JITTER REDUCTION CIRCUITS TO REDUCE THE BIT-ERROR RATE OF HIGH-SPEED SERIALIZER-DESERIALIZER (SERDES) CIRCUITS

BY HARI VIJAY VENKATANARAYANAN

A dissertation submitted to the
Graduate School—New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
Graduate Program in Electrical and Computer Engineering

Written under the direction of
Prof. Michael L. Bushnell
and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2008

ABSTRACT OF THE DISSERTATION

Jitter Reduction Circuits to Reduce the Bit-Error Rate of High-Speed Serializer-Deserializer (SERDES) Circuits

by Hari Vijay Venkatanarayanan

Dissertation Director: Prof. Michael L. Bushnell

A new jitter reduction technique is proposed for reducing the timing jitter in a *serializer-deserializer* (SERDES) circuit. The technique involves transmit and receive side jitter reducer circuits made of only 14 and 20 transistors, respectively. They reduce the jitter in the clock generated by the *phase-locked-loop* (PLL) at the transmit side, and the jitter between the recovered clock and the serial data at the receive side. The jitter reducers are designed using $70\,nm$ Berkeley Predictive process models and tested with various types of input jitter. In the case of the transmit side jitter reducer, the jitter is reduced, on average, by 62.24%. The performance of the jitter reducer is compared with the adaptive PLL technique proposed by Xia *et al.* [39] in terms of the peak-to-peak jitter reduction. The peak-to-peak jitter is reduced, on average, by 45.51% using the transmit side jitter reducer. For the receive side jitter reducer, the jitter is reduced, on average, by 35.88%. The SERDES circuit is then tested for its jitter performance under three conditions: (1) no jitter reducers are present, (2) the receive side jitter reducer is present and (3) both transmit and receive side jitter reducers are present. In

each of these cases, the *bit-error rate* (BER) is computed probabilistically and is shown to improve from 8.3×10^{-2} to 6.44×10^{-20}, for input RMS *periodic jitter* (PJ) of 71.77 ps. Finally, a SERDES test scheme is used to test the jitter reducers for their stuck-at faults and then to perform the receiver jitter tolerance and BER tests.

Acknowledgements

I am grateful to Prof. Michael L. Bushnell for taking me as his student and for guiding me in my endeavors. I am thankful to him for his support in the successful completion of my doctoral research. In my six years as a graduate student, I was able to hone my skills through his mentorship, and finally, I would like to thank him for providing me the financial support that helped me in concentrating on my research.

I would like to thank my sister Sunitha and my brother-in-law Karthik for providing me moral support and also my friends: Giri, Ashok, Bharath, Rajamani, Omar, Krishna and Karthikeya. I would also like to thank my fellow lab mates: Baozhen, Roy, Aditya, Sharanya, Raghuveer and Shiva for providing a healthy environment for research.

I would also like to thank Dr. Tapan J. Chakraborty for providing valuable advice for my doctoral research. Last but not the least, I would like to thank the CAIP staff members for providing support in running the various tools in our laboratory.

Dedication

To my parents, sister, brother, niece and friends

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

SERDES is a high-speed serial data link. A growing number of *application specific integrated circuits* (ASICs) and programmable *integrated circuits* (ICs) provide integrated SERDES interfaces [21]. A typical high-speed serial data link is shown in Figure 1.1. Its purpose is to quickly and reliably transfer data from one physical location to another. The data, often in parallel bus form, is serialized to a single high-speed signal. This signal is transferred across a path medium that is ideally a high quality transmission line path to the new location. Included in SERializer and DESerializer functions are embedded clock and *clock-data recovery* (CDR) circuitry, needed to create a high-speed serial path.

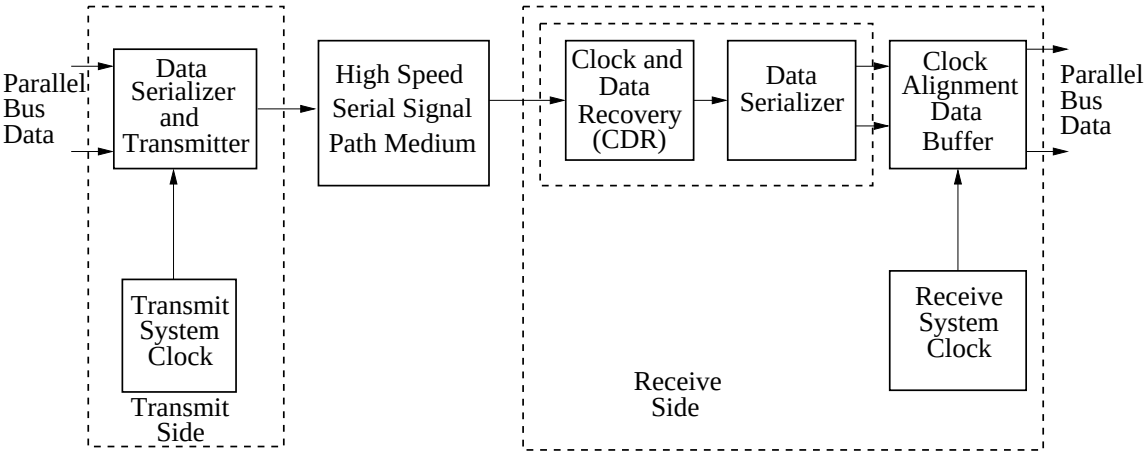


Figure 1.1: SERDES Block Diagram [21]

At the receive end of the path, a *clock and data recovery* (CDR) circuit receives the signal and extracts a properly timed bit clock from the data flow. The data signal is then deserialized down to a lower speed parallel data interface.

Chip-to-chip communication had previously been almost exclusively a parallel domain [6]. The amount of logic needed to serialize and deserialize far outweighed any savings that come from pin count reduction. But, with deep sub-micron geometry, an incredible amount of logic can be achieved in a very small area of silicon. SERDES can be included on parts for a very low silicon cost. Add to that the ever increasing need for I/O bandwidth, and SERDES quickly becomes the logical choice for moving any significant amount of data chip-to-chip. Consider the following benefits of SERDES chip-to-chip communication:

- Pin Count: Smaller, cheaper packages.
- Pin Count: Fewer layers and pins on *printed circuit board* (PCB) assemblies.
- Smaller Packages: Smaller, cheaper boards and more compact designs.
- *Simultaneous Switching Output* (SSO): When multiple output drivers switch simultaneously, they induce a voltage drop in the chip/package power distribution [4]. The simultaneous switching momentarily raises the ground voltage within the device relative to the system ground. This apparent shift in the ground potential to a non-zero value is known as *simultaneous switching noise* (SSN) or, more commonly, ground bounce. The ground bounce voltage is related to the inductance present between the device ground and the system ground. Problems may arise when this ground bounce gets transferred to the outside through output buffers driving a logic low value. If the bounce is higher than the V_{IL} threshold of the input being driven, there is a possibility that the glitch will be recognized as a legal logic '1'. Fewer pins and differential signaling eliminate the SSO problem.
- Power: Usually a high-speed serial link will use less power than a parallel link. This is especially true of some of the actively biased/terminated high-speed parallel standards such as *high-speed transistor logic* (HSTL).
- Control Lines Included: Often a parallel interface needs a few lines for

control and enable in addition to the data lines. Serial links have enabling and control capabilities built into most protocols.

A *phase locked loop* (PLL) is used to keep time for the serializer deserializer pair [1]. The PLL is internal to each device and is required to lock to the input clock frequency, perform the correct multiplication factor and maintain its output with minimal jitter. Jitter is the deviation of a signal's timing event from its intended (ideal) occurrence in time. A PLL is used because of its inherent feedback path allowing constant correction if a minor change is seen in the input signal edge position or period. To understand the SERDES technology, it is important to have a basic understanding of how a PLL operates. All SERDES PLLs have an input frequency and an internal core frequency that needs to be synchronized with this frequency. The internal frequency is responsible for the serialization timing. For, without the PLL running, data compression is not possible. There are several key factors to keep in mind for PLL operation: the time it takes to lock, the power consumed, the resolution of each loop correction factor and the effect that jitter has on the circuit.

1.1 Motivation

1.1.1 Problems with SERDES

The main problem in SERDES is *jitter*, the time deviation of the actual signal transition from the expected. The *phase locked loop* (PLL) at the transmit side of the SERDES generates a fast clock signal, which has timing jitter. When this clock signal drives the serializer, the jitter is passed on to the serial data. More jitter is added to the serial data as it propagates through the transmission path. At the receive end the CDR circuit does not properly track the jitter present in the received serial data signal. As a result, when the data is sample using the recovered clock from CDR unit, it results in bit errors.

1.1.2 Problems with PLL Circuit in Microprocessors and Wireless Transceivers

On a broader spectrum, the clock signals generated by the PLL circuits in microprocessors and wireless transceivers suffer from the same timing jitter problem. In the case of microprocessors, it affects the synchronization of the various signals in microprocessors with the jittered clock signal and in the case of wireless transceivers, the problem is similar to that of the SERDES circuit, i.e., a wrong data bit is latched resulting in bit-errors.

1.1.3 Why Do We Need a New Jitter Reduction Technique?

A jitter reduction technique was proposed by Tian Xia *et al* [39]. In this work a jitter test circuit was employed to monitor the PLL jitter performance. A digital control unit was used to calibrate the loop filter parameters dynamically to reduce jitter. The drawback in this technique is the chip area overhead, which consists of capacitors and digital counters. Apart from this there are several other techniques that try to reduce the jitter in the signal generated by the PLL circuit by modifying its loop bandwidth dynamically. The main problem with this is that the settling time of the PLL circuit is affected. To overcome this problem, we need to propose a circuit that reduces jitter in the PLL signal externally.

1.2 Problem Statement

Our goal is to design a jitter reduction circuit for the transmit side PLL, which determines the jitter present and reduces it using only a few logic gates. At the receive side, we propose to extend the idea to reduce the jitter between the recovered clock and data signal by aligning the serial data with the clock. The proposed hardware should have very little area overhead.

1.3 Original Contribution of the Dissertation

The main contribution of this work is the jitter reduction methodology that basically uses the concept of error estimation and reduction. For the transmit side, the error is estimated using the jittered clock signal and the looped-back signal and for the receive side, the error is estimated using the recovered clock and the incoming serial data signal. The reduction is based on the notion of pulse shaping, where the jittered signal pulse shape is changed according to the reference signal. The accuracy of pulse shaping depends on how accurate the reference signal is.

1.4 Summary of Results

In the case of the transmit and the receive side jitter reducer, the jitter is reduced, on average, by 62.24% and 35.88%, respectively and also the BER computed probabilistically is improved from 8.3×10^{-2} to 6.44×10^{-20}.

1.5 Organization of the Dissertation

The dissertation is organized as follows. The introduction about the new jitter reduction work is given in Chapter 1, where the problem is defined. The prior work is split into two chapters. The SERDES architecture and the PLL circuitry are explained in Chapter 2. In Chapter 3, various jitter reduction and testing schemes are explained. The jitter reduction technique for the transmit side is explained with results in Chapter 4 and in Chapter 5 the jitter reduction technique for the receive side and the performance of the SERDES circuit in the presence of both the transmit and the receive side jitter reducers are explained. In Chapter 6, the circuit design issues are addressed and also a SERDES test scheme is presented. Finally, Chapter 7 gives the conclusion and future work.

Chapter 2

Concepts on Serializer-Deserializer (SERDES) and Phase-Locked Loop (PLL) Circuits and Timed Boolean Functions

2.1 SERDES – Introduction

As data rates reach more than 1 Gb/s its becoming difficult to transmit data across multi-drop parallel bus such as the PCI or PCI-X [9] buses. The primary problem is the tolerance in timing skew between parallel wires in these bus standards. To overcome the problem, the parallel bus standards are being replaced by their serial equivalent, such as PCI Express. Timing skew [3] is a problem that can occur on many kinds of computer buses. When signals are transmitted down parallel paths, they will not arrive at exactly the same time due to unavoidable variations in wire transmission properties and transistor sizing, but the signals will arrive close to each other in time. As the frequencies of these circuits increase, this variation will become more and more erratic. If the timing skew is large enough, the clock signal may arrive while the data signal is still transitioning between the previous and current values. If this happens, it will be impossible to determine what value was transmitted from the detected value, resulting in a bit-error. The timing skew arises in parallel buses due to cross talk and signal reflections in the wires, which are very difficult to control at higher frequencies.

In the serial bus, a device called *serializer-deserializer* (SERDES) is used to transmit and receive data over a serial link. The SERDES can be either a standalone device or an ASIC. In essence, a SERDES is a serial transceiver that converts parallel data into a serial data stream on the transmitter side and converts the

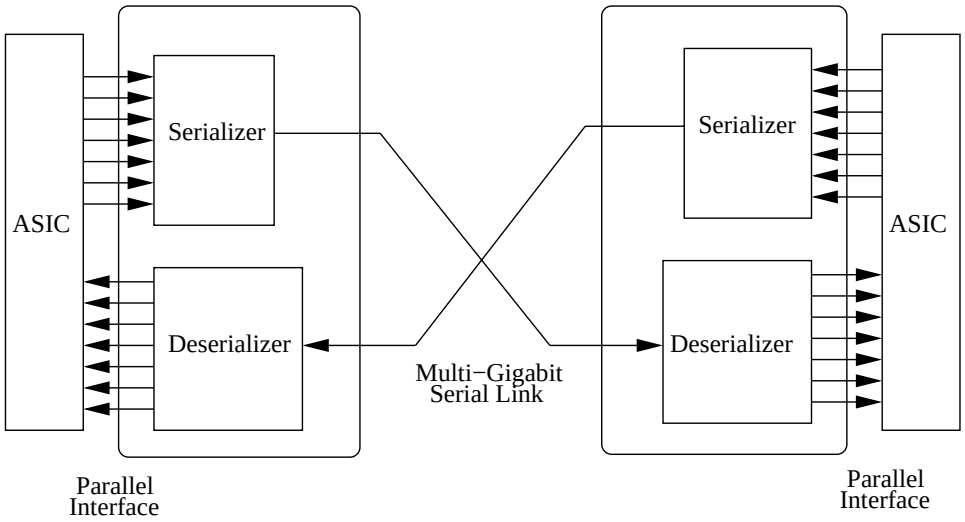


Figure 2.1: SERDES in Serial Data Communications [9]

serial data back into parallel data on the receiver side. The timing skew problem encountered in a parallel bus is solved by embedding the clock signal into the data stream. Since there is no separate clock signal in a serial bus, timing skew between clock and data no longer exists. As a result, a serial bus can usually operate at much higher data rate than a parallel bus in a comparable system environment. Figure 2.1 shows a typical *application specific integrated circuit* (ASIC) application where a SERDES circuit is used. The SERDES serializer and deserializer circuits are placed in the transmit and receive sides of the ASIC, respectively.

2.2 SERDES Architecture

Figures 2.2 and 2.3 show the functional blocks of the SERDES designed for PCI Express and a simple CDR circuit used in the receive side of the SERDES. The parallel data is encoded into serial data using the 8b/10b encoding scheme, where the lower 5 bits are converted into a 6-bit group and the upper 3 bits into a 4-bit group. These groups are concatenated to form 10-bit code word. The 8b/10b encoding is used for DC balancing, i.e., to maintain an equal number of one's and zero's in the transmitted data stream. The data symbols are often referred to as $Dxx.y$, where xx ranges from $0-31$ and y from $0-7$ [2]. Because 8b/10b

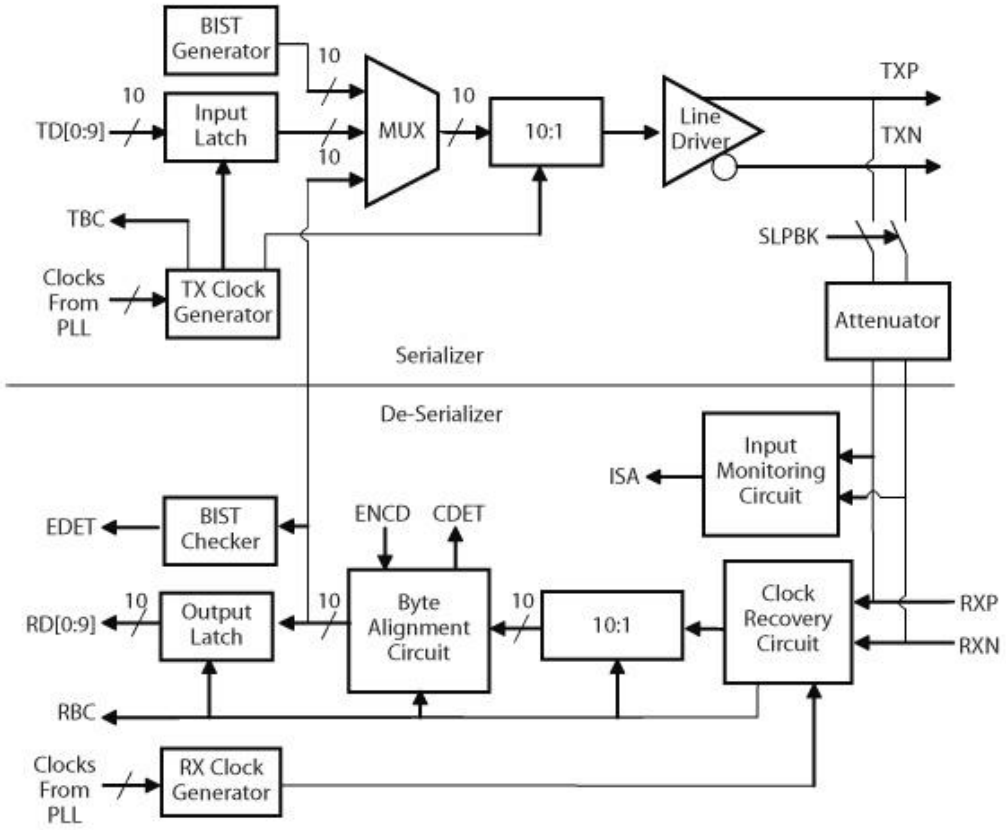


Figure 2.2: Functional Blocks of SERDES [9]

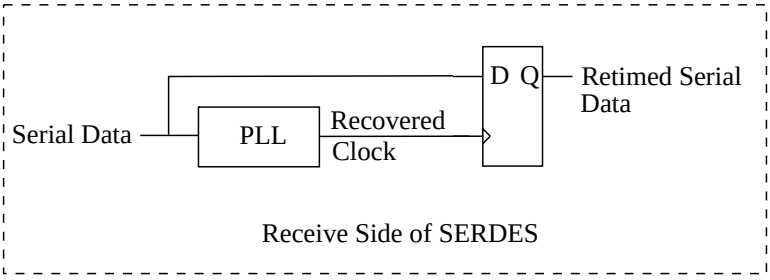


Figure 2.3: A Clock and Data Recovery Circuit

encoding uses 10-bit symbols to encode 8-bit words, each of the 256 possible 8-bit words can be encoded in two different ways, one the bit-wise inverse of the other.

