \Phi_1 - \Phi_{mean}$$

$$J((N + 1)T) = \Phi_2 - \Phi_{mean}$$

where $J(NT)$ refers to the phase jitter at cycle N , and Φ_{mean} refers to the mean value of the phase offset. Phase jitter is very important to quantify in applications such as communication systems. For example, too much phase jitter can often result in lost information in serial data transmission.

1.3.2 Period (RMS or Peak-to-Peak) Jitter

Period jitter refers to the deviation of each period with respect to its ideal or average period and is depicted in Figure 1-3 below.

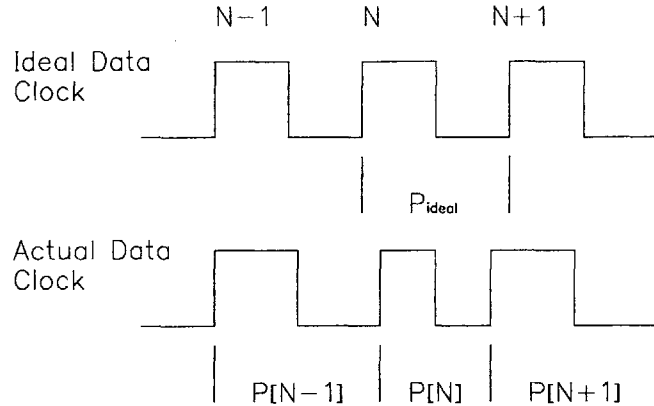


Figure 1-3: Example of period jitter

The period jitter of a particular cycle is defined as

$$J_p[N] = P[N] - P_{ideal}$$

where $J_p[N]$ is defined as the period jitter at cycle N and P_{ideal} is defined as the ideal period of the data clock.

To obtain a good idea about the characteristics of the period jitter, often many samples of a period must be taken. Two types of period jitter, RMS or peak-to-peak, can be extracted from this data. RMS jitter is defined as the standard deviation of the period data. Assuming you know the number of samples N and the period values $P[n]$, the RMS jitter value can be determined fairly easily.

$$P_{avg} = \sum_{n=1}^N \frac{P[n]}{N}$$

$$J_{rms} = \sqrt{\sum_{n=1}^N \frac{(P[n] - P_{avg})^2}{N}}$$

P_{avg} is defined as the average period of the set of samples of $P[n]$, and J_{rms} is the RMS jitter. Peak-to-peak jitter is defined as the largest swing in period time possible and is simple to calculate since there is a finite sample size.

$$J_{pp} = \max \left(P[1], P[2], \dots, P[N] \right) - \min \left(P[1], P[2], \dots, P[N] \right)$$

Here J_{pp} is defined as the peak-to-peak jitter of the set of samples of $P[N]$.

Note that period jitter data can also be easily related to phase jitter. The relation is $J_p[N] = J((NT) - J((N - 1)T))$ where $J_p[N]$ is the period jitter at interval N .

1.3.3 Cycle-to-Cycle Jitter

Cycle-to-cycle jitter refers to the deviation in period length of adjacent cycles of a signal and is depicted in Figure 1-4. Cycle-to-cycle jitter is calculated by using the definition

$$J_{cc}[N] = P[N] - P[N - 1]$$

where $J_{cc}[N]$ is defined as the cycle-to-cycle jitter at sample n . In this example, the clock has an ideal frequency of 1 GHz with a period of 1 ns. During cycle N we see that the period is 1.05 ns and during cycle $N + 1$ we see that the period is 0.95 ns. Thus, the cycle-to-cycle jitter for this cycle is -100 ps. However, in cycle $N + 2$ the period is 1.05 ns, so the cycle-to-cycle jitter for this cycle is +100 ps. Therefore, the overall cycle-to-cycle jitter for this observed portion of the waveform is +/- 100 ps.

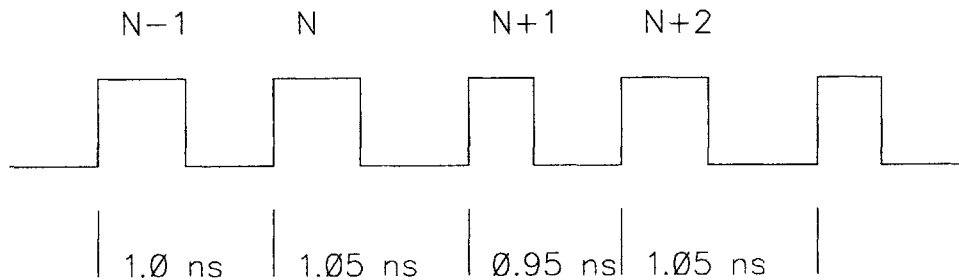


Figure 1-4: Example of cycle-to-cycle jitter

1.3.4 Long Term Jitter

Long term jitter gives an indication of the variation of a rising or falling edge compared to its ideal position after many cycles of measurement and is depicted in Figure 1-5. This parameter can be a very important metric in many applications, for example in PLLs when a large divider is used to synthesize a much higher output frequency than the input frequency. Long term jitter is defined as

$$J_{lt} = \sum_{n=1}^N (P[n] - P_{ideal})$$

where J_{lt} is the long term jitter.

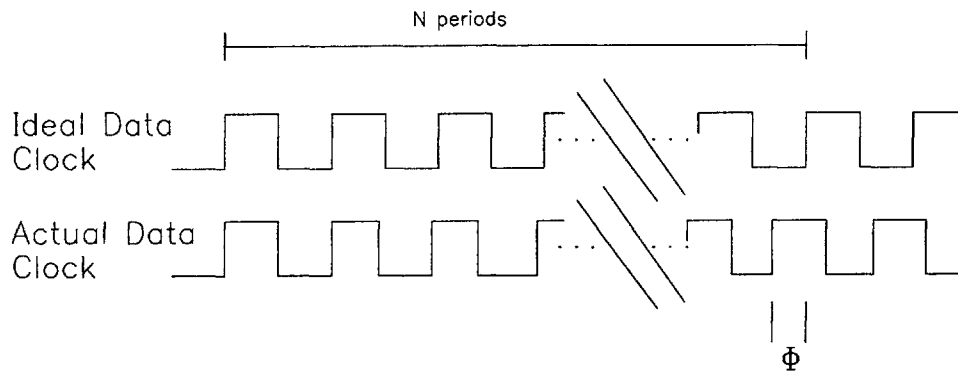


Figure 1-5: Example of long term jitter

1.4 Measuring Jitter

The first step in solving any problem is being able to accurately characterize it. Thus, the first step in solving jitter issues in designs is first being able to accurately measure it. For a 100 MHz clock, a jitter measurement with 100 ps accuracy offers 1% resolution, but for a 1 GHz clock the same measurement only gives a 10% measurement resolution. Thus, a major issue is developing jitter measurement techniques that have resolution in the order of picoseconds.

There are several methods that have conventionally been used to make jitter measurements [4].

1. High Speed Sampling Oscilloscopes
2. Automated Test Equipment (ATE)
3. Time Interval Analyzers
4. Dedicated Hardware

These are all off-chip methods that use various techniques to effectively measure jitter. However, there are several downsides to these and other off-chip techniques. The first is that as signals get faster, achieving high resolution measurements off-chip become harder and more expensive. Noise in the signal wires from the chip to its associated tester can significantly degrade the achievable precision. Another drawback is that for high volume production testing, these methods become prohibitively expensive or simply too time-consuming to setup. The last reason is that these methods do not offer an easy way to obtain access to other internal nodes of the PLL, which could prove useful in obtaining jitter measurements.

Thus, the new trend in jitter measurement has been towards on-chip devices, referred to as built-in self-test (BIST) circuitry. BIST circuits can be physically very close to the circuit under test and can have very easy access to internal nodes. BIST circuits can also lower testing cost compared to the aforementioned methods, as testing time can be accelerated and no additional testing hardware would be required. Unfortunately, BIST solutions have previously been avoided due to their high silicon overhead as well as poor measurement capabilities. However, due to the dramatic reduction of transistor sizes and new innovations in jitter measurement techniques, BIST solutions are now becoming a viable alternative to measuring jitter.

1.5 Jitter Measurement Techniques

There are many different ways of measuring jitter, but all methods involve transforming time data into an analog or digital signal. Thus, the concept of time interval measurement is central to jitter measurement techniques. In this type of design, there

is always a tradeoff between the circuit precision, silicon area, and testing time. Thus, each method must be analyzed to ensure that the optimal choice is made. The most primitive time interval measurement technique is edge counting [5]. This method involves triggering a high frequency clock at the start of the interval and counting the number of transitions until the end of the interval. To further explain the idea let us assume a 100 MHz clock is used as the reference and assume that we are only counting the rising edge transitions. If the count is 7 after the interval has ended, we can calculate the period length to be $7 \times \frac{1}{10^8} \text{s} = 70 \text{ ns}$. However, a severe limitation of this method is that the highest achievable resolution is at most half the period length (if both rising and falling transitions are counted), which in this case is 5 ns. Several other clever time interval measurement techniques have been developed for BIST implementation that offer significantly improved resolution.

1.5.1 Time to Amplitude, Analog to Digital Conversion

This method is an analog technique that relies upon the charging of a capacitor, then a conversion of the voltage on the capacitor to a digital value [6]. An illustration of this method is shown in Figure 1-6. At the start of the time interval, a voltage controlled current source will turn on and begin to charge a capacitor. If the voltage is held constant, the current will also be constant and the capacitor voltage will charge according to its state equation

$$I = C \frac{\Delta V_c}{\Delta t}$$

$$\Delta V_c = I \frac{\Delta t}{C}$$

Once the time interval ends, the voltage input goes to zero and the current also goes to zero, thus stopping the charging of the capacitor. Then an analog to digital converter samples the capacitor voltage and outputs a digital code based on the value. The accuracy of this code depends primarily on the linearity of the current source and the resolution of the analog to digital converter. This method has good precision and moderately fast testing time, but suffers from analog layout issues and difficulty in

component calibration.

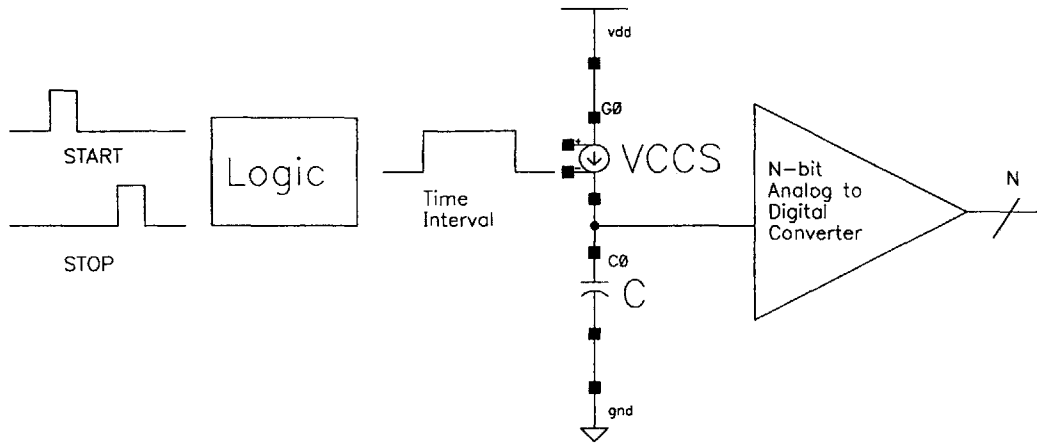


Figure 1-6: Time to amplitude and A/D conversion

1.5.2 Dual-Slope Method

A method similar to the time to amplitude converter is the dual-slope method [7, 8, 9]. An illustration of this method is shown in Figure 1-7.

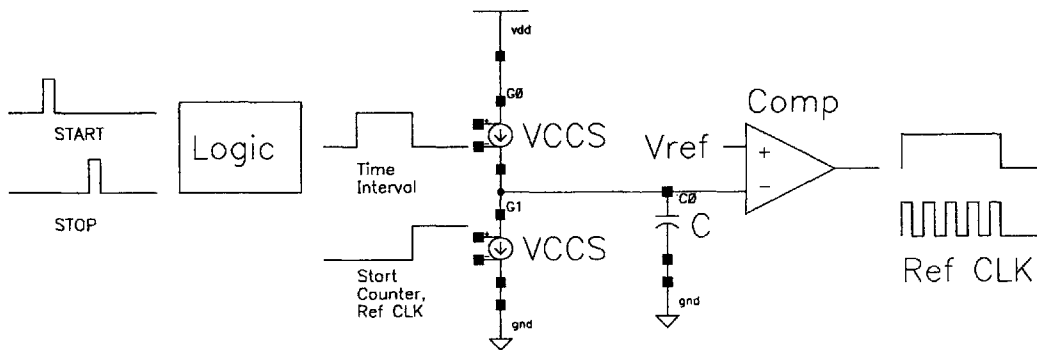


Figure 1-7: Dual slope method

In the Figure there are two voltage controlled current sources, one to charge the capacitor and one to discharge the capacitor. The discharging rate is some exact fraction of the charging rate. Once the discharging current source goes active, a

high frequency counter starts. The counter stops when the output of the comparator switches, signifying the voltage on the capacitor is back to its original value. The possible resolution of this method is the period of the high frequency reference divided by the current ratio. This method has good precision and a moderately small area, but suffers from long testing times.

1.5.3 Delay Line Method

An illustration of the delay line method is shown in Figure 1-8. This method uses a string of delay elements triggered at some point in time, usually at the start of the time interval [9, 10, 11, 12]. The signal is then sent through a number of delay elements, and the output of each delay element is compared with a reference signal, usually representing the end of the time interval. In this manner, a thermometer code is generated that shows at which delay element the signals no longer overlap. Since the delay through each element will be known, it is simply a matter of counting the number on the thermometer code and multiplying it by the delay of each element to determine the time measurement. This method offers fast testing time, but its precision is limited by the delay of a delay element and it has a large area requirement because of all the delay cells required.

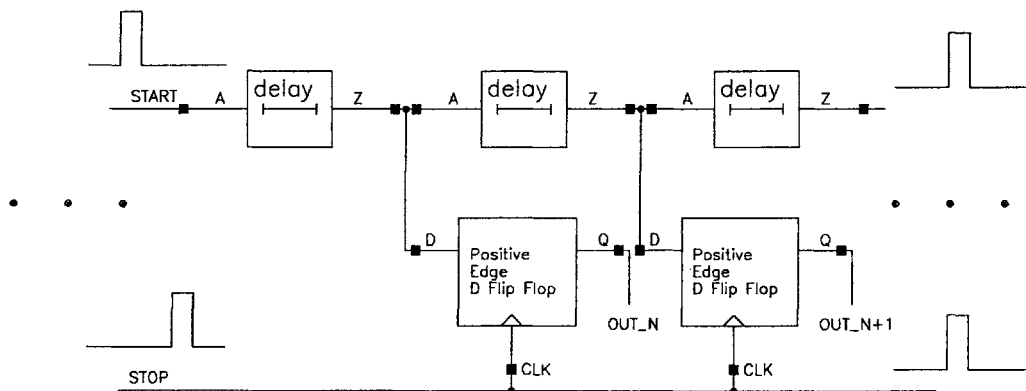


Figure 1-8: Standard delay line

1.5.4 Vernier Delay Line

If additional resolution is required, the Vernier interpolation principle has proven to be a successful method for achieving sub-gate delay resolution [13, 14]. The Vernier delay line method relies on the small delay differences in two elements. An illustration of the Vernier delay line is shown in Figure 1-9. The delay elements in the STOP signal path are shorter than the elements in the START signal path. Thus, although START is triggered earlier than STOP, eventually the STOP signal catches up. A thermometer code can also be used to determine when this occurs, which can be formed at the output of each register stage. In this situation, the maximum possible resolution attainable is $\tau_\alpha - \tau_\beta$. This method has fast testing time and very good precision, but like the delay line method it suffers from large area overhead.

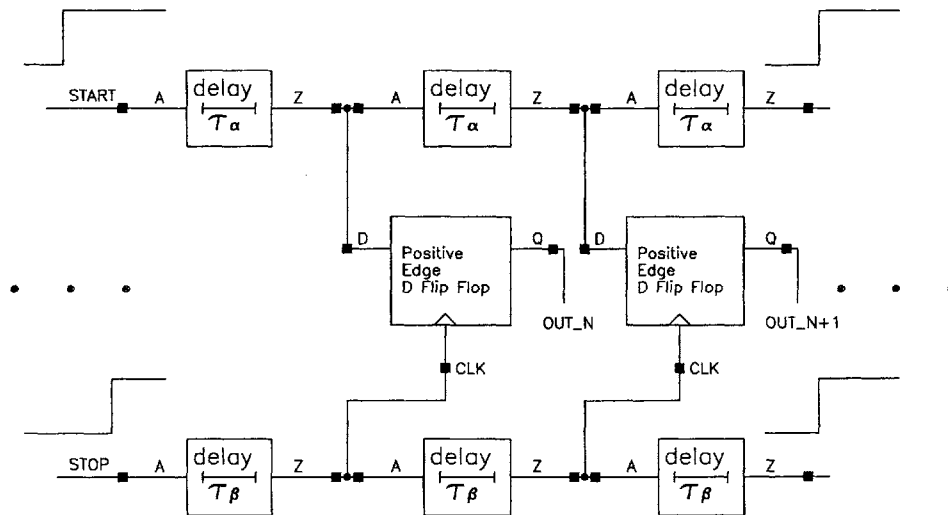


Figure 1-9: Vernier delay line

1.5.5 Dual Vernier Oscillator

The dual Vernier oscillator system is another clever way to achieve sub-gate time measurement resolution [4, 15]. A diagram of the theory of operation is shown in Figure 1-10. There are two oscillators used for the measurement, and the circuit relies on the small difference in frequency of the oscillators. The slower oscillator is

triggered by the start of the time interval, and the faster oscillator is triggered by the end of the time interval. When the fast oscillator catches up to the slower one, the time interval can be accurately calculated. If we define the period of the fast oscillator as T_β , the period of the slow oscillator as T_α , the number of rising edges of the fast oscillator as A and the number of rising edges of the slow oscillator as B , then the value of the time interval τ is $\tau = A \times T_\alpha - B \times T_\beta$. This method has good precision and a decently small area, but suffers from long testing time, because each measurement requires many cycles of the oscillators.

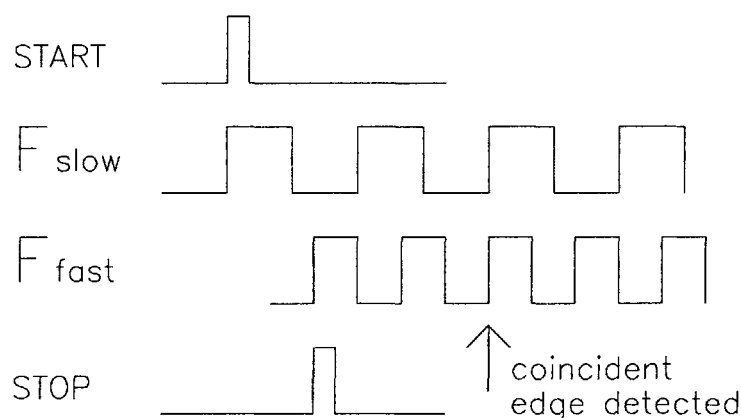


Figure 1-10: Timing diagram for Vernier oscillator method

1.5.6 Other Methods

Of course, there are many ways to implement a jitter measurement system besides the ones described. [16] attempts to determine RMS jitter by manipulating a variable delay element and statistical characteristics of the jitter profile. [17] proposes a method to measure multiple periods per sample to reduce computation time and circuit complexity. [18] uses undersampling to ease the speed and area requirement on the BIST circuitry. Each method offers its own particular advantages and disadvantages.

1.6 Outline

This Chapter dealt with the motivations behind developing an on-chip BIST circuit to measure jitter, as well as discussed various definitions of jitter and ways that have been developed to measure them. Chapter 2 discusses the design goals for the project. Chapter 3 explains the design of the BIST circuit. Chapter 4 talks about design complications that arose during the development process. Chapter 5 presents the simulation results for the BIST circuit. Finally, Chapter 6 provides some concluding remarks and offers ideas for future work.

Chapter 2

Design Goals

In the design of the BIST circuit, there were several goals that guided the design effort towards the end product, which are discussed here.

- Resolution of < 15 picoseconds.

The ability to resolve time differences on the order of picoseconds is becoming crucial for development in modern technologies. As discussed earlier, picosecond resolution is already essential for effectively characterizing jitter measurements. Thus, a design goal for the jitter measurement BIST circuit is to achieve a time resolution of less than 15 ps, which is less than the delay of an inverter gate in IBM's 90 nm process. The maximum amount of jitter in the input signal should be no more than ± 150 ps.

- Offers more than one type of jitter measurement.

For the jitter measurement BIST to be a viable option for designers looking to test their chips, the circuit must offer several types of jitter measurements. Depending on the application, one type of jitter measurement may be more meaningful than another. Thus, developing a circuit that could reuse some of its resources to test different types of jitter with minimal overhead would be a very desirable trait and have the broadest appeal. A major drawback

of many of the previously implemented BIST solutions is that none of them have been able to offer real time period or cycle-to-cycle jitter data because of the fast signals they are trying to measure [2]. As a result, most real-world methods make a measurement of long-term accumulated jitter, to ease the timing requirement on the BIST circuitry. This circuit will measure both phase jitter as well as cycle-to-cycle jitter.

- Functional for a wide range of input frequencies.

The jitter measurement BIST circuit must be functional for a wide range of input frequencies. In a testing situation, a designer often wants to know how the PLL performs at various frequencies, and the design of the jitter measurement BIST circuit must be robust enough to handle this request.

- Relatively small silicon area and fast testing time.

It was stated earlier that several reasons BIST implementation has not been widely used is because they are either too big or take too long. Both time and area mean money in VLSI design. Ideally, the jitter measurement BIST circuit will take up only a fraction of the area that the PLL will take, so the size of the BIST circuit will not greatly affect the amount of PLLs that could be produced on a single die. Some previous BIST methods have taken areas on the order of square millimeters, which is unacceptable in terms of high volume production. Also, the testing time for the circuit should be fast, ideally on the order of microseconds. The jitter measurement BIST circuit may be required to operate several times per PLL under test, perhaps testing the circuit at multiple frequencies. In high volume production, long testing times can become a bottleneck by increasing chip cost due to the need of additional testers to meet a specific deadline. Thus, testing times should be minimized as much as possible.

- First pass usable silicon

Rigorous simulations and checks are done at all levels of the design process in an attempt to ensure correct performance from the manufactured silicon. Unfortunately, the manufactured silicon does not always perform as intended. This project is approaching this issue in two ways. First, the hope is that the BIST circuit itself will be functional after the first manufacturing pass. Since the jitter measurement BIST circuit will be fabricated for testing, a lot of effort will be devoted to functionality at the schematic level as well as the layout extracted netlist level. Design, simulation, and redesign are hammered out until the product is ready for production. Second, the BIST circuit will hopefully help IBM's PLLs achieve a higher first manufacturing pass success rate. If IBM can successfully implement the jitter measurement BIST circuit with the production of its PLLs, then IBM can test jitter in these PLLs before shipping them out to clients. Thus, IBM can ensure that its PLLs are within the design specifications, and thus have an easier time ensuring quicker turnaround to its clients.

The items discussed in this Chapter are designated as five major goals for the BIST circuit to accomplish. The next two Chapters will delve into the design of the BIST circuit and show how these goals are addressed.

Chapter 3

Design

This Chapter discusses the design of the BIST circuit by describing the design and theory of operation of each component in the new architecture. It specifically discusses the two major blocks involved in the BIST circuit, the digital delay locked loop and the Vernier delay line. These blocks act as the coarse tune and fine tune portions of the BIST circuit, respectively. Figures and schematics are included to help in the understanding of the design.

The name of the circuit is (VDL)², which is an acronym that stands for **VVDDLL** - **V**ariable **V**ernier **D**igital **D**elay **L**ocked **L**ine. Variable because the resolution of the circuit changes from chip to chip (which will be explained later). Vernier Delay Line because the primary method implemented in this project involves an application of the Vernier delay line (VDL) method. Digital because the circuit is created in a straightforward digital CMOS implementation. Delay Locked because the circuit utilizes a delay locked loop (DLL).

This method was chosen over the analog methods because it offers several distinct advantages. One is that implementing the Vernier method offers the possibility of an all-digital design, which is promising for reusability and technology migration. The second is that careful layout considerations are much more important in the capacitor charging methods than in the Vernier method. The third is that with the Vernier method, incorporation into a test methodology such as level sensitive scan design

(LSSD) test is a fairly simple task.

The biggest problem with the VDL method is the huge area overhead associated with its implementation. If measuring period jitter is the objective, with a Vernier resolution of α ps, and a period of T , the Vernier chain would need to be at least $\frac{T}{\alpha}$ cells long. For example, if this method were used to measure the period jitter of a 100 MHz signal with a resolution of 10 ps, it would take over 1000 Vernier delay blocks to simply delay a signal long enough to obtain a meaningful period measurement! This is simply unacceptable. Some designs have tried to overcome this limitation by implementing two stages of time measurement, a coarse measurement and a fine measurement, such as in [19]. However, these methods still suffer from several drawbacks. Many of these methods require an unacceptably large area, so although they function well, they are impractical to use in industry. Most of them also require the use of an external jitter-free clock as a reference, which either is inconvenient or impossible to obtain during testing, may be corrupted by the I/O, and may not actually be jitter-free. A last concern is that it is difficult to calibrate the Vernier delay line unless there are external signals available with a known delay difference during the test. [20] has recently developed a method to provide better calibration, but this method has yet to be developed beyond software simulations. [21] has implemented a clever technique of two cascaded delay lines to eliminate the need for an external clock, but this technique still suffers from large area overhead in its coarse tune and storage elements and difficult real world calibration. It also has not been developed beyond simulations.

Thus, (VDL)² has been developed with these limitations in mind, and a simplified block diagram is shown in Figure 3-1. (VDL)² uses a custom digital DLL as a coarse tune and an implementation of the VDL concept as a fine tune. Some of the key functionality specifications are:

1. **The circuit requires no external clock.** Because it requires no external clocking circuitry, a lot of testing complexity and noise issues can be avoided. The circuit acquires the only clock it needs from the PLL and does all its measurements using that signal.

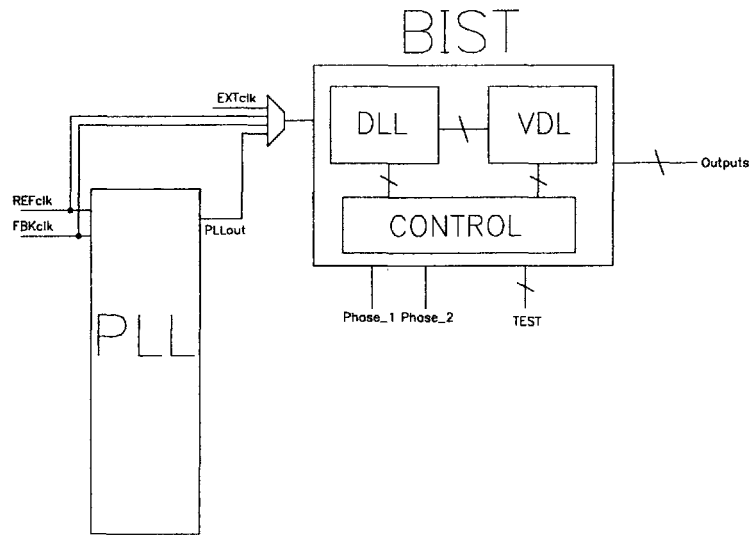


Figure 3-1: Block diagram of proposed (VDL)² design

2. **It can accept frequency inputs over the range of 100 MHz - 500 MHz.**
3. **Variable, post-processed resolution determination solves the problem of calibration as well as the overhead associated with it.** Post-processed resolution determination is achieved by the clever use of the coarse tune circuit and a matching of the elements in the coarse tune and the fine tune portions of the circuit. The issue of calibration will be discussed further in Section 4.1.
4. **Low jitter generation.** It is important that the jitter measurement circuit adds a minimal amount of jitter to the PLL signal. If the jitter measurement circuit adds too much jitter, it corrupts the data, and when reading the output it would be unclear whether the data represents the jitter in the PLL signal or the BIST circuit jitter.
5. **Small area overhead on the PLL.** The footprint of this BIST circuit is around 5-10% the area of a single PLL, which is fairly small. Also, the area overhead can be amortized by the fact that the BIST circuit can be used to test multiple PLLs on a chip via the introduction of a simple mux.