Input		RD = -1	RD = +1
	HGF	fghj	
D.x.0	000	1011	0100
D.x.1	001	1001	
D.x.2	010	0101	
D.x.3	011	1100	0011
D.x.4	100	1101	0010
D.x.5	101	1010	
D.x.6	110	0110	
D.x.P7	111	1110	0001
D.x.A7	111	0111	1000

Figure 2.4: 3b/4b Encoding Table

Figures 2.4 and 2.5 show the tables for both 3b/4b and 5b/6b encoding, where $RD-$ and $RD+$ represent the two different ways for encoding and $RD-$ is the bit-wise inverse of $RD+$.

2.2.1 Transmit Section

The transmit side of the SERDES consists of the *built-in self-test* (BIST) generator, a 10-to-1 multiplexer (10:1) and a line driver. The BIST pattern generator is used to generate various test patterns to perform system level and diagnostic tests. The multiplexer converts the 10-bit parallel data to serial data and is driven by a high speed clock generated using an analog *phase-locked loop* (PLL) circuit. The PLL takes a low frequency clock signal from a crystal oscillator as a reference input. The PLL has to generate a clock signal that has very low jitter in it. For the PCI Express, the amount of jitter allowed is $120ps$ for a clock period of $400ps$. The serial data is then transmitted using a line driver across a 100Ω differential

Input		RD = -1	RD = +1	Input		RD = -1	RD = +1
	EDCBA	abcdei			EDCBA	abcdei	
D.00	00000	100111	011000	D.16	10000	011011	100100
D.01	00001	011101	100010	D.17	10001	100011	
D.02	00010	101101	010010	D.18	10010	010011	
D.03	00011	110001		D.19	10011	110010	
D.04	00100	110101	001010	D.20	10100	001011	
D.05	00101	101001		D.21	10101	101010	
D.06	00110	011001		D.22	10110	011010	
D.07	00111	111000	000111	D.23	10111	111010	000101
D.08	01000	111001	000110	D.24	11000	110011	001100
D.09	01001	100101		D.25	11001	100110	
D.10	01010	010101		D.26	11010	010110	
D.11	01011	110100		D.27	11011	110110	001001
D.12	01100	001101		D.28	11100	001110	
D.13	01101	101100		D.29	11101	101110	010001
D.14	01110	011100		D.30	11110	011110	100001
D.15	01111	010111	101000	D.31	11111	101011	010100
				K.28		001111	110000

Figure 2.5: 5b/6b Encoding Table

terminated *printed circuit board* (PCB) trace.

2.2.2 Receive Section

At the receive side there is an input monitoring circuit that senses the differential line to find whether the differential voltage is greater than $175mV$. For the receive section to be idle the differential signal should be less than $65mV$ according to the PCI Express specification. The received data is retimed with the clock recovered using the *clock and data recovery* (CDR) circuit. The PCI Express allows a timing jitter of 60% of the input bit time. The CDR circuit has to have a bandwidth wide enough to track this timing jitter or should filter out the high frequency jitter. Once the data is retimed, the serial data is latched into the demultiplexer using the high speed recovered clock. The demultiplexer then uses a low frequency clock that has a constant phase relation with the high speed recovered clock to provide the parallel data, which is at its original speed. A byte alignment circuit is used to align the encoded data at its 8b/10b encoded byte boundaries. During the encoding process the original serial data is appended with special start and stop bits. The alignment circuit looks for these special characters and aligns the parallel data with these special characters and transmits the aligned data to the ASIC. In the test mode the parallel data is compared with the expected data for performing the diagnostic tests.

2.3 SERDES Design Features

2.3.1 Jitter Performance

A good SERDES design is judged based on its jitter performance. The PCI Express allows a maximum jitter of $120ps$ for the serializer and $240ps$ for the deserializer, for a clock period of $400ps$. A small jitter in the serializer output means that the received data will have a low *bit error rate* (BER). The serializer jitter is mainly due to the high-speed clock generated from the PLL circuit. The

PLL jitter has to be reduced using an on-chip jitter tracking circuit that monitors the loop bandwidth and tunes it accordingly to reduce the jitter in the clock generated. At the receive side the jitter performance is judged by the jitter present in the incoming the serial data. The clock recovered by the CDR circuit retimes the serial data, so a bit error will occur if either the clock or data is too early or late. Therefore, a good jitter reduction mechanism should reduce the timing jitter between the clock and data signal.

2.3.2 Power and Area

In applications where there are multiple SERDES cores, each requiring its own PLL circuit to generate the fast clock, power and area can be saved by driving a group of SERDES cores in parallel using the same PLL circuit as shown in Figure 2.6.

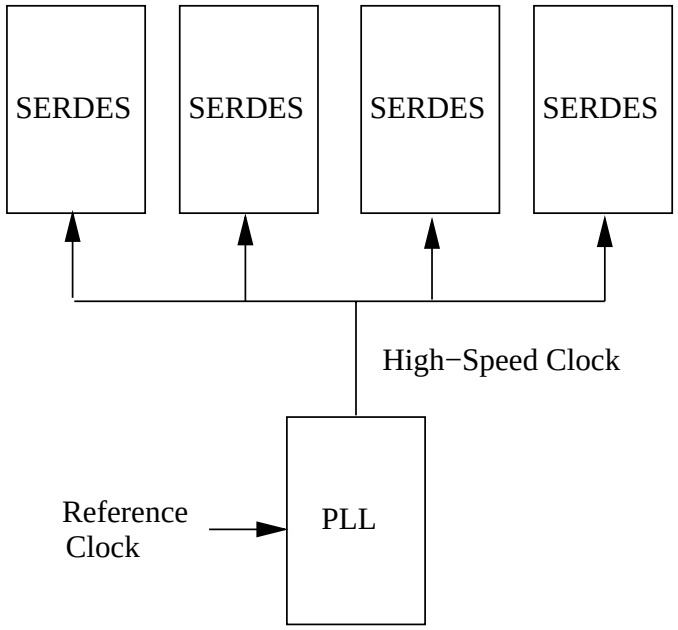


Figure 2.6: Parallel SERDES Cores Driven by the Same PLL Circuit Clock [9]

At the receive side, to reduce power and area, instead of using a PLL circuit, a *delay-locked loop* (DLL) circuit can be used. The DLL circuit uses a *voltage*

controlled delay line (VCDL) to change the frequency of its local clock signal so that it is phase locked to the incoming data signal. The DLL circuit requires a high-speed clock for its operations. Both the serializer and the deserializer can uses the same PLL circuit that generates the high-speed clock, thereby reducing the area and power.

2.3.3 SERDES Test

The current *automatic test equipment* (ATE) available can operate at $1.5GHz$, but are not capable of testing the SERDES cores at the required speed, which is either $10Gb/s$ for Ethernet or $40Gb/s$ for *synchronous optical network* (SONET). So, the ASIC designers provide BIST pattern generators that generate *pseudo random bit sequence* (PRBS) patterns and the corresponding pattern checker. To perform the jitter tolerance test at the receive side, the PRBS pattern is looped back to the receiver through an on-chip or external jitter injection circuit. The receiver is now tested with this input jittered signal and the jitter transfer function is determined.

2.4 Phase-Locked Loop – Introduction

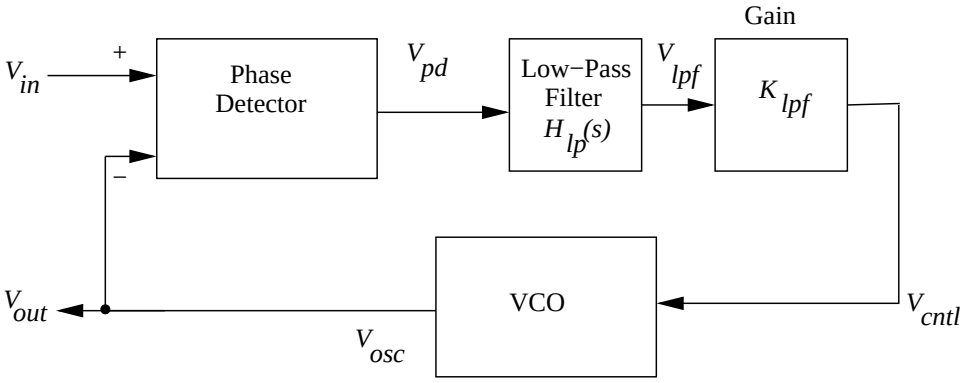


Figure 2.7: Basic PLL Architecture [16]

Figure 2.7 shows the basic architecture of a PLL circuit [16], which consists of a phase detector, a low-pass filter, a gain stage, and a *voltage controlled oscillator* (VCO). The phase detector produces an output proportional to the phase

difference of the input and the feedback signal. The low-pass filter extracts an average voltage, which is amplified by the gain stage to give the control voltage for the VCO.

Let V_{in} be an input sinusoidal signal. The phase detector is a analog multiplier with a relationship given by:

$$V_{pd} = K_M V_{in} V_{osc} \tag{2.1}$$

where V_{osc} is the VCO output and K_M is the multiplication constant. Let V_{in} and V_{osc} be given as follows:

$$V_{in} = E_{in} \sin(\omega t) \tag{2.2}$$

$$V_{osc} = E_{osc} \sin(\omega t - \phi_d) \tag{2.3}$$

The term ϕ_d represents the phase difference between the input and the oscillator signal and the reason for having 90^o phase shift is that when the phase difference is zero, the average output of the phase detector is zero. The output of the phase detector is given by the following equation:

$$V_{pd} = K_M V_{in} V_{osc} = K_M E_{in} E_{osc} \sin(\omega t) \cos(\omega t - \phi_d) \tag{2.4}$$

Using a trigonometric identity, we have:

$$V_{pd} = K_M \frac{E_{in} E_{osc}}{2} [\sin(\phi_d) + \cos(2\omega t - \phi_d)] \tag{2.5}$$

The function of the low-pass filter is to remove the high frequency component and the output of the low-pass filter is given as follows:

$$V_{lpf} = K_{lpf} V_{pd} \tag{2.6}$$

where K_{lpf} is the gain of the low-pass filter. The VCO control voltage is then given as follows:

$$V_{cntl} = K_{lpf} K_M \frac{E_{in} E_{osc}}{2} \sin(\phi_d) \tag{2.7}$$

For small ϕ_d, the control voltage is approximated as follows:

$$V_{cntl} = K_{lpf} K_{pd} \phi_d \tag{2.8}$$

From the above equation we see that the control voltage is directly proportional to the phase difference, and the constant K_{pd} is given as follows:

$$K_{pd} = K_M \frac{E_{in} E_{osc}}{2} \tag{2.9}$$

2.4.1 Operation of PLL Circuit

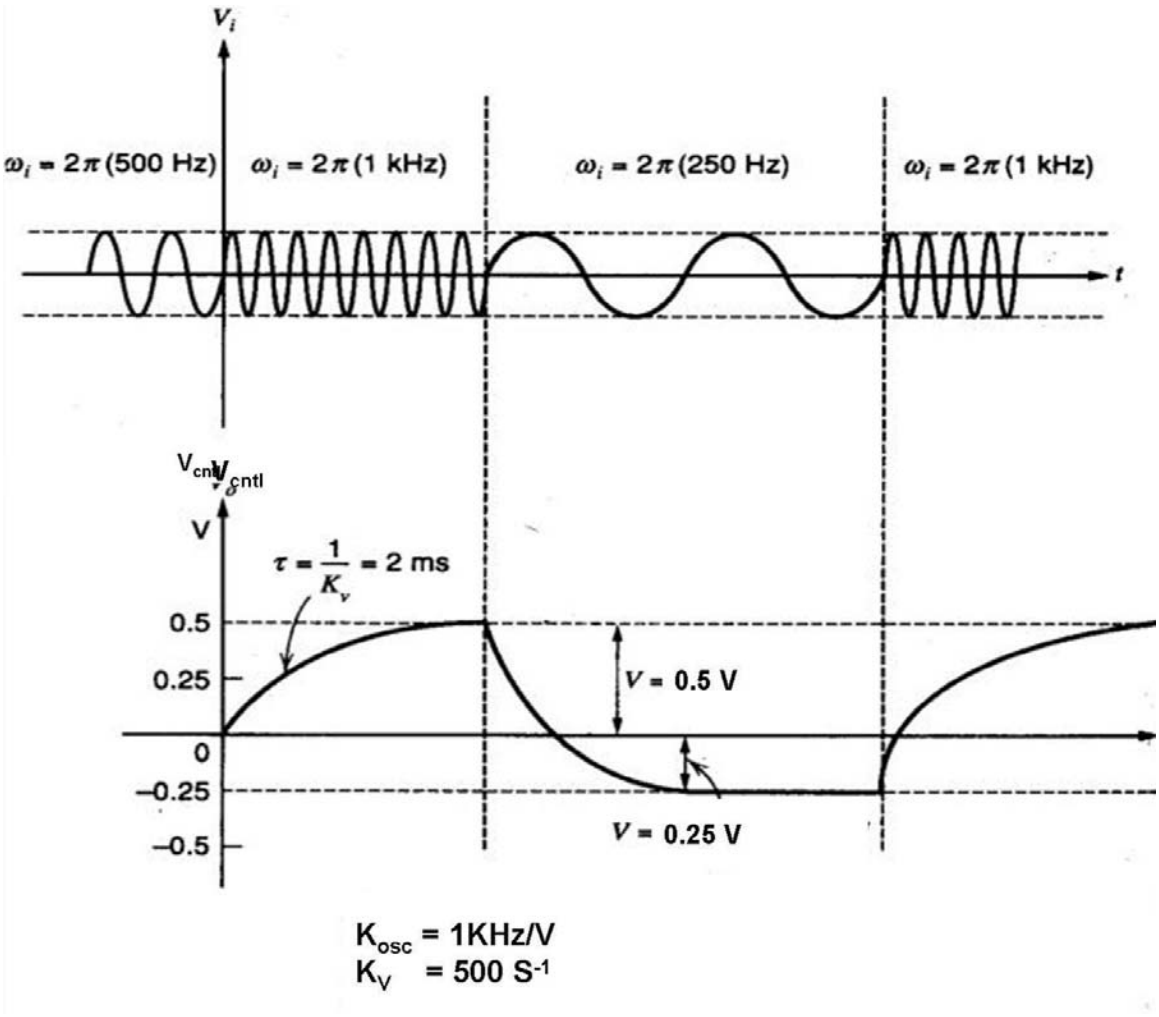


Figure 2.8: Response of a PLL to an Input Analog Signal of Varying Frequency [10]

Assume that the phase difference ϕ_d is initially zero and that the input signal is locked with the VCO signal frequency. As the input signal frequency increases, it starts to lead the feedback signal and the phase detector produces an average

positive voltage that is filtered by the low pass filter and is applied to the VCO, thereby increasing its frequency. The opposite happens when the input signal frequency reduces, producing an average negative voltage. The VCO reduces its frequency accordingly. The oscillator's output frequency is given as follows:

$$\omega_{osc} = K_{osc} V_{cntl} + \omega_{fr} \tag{2.10}$$

where ω_{fr} is the free-running frequency of the VCO when its control voltage is zero and K_{osc} is a constant relating the change in frequency to control voltage. The control voltage is now given as:

$$V_{cntl} = \frac{\omega_{in} - \omega_{fr}}{K_{osc}} \tag{2.11}$$

where ω_{in} is the frequency of the input signal, which is equal to the frequency of the oscillator output (ω_{osc}). Finally the phase difference is determined as follows:

$$\phi_d = \frac{V_{cntl}}{K_{lp} K_{pd}} = \frac{\omega_{in} - \omega_{fr}}{K_{lp} K_{pd} K_{osc}} \tag{2.12}$$

Figure 2.8 shows the response of the PLL circuit to a varying analog input signal. For a VCO free-running frequency of $500KHz$ and an input signal frequency of $500KHz$, the corresponding control voltage is zero. As the input signal frequency increases to $1KHz$, the control voltage also increases to $0.5V$ to increase the VCO frequency and when the input signal frequency reduces to $250KHz$, the control voltage changes by $-0.75V$ from its previous point to decrease the VCO frequency. The time constant τ is $2ms$, which is the inverse of the loop bandwidth K_v, which in turn is the product $K_{pd} K_{lp} K_{osc}$.