The BIST circuit was deployed on an ASIC test site. The initial application of this circuit will be during manufacturing test of the PLL. During the manufacturing test, the only indicator concerning the performance of the PLL is the lock indicator, so the hope is that this BIST circuit will be able to give significantly more information during that time.

3.1 Modes of Operation

There are several different modes of operation for the BIST circuit, controlled by test bits and some decode logic. The BIST circuit makes two types of jitter measurements, phase jitter and cycle-to-cycle jitter measurements. Since this circuit has been deployed on a test site, it was important to provide external inputs to be able to fully characterize the circuit. The TEST signal configuration in Table 3.1 shows which setting corresponds to which type of measurement, and Figure 3-2 illustrates the overall setup.

TEST	Type of Measurement
000	<i>PLLout</i> cycle-to-cycle jitter
001	<i>FBKclk</i> cycle-to-cycle jitter
010	<i>REFclk</i> cycle-to-cycle jitter
011	External cycle-to-cycle jitter
100	<i>REFclk</i> - <i>FBKclk</i> phase jitter
110	External phase jitter

Table 3.1: Possible TEST configurations

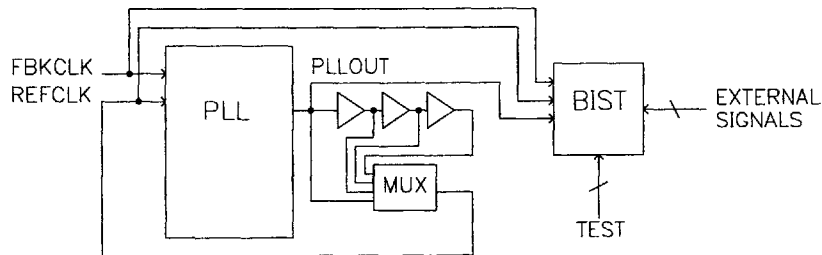


Figure 3-2: PLL and BIST signal routing

3.1.1 Phase Jitter

Implementation of a phase jitter measurement primarily utilizes the VDL. The signals used to make this measurement are *FBKclk* and *REFclk* as depicted in Figure 3-2. The variation of the phase difference between the rising edges of these signals denotes the phase jitter of the system. These signals are sent to the VDL after some delay logic that centers the output code in the middle of the chain. The basic operation of the VDL was described in Section 1.5.4. Both *FBKclk* and *REFclk* signals propagate down the Vernier chain for as many periods of the *PLLout* clock as determined by the tester. The data is accumulated for peak-to-peak, cycle-to-cycle jitter data using “sticky logic,” which is inspired by the Skitter Macro [22]. Then, after the data is extracted by the test engineer via the LSSD test methodology, the system is reset and put on idle.

3.1.2 Cycle-to-Cycle Jitter

Recall from Section 1.3.3 that cycle-to-cycle jitter refers to the difference in adjacent period lengths of a signal. The cycle-to-cycle jitter measurement is implemented in two stages. The BIST circuit takes either *PLLout*, *REFclk*, or *FBKclk* as the input, depending on the value of the TEST bits. This input is sent to the first stage, the digital DLL, which delays the signal to almost 180° out of phase and outputs *DLLout*. An example of this type of measurement is shown in Figure 3-3. It is done using bang-bang phase detection logic, a digital counter, and an adjustable delay line, as shown in the basic diagram in Figure 3-4. Once *DLLout* has the appropriate amount of delay, the DLL becomes locked, and triggers the second stage, the VDL. The DLL signal and the inverted input signal both propagate down the Vernier chain in the same manner as *FBKclk* and *REFclk* in the previous section.

Note that it may seem as if this circuit operates on duty cycle jitter as opposed to cycle-to-cycle jitter, because of the 180° shift instead of a 360° shift. Because the VCOs in IBM’s PLLs operate very quickly, a divide by two is incorporated within the VCO circuit to slow down the oscillation. Thus, although it may seem as if the BIST

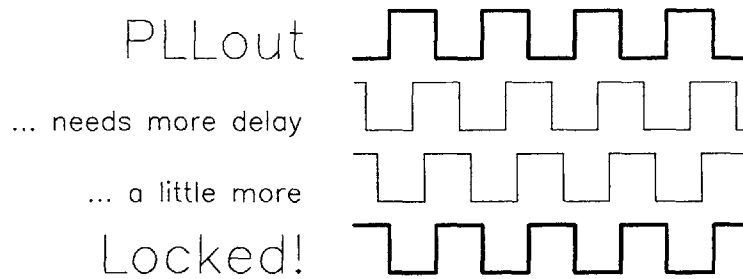


Figure 3-3: DLL waveform example

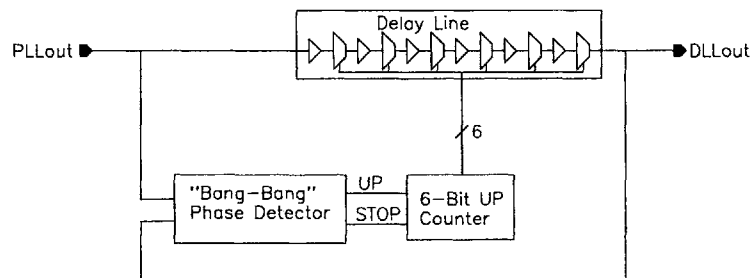


Figure 3-4: Block diagram of DLL

circuit is only measuring duty cycle jitter, since the actual period of the VCO output is half that of *PLLout*, this technique ultimately does measure cycle-to-cycle jitter.

3.2 Delay Elements

The delay elements are critical to the performance and precision of the BIST circuit. As shown in Chapter 1, the resolution of the circuit depends upon a delay mismatch in the two signal paths. To create that mismatch, differently sized buffer delays were created using a cascade of active inverters. Ideally, the delay blocks would be comprised of elements that are more insensitive to process/voltage/temperature variations, but there are a couple of reasons that this is unnecessary for the circuit. The first is that, as described above, the initial application of the BIST circuit will be during manufacturing test. In this scenario, the PLL is tested while the rest of the digital circuitry remains in an idle state. The digital VDD and GND will be

very clean since the circuits that normally load them down will not be switching and since the PLL operates on its own power supply. Thus, we could consider connecting the BIST power supply to the digital power supply without much concern about VDD noise or GND bounce, and thus can get a reliable and repeatable response from these delays. Another reason is that the primary purpose of this circuit is proving the concept in hardware. Once we know that (VDL)² works, we can substitute the current delay elements for any other type of active delay elements, providing we still get a reliable difference between two delays. The last and most practical reason was the time constraints imposed on this project.

3.3 Delay Locked Loop

The delay locked loop used in this circuit is much simpler than those found as commercial stand-alone units because of the area requirement and the specific function it performs. The DLL is implemented in a self-timed, all digital fashion to eliminate delay perturbations once locked. In analog DLLs, there is often a charge pump and low pass filter that connects to a voltage controlled delay element (similar to the PLLs described in Section 1.2). However, since the charge pump still charges and discharges even after the lock condition is reached, there would always be fluctuations of the voltage on the capacitor. This situation leads to fluctuations of the delay values and consequently a jittery signal. A digital DLL does not suffer from this jitter problem because once the lock condition is reached, the system truly exhibits a constant delay. In other words, a typical DLL will continue to make corrections even after the lock condition is reached, resulting in quantization jitter. The DLL in this design freezes the DLL once locked, so this quantization error is eliminated. Many digital DLLs in commercial parts use complex techniques to achieve very accurate locks, such as phase blending or using multiple delay lines. However, this extra complexity and area cost is unnecessary for this application, because the DLLs centering ability is sufficient as a coarse tune.

As shown in Figure 3-4, the DLL has three major components, the phase detector,

the counter, and the delay line.

3.3.1 Phase Detector

The phase detector is implemented with simple bang-bang logic. A schematic of the phase detector is shown in Figure 3-5. A single rising edge D flip flop is used to determine whether or not *DLLout* requires more delay. The *CLK* input to the flip flop is the inverted PLL signal, denoted as *PLLout.B*, while the *D* input is simply the output of the delay line, *DLLout*. The PLL signal is inverted because the phase detector is looking for a 180° lock, as described in Section 3.1.2. Until the rising edge of *DLLout* occurs after the rising edge of *PLLout.B*, the *UP* signal will continue to pulse. A timing diagram of this is depicted in Figure 3-6.

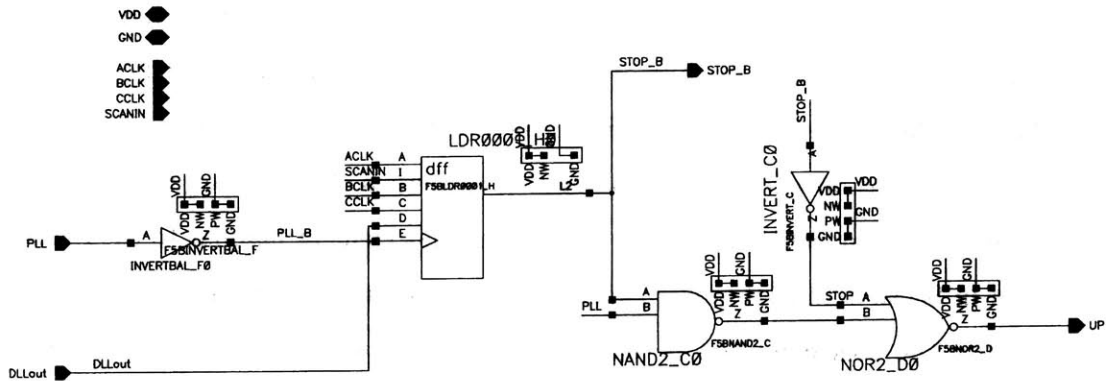


Figure 3-5: Schematic capture of phase detector

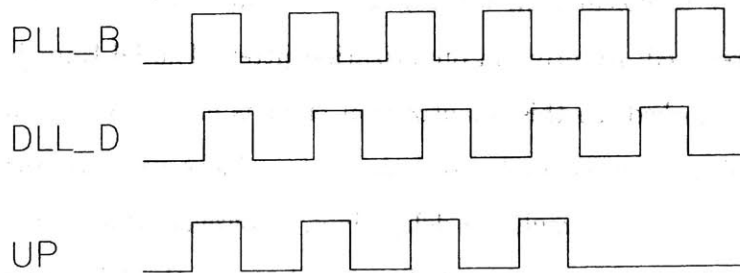


Figure 3-6: Timing diagram of phase detector

3.3.2 The Counter

The counter is a simple synchronous 6-bit up counter with RESET and HALT, implemented as shown in Figure 3-7. Each module represents a one-bit adder. The counter has three modes of operation. The first mode is the resetting of the counter. The D flip flops used in this design do not have internal set/reset functionality (but do have LSSD test capability), so the counter was designed with this in mind. The reset is enabled via a muxing of the clock and data inputs, which is decided by the control logic. The second mode is the count mode, used when the DLL is in operation. The circuit counts incrementally from 0 to 63 and adds the appropriate amount of delay depending on the count. The third mode is the halt mode, which occurs when the DLL has locked onto the input signal and the VDL is in operation. This mode forces the counter to hold its outputs, thus the counter has logic to prevent them from incrementing, even if there is an *UP* pulse indicating that it should do so.

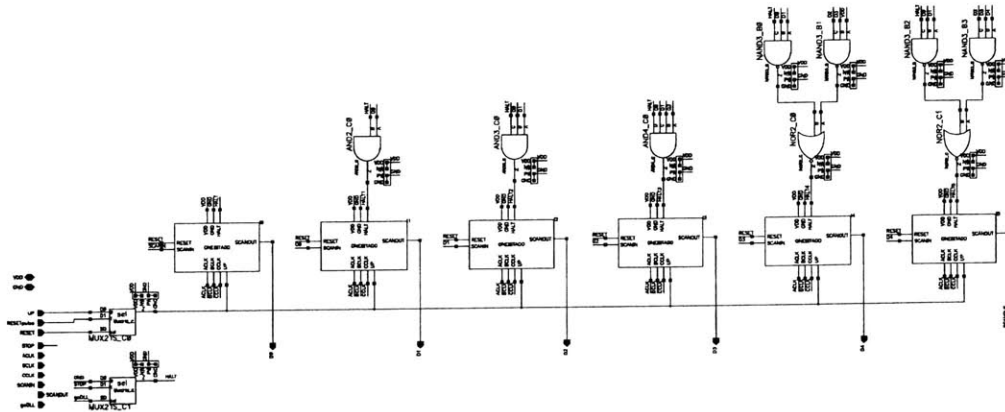


Figure 3-7: Schematic capture of counter

3.3.3 The Delay Line

The delay line is a 64-tap binary weighted delay element. The amount of delay depends upon the 6-bit count output of the counter, and a diagram that illustrates the operation of the delay line is shown in Figure 3-8. For the sake of simplicity, let us assume that the delay of each delay element has a unit value and that the wire

and propagation delay in the muxes are negligible. The delays are created by using mux elements which are controlled by the counter outputs. Looking at Figure 3-8, we see that the mux which chooses between a delay of 32 and a delay of 0 is controlled by the MSB of the counter output, D5. We see that the next mux chooses between a delay of 16 and a delay of 0, and is controlled by D4. This pattern continues until the signal finally propagates out of the last mux.

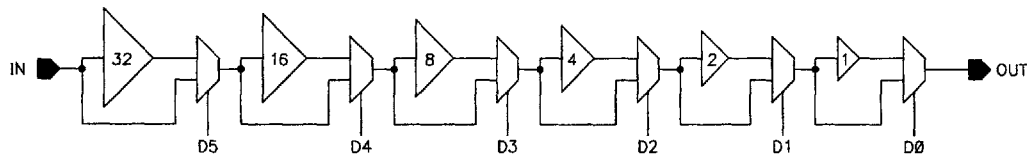


Figure 3-8: Diagram of delay line

Note that in actuality, the propagation delay of the muxes as well as the intrinsic wire delays can contribute significantly to the overall delay in the delay line. As will be shown later, for calibration purposes it is crucial that the delay introduced in the delay line is a result of only the delay elements. Thus, the propagation delay of the muxes are matched in the PLL signal path, and careful layout must be considered as well.

3.3.4 Theory of Operation

The timing diagram in Figure 3-9 illustrates the operation of the DLL. The delay through the delay line is initialized to zero to eliminate the problem of locking on the wrong edge, for example locking 180° or 360° away from the desired target. When the control circuitry enables the DLL, *DLLout* is slightly delayed with respect to the input, *PLLout*. At this time, the phase detector realizes that *DLLout* requires more delay so it begins sending *UP* pulses to the counter, which controls the delay line and increases the amount of delay between *DLLout* and *PLLout*. The *UP* signal continues to pulse until *DLLout* is almost 180° out of phase with respect to *PLLout*. When this occurs, the *UP* signal will stop pulsing, and a *STOP* signal is then asserted and sent to the control unit indicating the DLL has locked. In the

diagram, note that *STOP* is active low. As mentioned earlier, the reason the DLL is completely frozen once locked is to eliminate the quantization jitter that occurs in typical DLLs. Although this reduces the accuracy of its lock, the lock is still sufficient, since the DLL operates as a coarse tuner for the (VDL)².

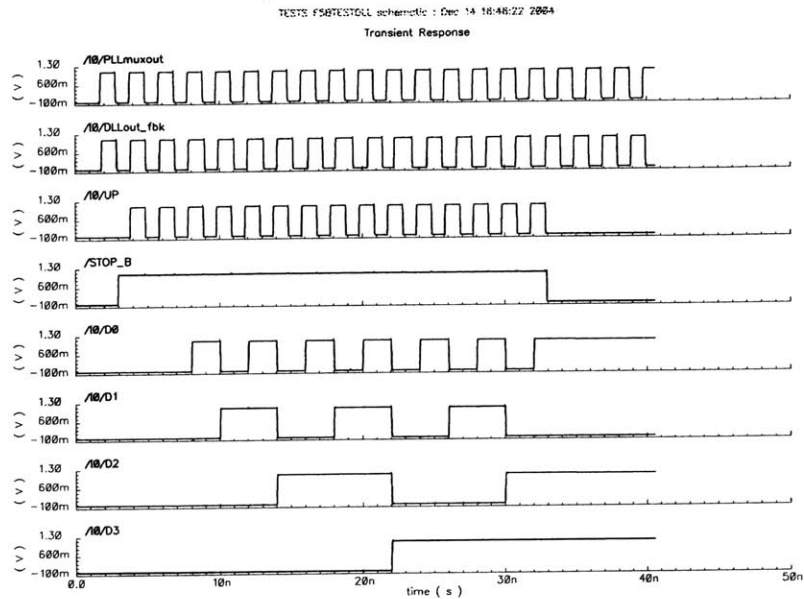


Figure 3-9: Timing diagram of DLL

3.4 Vernier Delay Line

The VDL is comprised of 60 Vernier cells connected serially. As described in Section 1.5.4, the Vernier method offers a resolution that is the difference between two delay cells. However, unlike the diagram in Figure 1-9 which requires a *START* and *STOP* signal, the VDL implemented in this design operates on two clock signals, *PLLout_B* (the *top* signal) and *DLLout* (the *bot* signal). Note that the DLL locks at 180° out of phase, which is why the inversion is required.

A schematic of the Vernier cell is shown in Figure 3-10. The two delay elements are contained within the block at the top of the figure. The *top* signal is sent into the *CLK* input of a flip flop, and the *bot* signal is sent into the *D* input. There are

two flip flops in this circuit, and the truth table logic for this circuitry can be found in Appendix A.1 and A.2. The output of this flip flop will generate a 1 if the rising edge of *bot* occurs before *top* and will generate a 0 if the rising edge of *top* occurs before *bot*. When these cells are chained together, it is evident that this configuration will generate a thermometer code for each pair of rising edge transitions. If there is a jittery input signal, this will cause a change in the temporal location of the rising edge of the *top* signal and will lead to a change in the temporal location of the transition point in the thermometer code. Measuring the change in this transition point gives information about the amount of jitter in the input signal.

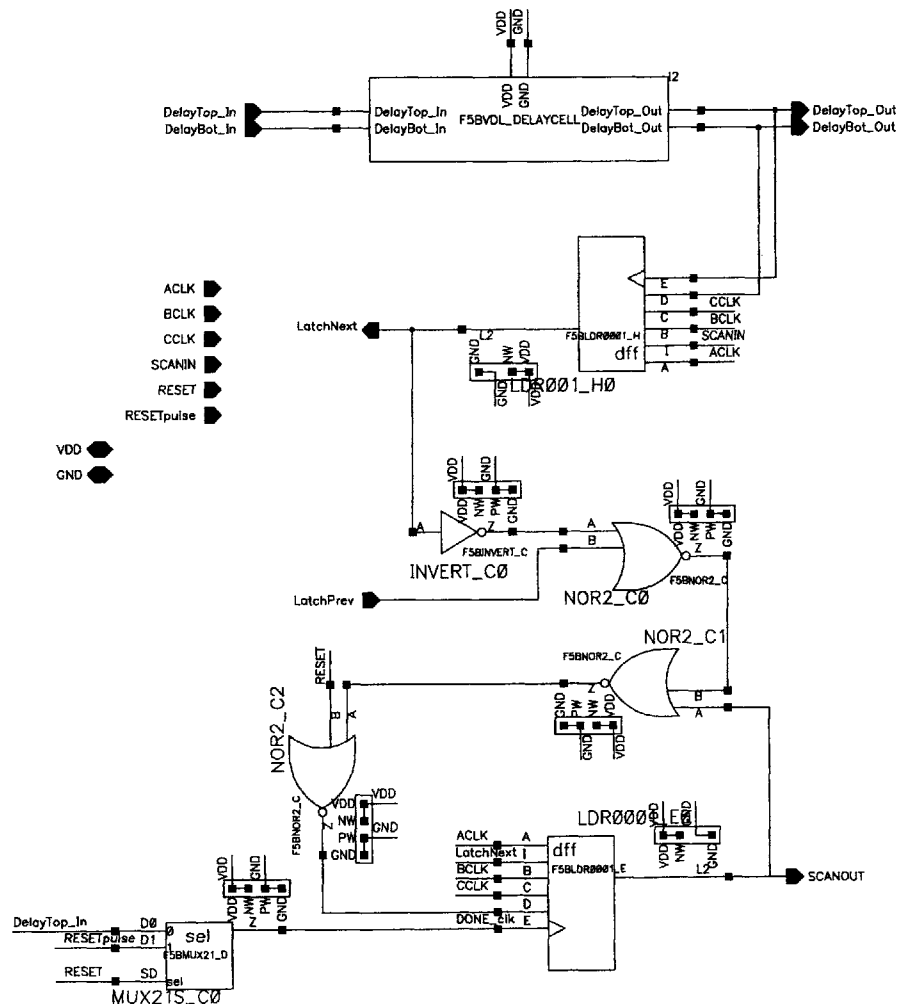


Figure 3-10: Schematic of Vernier cell

3.4.1 Data Storage

Each Vernier cell uses what is called “sticky” logic to store the collected data. This logic is implemented as follows: XOR-type logic is used after the first flip flop to convert the thermometer code into a code that locates the transition point. This data is fed into some combinational logic and another flip flop which is used to acquire the peak-to-peak jitter data. This logic remembers where each transition point happens for the duration that the test is active as described with an example as shown in Figure 3-11.

```
OutCode 00000000000000000000  Initialized
  
HitCode 11111111000000000000  After 1 rising edge
OutCode 00000000100000000000
  
HitCode 11111111111100000000  After 2 rising edges
OutCode 00000000100010000000
  
HitCode 11111111111100000000  After 3 rising edges
OutCode 00000000100110000000
  
OutCode 00000011111111110000  After N rising edges
```

Figure 3-11: Example of Vernier cell

This example shows a VDL that is 20 cells long. *HitCode* refers to the output of the first flip flop, and *OutCode* refers to the output of the sticky logic. When the circuit is initialized, the control logic resets *OutCode*. After the first rising edge appears, the thermometer code from *HitCode* is translated into a single transition point on *OutCode*. Then, assuming a jittery input signal, the transition point will occur at a different cell location on the second rising edge. However, because of the sticky logic mechanism, the first transition point is still stored on *OutCode*. Thus, after N rising edges, *OutCode* will have a string of 1's which represent different transition points of the input signal.

One drawback of the sticky logic method is that it only provides a worst case estimation for the jitter. Since the BIST circuit is measuring cycle-to-cycle jitter,

this is a peak-to-peak, cycle-to-cycle jitter measurement. However, it is very area efficient, because it only requires one flip flop and some combination logic per Vernier cell. The other end of the data storage spectrum is the idea of using large counters to store the transition data of each bit. By using large counters and exercising the VDL over many cycles, a probability distribution function can be generated, and external processing software can obtain many valuable results from the data, such as RMS and peak-to-peak jitter and possibly even the deterministic jitter and random jitter components. However, these large counters require a significant amount of area, which would have been unacceptable in this implementation and is the reason that the sticky logic method is used.

3.5 Data Acquisition

Data acquisition is done through the scan chain via the LSSD, or Level Sensitive Scan Design, protocol.¹ This data acquisition is implemented with little design modification and allows for all the data generated in the (VDL)² to be extracted serially through the scan chain instead of using an I/O pin for each signal, which is infeasible. In this design, note that LSSD is used solely for data acquisition and not for extensive testing of the circuit. There are two major modifications required over the original design. The first is that specially designed LSSD registers are required for the data acquisition, which is available in IBM's ASIC library. The second is that routing of the scan chain and A, B, and C clocks required for LSSD test must be implemented carefully.

3.6 Control Block

The control block is an FSM that describes the process flow of the system. A state transition diagram that illustrates this flow is shown in Figure 3-12, with the corresponding truth table shown in Appendix A.3.

¹LSSD is extensively covered in the literature and the methodology will not be discussed in detail in this report. Information about LSSD can be found in [23].

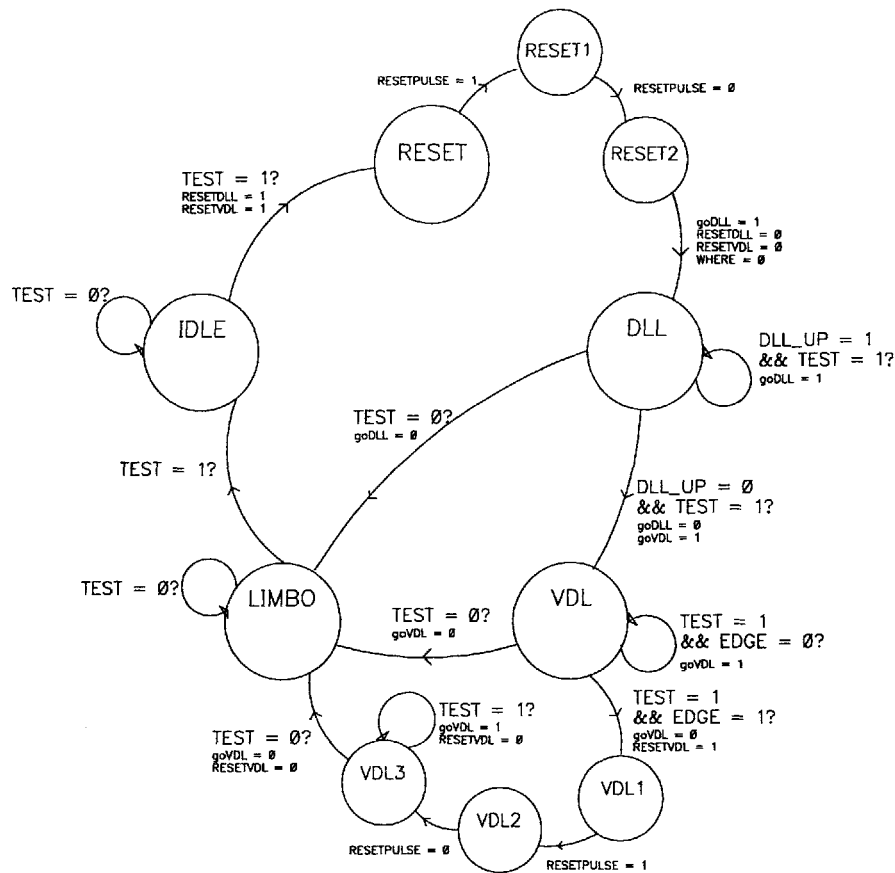


Figure 3-12: State transition diagram of control logic

The registers are clocked by the *PLLout* signal instead of a global reference clock. The system is designed such that regardless of the start up conditions, the circuit will ultimately initialize at the *IDLE* or *LIMBO* state before the test begins. At the rising edge of the *TEST* signal, the system will begin its reset sequence that will ensure the delay in the *DLL* delay line is at zero and that the Vernier cell outputs are all at zero as well. Then the system will enter the *DLL* state. During this state, all the reset signals are asserted low, and the *goDLL* signal is asserted high, which triggers the *DLL*. This state is valid until either the *TEST* signal is asserted low or the *DLL* has completed its task. In normal operation, after the *DLL* has locked, we would have *TEST* = 1 and *DLL_UP* = 0, which will cause a transition into the *VDL* state. The *VDL* state will freeze the *DLL* output by forcing *goDLL* = 0 and will turn on the

VDL by asserting $goVDL = 1$. The VDL will continue to run and collect data until the tester tells it to stop by making $TEST = 0$ or if an edge case is detected. If an edge case is detected, the FSM will enter the smaller loop that will reset the values in the VDL and then begin again, as indicated by states VDL1, VDL2 and VDL3. Edge cases are discussed later in Section 4.2.

This Chapter discussed in detail the design of the (VDL)². Two major components were introduced. The first was the digital delay locked loop, a modified and streamlined DLL used as a coarse tune circuit for centering the transition edges of the measured signal. The second was the Vernier delay line, a 60 stage VDL developed and optimized for linearity and minimal area. The next Chapter will discuss issues of these two components as well as other design issues that must be addressed.

Chapter 4

Other Design Considerations

Chapter 3 discussed on a high level the design of the BIST circuit. However, there are many challenges associated with implementing the system, from an architectural level as well as from a circuit and physical level. Chapter 4 will address these challenges and discuss their possible solutions.