2.4.1.1 Analysis of PLL Circuit in Locked Condition

Consider a PLL in the locked condition [16] and assume the input and output waveform can be expressed as follows:

$$V_{in}(t) = V_A \cos \omega_1 t \tag{2.13}$$

$$V_{out}(t) = V_B \cos(\omega_1 t + \phi_o) \tag{2.14}$$

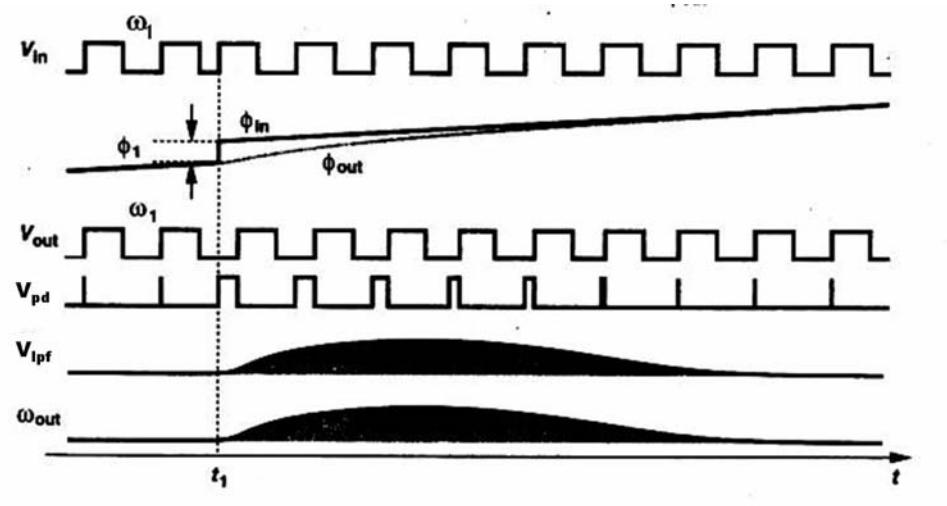


Figure 2.9: Response of a PLL to a Phase Step [16]

where ϕ_o is the static phase error. Suppose as shown in Figure 2.9, the input has a phase step of ϕ_1 at $t = t_1$, then the total input phase at t_1 is given as follows:

$$\phi_{in} = \omega_1 t + \phi_1 u(t - t_1)^2 \tag{2.15}$$

Since the output of the LPF (V_{lpf}) does not change instantaneously, the VCO initially continues to oscillate at ω_1. The growing phase difference between the input and the output then creates wide pulses at the output of the PD (V_{pd}), forcing V_{lpf} to rise gradually. As a result, the VCO frequency begins to change, attempting to minimize the phase error. The loop is not locked during the transient phase because the phase error varies with time. Since ϕ_{in} has changed by ϕ_1, the variation in the VCO frequency is such that the *area* under ω_{out} provides an additional phase of ϕ_1 in ϕ_{out};

$$\int_{t_1}^{\infty} \omega_{out} dt = \phi_1 \tag{2.16}$$

Thus, when the loop settles, the output becomes equal to:

$$V_{out}(t) = V_B \cos\left[\omega_1 t + \phi_o + \phi_1 u(t - t_1)\right] \tag{2.17}$$

Consequently, as shown in Figure 2.9, ϕ_{out} gradually approaches ϕ_{in}.

2.5 Building Blocks of PLL Circuit

In this section, we will see how to design a phase/frequency detector with a charge pump to generate the control voltage and a voltage controlled oscillator.

2.5.1 Phase/Frequency Detector and Charge Pump

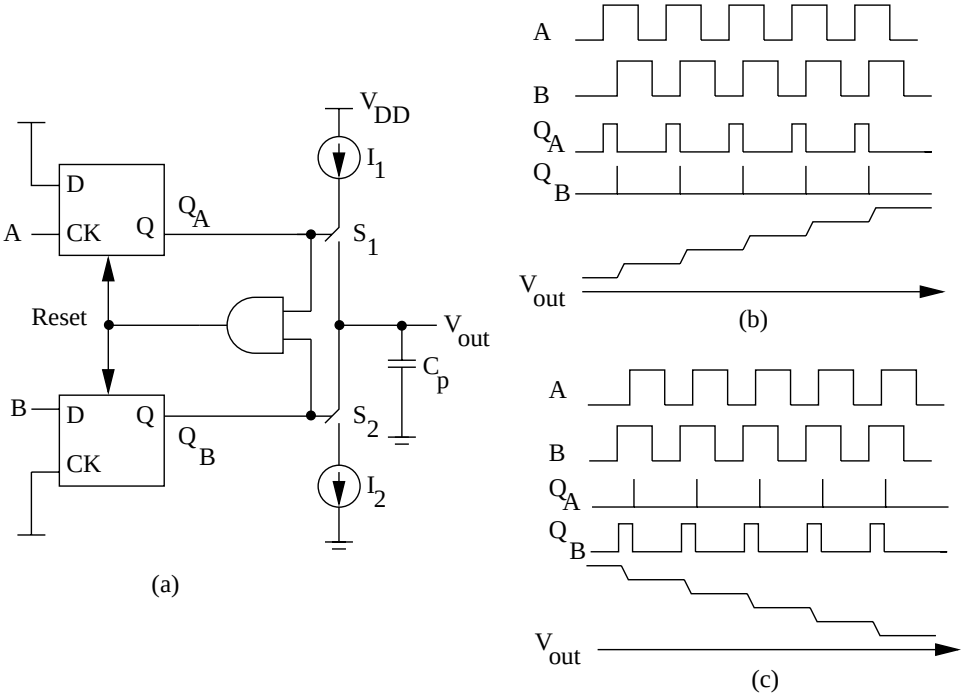


Figure 2.10: PFD with Charge Pump [30]

The *phase/frequency detector* (PFD) is implemented using sequential logic as shown in Figure 2.10(a) [30]. When signal A is leading B or when its frequency is higher than the frequency of the signal B as shown in Figure 2.10(b), the signal Q_A goes high and remains high as long B is not rising. When signal B rises, momentarily both Q_A and Q_B are high and at that instant the reset signal goes high and both Q_A and Q_B are reset. The opposite is true when either the signal A is lagging signal B or has a lower frequency than B as shown in Figure 2.10(b) then, first the signal Q_B goes high and when signal A rises, both Q_B and Q_A

are high. At this instant the reset signal goes high and resets both Q_B and Q_A. The width of the signals Q_A and Q_B depends on the phase difference $\phi_A - \phi_B$. The signals Q_A and Q_B are known as UP and DOWN signals and they control the flow of charge across the capacitor C_p. The UP signal controls the switch S_1 and the DOWN signal controls the switch S_2. I_1 and I_2 are current sources that either supply a constant current to charge the capacitor C_p or draw a current from the capacitor, thereby discharging it. The timing waveform shows both the cases where either A is leading or lagging B . The pulse Q_A or Q_B is proportional to the phase difference and for that period either the switch S_1 or S_2 is turned on, the current from the current source I_1 charges the capacitor or the charge in the capacitor is discharged through the current source I_2 producing the step voltage V_{out}.

2.5.2 Gilbert Cell as Phase Detector

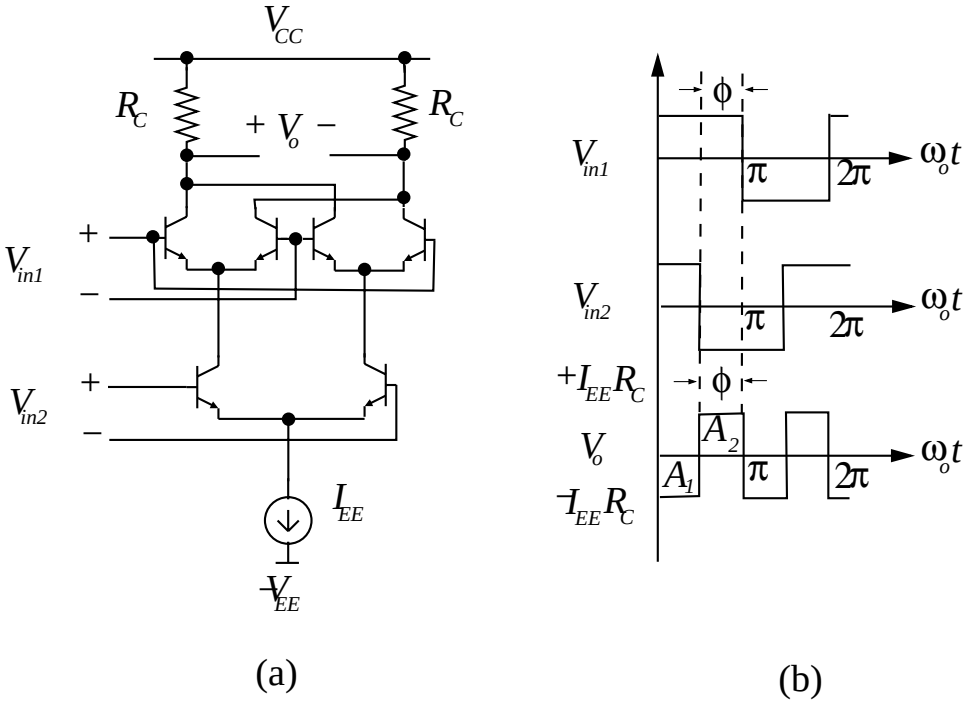


Figure 2.11: Gilbert Phase Detector with Input and Output Waveforms [10]

In frequency translation, signals at two different frequencies are applied to the two inputs of the Gilbert cell, and the sum or the difference frequency component is taken from the output [10]. If unmodulated signals of identical frequency ω_o are applied to the two inputs, the circuit behaves as a phase detector (Figure 2.11(a)) and produces an output whose DC component is proportional to the phase difference between the two inputs. For example, consider the two input waveforms in Figure 2.11(b), which are applied to the Gilbert cell. The DC component of the output waveform is given by:

$$V_{average} = \frac{1}{2\pi}\int_0^{2\pi} V_o(t)d(\omega_o t) \tag{2.18}$$

$$= \frac{-1}{\pi}(A_1 - A_2) \tag{2.19}$$

where A_1 and A_2 are as shown in Figure. Thus,

$$V_{average} = -\left[I_{EE}R_C\frac{(\pi-\phi)}{\pi} - I_{EE}R_C\frac{\phi}{\pi}\right] \tag{2.20}$$

$$= I_{EE}R_C\left(\frac{2\phi}{\pi} - 1\right) \tag{2.21}$$

Figure 2.12 shows the equivalent CMOS Gilbert cell.

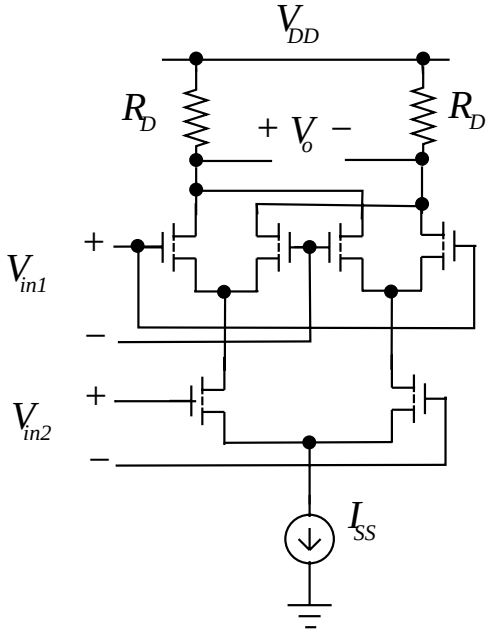


Figure 2.12: CMOS Gilbert Cell [30]

2.5.3 Voltage Controlled Oscillator

Figure 2.13 shows the most common way of implementing a VCO using digital inverters. These five inverters form a ring oscillator whose frequency of oscillation can be controlled by a voltage. Each of the n inverters offer a 90^o phase shift at unity gain frequency, and is guaranteed to provide a 180^o phase shift when the gain of the oscillator is greater than unity. Thus, according to the Barkhausen

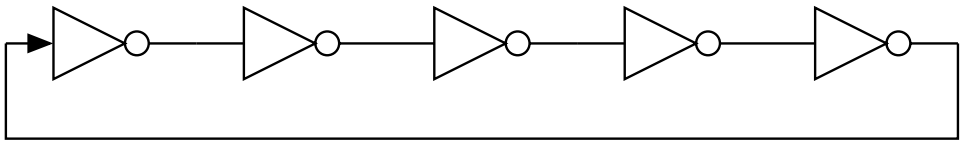


Figure 2.13: A Ring Oscillator Using Five Digital Inverters [16]

criteria [16] the ring oscillator will start to oscillate with a frequency given as follows:

$$f_{osc} = \frac{1}{T} = \frac{1}{2n\tau_{inv}} \tag{2.22}$$

where T is the time period of the periodic signal and τ_{inv} is the inverter delay. Thus the free running frequency of the oscillator can be changed by changing the number of inverter stages used.

2.5.3.1 LC Cross Coupled Voltage Controlled Oscillator

Figure 2.14 shows the LC cross-coupled voltage controlled oscillator and it consists of two cascaded *common source* (CS) stages [10]. The load for each stage is made of an inductor L, a *variable capacitor* (varactor) C and a resistor R [10]. At the resonance frequency given by the following equation:

$$f_{osc} = \frac{1}{2\pi\sqrt{LC}} \tag{2.23}$$

the phase shift due to the inductor L is canceled by the phase shift of the capacitor C and so, at resonance, the total phase shift around the loop is 360^o with each CS stage contributing a 180^0 phase shift. According to the Barkhausen criteria,

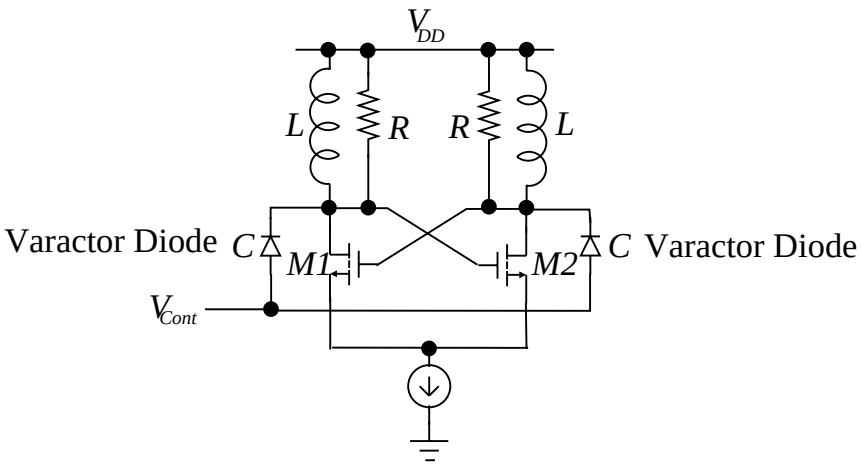


Figure 2.14: LC Cross Coupled VCO [10]

the circuit starts oscillating at resonance if the gain of the circuit is greater than 1, i.e., the following condition must be satisfied:

$$gm_1 \; R \; gm_2 \; R \geq 1 \tag{2.24}$$

where gm_1 and gm_2 are the transconductances of transistors $M1$ and $M2$, respectively. The frequency of oscillation can be changed by changing the value of the capacitance C and hence, a varactor (reverse biased pn junction diode) is used. The capacitance value can be changed according to the following equation:

$$C_{var} = \frac{C_o}{\left(1 + \frac{V_R}{\phi_B}\right)^m} \tag{2.25}$$

where C_o is the zero-bias value, V_R the reverse-bias voltage, ϕ_B the built-in potential of the junction and m a value typically between 0.3 and 0.4, The control voltage V_{cont} is used to change the reverse-bias voltage V_R of the varactor, thereby changing the capacitance C and the frequency of oscillation

2.6 Delay-Locked Loop

Figure 2.15(a) shows the block diagram of a delay-locked loop used for generating four clock signals, each separated by $\triangle T$ seconds from a reference clock signal [16].

The same four clock signals can be generated using a PLL circuit, but the noise in the clock signals will be difficult to minimize as compared to the noise in the clock signals generated by the DLL circuit. Primarily this is because the PLL has

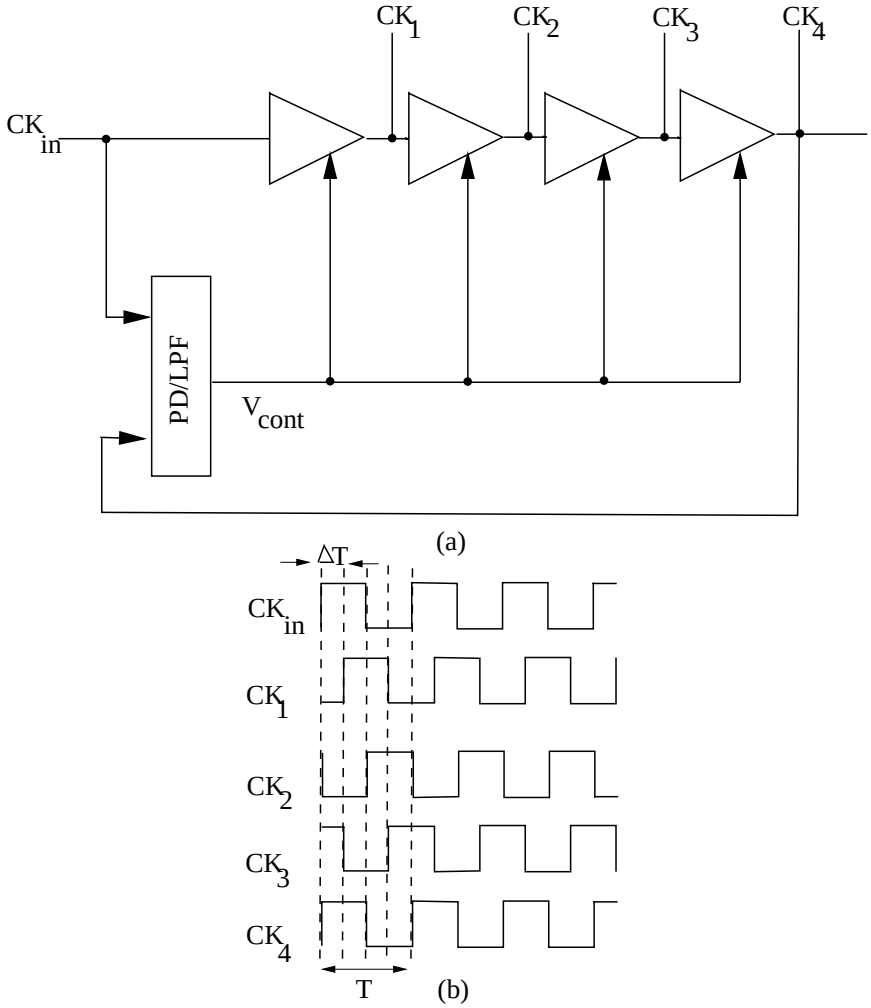


Figure 2.15: Delay-Locked Loop Generating Clock Edges [16]

a internal oscillator whose noise is recirculated back to the input, whereas in the case of the DLL the noise in the input disappears once the clock signal reaches the output and is not recirculated. The other major difference is that, since the DLL has no internal oscillator, it cannot be used in frequency synthesis, i.e., it cannot generate signals of frequency other than the input reference signal frequency and it can be used only to generate phase shifted signals. The timing waveform in Figure 2.15(b) shows that each of the four clock edges CK_1, CK_2, CK_3 and CK_4 are shifted by a time period of $\triangle T$ seconds from each other. The clock edge CK_4

is in phase with the reference clock signal CK_{in}. The delay $\triangle T$ varies due to temperature and process variations and therefore, to perfectly control the delay, the fourth clock signal is looped back as in a PLL circuit to a phase detector.

The phase detector determines the phase error and accordingly produces a proportional voltage, which is low-pass filtered and applied to each of the buffers producing the respective clock signals. The buffers are built using simple inverter gates, whose delay is varied by varying their drain current. So, as the control voltage changes, the drain current also changes and so the delay of each stage changes. The DLL circuit borrows the principle of phase error detection and correction from the PLL operation. The individual stages used in the DLL circuit are known as the *voltage controlled delay line* (VCDL).

2.7 Timed Boolean Functions – Introduction

In this section two models for representing the timing information of the digital circuits are presented. McCluskey was the first to use Boolean expressions along-with the timing information to represent a given circuit and the model is known as a *transient output function* (TOF) [25].

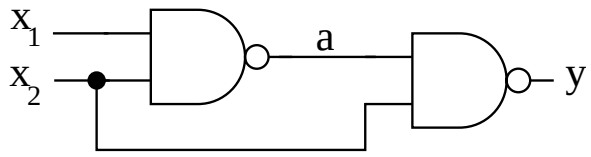


Figure 2.16: Modeling with TOF

Consider Figure 2.16, where the TOF of the circuit is given below:

$$y(t) = (x_1(t - \tau_{x_1ay}) \cdot x_2(t - \tau_{x_2ay})) + \overline{x_2(t - \tau_{x_2y})} \tag{2.26}$$

where τ_{x_1ay}, τ_{x_2ay} and τ_{x_2y} are the delays of the paths x_1ay, x_2ay and x_2y.

The second model known as the *timed Boolean function* (TBF) was proposed by Lam and Brayton [19] and they extended the work done by McCluskey. In our

new jitter reduction work explained in Chapter 4, we use the same notation used by Brayton to derive the timed Boolean expressions for our proposed circuits, since it is simple and easy to understand. The timing formalism proposed by Brayton has time as an argument and has the following properties:

1. This formalism representing the circuit at a particular time t gives the output values of the circuit at t.

2. This formalism reduces to the ordinary Boolean function for the circuit at t greater than or equal to the settling time of the circuit.

Consider a buffer with equal rising and falling delay. Let $x(t)$ and $y(t)$ represent the input and the output of the buffer at time t, respectively, and $\hat{x}$ and $\hat{y}$ represent the steady state input and output, respectively. The output and the input of the buffer are related through the equation $y(t) = x(t-d)$, implying that the buffer delays the input by d, which is the settling time of the buffer. If the last transition at the input occured at $t = 0$, then after delay d, $y(t) = x(t-d)$ involves the input at $t \geq d$, i.e., the steady state input and hence, for $t \geq d$, $y(t) = x(t-d)$ reduces to the ordinary Boolean function, namely, $\hat{y} = \hat{x}$

The definition for the timed Boolean function is given below:

1. A *binary signal space* $B(t)$ is a collection of mappings $f : R \rightarrow B$, where R is the set of real numbers and $B = 0, 1$.

2. A TBF is any function with domain $B^n(t)$ and range $B(t)$. TBF $F : B^n(t) \rightarrow B(t)$ satisfies the following properties:

   - The identity function F (i.e., $F(v)(t) = v(t), v(t) \epsilon B(t)$) is a TBF.
   - If $G(t) : B^{n1}(t) \rightarrow B(t)$ and $H(t) : B^{n2}(t) \rightarrow B(t)$ are TBF's, then, $\overline{G}, G \cdot H$, and $G + H$ are also TBF's.
   - If $F(t)$ is a TBF, then, for any function $\phi : R^n \rightarrow R$, $F(\phi)$ is also a TBF.

2.7.1 Modeling Timing Behaviors

The TBF of a given circuit is obtained by first decomposing the complex gates into simple gates and deriving their TBF representations. A few examples are given below:

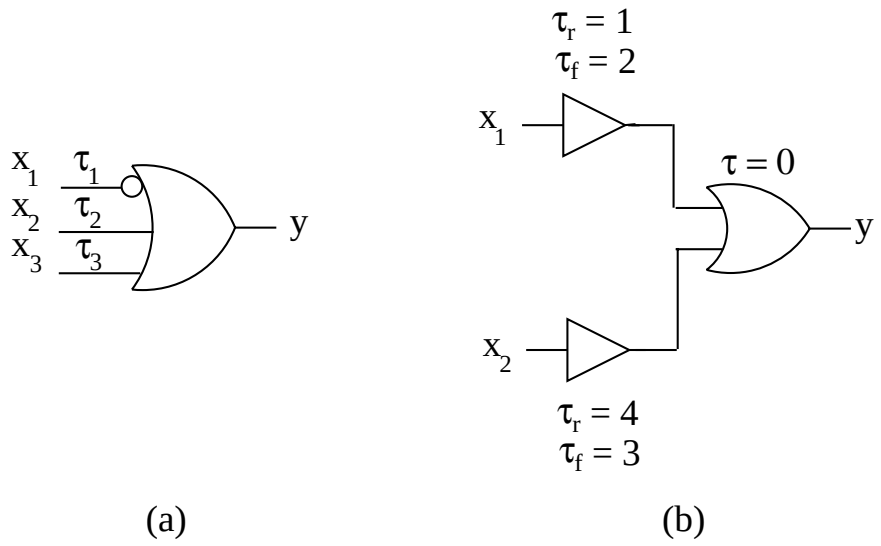


Figure 2.17: Modeling with TBF's [19]

1. *Gates characterized by a single delay for each input-output pair:* In this case, a gate's delay is modeled by the delays of input-output pairs. The complex gate in Figure 2.17(a) has three inputs; input x_i has a delay τ_i to the output, both rising and falling. This gate is modeled by the following TBF:

$$y(t) = \overline{x_1}(t - \tau_1) + x_2(t - \tau_2) + x_3(t - \tau_3) \tag{2.27}$$

2. *Buffer with different rising and falling delays*: Let τ_r and τ_f be the rising and the falling delay of a buffer. The TBF for the buffer with rising delay greater than the falling delay ($\tau_r > \tau_f$) is given as follows:

$$y(t) = x(t - \tau_r) \cdot x(t - \tau_f) \tag{2.28}$$

For the case $\tau_r < \tau_f$, the TBF is given as follows:

$$y(t) = x(t - \tau_r) + x(t - \tau_f) \tag{2.29}$$

3. *Gates with different rising and falling delays for each input-output pair:* In Figure 2.17(b) an OR gate is shown with two inputs: input x_1 has a rising delay of 1 and a falling delay of 2, while input x_2 has a rising delay of 4 and a falling delay of 3. The buffer modeling input 1 has the following TBF:

$$x_1(t-1) + x_1(t-2) \tag{2.30}$$

and the buffer modeling input 2 has the following TBF:

$$x_2(t-4) \cdot x_2(t-3) \tag{2.31}$$

Therfore, the OR gate has the following TBF:

$$y(t) = x_1(t-1) + x_1(t-2) + x_2(t-4) \cdot x_2(t-3) \tag{2.32}$$

2.7.2 Circuit Formulation

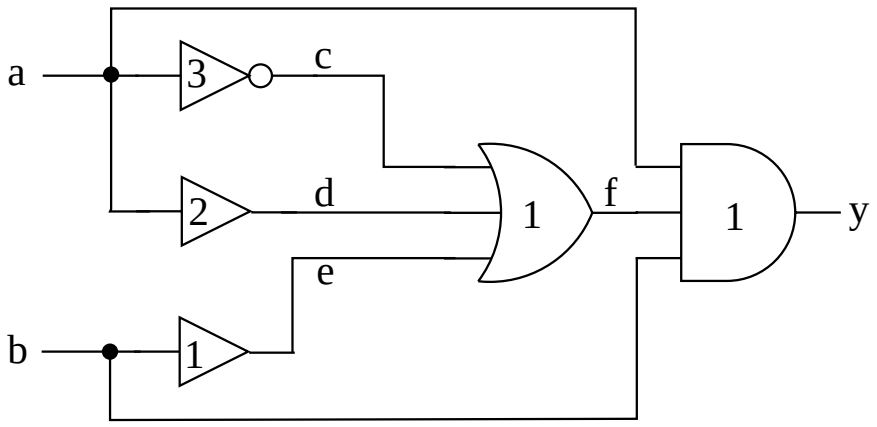


Figure 2.18: An Example Circuit for TBF [19]

Once all of the components are represented by TBF's, the TBF for the circuit can be derived by indentifying the timed variables corresponding to the ports

connected to the same net. Consider Figure 2.18, where the TBF at node f is obtained by composing the TBf's of the inverter and the buffers with that of the OR gate. Thus the TBF at f is:

$$f(t) = a(t-3) + \overline{a}(t-4) + b(t-2) \tag{2.33}$$

The TBF at node y is obtained by composing the TBF at f with that of the AND gate giving:

$$\begin{aligned} y(t) &= a(t-1)b(t-1)f(t-1) && (2.34) \\ &= a(t-1)b(t-1)(a(t-4) + \overline{a}(t-5) + b(t-3)) && (2.35) \end{aligned}$$

2.8 Summary

In this section various building blocks of the PLL circuit were explained with figures and examples showing how the PLL circuit captures and locks onto a reference signal and also the SERDES architecture was explained. In addition, the formulation of a circuit with timing information using timed Boolean functions was also explained.

Chapter 3
Jitter Fundamentals – Reduction and Testing

In a multi-gigabit SERDES, its jitter performance is one of the most important parameters for judging the robustness of the design since the bit error rate is directly affected by the jitter performance [9]. For the serializer, a small amount of jitter means that it is less likely that a bit error will occur when the data is received by the Deserializer. There are many factors that can affect a Serializer's output jitter but the key is to keep the high-speed clock that is used for clocking out the serial data as jitter-free as possible. On the receiver side, the deserializer's jitter performance is judged by the maximum amount of jitter riding on the incoming data stream that it can tolerate. Since the received data is retimed by latching the data with the recovered clock, a bit error can occur only if either the clock or data is too early or late. So, in general the deserializer's input jitter tolerance can be improved by making the clock less likely to be early or late and/or making the data less likely to be early or late. In other words, the jitter on the recovered clock and the received data has to be reduced.

3.1 Jitter Fundamentals

Jitter is the deviation of a signal's timing event from its intended (ideal) occurrence in time, as shown in Figure 3.1. Jitter is expressed in absolute time or normalized to a *unit interval* (UI). A UI is the ideal or average time duration of a single bit or the reciprocal of the average data rate [28]. *Total jitter* (TJ) has two subcategories, *deterministic jitter* (DJ) and *random jitter* (RJ). The classification of TJ is shown in Figure 3.2. The TJ's *probability density function* (PDF) is equal

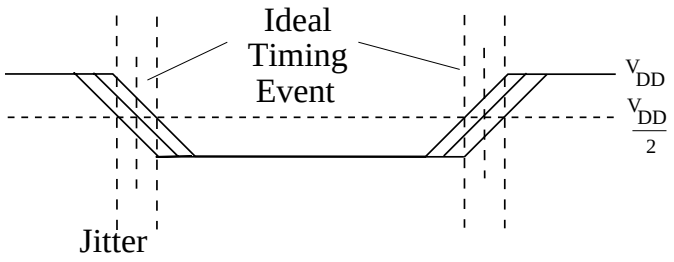


Figure 3.1: Timing Jitter [28]

to the convolution of its RJ and DJ components.

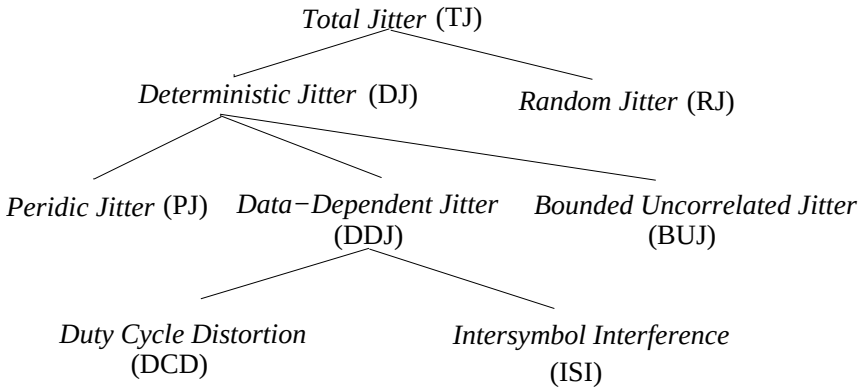


Figure 3.2: Jitter Classification [28]

DJ in turn comprises several subcomponents [28]. *Sinusoidal jitter* or *periodic jitter* (PJ) refers to periodic variations of signal edge positions over time. Possible causes of PJ are electromagnetic interference sources such as power supplies. *Bounded uncorrelated jitter* (BUJ) is typically due to coupling, for example, from adjacent data-carrying links or on-chip random logic switching. *Data-dependent jitter* (DDJ) corresponds to a variable jitter that depends on the bit pattern transmitted on the link under test. DDJ does not describe jitter induced by crosstalk resulting from coupling with other signal paths. DDJ in turn has two subcomponents. The first DDJ subcomponent, *duty-cycle distortion* (DCD), describes a jitter amounting to a signal having unequal pulse widths for high and low logic values. Causes of DCD can be voltage offsets between the differential inputs, and

differences between the system's rise and fall times. The second DDJ subcomponent, *intersymbol interference* (ISI) is a most common type of jitter that occurs in a typical wireless communication environment. The data from the transmitter is distorted due to the bandwidth limitation of the channel. The ISI depends on the transmitted bit pattern. With ISI, the timing of each edge of the transmitted signal depends on the bit pattern preceding this edge. Different edge patterns have different frequency components. Fast-changing edge patterns behave as high-frequency signals; slow-changing edge patterns behave as slow-frequency signals. Because of the channel's filtering effects, different edge patterns propagate at different speeds through the channel. The difference in propagation speeds cause bits to smear into adjacent bits, resulting in ISI.