4.1 Variable Resolution and Post-Processed Calibration

Calibration of delay elements has always been a serious issue in designing precise jitter measurement BIST circuits. Process, temperature, and voltage variations can alter delay elements by a factor of 4-5 in magnitude. Many designs of delay elements have attempted to set a specific attainable resolution and have built feedback systems for their delay elements to guard against environmental changes [24]. However, such efforts have been shown to significantly increase the circuit area and complexity with mixed results. As mentioned earlier, [20] shows promise in calibrating the delay lines accurately but has not been developed beyond software simulations. In the (VDL)² design, the decision was made that the smallest possible area was of utmost importance for the circuit to be valuable as a BIST implementation. Thus, a dynamic time resolution is used that has a dependence on process parameters.

One natural question that arises is: Since the resolution is variable, how does one know what the resolution actually is given a specific hardware implementation? This is done through a design trick by using identical delay elements in the DLL and the VDL. Using the LSSD scan chain, it is easy to obtain data out of all the registers in the circuit, including those of the counter in the DLL. The counter data gives information about the amount of delay elements inserted into the DLL delay line to delay the input signal by half a period. Thus, if the frequency of the input signal is known, it is easy to estimate the delay value of each delay element by the following equation

$$t_{PD_DLLDELAY} = \frac{\frac{1}{2f}}{COUNT(0-5)}$$

with $t_{PD_DLLDELAY}$ referring to the delay of each element in the DLL, f (or $\frac{1}{f}$) referring to the frequency (or period) of the input signal, and $COUNT(0-5)$ referring to the decimal value of the 6 count bits. Now that we know what $t_{PD_DLLDELAY}$ is, we also know what the delay through one element in the bottom chain, ($t_{PD_BOTDELAY}$), is, since the circuit is designed such that $t_{PD_DLLDELAY} = 2 \times t_{PD_BOTDELAY}$. Although we still do not have concrete information about the actual resolution, i.e. $|t_{PD_TOPDELAY} - t_{PD_BOTDELAY}|$, having information about the delay of one type of delay element can accurately give information about the difference between the delay of two types of delay elements. The simulation results of this hypothesis can be seen in Section 5.2.

4.2 Edge Cases and Centering

Here is an interesting problem that could occur with the BIST circuitry. Imagine that after the test is completed, a designer looks at a 20 bit *OutCode* and sees this:

OutCode 00000000000011111111

If our data storage method had used counters for each bit, this error would not be a big issue, as we would probably still be able to generate a histogram with the

data. However, using the sticky logic method requires knowledge about the length of the string of 1's. All we know about the above data is that the peak-to-peak jitter is at least 8 bits (times the circuit resolution) long. However, if we had an *OutCode* string that was deeper, we could see the entire distribution and know the actual peak-to-peak jitter data. If the data was centered perfectly, the results might look like:

OutCode 00001111111111110000

Here we would see that our distribution was actually 12 bits long. There are two issues that need to be addressed to solve this problem. One is keeping the data as centered as possible and the other is ensuring that the depth of *OutCode* is large enough that it can portray the entire jitter profile. Both techniques have been used in this circuit to ensure that the observable data is valuable data.

The maximum amount of measurable jitter desired is +/- 150 ps. If we assume the resolution of the circuit is 10 ps, then in order to make a meaningful measurement, the VDL needs to be (at the bare minimum) 30 cells deep. However, this depth assumes that the data will be perfectly centered, which is an incorrect assumption. Despite accurate centering logic, the coarse resolution of the DLL as well as the inherent jitter in the signal will cause the transitions to always be at least slightly mis-centered. Thus, the VDL has been made to be 60 cells deep to compensate for these uncertainties. However, this buffer space is not enough to ensure that the jitter distribution will not exceed the bounds of *OutCode*.

To account for this error fully, edge detection logic has been developed for cases such as the first example and is depicted in Figure 4-1. The logic monitors the activity of the first and last bits in the *OutCode* string, which should not flip to 1. If either of them do, it signifies that either the signals are not appropriately centered or that the input signal has significantly more jitter than expected. In any case the circuit will attempt a one-time correction by resetting *OutCode* and adjusting the delay logic for recentering. Resetting *OutCode* is necessary so that the old uncentered data does not contaminate the new data. This solution will solve the problem of edge cases and

centering except for the situation when the measured jitter is significantly larger than +/- 150 ps. In this case, the correction logic may be unable to effectively capture the entire jitter profile. However, this fact is inconsequential, because when looking at the outputs for this case it will be clear that the jitter is much larger than appropriate for the device under test.

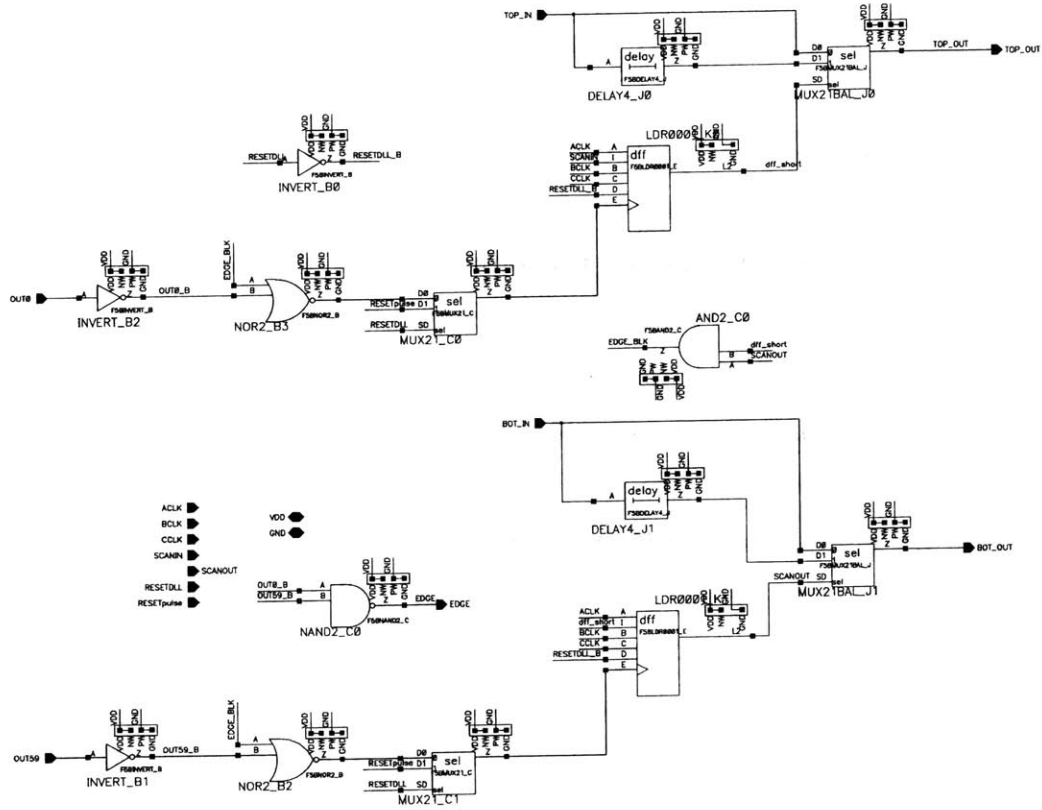


Figure 4-1: Schematic capture of edge detection circuitry

4.3 Metastability

Metastability is the fundamental unsolvable problem and occurs, for example, when the inputs to a sequential element violate the setup or hold time requirements of that element. As is evident in the design of this circuit, there can often be issues of metastability when the rising edge of one signal coincides with the rising edge of

another signal. The only way to avoid conditions of metastability is to either design around it or to allow a lot of settling time for the potentially metastable elements. There are two possible locations where metastability could be an issue, in the phase detector and in the VDL logic circuitry.

The phase detector output will have a very small chance of producing a metastable output because it has a significant amount of settling time available. Assuming T is the period of PLL_{out} , t_{PD} denotes propagation delay of an element, and t_{SU_FF} denotes the setup time of a flip flop, we have $t_{SETTLE} = T - (2t_{PD_FF} + 2t_{PD_LOGIC}) - t_{SU_FF}$. Generally, the entire contribution of the propagation delay comprise only a small fraction of T , which is on the order of several nanoseconds. Also, since the DLL acts as the coarse tune circuit and adds delay in a digital fashion, the probability that the rising edges of both inputs to the phase detector occur simultaneously are slim. Thus, the flip flop in the phase detector is fairly resilient against metastability.

The metastability in the VDL is a more serious issue because of the fact that through every cycle in the VDL there will be cells that have to deal with close rising edge transitions. The solution developed involves adding a little more settling time, and relies on the unique timing logic implemented for the VDL. Since the VDL is 60 stages long, the propagation delay of a rising edge from input to output would take approximately $t_{PD} = \max\{60 \times t_{PD_TOP}, 60 \times t_{PD_BOT}\}$, with t_{PD_TOP} representing the delay through the top delay element, and t_{PD_BOT} representing the delay through the bottom delay element. However, this is unacceptable, because the period of an incoming signal could be as short as 2 ns. Thus, the problem exists of when to notify the storage flip flop that data is valid on its inputs. This notification is done by using the signal path itself as a timer and by pipelining the data.

Recall from Figure 3-10 that there are two flip flops used in each VDL cell. One is used to generate the thermometer code data, and another is used to store that data using sticky logic. Both flip flops are clocked by the *top* input signal instead of a global clock as is common in digital designs. Thus, we know that each flip flop has a minimum of 2 ns to complete its computation (if the frequency being tested is at the maximum 500 MHz) and a maximum of 10 ns (if the frequency being

tested is at the minimum 100 MHz). The flip flop that generates the thermometer code data is the one that is susceptible to metastability, because the *top* and *bot* rising edge transitions could be very close together. The amount of settling time for this circuit to guarantee a correct transition at the second flip flop is $t_{SETTLE} = T - (3t_{PD_NOR} + t_{PD_INVERT} + t_{PD_TOPDELAY}) - t_{SU_FF}$. If the data is not valid within the settling time then the *OutCode* data bit corresponding to that VDL cell could be corrupted.¹ In general, this leniency should be adequate for the current purpose. However, if there appear to be bits which have metastable outputs, these bits are assumed to be at logical 1.

One other possible solution to the metastability issue with the VDL is to introduce dividers between the DLL and VDL stage, as shown in Figure 4-2.

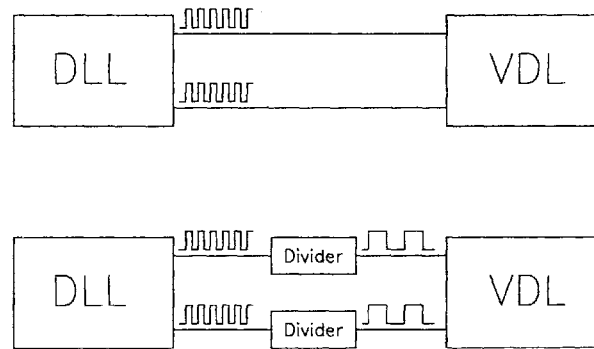


Figure 4-2: Top: Current implementation, Bottom: Implementation using dividers

Introducing dividers will increase the amount of settling time given to the thermometer code flip flop, because each rising edge transition in *top* and *bot* will be less frequent. Note however that this will slightly change the type of measurement being made, as the circuit will no longer make continuous measurements of the jitter, but will make measurements depending upon the amount of division. For example, if the divide ratio was 4, then only every 4th pair of transitions on *top* and *bot* would be measured by the VDL.

¹This will occur unless later on in the test cycle the same VDL cell sees a transition which satisfies the settling time criterion. In this case the bad data will be flushed out and the metastability condition eliminated.

A metastable output will most likely cause a corrupt *OutCode*, as described in the next section.

4.4 Corrupt OutCodes

It is possible that a corrupt *OutCode* could occur due to metastability issues, which would throw off measurement results, but only slightly. For example, the thermometer code (*Hitcode*) of a 20 stage VDL could be

HitCode 11111111110100000000

which signifies a false transition. How does the (VDL)² deal with this type of error? If this was the only measurement recorded, the output would be

OutCode 00000000001010000000.

Since the logic only compares two adjacent *HitCode* bits, this measurement would show two transitions on *OutCode*, because the data would appear as if there had been two transitions. However, a false transition would only affect the results of a test if it occurred near the ends of the jitter profile. For example if the *OutCode* distribution is

OutCode 00000011111111000000

and the next thermometer code is

HitCode 1111111101100000000

there would be no change in *OutCode*. However, if the thermometer code instead was

HitCode 111111111101000000

then the *OutCode* distribution would look like

OutCode 0000001111111100000

So, the effects of a corrupt *OutCode* due to metastability, if they occur, are inconsequential unless they occur near the edges of a jitter distribution, in which case the results are slightly more pessimistic than they appear.

4.5 Implementing Internal Reset

Something which may have been evident from observing the schematic captures of Chapter 3 are the seemingly unnecessary muxes preceding the D inputs to registers. However, these muxes are very necessary for implementing resets and start-up in the system. Because the circuit relies heavily on LSSD testing to extract data, special LSSD compatible registers were required and were taken from IBM's standard cell ASIC library. However, these registers were not built with any type of reset functionality, and as a result a larger module was designed which included these muxes, as shown in Figure 4-3.

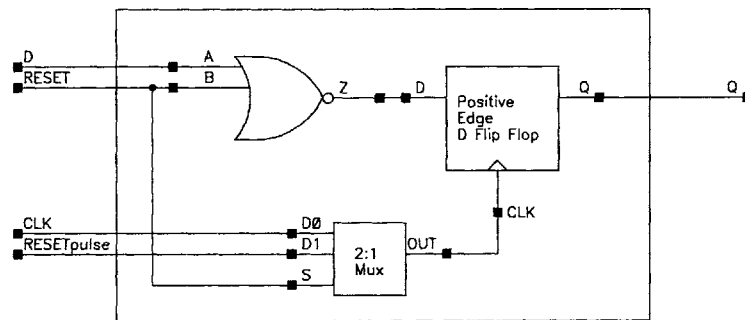


Figure 4-3: Flip flop module with embedded reset

Figure 4-3 can be best explained by visually examining the timing diagram in Figure 4-4. The signals *RESET* and *RESETpulse* are controlled and generated by the control logic. Initially the output of the flip flop is set to 1. When a reset is desired, *RESET* is set to 1, which will both mux out the *CLK* input to the internal flip flop as well as cause the internal D input to always be 0. Then on the next cycle, *RESETpulse* is set to 1, which triggers the internal flip flop to sample the internal D input. This action will logically cause the flip flop output to settle at 0. Then the reset signals are de-asserted, and the module once again acts as a traditional D flip flop.