3.1.1 Random Jitter

RJ comes from device noise sources, for example, thermal effects and flicker [28]. An example of device noise is shot noise, which is related to a transistor's fluctuation in current flow. Thermal noise is a component of device noise. Electron scattering causes thermal noise when electrons move through a conducting medium and collide with silicon atoms or impurities in the lattice. Higher temperatures result in greater atom vibration and increased chances of collisions. *Flicker noise*, or 1/frequency noise, results from the random capture and emission of carriers from oxide interface traps, which affects carrier density in a transistor. Engineers commonly model RJ by the Gaussian distribution function:

$$J_{RJ}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x)^2}{2\sigma^2}} \tag{3.1}$$

where $J_{RJ}(x)$ denotes the RJ PDF, σ is the standard deviation of the Gaussian distribution and x is the time displacement relative to the ideal time position. Hence, a Gaussian RJ is completely specified by a single parameter – its standard deviation.

3.1.2 Deterministic Jitter

DJ arises from the interaction of different system components [28]. Its major causes include electromagnetic interference, crosstalk, signal reflection, driver slew rate, skin effects and dielectric loss. Electromagnetic interference is the interference from radiated or conducted energy that comes from other devices or systems. Such radiation can induce currents on signal wires and power rails, and alter the signal voltage biases or the reference voltages. Impedance mismatch between the cables or traces and a terminating resistor contributes to signal reflections. As a signal propagates and reaches the receiver, part of the signal energy reflects back toward the transmitter. It is possible to estimate the percentage of reflected energy relative to signal energy. Mismatches in the terminating resistance cause electrons to literally bounce back to the transmitter. This corrupts the succeeding bits and reduces the signal-to-noise ratio. The reflected signal energy bounces back and forth until it dissipates completely. As it bounces, it adds to the original signal out of phase, resulting in jitter.

Above a certain frequency, transmitting conductors experience a skin effect. This is a phenomenon whereby at high frequencies conductor self-inductance causes the current flow to concentrate on the surface of a conducting medium. The onset frequency is a function of the conductor's cross-sectional area, impedance and other material physical parameters. The skin effect increases the conductor's resistance because of the reduction in effective cross-sectional area and leads to increased attenuation of a signal's high-frequency contents. The results are longer rise and fall times, and degraded signal amplitudes. Dielectric loss results from the delay of polarization in the dielectric material when it is subject to a changing electric field. In an ideal lossless material, the current leads the voltage by 90 degrees. But in real material, the delay in polarization creates a phase lag between the external electric field and the resonating molecules, which leads to a phase difference in current, thus causing a power loss. Above some frequencies, dielectric losses dominate skin effect losses because dielectric losses are proportional to

the frequency, while skin effect losses are proportional to the frequency's square root.

The signal slew rate depends on the signal driver's ability to drive its load. A strong driver can provide a fast slew rate and drive higher-frequency signals. When a high-frequency signal's driver is weak, the signal at the opposite end of the wire might not have enough time to rise or fall to the desired signal high or low value.

3.1.2.1 Duty-cycle Distortion (DCD) Model

The sum of two δ functions can represent the jitter due to DCD.

$$J_{DCD}(x) = \frac{\delta(x - \frac{W}{2})}{2} + \frac{\delta(x - \frac{W}{2})}{2} \tag{3.2}$$

where $J_{DCD}(x)$ is the DCD PDF, W is the peak-to-peak DCD magnitude, and x is the time displacement relative to the ideal time position. The two δ functions represent the rising and falling edges of the signal. The magnitude of each δ function is 1/2 because the equation assumes that there are equal numbers of rising and falling transitions in the transmitted signal.

3.1.2.2 Periodic Jitter (PJ) Model

A summation of cosine functions with different phases and amplitudes provides a model for PJ:

$$PJ(t) = \sum_{i=0}^{N} A_i cos(\omega_i t + \theta_i) \tag{3.3}$$

where $PJ(t)$ denotes the total periodic jitter, N is the number of cosine components (tones), A_i is the corresponding amplitude, ω_i is the corresponding angular frequency, t is the time and θ_i is the corresponding phase. The following equation describes the PDF of a single-tone PJ:

$$J_{PJ}(x) = \begin{cases} \frac{1}{\pi\sqrt{A^2 - x^2}} & \mid x \mid < A \\ 0 & \mid x \mid \geq A \end{cases} \tag{3.4}$$

where A is the amplitude of the PJ sinusoidal component and x is the time displacement relative to the ideal position. Assume that there is only PJ in the signal. The resulting jitter PDF will then have a concave shape because there will be a higher proportion of samples having jitter magnitudes closer to the sinusoidal peaks than those with smaller jitter magnitudes.

3.1.3 Eye Diagram

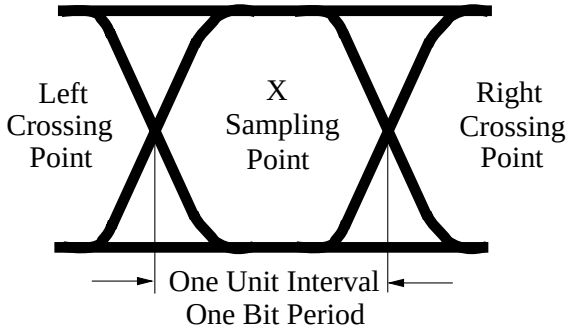


Figure 3.3: Eye Diagram [5]

The most fundamental intuitive view of jitter is provided by the eye diagram [5]. An eye diagram is a composite view of all bit periods of a captured waveform superimposed upon each other. In other words, the waveform trajectory from the start of period 2 to the start of period 3 is overlaid on the trajectory from the start of period 1 to the start of period 2, and so on for all bit periods. Figure 3.3 shows an idealized eye diagram, very straight and symmetrical with smooth transitions (left and right crossing points), and a large, wide-open eye to provide an ideal location to a sample a bit. At this point the waveform should have settled to its high or low value and is least likely to result in a bit error.

3.1.4 Bit-Error-Rate (BER)

BER is the *cumulative distribution function* (CDF) of the TJ PDFs of the left and right eye crossings over the time interval in which a bit error occurs. In

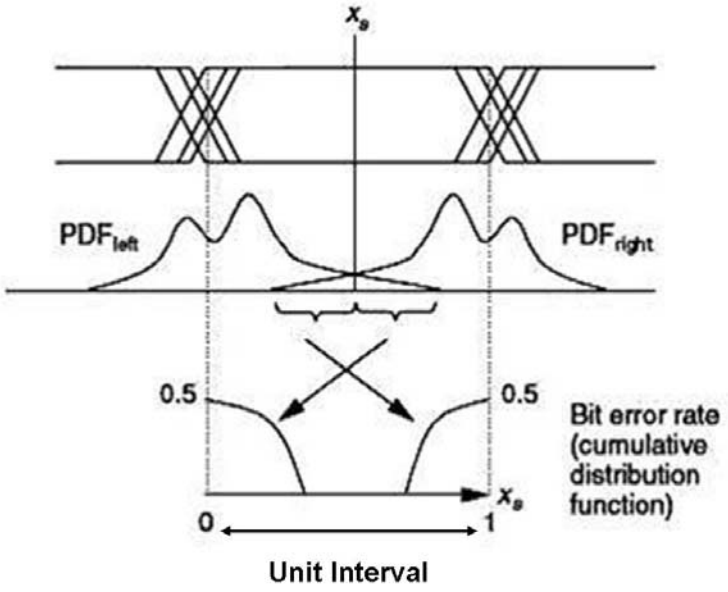


Figure 3.4: Bit Error Rate [28]

Figure 3.4, the erroneous time interval is that to the right of sampling instant X_s for the left eye crossing and that to the left of X_s for the right eye crossing. Integrating the PDFs of both eye crossings over their respective time intervals produces the BER function:

$$BER(X_s) = CDF(X_s) = \frac{1}{2}[1 - \int_{-\infty}^{X_s} PDF_{Left}(\triangle x)d(\triangle x) + \int_{-\infty}^{X_s} PDF_{Right}(\triangle x)d(\triangle x)] \quad (3.5)$$

Figure 3.4 illustrates the relationship between the TJ PDF and the BER function. The BER at the bottom of the figure is also known as a bathtub curve.

3.1.5 Bath Tub Curve

Another viewpoint of jitter is provided by the bathtub plot, depicted in Figure 3.5, where T_B is the bit period and T_L^{DJ} and T_R^{DJ} are the maximum deterministic jitter of the left and right crossing edges, respectively. It is so named because its characteristic curve looks like the cross-section of a bathtub [5]. A bathtub

curve is a graph of BER versus sampling point throughout the Unit Interval. A bathtub plot typically shows the functional relationship between sampling time and BER, starting from a value of 0.5, which will be the probability of a bit-error occurring if the sampling point is at the transition edge.

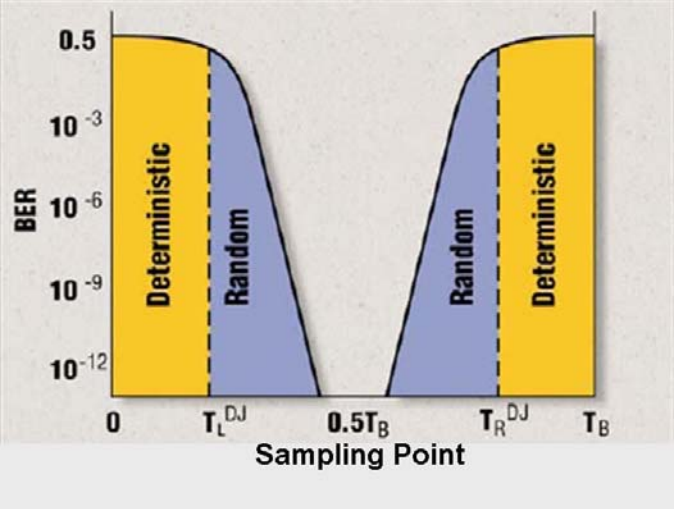


Figure 3.5: Bathtub Curve [5]

When the sampling point is at or near the transition points, the curve is fairly flat and is dominated by deterministic jitter phenomena. As the sampling point moves inward from both ends of the unit interval, the BER drops off precipitously. These regions are dominated by random jitter phenomena and the BER is determined by the σ's of the Gaussian processes producing the random jitter. As one would expect, the center of the unit interval provides the optimum sampling point. Note that there is measured BER for the middle sampling times. Again with an eyeball extrapolation we can estimate that the curves would likely exceed 10^{-18} BER at the 0.5 point of the unit interval. In this case, even for a $10Gb/s$ system it would take over 3×10^8 *seconds* to obtain that value. The curves of the bathtub plot readily show the transmission–error margins at the BER level of interest. The further the left edge is from the right edge at a specified BER, the more margin the design has to jitter. The closer these edges become, the less margin is available. The bathtub plot can also be used to separate random and

deterministic jitter and determine the sigma of the random component.

3.1.6 Jitter Tolerance and Jitter Transfer

Jitter tolerance is a measure of how a known amount of input jitter affects the BER of a device [5]. The measurement sequence requires an instrument such as a pattern generator that can supply a signal with precise amounts of jitter, and also a means to measure the raw bit error rate at the output. The test provides insight into how the *device under test* (DUT) clock recovery circuits or PLLs respond to jitter. Jitter transfer is a measure of the jitter gain of a device. The subsystems pass on the characteristics of the input, so jitter gain in these devices can multiply through the entire network. Jitter transfer is important for characterizing the PLL response of clock recovery devices.

3.2 PLL Jitter Reduction Techniques

3.2.1 Phase-Locked Loop Architecture for Adaptive Jitter Optimization

Vamvakos *et al.* present a PLL architecture that allows adaptive optimization of tracking jitter by using an on-chip jitter estimation block [38]. The jitter estimation circuit operates at the PLL reference clock frequency and is composed of digital blocks, improving the robustness of the overall architecture. The jitter estimates may be used to adaptively tune the PLL loop parameters to achieve minimum jitter operation.

The block diagram of the jitter estimation circuit is shown in Figure 3.6. It consists of two *voltage-controlled delay lines* (VCDLs) whose outputs are delayed versions of the PLL reference clocks. Each of the VCDL outputs is fed into an edge comparison circuit along with the PLL output clock whose jitter is to be measured. The top (bottom) edge comparator produces a 1, if the PLL edge occurs before V_{REF1} (after V_{REF2}). The number of hits H is counted over a time

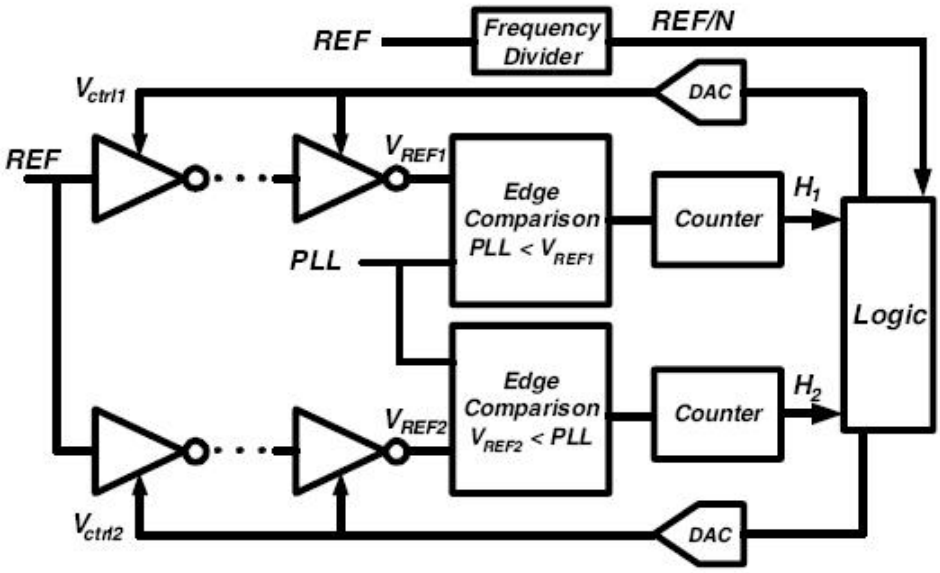


Figure 3.6: Jitter Estimation Circuit [38]

interval equal to N reference clock periods and compared to a target value M and a *hit* is defined as the event when either the top or bottom comparator detects that the PLL edge has occurred before or after V_{REF1} and V_{REF2}, respectively. The difference is used to adjust the VCDL control voltages in such a manner as to decrease the difference between H and M. The procedure is repeated until a convergence criterion is met. The end result is the creation of a dead-zone, the width of which gives an estimate of the PLL output jitter at the current operating conditions.

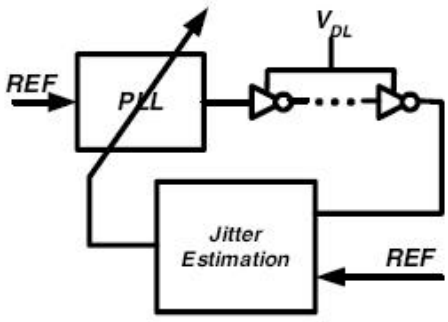


Figure 3.7: System Architecture [38]

The overall system architecture is shown in Figure 3.7. Before the operation of the jitter estimation block begins, the DAC codewords are initialized so that both VCDL delays are equal to half of the delay range. The delay line control voltage V_{DL} is subsequently adjusted so that the edges of the PLL output clock and the VCDL outputs are aligned. This procedure provides the maximum dynamic range for the jitter measurement.

3.2.2 Jitter Minimization in Digital Transmission Using Dual Phase-Locked Loops

In this work by Telba *et al.* [33], a new method for minimization of timing jitter due to phase-locked loops is described. The timing jitter can be minimized using two phase locked loops connected in cascade, where the first one has a *Voltage Controlled Crystal Oscillator* (VCXO) to eliminate the input jitter and the second is a wide band phase-locked loop.

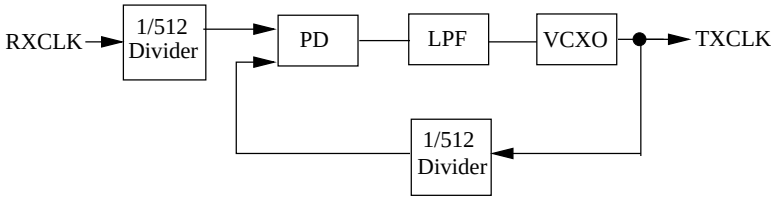


Figure 3.8: Block Diagram of PLL De-Jitter Circuit [33]

Figure 3.8 shows the de-jitter PLL circuit, where PD stands for the *phase detector*. The design objective of this circuit is to generate a stable, low-jitter clock based on either the recovered receive clock or the transmit clock input. But a problem with this design is that they have to use a VCXO that has the same center frequency as the input reference frequency. To avoid this problem, the proposed circuit in Figure 3.9 uses two-cascaded PLLs, and the first one uses a VCXO with a center frequency f_x, not necessarily equal to f_{in}, as in Figure 3.8, where LPF stands for the *low pass filter*. The second one is a narrow band PLL

with wide sweep range.