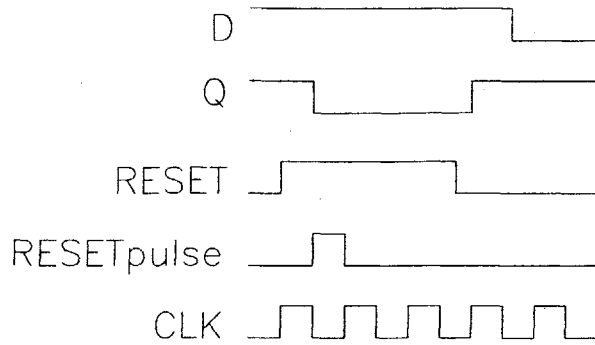


Figure 4-4: Modified flip flop behavior

4.6 Unwanted Jitter Generation

Jitter produced in the $(VDL)^2$ circuit itself is obviously something that should be minimized as much as possible. Such jitter will make the $(VDL)^2$ less accurate because there would be uncertainty as to whether the observed jitter originated from the input signal or from the testing circuitry. As mentioned in Section 3.2, the primary use of this circuit will be during manufacturing test, when the power supply for the circuit is very stable. Thus, assuming that most of the jitter would be coming from noisy power supplies, the other concern becomes jitter from temperature variation and transistor noise in the signal path. During circuit characterization the noise floor will be determined and taken into account in the future measurements.

4.7 Layout

Careful layout is essential in designing the $(VDL)^2$, as the architecture relies on closely matched signal paths to minimize timing errors. In consideration of time constraints, the control block and portions of the DLL were implemented using place and route technology as they did not contain any timing critical or noise sensitive signal paths. However, the actual signal path of the input signal and the delayed signal were designed and matched carefully in the VDL and the DLL delay line.

This Chapter presented the important design considerations associated with the (VDL)². Issues of resolution determination, edge cases and metastability are some of the problems addressed here. The next Chapter will show the results of testing of the (VDL)².

Chapter 5

Results

This Chapter presents the results found through simulation and analysis of the (VDL)². It discusses the performance of the two major components, the delay locked loop and the Vernier delay line, as well as overall characteristics such as calibration and power dissipation.

5.1 Overall Results

5.1.1 DLL Performance

DLL performance was tested under nominal, slow, and fast process and temperature variations with a 1.1 V supply. The DLL offers a dynamic capture range of 100 MHz - 500 MHz under these variations. Figure 5-1 shows a complete sweep of the DLL from code 0 to code 63 under nominal conditions and relates the amount of delay introduced into the signal with the COUNT value of the counter. The plot was generated by sending a 50 MHz signal as input to the DLL and measuring the delay generated with respect to that signal after each increment of the counter. It is plotted against a best-fit line for illustration of linearity.

The instantaneous slope of the best-fit line is 108.5542, which represents the ps/code, meaning the amount of delay introduced by each delay element.

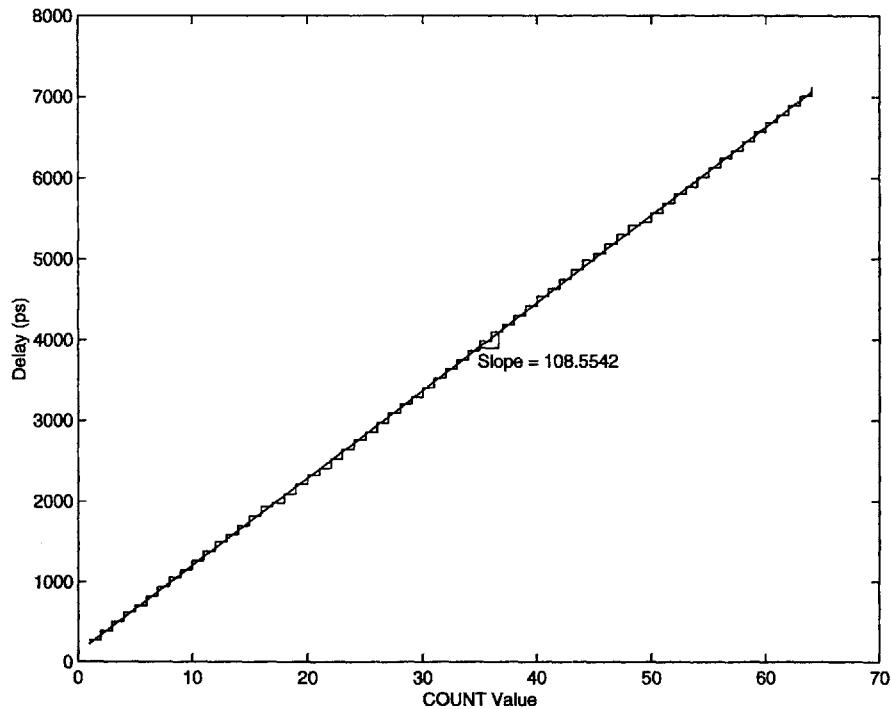


Figure 5-1: Entire DLL sweep

Figure 5-2 is a plot of the differential non-linearity (DNL) and integral non-linearity (INL) of the data from Figure 5-1. DNL refers to the difference between a measured code width and the ideal code width, and INL refers to the difference between the measured code transition point and the ideal code transition point.

As mentioned in Chapter 3, the DLL acts as a coarse tune portion of the circuit, and its major functions are to provide approximately $\frac{1}{2}$ period delay as well as to provide calibration data. Observing the DNL and INL data shows generally satisfactory performance except at two locations, which correspond to COUNT code transitions 15-16 and 47-48. In this situation COUNT bits 0-3 are changing from 1 to 0 while COUNT bit 4 is changing from 0 to 1. This error is introduced because of a mismatch in the wiring delay for the mux that controls the delay of 16. Unfortunately the mismatch was discovered too close to the tape out date, so implementing the necessary changes was infeasible. However, this non-linearity does not affect the DLL's locking capability, only its calibration capability, and knowing the results from this simulation can be very useful. For example, when doing calibration for a signal

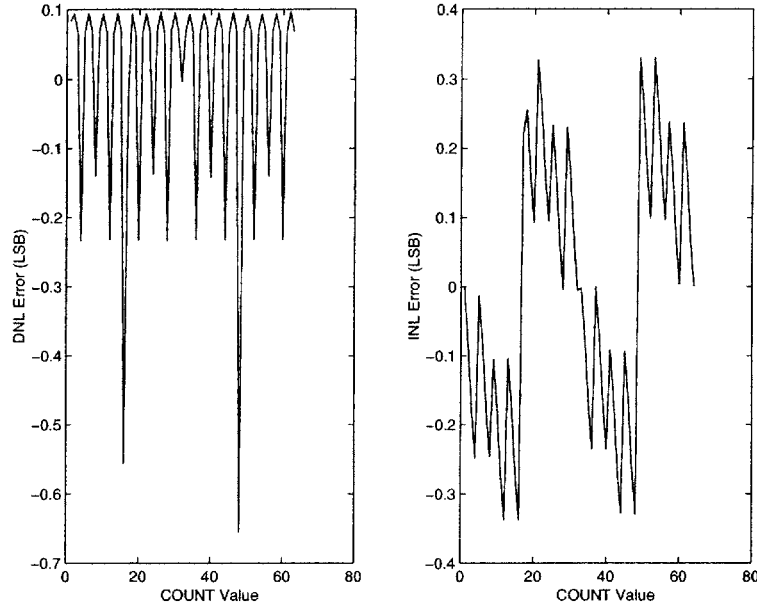


Figure 5-2: Left: DNL (in LSB) of the DLL, Right: INL (in LSB) of the DLL

that was delayed by 0-15 COUNT codes, no adjustment is necessary in the resolution calculation. If the signal was delayed by 16-47 COUNT codes, the number of bits used in the calculation could be subtracted by $\frac{1}{2}$ to take the error into account. Finally, if the signal was delayed by 48-63 COUNT codes, the number of bits would be subtracted by 1 in the calculation. In this manner the effects of the errors introduced at these two locations can be minimized.

5.1.2 VDL Performance

The VDL was tested at all process corners to ensure functionality. Figure 5-3 shows the VDL response for nominal, fast and slow process corners. The test was developed by sending in two signals with slightly different periods and running the test until every bit in the VDL chain had transitioned. Developing an effective physical layout for the VDL proved difficult, as simulation of the physical layout often showed deviation from the schematic simulation, and getting it right was a fairly time-consuming endeavor.

Each run was swept by using a period difference of 4 ps. An ideal sweep would use

a period difference much smaller to get a significantly more accurate representation of the resolution distribution, but each simulation was very time-consuming and the simulator tended to fail with significantly longer test runs. Regardless, the plots still demonstrate the monotonicity of the VDL as well as a fairly linear response.

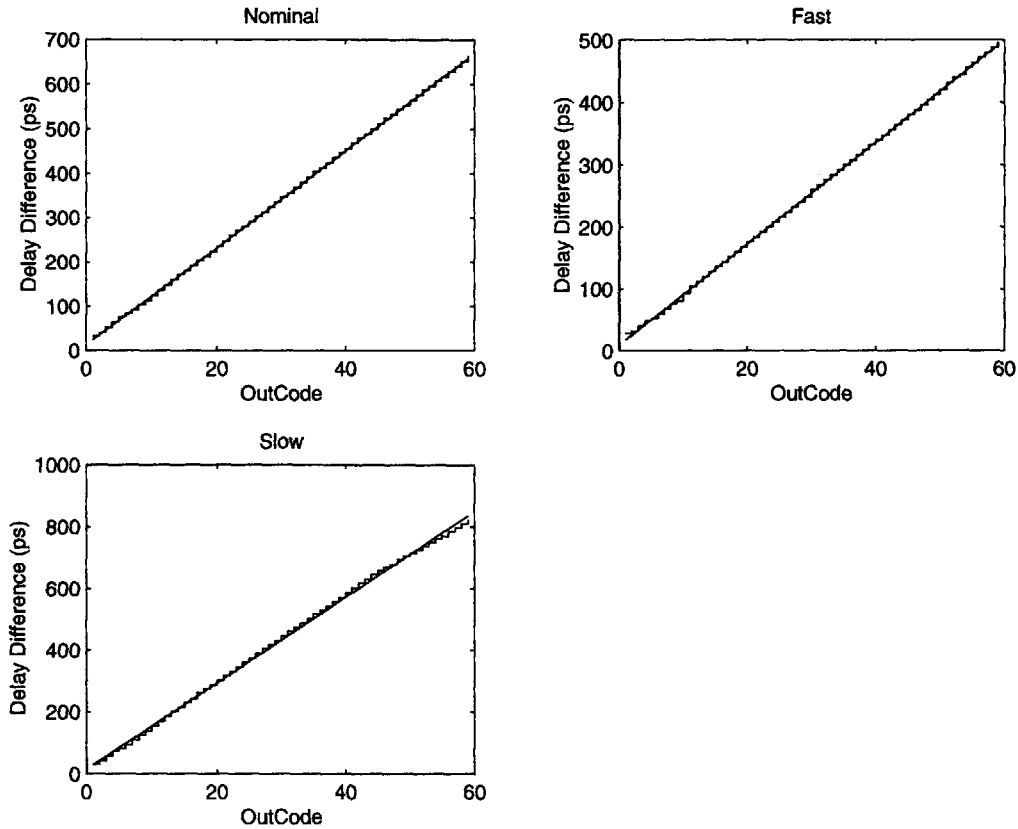


Figure 5-3: Full sweep under three process corners

Resolution

Plotting the best-fit lines of each sweep in the previous section on one graph produces Figure 5-4. The slopes of the best fit lines are shown as well. These slopes represent the resolution of the circuit for the given process. The fast process has a resolution of 8.1744 ps, the nominal process a resolution of 10.9372 ps, and the slow process a resolution of 13.9212 ps, all satisfying the design goal of 15 ps. The resolution of each circuit has a clear dependence on process variation, so on a faster process the

resolution of the circuit will be better than on a slower process.

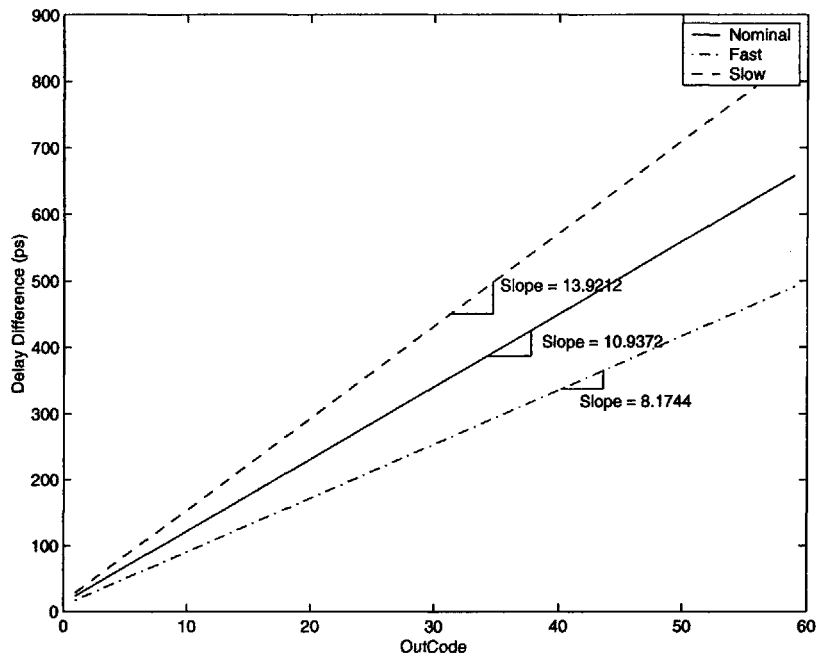


Figure 5-4: Three VDL sweep variations highlighting the resolution difference

Edge Detection

The edge detection circuit works as a one-shot method to correct a drastic centering error from the DLL. An example of this is shown in Figure 5-5.¹ This example shows how the detection of a transition on *Out0* causes the circuit to believe the DLL is miscentered, thus introducing a delay to attempt to recenter the distribution. The delay of 108 ps introduced in this example causes an *OutCode* shift of 10-11 closer to the center, and this one-time adjustment is enough to ensure the results will be bound by the 60 bit code. If *Out0* or *Out59* show a transition even after this adjustment, this suggests that the signal jitter is much larger than the maximum specified +/- 150 ps, and this would be easily verified by observing the output.