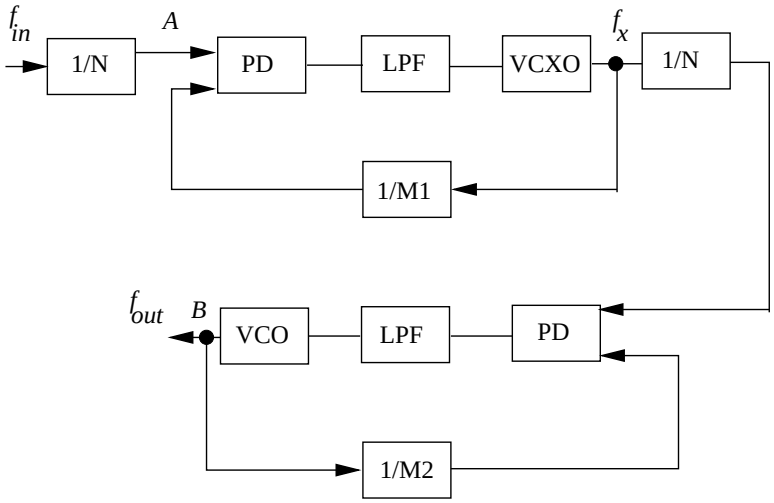


Figure 3.9: Proposed De-Jitter Circuit Using Two Cascaded PLL Circuits [33]

The relation between f_x and f_{in} when the first loop is in locked condition is written as follows:

$$\frac{f_{in}}{N} = \frac{f_x}{M_1} \tag{3.6}$$

With the second loop in locked condition, the relation is written as follows:

$$\frac{f_{out}}{M_2} = \frac{f_x}{N} \tag{3.7}$$

$$f_{out} = f_{in}\frac{M_1 M_2}{N^2} \tag{3.8}$$

If $M1 = M2 = N$, then $f_{in} = f_{out}$ independent of the value of f_x. Since f_x is a low jittered signal as it is produced using the PLL with a VCXO, f_{out} will keep at least the same jitter. On the other hand, more reduction in jitter is obtained if the second PLL is well designed. The filter design in this case is easy since the input signal is already de-jittered. Reducing the PLL bandwidth without using a VCXO is unacceptable because in this case the PLL will not be able to get a locking condition while trying to track the phase variations embedded in the signal, if it is taken directly from the clock recovery circuit.

3.2.3 A Low Jitter Phase-Locked Loop Based on a New Adaptive Bandwidth Controller

An analog adaptive PLL architecture with a new adaptive bandwidth controller to reduce locking time and to minimize jitter in PLL output for wireless communication is presented by Hur *et al.* [15]. It adaptively controls the loop bandwidth according to the locking status. The adaptive bandwidth control is implemented by controlling the charge pump current depending on the locking status.

Figure 3.10 shows the proposed low jitter phase-locked loop based on a new adaptive bandwidth controller. It adaptively controls the loop bandwidth according to the locking status. When the phase error is large, the PLL increases the loop bandwidth and reduces locking time. When the phase error is small, the PLL decreases the loop bandwidth and minimizes output jitter. The loop bandwidth is controlled by changing the magnitude of charge pump current using the adaptive bandwidth controller shown on Figure 3.11.

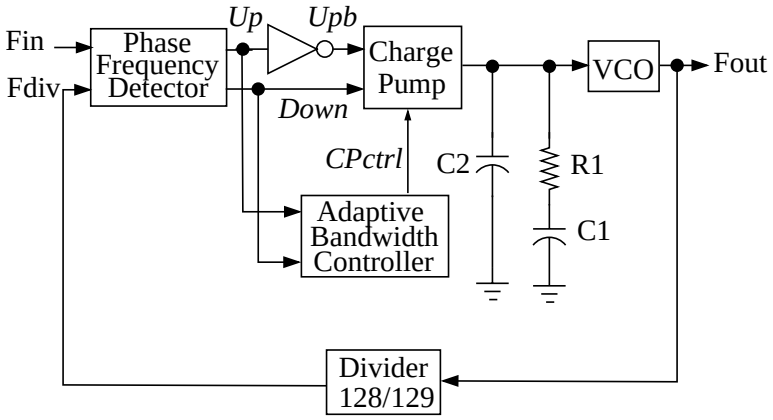


Figure 3.10: Block Diagram of the Proposed PLL Circuit [15]

Up and *Down* pulses obtained from phase frequency detector through the EX-OR gate determine the status of transistors, $MN1$ and $MP1$. When the PLL is out of lock, $MN1$ turns on and $MP1$ turns off. The voltage (CP_{ctrl}) increases the current of $MN3$ and $MN4$, and, subsequently, the currents (I_p and I_n) of the charge pump. When the PLL is locked, on the other side, $MP1$ turns on and $MN1$ turns off. The voltage (CP_{ctrl}) on the capacitor decreases. Then, the

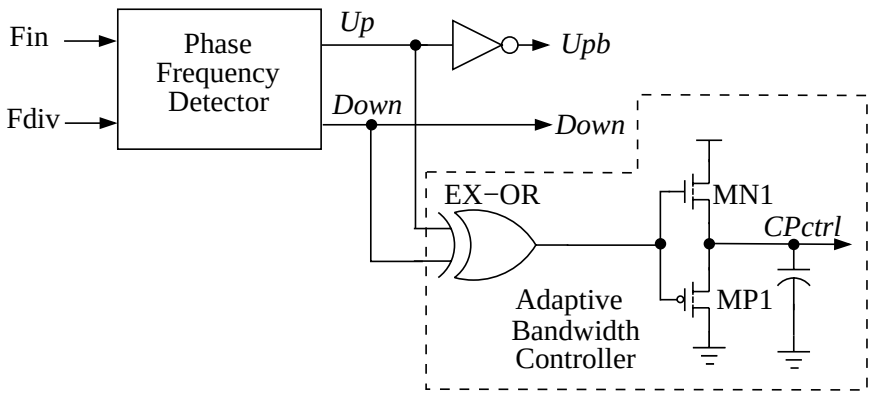


Figure 3.11: Adaptive Bandwidth Controller Circuit [15]

currents of the charge pump decrease. When the PLL is out of lock, the loop bandwidth is wide with the large charge pump current. When the PLL is locked, the loop bandwidth is narrow with small charge pump current. Therefore, the proposed PLL can achieve fast locking with a low jitter characteristic because it controls the magnitude of charge pump current depending on the locking status. When the PLL is out of lock, the EX-OR generates an output proportional to the difference between the *Up* and *Down* pulses. When the PLL is locked, the width of the *Up* and *Down* pulses is the same. Then, the EX-OR gate generates no signal. The effect of the mismatches can be minimized by reducing the charge pump current in the locked state. Therefore, a low jitter PLL can be designed while keeping fast locking.

3.2.4 Other Jitter Reduction Techniques

Mansuri *et al.* [23] proposed a run-time adaptive method of minimizing jitter for a PLL circuit. Various techniques have been reported for designing low jitter clock recovery circuits, for example, modifying the filter design to narrow the PLL bandwidth and make the phase noise at the VCO input as low as possible [35, 36], reducing power supply noise [11, 13, 37], eliminating ground bounce [13] and using a *voltage controlled crystal oscillator* (VCXO) [11, 34].

3.3 Various Test Techniques for SERDES and Jitter

3.3.1 Jitter Test with External Test Equipment

3.3.1.1 BER Estimation for Serial Links Based on Jitter Spectrum and Clock Recovery Characteristics

High performance serial communication systems often require the BER to be at the level of 10^{-12} or below. The excessive test time for measuring such a low BER is a major hindrance in testing communication systems cost-effectively [14]. Hong *et al.* proposed a new technique for accurate and efficient estimation of the BER. The proposed technique estimates the BER based on the spectral information of jitter and the characteristics of the clock and data recovery circuit.

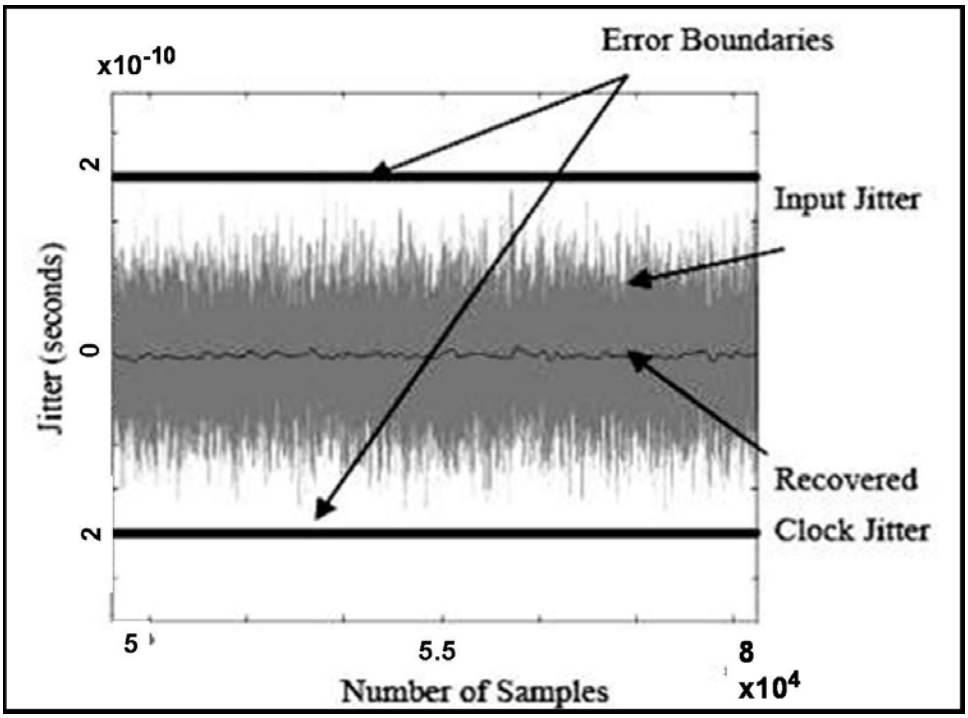


Figure 3.12: The Input Jitter and the Recovered Clock [14]

If only random jitter is present in the transmitted data, the BER can be easily estimated. The CDR circuit cannot track rapidly varying input RJ, because the CDR circuit has a low-pass filter characteristic for the input jitter (so it will filter

out all high frequency RJ). Figure 3.12 shows the input jitter of the transmitted data with only the RJ component and the jitter of the recovered clock produced by the CDR circuit (from simulation). As observed, the recovered clock does not track input jitter at all (i.e., the jitter of the recovered clock is close to zero whereas RJ in the transmitted data is significant). Thus, errors occur when the input jitter is larger than than the 0.5 *unit interval* (UI) or less than the −0.5 UI (indicated as the Error Boundaries in Figure 3.12). The RJ is commonly characterized by a zero-mean Gaussian distribution function. Therefore, the probability that the RJ exceeds a certain threshold can be calculated using the Q-function, which is defined as:

$$Q(x) = \left[\frac{1}{(1-a)x + a\sqrt{x^2+b}}\right]\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} \tag{3.9}$$

where $a = 1/\pi$ and $b = 2\pi$. This Q-function can be used to calculate the probability that the random component, which has zero mean and unity standard deviation, is larger than any given value x. For the case we are interested in, errors occur when the magnitude of the RJ is larger than $T/2$ (T is the Unit Interval) and the variance σ^2 of the RJ is not unity. The threshold value x would be:

$$x = \frac{T}{2\sqrt{\sigma^2}} \tag{3.10}$$

Therefore, the BER can be estimated as:

$$BER = 2Q(\frac{T}{2\sqrt{\sigma^2}}) \tag{3.11}$$

The Q-function is multiplied by 2, because the error occurs on both sides (i.e., when jitter is greater than the threshold or less than ($-1\times$ threshold). Thus, if only RJ is present, the BER can be estimated using Equation 3.11 by measuring the variance of the RJ.

The CDR circuit has a low pass filter characteristic for the input jitter. The basic block diagram of the CDR circuit is shown in Figure 3.13. This characteristic results in higher BER when the frequency of the PJ increases. A CDR circuit is

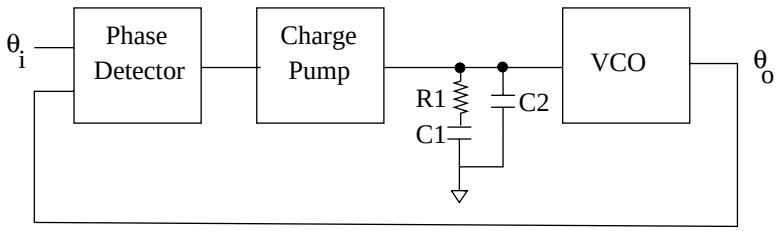


Figure 3.13: The Clock and Data Recovery Circuit [14]

commonly implemented using the architecture of a PLL. The closed loop transfer function of the CDR is:

$$H(s) = \frac{2\xi\omega_n + \omega_n^2}{m\frac{2\xi}{\omega_n}s^3 + (m+1)s^2 + 2\xi\omega_n s + \omega_n^2} \tag{3.12}$$

where m is the capacitance ratio $C1/C2$, ξ is the damping ratio and ω_n is the natural frequency of the ripple in the control voltage. The equations for ξ and ω_n are given as follows:

$$\omega_n = \sqrt{\omega_{LPF} K_{PD} K_{VCO}} \tag{3.13}$$

$$\xi = \frac{1}{2}\sqrt{\frac{\omega_{LPF}}{K_{PD} K_{VCO}}} \tag{3.14}$$

where K_{PD} and K_{VCO} are the gains of the phase detector and VCO, respectively and ω_{LPF} is the cut-off frequency of the low-pass filter. The frequency response of the CDR circuit is divided into four regions based on the magnitude and phase responses:

1. Region 1 (0 to $70KHz$): The magnitude gain is 1, and the phase curve is flat. The PJ is perfectly tracked by the CDR in this region, so it does not affect the BER. Only the RJ contributes to the BER. The PJ is tracked by the CDR circuit with certain delay introduced into the recovered clock. This time delay also shifts the error boundaries, thus increasing the BER.

2. Region 2 ($70KHz$ to $2MHz$): The magnitude gain is 1, and the phase curve has a non-zero slope. The recovered clock has a certain delay, and its

magnitude is compressed. Since the time delay is significant, the input PJ and the recovered clock jitter are out of phase.

3. Region 3 ($2MHz$ to $40MHz$): The magnitude gain is less than 1, and the phase curve has a non-zero slope. The PJ component is not tracked at all. When the input PJ has maximum value, the recovered clock jitter could be almost at a minimum value. The BER of region 3 is worse than any other region.

4. Region 4 ($40MHz$ to ∞): The magnitude gain is negligible.

The variance σ^2 is determined for these four regions and is used in the BER estimation.

3.3.1.2 Extraction of Peak-to-Peak and RMS Sinusoidal Jitter Using an Analytic Signal Method

A new method based on analytic signal theory for extracting both instantaneous and *root mean square* (RMS) sinusoidal jitter from the PLL output signals is proposed by Yamaguchi *et al.* [40]. The method relies on the extension of a real signal into an analytic signal by utilizing the Hilbert transform. The Hilbert transform of a time function $\hat{x}_a(t)$ is defined by:

$$\hat{x}_a(t) = H[x_a(t)] = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x_a(\tau)}{t-\tau}d\tau \tag{3.15}$$

Thus $\hat{x}_a(t)$ is the convolution of the function $x_a(t)$ and $(1/\pi\mathrm{t})$. The Hilbert transform is equivalent to passing $x_a(t)$ through an all-pass filter, in which the magnitudes of the spectral components are unchanged but their phases are shifted by $\pi/2$. The analytic signal $z(t)$ associated with a real signal $x_a(t)$ is defined as the complex signal:

$$z(t) \equiv x_a(t) + j\hat{x}_a(t) \tag{3.16}$$

where the imaginary part $\hat{x}_a(t)$ is the Hilbert Transform of the real part $x_a(t)$. From this, the total instantaneous phase $\phi(t)$ of the real signal $x_a(t)$ is:

$$\phi(t) = \tan^{-1}\left[\frac{\hat{x}_a(t)}{x_a(t)}\right] \tag{3.17}$$

A jitter-free PLL output is a square wave of fundamental frequency f_o. This signal can be decomposed by Fourier analysis into a sum of sine harmonics of frequency f_o, $3f_o$, $5f_o$, etc. With jitter added, the fundamental sinusoidal component with amplitude A and frequency f_o can be written as:

$$A\cos(\phi(t)) = A\cos(2\pi f_o t + \theta + \triangle\phi(t)) \tag{3.18}$$

Notice that the total instantaneous phase function $\phi(t)$ has been written as the sum of three components:

- The linear phase component, which contains the fundamental frequency f_o;
- A constant phase component θ, which can be normalized to zero for computational convenience; and
- The phase modulation component $\triangle\phi(t)$, which is the timing jitter.

The analytic signal $z(t)$ corresponding to the signal is:

$$z(t) = A\cos(2\pi f_o t + \theta + \triangle\phi(t)) + jA\sin(2\pi f_o t + \theta + \triangle\phi(t)) \tag{3.19}$$

The RMS timing jitter and peak-to-peak timing jitter are computed from $\triangle\phi(t)$ as follows:

- At the discrete times nT corresponding to the square wave signal edges or zero-crossings, the phase modulation values are $\triangle\phi(nT)$.
- RMS timing jitter $\triangle\phi_{RMS}$ is calculated as the RMS value of $\triangle\phi(nT)$. This corresponds to the timing jitter value obtained by the phase detector method:
$$\triangle\phi_{RMS} = \sqrt{\frac{1}{N}\sum_{k=0}^{N-1}\triangle\phi^2(kT)} \tag{3.20}$$
- The peak-to-peak timing jitter $\triangle\phi_{pp}$ is calculated by the difference: $[max\ \triangle\phi(nT) - min\ \triangle\phi(nT)]$

The jitter values $J(nT)$ are computed from Equation 3.21 and the RMS and peak-to-peak jitter values are computed as follows:

$$J(nT) = \triangle\phi((n+1)T) - \triangle\phi(nT) \tag{3.21}$$

- The RMS jitter J_{RMS} is simply the RMS value of the set $J(nT)$.
- The peak-to-peak jitter J_{pp} is calculated as $[max\ J(nT) - min\ J(nT)]$.