¹Note that this simulation does not show the reset effects that would normally occur after an edge detection.

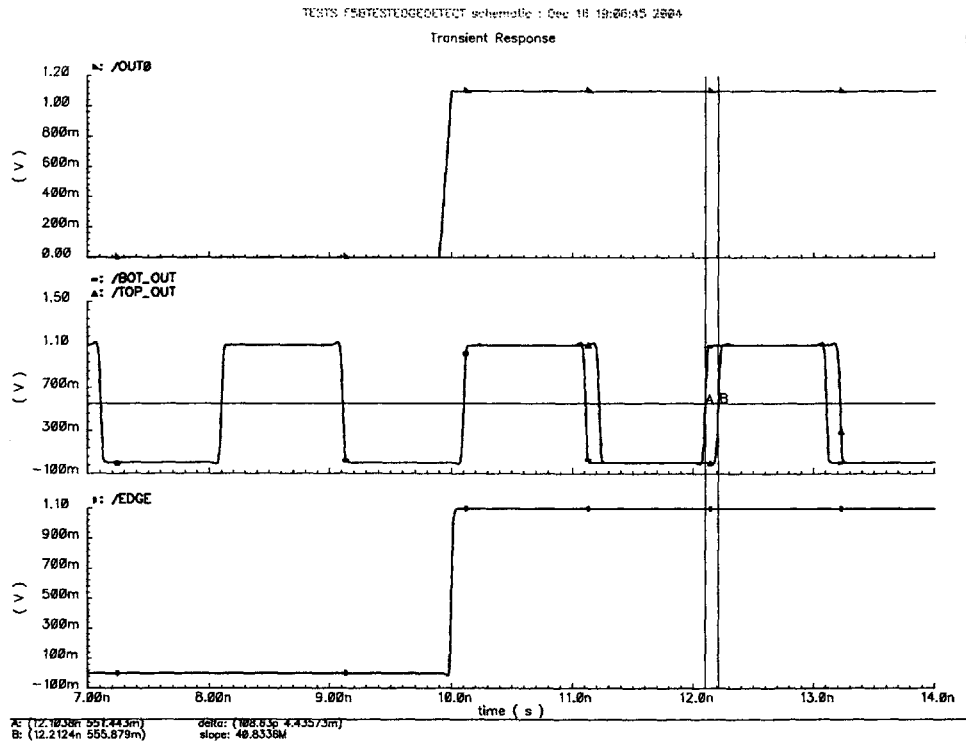


Figure 5-5: Edge detection causing a delay change

5.2 Calibration

The hypothesis proposed earlier in Section 4.1 was that in order to minimize chip area (which is a crucial specification in a BIST implementation), calibration of the $(VDL)^2$ could be done after the completion of a test, by taking advantage of the fact that the delay of a delay element correlates to the delay difference of two delay elements. Simulation data has been observed and is shown in Figure 5-6. Each plot shows a delay value in ps versus a specific process corner. Each process corner number in one graph represents the same conditions in the other, and each process corner number corresponds to the environmental conditions as shown in Table 5.1.

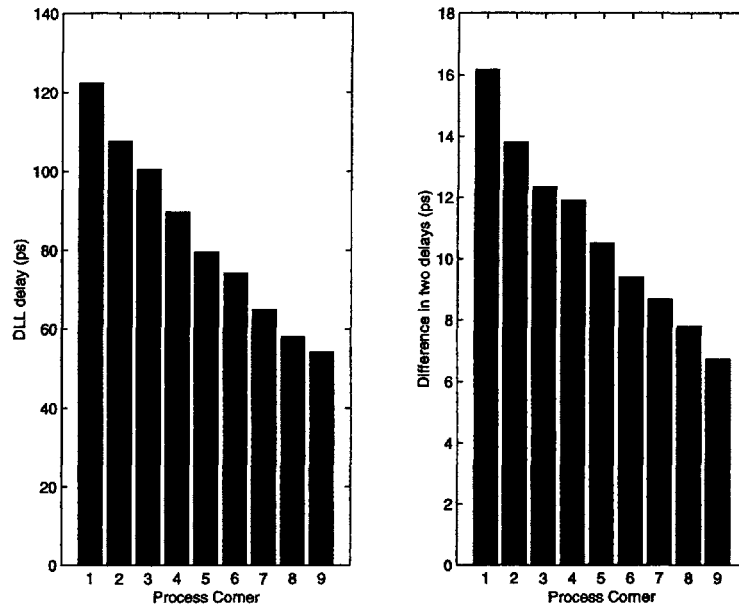


Figure 5-6: Left: Delay of an element over circuit variations, Right: Delay difference of two elements over circuit variations

Process	1	2	3	4	5	6	7	8	9
VDD	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1
Temp	125	27	-25	125	27	-25	125	27	-25
Process	Slow	Slow	Slow	Nom	Nom	Nom	Fast	Fast	Fast

Table 5.1: Relation between process variation number and environmental condition

Figure 5-7 shows both of these results normalized to its nominal case and plotted on the same graph. As you can see, it is clear that having information about the delay of a delay element will give valid information about the delay difference of two elements. This data can be easily obtained during circuit characterization and used as a reference during manufacturing test.

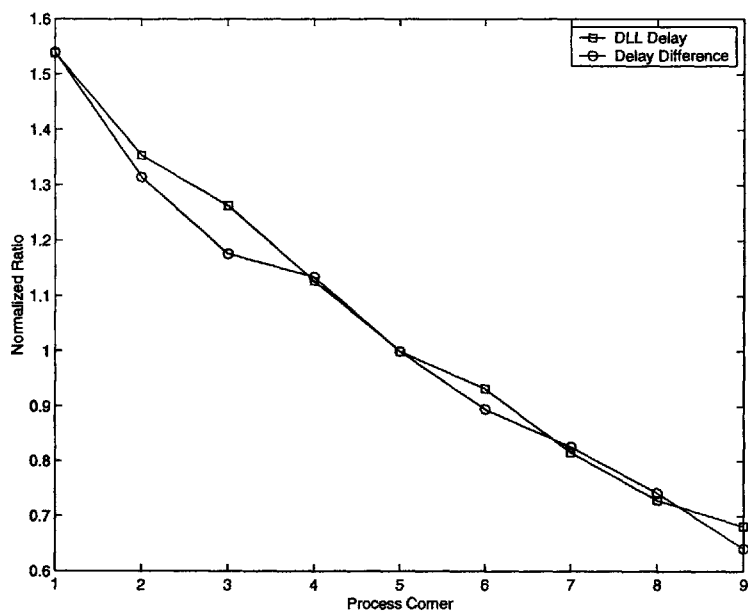


Figure 5-7: Correlation between the two delays

5.3 Power Dissipation

The power dissipation of the BIST circuitry is low and has a negligible impact on the voltage levels of the power supply. A plot of the current draw over an entire run of the BIST circuitry is shown in Figure 5-8.

The input to the BIST is a jitter-free 500 MHz clock signal. From 0 ns to 6 ns, the circuit is initializing and the control logic is settling into *IDLE* state. At 6 ns, the *TEST* signal is asserted and the system reset begins. This system reset continues until approximately 12 ns, when the *goDLL* signal is asserted and the DLL begins delaying the input signal. From 12 ns until 42 ns the DLL slowly increments its count and introduces more delays into the delay line, until the desired phase shift is achieved. At 42 ns the DLL is locked, and the VDL is triggered, and the circuit stays in this mode until the *TEST* signal is de-asserted. The circuit is running on a 1.1 V power supply, so the effective power dissipation under nominal conditions is approximately $1.1 \text{ V} \times 1.3 \text{ mA} = 1.43 \text{ mW}$ during lock acquisition and $1.1 \text{ V} \times 6.1 \text{ mA} = 6.71 \text{ mW}$ during the VDL operation.

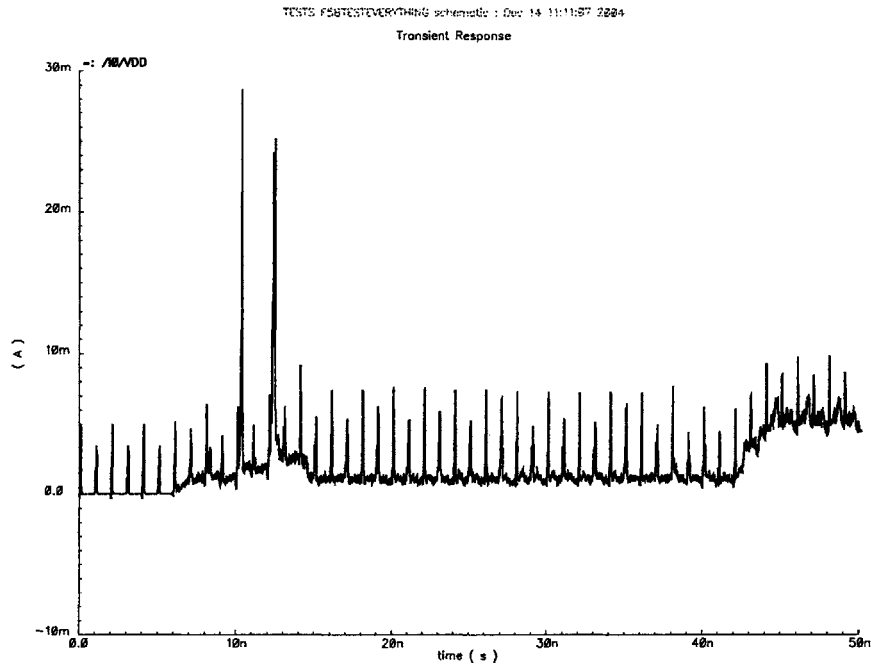


Figure 5-8: Current draw for the (VDL)² over a complete test run

Note the large spiking in the waveform that is typical in digital designs. The dominant periodic spikes are due to the internal switching in the control logic caused by the input signal, while the two large spikes correspond to the global reset sent to all registers. The periodic spikes could be mitigated in a future design by using less power hungry registers, but LSSD ones were used for testing purposes.

5.4 Chip Level Issues

An IBM ASIC test site launch consists of about a dozen chips, with each chip being around 4 mm x 4 mm. The PLL and Clocking Development department owns two of these chips, and divides each chip into 4 quadrants, generally with one module or macro in a single quadrant. (VDL)² shares two quadrants with a standard IBM PLL that has already been fully specified. The two circuits are connected together as described earlier in Figure 3-2. In actual implementation, the (VDL)² will be as physically close to the PLL as possible, but for the test site the decision was made to

keep the (VDL)² and the PLL in their own quadrants. The test site is designed such that there are three spots in each quadrant where sensitive or high frequency signals can be transmitted off-chip using high quality SMA connectors and cable. Since the PLL already uses the available I/O for its quadrant, a tradeoff exists between keeping the signals between the PLL and (VDL)² as short as possible and keeping the signals between the (VDL)² and the off-chip tester as short as possible. By keeping the (VDL)² in its own quadrant, the signal paths to the testers are minimized. This is more important than the signals between the PLL and the (VDL)² because the characterization work will mostly involve transmitting signals with various known jitter values from a tester to the (VDL)². All sensitive signals are balanced to be minimized and matched in capacitance as closely as possible.

5.5 Power Ring

The power distribution network for this circuit was fairly straightforward due to both the small footprint as well as the low power requirements. 7 μm wide M1 and M2 metal is used for both VDD and GND supplies, with frequent 0.7 μm wide straps along the circuit to ensure excellent coverage. These numbers were chosen by examining the average and maximum power dissipation of the BIST circuit and utilizing results from prior designs within the ASIC group.

For the purposes of the ASIC test site, the power supply of the BIST circuit is independent of the supplies of the other circuits on the chip. Power is distributed externally to the power ring through a 25 μm wide metal.

5.6 Final Layout

The final layout of the (VDL)² sent to be fabricated is shown in Figure 5-9, and the chip level layout showing the PLL and the (VDL)² can be seen in Figure 5-10.

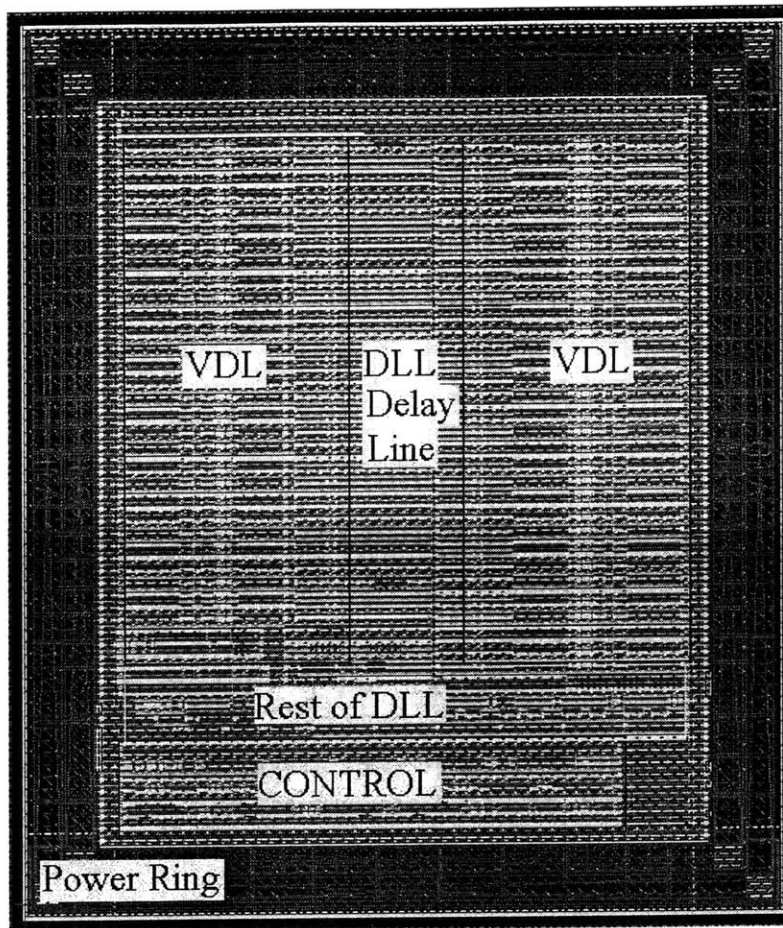


Figure 5-9: Final layout of $(VDL)^2$

This Chapter discussed the results of the $(VDL)^2$ testing. Performance of the major components was described and analyzed, as well as the performance of the entire circuit. Environmental variations were introduced during simulations to obtain data at all process corners. Testing showed functionality on all process corners with low power dissipation, but with two isolated nonlinearities in the DLL. Data gathered on VDL variable resolution and edge detection circuitry correlated well with theory.