3.3.1.3 Jitter Spectral Extraction for Multi-Gigahertz Signal

A method for extracting the spectral information of a multi-gigahertz jittery signal is proposed by Ong *et al.* [26]. This method may utilize existing on-chip, single-shot period measurement techniques to measure the multi-gigahertz signal periods for spectral analysis. This method does not require an external sampling clock, nor any additional measurement beyond existing techniques. To extract any signal spectral information, the signal amplitude needs to be periodically sampled over a given time for spectral analysis of the signal's jitter. However, one great challenge is to sample periods of a multi-gigahertz signal at absolute periodic intervals. A jitter-free sampling clock signal would be required to trigger the *time measuring unit* (TMU) to perform period measurement.

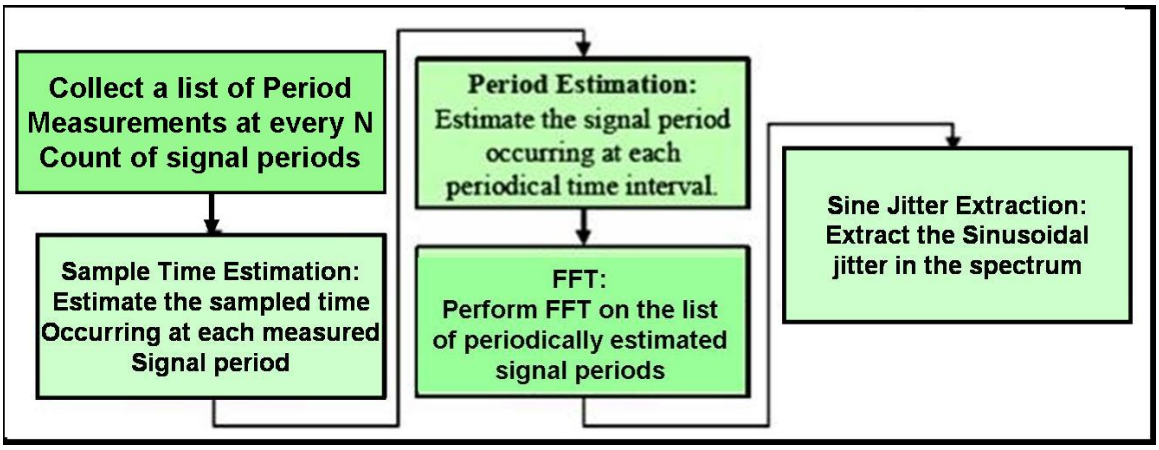


Figure 3.14: Simplified Technique Flow Overview [26]

An overview of the proposed technique is illustrated in Figure 3.14. The technique does not require a sampling clock signal to perform spectral analysis on the signal. Instead, it collects a list of signal periods by measuring each signal period width on every N count of signal periods as elaborated in Figure 3.15. With each sampled period value, the technique has a simple *Sample Time Estimation procedure* to estimate the time T_n at which each period is sampled. With this procedure, every sampled period can be associated with an estimated sampling time T_n as shown in Figure 3.15. The current sampling time T_n can be expressed

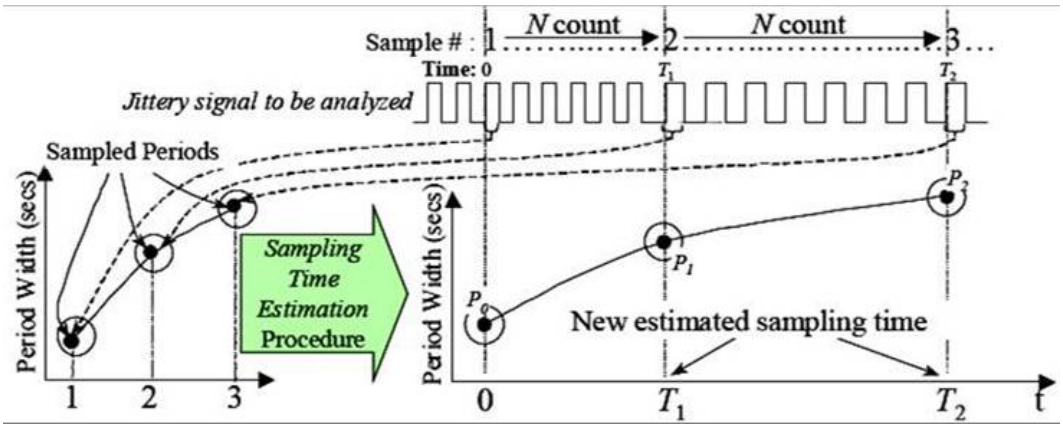


Figure 3.15: Estimate the Sampled Time of Each Measured Period [26]

in terms of the count N, the previous sampling time T_{n-1}, the current sampled period width W_{p_n}, and previous sampled period $W_{p_{n-1}}$, as follows:

$$T_n = T_{n-1} + \frac{W_{p_n}(N+1) + W_{p_{n-1}}(N-1)}{2} \tag{3.22}$$

The next procedure is to estimate the width of the signal period that occurs at periodic time intervals P_n to analyze the jitter spectrum. Since they have generated a list of measured signal periods with their respective time incidences T_n, estimating a list of signal periods that occur periodically at P_n can be easily accomplished through interpolation using the *Period Estimation* procedure as shown in Figure 3.16. The width w_{EP} of the estimated signal period EP, can be

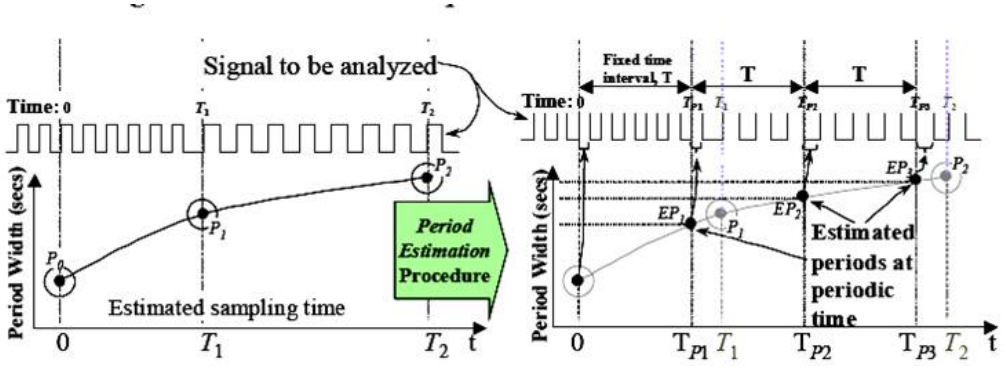


Figure 3.16: Estimate the Signal Periods at Periodic Time Interval [26]

determined using the following equation:

$$w_{EP} = W_{P1} + (T_{P1} - T_1)\frac{W_{p2} + W_{p1}}{T_2 - T_1} \tag{3.23}$$

where T_{P1} is the periodic time when EP occurs, while T_1 and T_2 are the immediate-adjacent sampling times estimated in the previous sampling time estimation procedure. W_{P1} and W_{P2} are the widths of the immediate-adjacent sampled periods P_1 and P_2, respectively. For simplicity, a random jitter component was not included in Figures 3.15 and 3.16. After generating a list of estimated signal periods at periodic intervals, the technique performs the *Fast Fourier Transform* (FFT) on the list to extract any sinusoidal/periodic jitter that may be found in the signal.

3.3.2 Jitter BIST

3.3.2.1 Circular BIST Testing the Digital Logic within a High Speed SERDES

A BIST method for testing the digital part of a SERDES is presented by Hetherington and Simpson [12]. Krasniewski and Pilarski invented circular BIST [17]. Circular BIST is a structural self-test method whereby some of the flip-flops of a design are upgraded with an enabled XOR on the D input, as shown in Figure 3.17. When the BIST enable signal is inactive (0), the flip-flop sees the normal

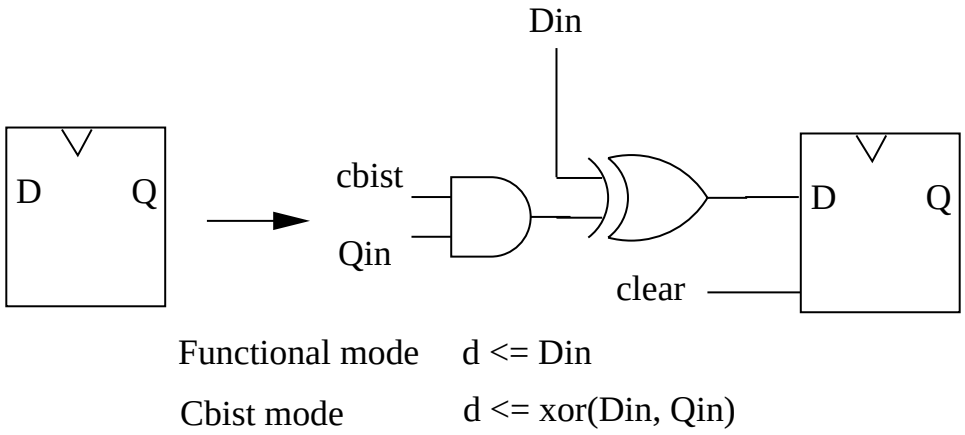


Figure 3.17: Circular BIST Flip-Flop [12]

functional *Din* input.

When the BIST enable signal is active (1), the flip-flop reads the XOR of the functional *Din* input together with, typically, the *Q* output of another circular BISTed flip-flop. The subset of flip-flops that are converted into circular BIST flip-flops are connected to form a circular path as in Figure 3.18. This BIST circuit is then operated by this sequence:

1. Reset all flip-flops.
2. Enable circular BIST mode.
3. Clock for *N* cycles.
4. Compare values in a subset of flip-flops with expected values.

The main advantages of circular BIST are:

1. It generates one pattern per clock, unlike scan-based BIST, which generates one pattern per scan.
2. Clocks can be used as-is. Derived clocks are used as in functional mode and no clock gating is required.
3. BIST control is simple to implement and small in overhead.

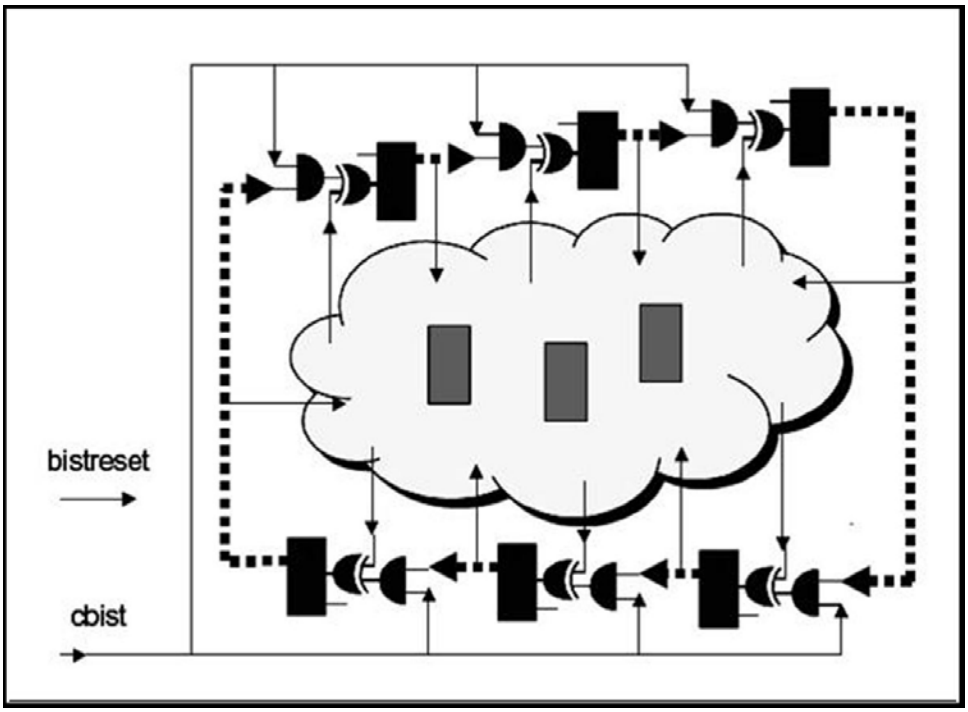


Figure 3.18: Circular BIST Path [12]

4. The BIST test can be run at-speed.

5. Full conversion of functional flip-flops into BIST flip-flops is not required [5].

The main disadvantages of circular BIST are:

1. Fault grades can be low due to *limit cycling*. This is where an inappropriate starting state for the circular BIST path leads to the BIST path repeatedly cycling through a limited number of states.

2. Fault grades can be low due to the register adjacency problem. This is where adjacent cells in the BIST path have the property that the output of the first cell is in the functional input cone of the second. The result is that the XOR gate of the second cell can always output zero and, hence, block fault propagation.

3. Fault grades must be obtained using slow sequential fault simulation.

Along with these specific disadvantages, circular BIST shares the requirement of all embedded BIST that inputs be bounded for controllability and outputs be bounded for observability.

3.3.2.2 Automated Calibration of Phase Locked Loop with On-Chip Jitter Test

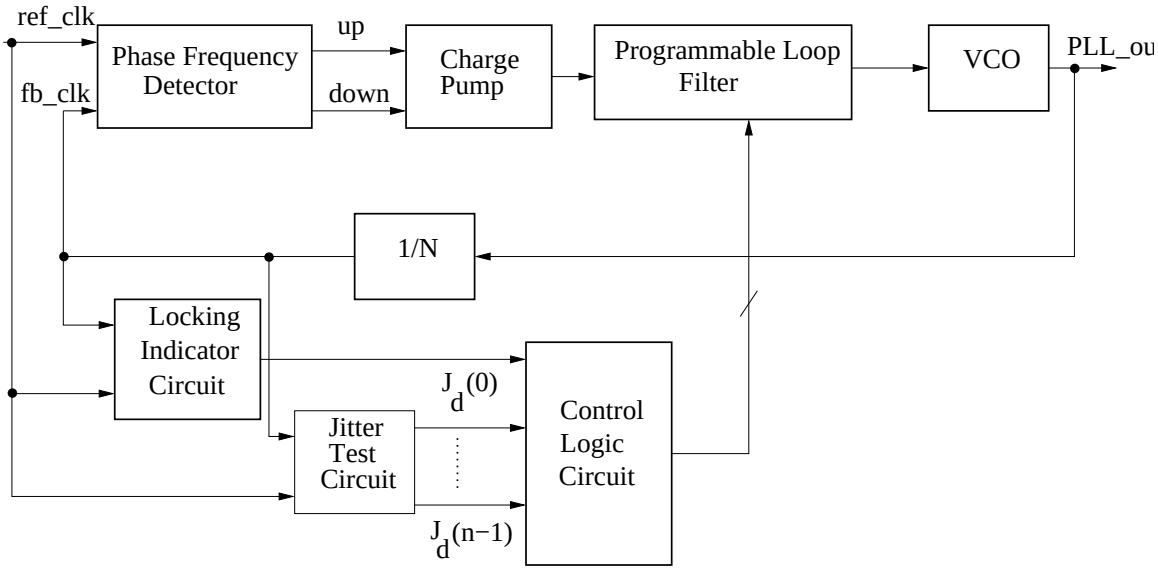


Figure 3.19: The Adaptive PLL [39]

A new adaptive PLL is implemented by Xia *et al.* [39]. The PLL employs a jitter test circuit to monitor the PLL jitter performance. Additionally, it uses a digital control unit to calibrate the loop filter parameters dynamically. Figure 3.19 shows the proposed adaptive PLL structure. Comparing with the conventional design, three extra functional components are added: (1) Jitter test circuit; (2) Locking indicator circuit; and (3) Control logic unit. The locking indicator circuit is used to monitor the PLL locking status. When the PLL is unlocked the control logic unit will program the loop filter to make the PLL have the widest loop bandwidth. When the PLL is locked, the jitter test circuit is activated to track the PLL jitter performance. If the jitter amplitude is larger than the design

specification, the control unit will configure the loop filter and narrow down the loop bandwidth.

3.3.2.3 On-Chip Jitter Measurement Using a Dual-Channel Undersampling Time Digitizer

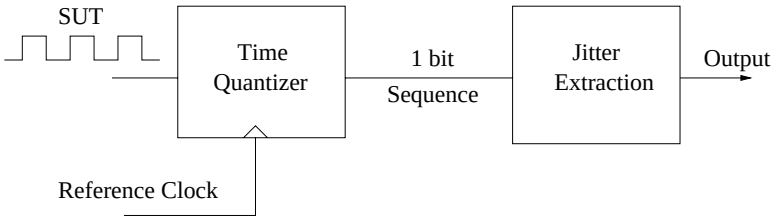


Figure 3.20: Single-Channel Architecture for Jitter Measurement [8]

Dou and Abraham use a clock signal of frequency f_s to undersample the *signal under test* (SUT) of frequency f_o [8]. The SUT is fed to a time quantizer, which is triggered by the reference clock as shown in Figure 3.20. The time quantizer functions as phase deference detector. The phase difference is in multiples of Δ, where Δ is the smallest possible difference. The phase difference is converted into bits and sent to a counter. The counter determines the number of hits for each event. Here an event is the probability of getting a difference of Δ, 2Δ, 3Δ, and so on. From this, the *cumulative distribution function* (CDF) and PDF are obtained. The PDF gives the jitter distribution, from which the jitter parameters can be determined.