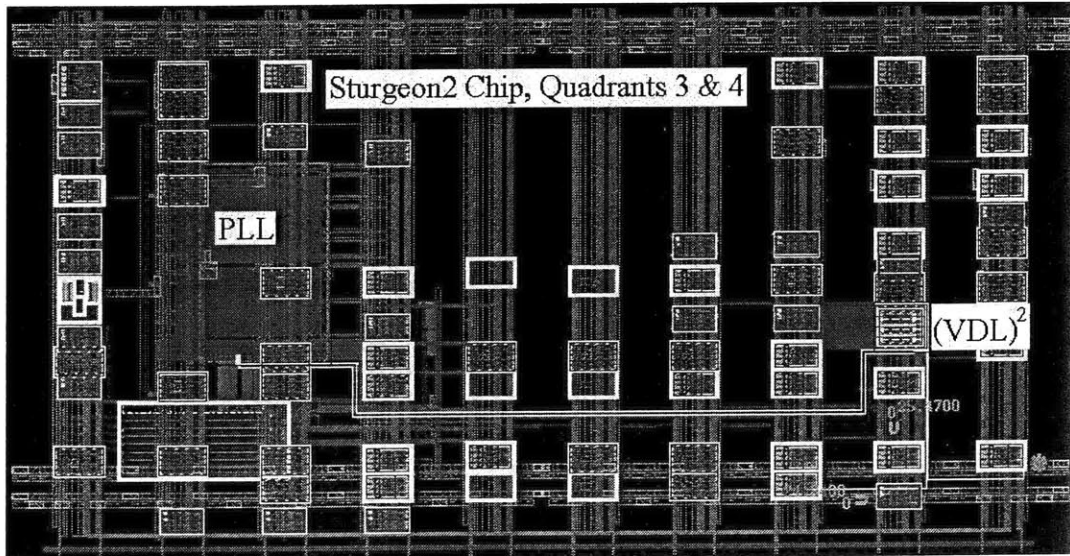


Figure 5-10: Final layout of Sturgeon2, quadrants 3 and 4

Chapter 6

Future Work and Conclusion

This Chapter discusses the substantial amount of future work that could be done on the (VDL)² circuit to make it an even more appealing solution for measuring jitter. Some of the more important improvements are listed below.

6.1 Future Work

6.1.1 Improve Delay Elements and Signal Matching

As mentioned earlier, the delay elements used for coarse delaying in the DLL and for generating a delay difference in two delay elements are various versions of active inverters. These active inverters have a poor power supply rejection ratio and intrinsic noise factors associated with the physical design. In the primary application, power supply noise is negligible because the test will be applied during the PLL manufacturing test when the digital circuitry is idle. However, as shown in Section 5.1.1, there are some non-linearity issues that should be addressed. Most of these issues can be solved through an iterative process towards balancing signal paths better, such as in the DLL delay line at codes 15 and 47, but some of the errors are introduced by the delay elements as well. Incorporating more environment-insensitive delay elements would help in the development of an entire system that is robust to noise, as described in the next section.

6.1.2 Improve Robustness to High Frequency Voltage Variation

If further development of the circuit is done in such a way to ensure the entire signal path is fairly resistant to VDD noise or GND bounce, then (VDL)² would become a macro that could measure jitter not only in PLLs on the manufacturing line, but could measure it anywhere in a circuit at any time. For example, in a microprocessor, such a circuit could be placed at the ends of a clock distribution tree and could feasibly be tested even while the processor is running! This type of application would spell disaster for the current implementation, as the power supply in a microprocessor can spike up to 200 mV at a time, which would alter the delays in the delay elements and throw off the jitter measurement.

6.1.3 Offer More Types of Jitter Measurements

Ideally designers would be able to obtain many various types of jitter measurements from its testing circuit. The most valuable type of jitter measurement that could be added would be a long term period jitter measurement. Being able to extract the long term period jitter data would give information on how lower frequency perturbations affect a phase locked loop's jitter performance. One key example where long term period jitter data would be valuable is in the case of a frequency synthesis of a much higher frequency than the input. For example, a frequency synthesis factor of 1000 would generate a 1 GHz signal from a 1 MHz input signal by using a divide by 1000 in the feedback loop. However, because the inputs to the phase detector are both at 1 MHz, the 1 GHz output signal only gets refreshed once every 1000 periods. This can lead to significant long term deviation in the output signal, as shown in Figure 6-1. In this situation, the cycle-to-cycle jitter measurement would be the same magnitude at each edge transition comparison, but after several transitions the actual period would have deviated far from its ideal length. Thus, incorporating measurements that capture low frequency components of jitter in addition to the high frequency components is very desirable.

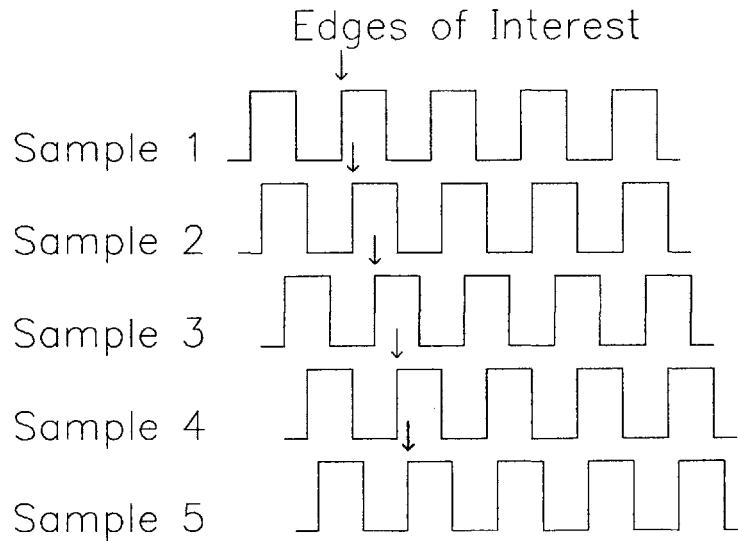


Figure 6-1: Constant cycle-to-cycle jitter but increasing long term jitter

6.1.4 Wider Frequency Range

The frequency range of 100 MHz - 500 MHz for the (VDL)² was determined around a decision that 250 MHz was the so-called "sweet spot" of measurements, as it is a frequency that can be generated by many different PLL architectures that run at different frequencies. However, as PLL development continues to push the envelope in terms of speed and performance, this sweet spot will eventually change. Thus, the best solution would be to increase the frequency input range of the circuit, to ensure the BIST circuit can effectively test various PLL designs without any internal changes. In the (VDL)², an even wider frequency range could have been achieved by developing speed-oriented custom registers and logic. In certain situations, LSSD registers were used to make data extraction easier but are not necessary for the functionality of the circuit. The LSSD registers are slow compared to a straightforward implementation of a D flip flop, and this restricts the upper bound on the frequency range.

6.2 Conclusion

The development of accurate, fast and cost-effective testing is an ongoing battle, and the concept of BIST is one promising solution. This design project attempts to satisfy these testing goals by developing a new type of BIST structure for measuring jitter in phase locked loops. It improves on prior attempts primarily by eliminating the need for an external clock signal, by offering more than just a phase jitter measurement, and by allowing for easy, area-friendly calibration and data acquisition. The development of jitter measurement BIST circuits is a new and growing research field, and in the near future, these types of BIST circuits may be the dominant test methodology of choice for designers.

Appendix A

Tables

DelayTop_In	DelayBot_In	LatchNext
1st Rising Edge	2nd Rising Edge	1
2nd Rising Edge	1st Rising Edge	0

Table A.1: Truth table for thermometer flip flop

Latch Next	Latch Prev	RESET	RESETpulse	DelayTop_In	SCANOUT_P	SCANOUT
X	X	0	0	Rising Edge	1	1
0	0	0	0	Rising Edge	0	0
0	1	0	0	Rising Edge	0	0
1	0	0	0	Rising Edge	0	1
1	1	0	0	Rising Edge	0	0
X	X	1	Rising Edge	X	X	0

Table A.2: Truth table for *OutCode* flip flop

S3-S0	DLL_UP	TEST	EDGE	D3-D0	RESET_DLL	RESET_VDL	RESET_pulse	WHERE_N	go_DLL	go_VDL	CLK_BLK
0	X	0	X	0	0	0	0	1	0	0	1
0	X	1	X	1	1	1	0	1	0	0	0
1	X	X	X	2	1	1	1	IN	0	0	0
2	X	X	X	3	0	0	0	0	1	0	0
3	x	0	X	5	0	0	0	0	0	0	0
3	1	1	X	3	0	0	0	0	1	0	0
3	0	1	X	4	0	0	0	0	0	1	0
4	X	0	X	5	0	0	0	0	0	0	0
4	X	1	0	4	0	0	0	0	0	1	0
4	X	1	1	8	0	1	0	0	0	0	0
5	X	0	X	5	0	0	0	0	0	0	1
5	X	1	X	0	0	0	0	0	0	0	0
8	X	X	X	9	0	1	1	0	0	0	0
9	X	X	X	A	0	1	0	0	0	0	0
A	X	0	X	5	0	0	0	0	0	0	0
A	X	1	X	A	0	0	0	0	0	1	0

Table A.3: Control logic truth table

Bibliography

- [1] Bezhad Razavi. *Monolithic Phase-Locked Loops and Clock Recovery Circuits: Theory and Design*, pages 1–39. IEEE Press, 1996.
- [2] Tian Xia, Jing Li, and Randy Wolf. Overview of jitter measurement for high speed digital signals. Talk at IBM Microelectronics in Essex Junction, VT, July 2004.
- [3] JESD65b: Definition of skew specifications for standard logic devices, September 2003. JEDEC Solid State Technology Association.
- [4] Bozena Kaminska. Bist means more measurement options for designers. *EDN Magazine*, December 2000.
- [5] Paul Horowitz and Windfield Hill. *The Art of Electronics*, chapter 15.10, pages 1022–1024. Cambridge University Press, second edition, 1989.
- [6] Abhijit Chatterjee and Sasikumar Cherubal. Method and apparatus for high resolution jitter measurement, September 2002. US Patent Pending no. 2002/0136337.
- [7] Tian Xia and Jien-Chung Lo. Time-to-voltage converter for on-chip jitter measurement. *IEEE Transactions on Instrumentation and Measurement*, 52(6):1738–1748, December 2003.
- [8] Elvi Raisanen-Ruotsalainen, Timo Rahkonen, and Juha Kostamovaara. Time interval measurements using time-to-voltage conversion with built-in dual-slope

- a/d conversion. *IEEE International Symposium on Circuits and Systems*, 5:2573–2576, June 1991.
- [9] Makoto Takamlya, Hiroki Inohara, and Masayuki Mizuno. On-chip jitter-spectrum-analyzer for high-speed digital designs. *IEEE International Solid-State Circuits Conference*, 1:350–352, February 2004.
- [10] Timo E. Rahkonen and Juha T. Kostamovaara. The use of stabilized cmos delay lines for the digitization of short time intervals. *IEEE Journal of Solid-State Circuits*, 28(8):887–894, August 1993.
- [11] Pramodchandran N. Variyam and Hari Balachandran. All digital built-in self-test circuit for phase-locked loops, December 2003. US Patent no. 6,661,266.
- [12] Larry D. Smith and Normal E. Abt. On-chip pll phase and jitter self-test circuit, March 1999. US Patent no. 5,889,435.
- [13] Piotr Dudek, Stanislaw Szczepanski, and John V. Hatfield. A high-resolution cmos time-to-digital converter utilizing a vernier delay line. *IEEE Transactions on Solid-State Circuits*, 35(2):240–247, February 2000.
- [14] Antonio H. Chan and Gordon W. Roberts. A jitter characterization system using a component-invariant vernier delay line. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(1):79–95, January 2004.
- [15] Arnold M. Frisch and Thomas H. Rinderknecht. Jitter measurement system and method, September 2001. US Patent no. 6,295,315.
- [16] Jui-Jer Huang and Jiun-Lang Huang. A low-cost jitter measurement technique for bist applications. *12th Asian Test Symposium*, pages 336–339, November 2003.
- [17] Chee-Kian Ong, Dongwoo Hong, Kwang-Ting Cheng, and Li-C Wang. A scalable on-chip jitter extraction technique. *Proceedings of the 22nd IEEE VLSI Test Symposium*, pages 267–272, April 2004.

- [18] Stephen Sunter and Aubin Roy. On-chip digital jitter measurement, from megahertz to gigahertz. *IEEE Design and Test of Computers*, 21:314–321, July 2004.
- [19] Chin-Cheng Tsai and Chung-Len Lee. An on-chip jitter measurement circuit for the pll. *12th Asian Test Symposium*, pages 332–335, November 2003.
- [20] Byron Nelson and Mani Soma. On-chip calibration technique for delay line based bist jitter measurement. In *Proceedings of the 2004 International Symposium on Circuits and Systems*, volume 1, pages 944–947, May 2004.
- [21] Kuo-Hsing Cheng, Shu-Yu Jiang, and Zong-Shen Chen. Bist for clock jitter measurements. In *Proceedings of the 2003 International Symposium on Circuits and Systems*, volume 5, pages 577–580, May 2003.
- [22] Phillip J. Restle, Robert L. Franch, Norman K. James, William V. Huott, Timothy M. Skergan, Steven C. Wilson, Nicole S. Schwartz, and Joachim G. Clabes. Timing uncertainty measurements on the power5 microprocessor. In *2004 IEEE International Solid-State Circuits Conference*, February 2004.
- [23] Robert W. Bassett, Mark E. Turner, Jeannie H. Panner, Pamela S. Gillis, Steven F. Oakland, and Douglas W. Stout. Boundary-scan design principles for efficient lssd asic testing. *IBM Journal of Research and Development*, 34(2):339–354, March 1990.
- [24] Federico Baronti, Diego Lunardini, Roberto Roncella, and Roberto Saletti. A self-calibrating delay-locked delay line with shunt-capacitor circuit scheme. *IEEE Journal of Solid-State Circuits*, 39(2):384–387, February 2004.