To reduce the reference clock uncertainty and the quantization noise, a dual channel configuration is used. Here two SUT's are triggered by the same reference clock signal. Therefore, the RJ from the clock signal and the quantization noise from the two time quantizers cancel each other, leaving only the RJ from the SUTs. The configuration is shown in Figure 3.21. Results indicate that the dual channel performs better than the single channel.

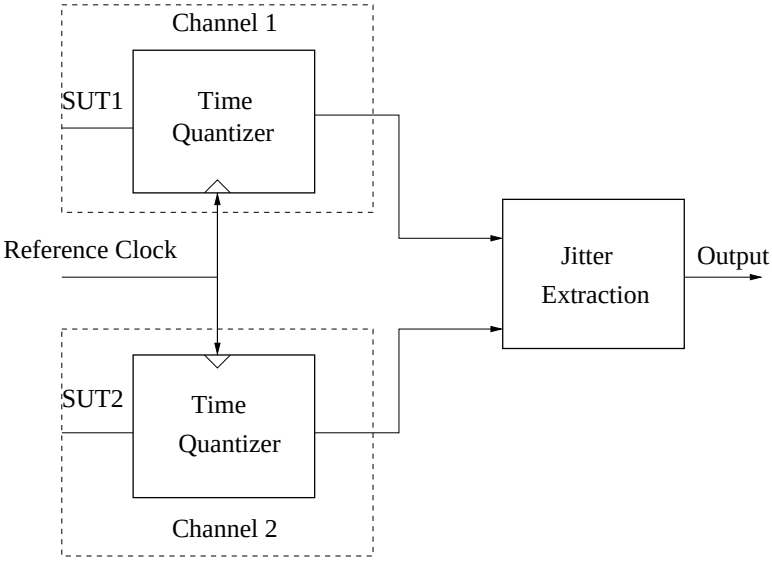


Figure 3.21: Test Architecture of a Dual-Channel Configuration [8]

3.3.3 Other SERDES and Jitter Test Solutions

Sunter *et al.* proposed an automated, structural test solution for SERDES that uses the receiver to demodulate the signal jitter to a low-speed bit stream, which is analyzed by a single-clock domain [31]. The technique is combined with logic BIST and 1149.6 boundary scan to completely test an IC. In the work by Takahiro *et al.*, a method for measuring jitter tolerance of a SERDES receiver using the timing misalignment between the jittered source clock and recovered clock is presented [42]. A sinusoidal jitter is injected into the serial bit stream. The method derives an equation for estimating BER accurately. Li *et al.* have found that utilizing a double delta function in BER estimation is inaccurate by conducting experiments and systematic simulations on the accuracy of jitter separation based on BER functions [22]. Analytic signal theory is used in the estimation of cycle-to-cycle period jitter in PLL outputs [41] and for measuring clock skews in the clock distribution network of microprocessors [43]. A technique for estimating the standard deviation of a Gaussian random jitter component in a multi-gigahertz

signal is proposed by Ong *et al.* [27]. The method utilizes existing on-chip single-shot measurement techniques to measure the multi-gigahertz signal periods for the estimation. Jitter models and measurement methods for high-speed serial interconnects are presented by Kuo *et al.* [18]. They describe the relationship between a jitter PDF and BER followed by a discussion on what causes jitter. Common jitter measurement methods are presented, along with an analysis of their respective advantages and disadvantages. A new jitter measurement technique utilizing a high bandwidth undersampling voltage measurement instrument is proposed by Wajih *et al.* [7]. A test methodology based on a passive filter technique to enhance the traditional loop back test for SERDES, by including jitter tests, is presented by Laquai *et al.* [20]. Finally, Taylor *et al.* propose a BIST method to measure jitter without external references [32].

3.3.4 Summary

In this section various techniques for reducing jitter by modifying the loop bandwidth of the PLL circuit and also various techniques for measuring jitter using external test equipment and on-chip BIST methods were presented.

Chapter 4
New Jitter Reduction Technique

In this chapter a new design methodology and jitter reduction hardware are proposed for reducing jitter in the transmit side section of high-speed SERDES circuits. The technique is then extended for reducing jitter in the receive side, which is explained in the next chapter. First, a complete SERDES architecture with *transmit* (Tx) and *receive* (Rx) jitter reducer circuits (shaded blocks) is shown in Figure 4.1 to give an idea as to where these proposed hardware blocks are to be placed to reduce jitter.

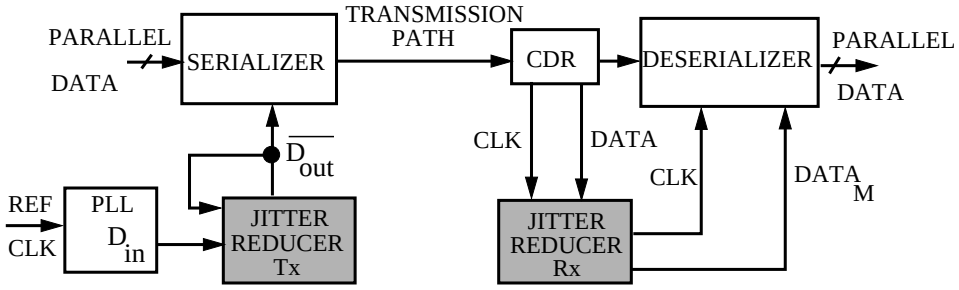


Figure 4.1: SERDES with Jitter Reduction Circuits

The circuit consists of a PLL that generates a fast clock of 1 GHz, a serializer (parallel-in-serial-out shifter) that converts 8-bit parallel data into serial data, a CDR circuit and a deserializer (serial-in-parallel-out shifter) that converts 1-bit serial data into 8-bit parallel data. The PLL is of Type 1 and consists of an XOR gate phase detector and a three stage ring oscillator. A Type 1 PLL has a single pole at the origin and is used to correct a step phase change in the input clock signal. The CDR circuit is made using a similar PLL and a flip-flop for retiming the recovered data. A *low-pass filter* (LPF) of 1 GHz cut-off frequency models

the magnitude and frequency behavior of the transmission medium.

Apart from Type 1 PLL circuit, there are Type 2 and Type 3 PLL circuits. Type 2 has two poles at the origin and corrects a step velocity (change of frequency) and a step phase change in the input signal. Finally, Type 3 has three poles at the origin and corrects a step acceleration (time variant frequency change), a step velocity and a step phase change in the input signal. Regardless of the type of the PLL circuits, the jitter reducer hardware can reduce the jitter, since it operates on the signal at the output of the PLL circuit and so, it does not depend on the internal circuity of the PLL circuit.

4.1 Jitter Reduction Technique for the Transmit Side Phase-Locked Loop

In this section a new jitter reduction technique is proposed for reducing the jitter present in the output of the PLL (D_{in}). A reference signal is used for determining the jitter that will be reduced from the jittered signal. The proposed jitter reduction circuit (Figure 4.2) does: (1) inversion to generate the output signal $\overline{D_{out}}$, which has the opposite polarity to that of the input reference signal (D_{ref}), (2) jitter reduction and (3) generation of the reference signal D_{ref}.

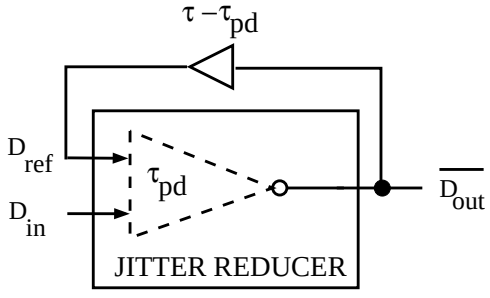


Figure 4.2: Jitter Reducer with Reference Signal $\overline{D_{out}}$

The jitter reduction circuit behaves as an *inverter* that toggles the signal D_{ref} every τ *seconds*, where τ is the bit period (period of either the logic '1' or '0' bit) of the clock signal D_{in}. The buffer delays the signal from the output of the

inverter to the input by $\tau - \tau_{pd}$ seconds, where τ_{pd} is the propagation delay of the jitter reducer circuit. The circuit takes the loop back signal D_{ref}, which is the modified input jittered signal, and the input jittered signal D_{in} and produces the jitter reduced signal $\overline{D_{out}}$ as shown in Figure 4.3.

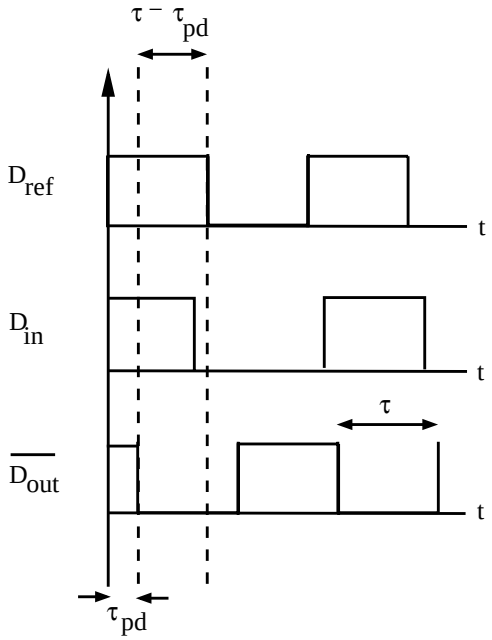


Figure 4.3: Jitter Reducer – Timing Waveforms

The additional information that is used in the jitter reduction process is that the bit period of the loop back signal depends on the circuit delay and not on the input signal and, therefore, the jitter in the input signal is not transferred completely to the loop back signal $\overline{D_{out}}$. The loop back signal is not an ideal signal and has some jitter in it that can be reduced by a good circuit design. This method is based on the principle of *auto-correlation*, where a signal correlates with its past to determine the error. In our case the correlating signals are the jittered input and the jitter reduced output signal. The reason we call it auto-correlation and not cross-correlation is because the jitter reduced signal is nothing but the same delayed input signal but with less jitter in it.

4.1.1 Process of Jitter Reduction

The difference in time when the signals D_{ref} and D_{in} rise and fall is obtained in the form of pulses. Once we have these pulses, we can do pulse shaping to get a reduced jitter signal.

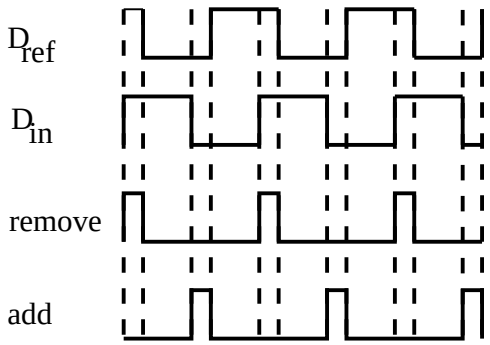


Figure 4.4: *Add* and *Remove* Pulses

The amount of jitter reduced depends on how accurate these pulses are as compared to those obtained using an ideal reference signal. For example, in Figure 4.4, D_{in} is leading D_{ref} and two kinds of pulses are generated. The *add* pulse is used to add a period of length τ_{add} to D_{in} and *remove* is used to remove a period of length τ_{remove} from D_{in}. Mathematically, if we add and remove pulses of period τ_{add} and τ_{remove} from D_{in} we should get a pulse of period τ, which is the expected period of $\overline{D_{out}}$.

4.1.2 Mathematical Theory with Examples

A mathematical proof is presented to show how jitter is reduced in the phase domain. Let D_{in} be the input jittered signal and D_{ref} be an ideal reference signal. We will represent these as analog signals [40] as follows:

$$D_{in}(t) = \cos(2\pi f_o t + \triangle\phi_{in}(t)) \quad (4.1)$$

$$D_{ref}(t) = \cos(2\pi f_o t) \quad (4.2)$$

where f_o is the fundamental frequency of the signal and $\triangle\phi_{in}(t)$ is the phase jitter.

Signal Flow Graph:

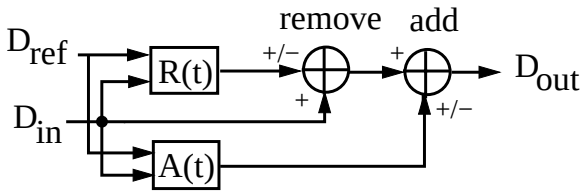


Figure 4.5: Signal Flow Graph of the Jitter Reducer

Figure 4.5 shows the signal flow graph of the jitter reduction process. Let $R(t)$ and $A(t)$ be the functions that extract a phase jitter of opposite polarity from the input signals, i.e., if D_{in} has positive jitter then either $R(t)$ or $A(t)$ will extract a jitter of equal magnitude and negative polarity, so that the jitter in D_{in} can be nullified. $R(t)$ extracts $\triangle\phi_R(t)$ that has to be removed from D_{in}, and $A(t)$ extracts $\triangle\phi_A(t)$ that has to be added to signal D_{in} to transform it into D_{ref}.

Four Jitter Types:

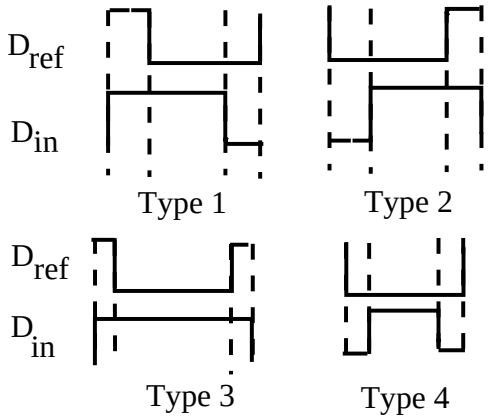
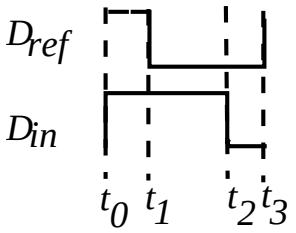


Figure 4.6: Four Types of Jitter Conditions

Let us denote a jitter as positive when it advances a signal transition and negative when it delays a transition with respect to the reference signal. There are four types of jitter conditions as shown in Figure 4.6. For Type 1, the jitter at the rising and the falling transitions is positive, for Type 2 it is negative, for

Type 3 the jitter at the rising transition is positive and at the falling one it is negative and, finally, for Type 4 it is negative and positive.

Type 1 Analysis:



TIME REGION	PHASE	
	D_{in}	D_{ref}
$[t_0, t_1]$	$2\pi f_o t + \triangle\phi_{in}(t)$	$2\pi f_o t$
$(t_1, t_2]$	$2\pi f_o t$	$2\pi f_o t$
$(t_2, t_3]$	$2\pi f_o t + \triangle\phi_{in}(t)$	$2\pi f_o t$

Figure 4.7: D_{in} and D_{ref} Phase in Three Time Regions

Let us consider the case where D_{in} is leading D_{ref} (Type 1). The table in Figure 4.7 shows the phase of D_{in} and D_{ref} in three time regions. The general equation for jitter reduction as inferred from the signal flow graph is:

$$\overline{D_{out}} = \cos\left(((2\pi f_o t + \triangle\phi_{in}(t)) \ \pm \triangle\phi_R(t)) \ \pm \triangle\phi_A(t)\right) \tag{4.3}$$

In the time interval $[t_o, t_1]$, a negative jitter has to be removed since D_{in} has positive jitter. Therefore, $\triangle\phi_R(t)$ is $-\triangle\phi_{in}(t)$ and $\triangle\phi_A(t)$ is 0, since only removing is required in this region.

$$\overline{D_{out}} = \cos\left(((2\pi f_o t + \triangle\phi_{in}(t)) - \triangle\phi_{in}(t)) \pm 0\right) \longrightarrow \cos\left(2\pi f_o t\right) \tag{4.4}$$

In the time interval $(t_1, t_2]$, $\triangle\phi_R(t)$ and $\triangle\phi_A(t)$ are zero, since there is no jitter in D_{in}.

$$\overline{D_{out}} = \cos\left(((2\pi f_o t + 0) \pm 0) \pm 0\right) \longrightarrow \cos\left(2\pi f_o t\right) \tag{4.5}$$

Finally in the time interval $(t_2, t_3]$, $\triangle\phi_A(t)$ is $-\triangle\phi_{in}(t)$ and $\triangle\phi_R(t)$ is 0, since only adding is required in this region.

$$\overline{D_{out}} = \cos\left(((2\pi f_o t + \triangle\phi_{in}(t)) \pm 0) - \triangle\phi_{in}(t)\right) \longrightarrow \cos\left(2\pi f_o t\right) \tag{4.6}$$

So in all the three regions the output signal is $2\pi f_o t$, which has the same phase as the phase of the jitter-free reference signal. But in our case, the reference signal used is the loop back signal, which will contain some jitter, therefore the jitter extracted by $R(t)$ and $D(t)$ will not be exactly the same as the jitter present in D_{in}. This implies that the signal at the output of the jitter reducer will still contain some jitter, but considerably less than the jitter in the input signal.

Theorem 4.1: *If the phase error of a output signal at a time instant t is* $\triangle\phi(t)$*, then it is reduced to zero by the jitter reducer, if either* $\triangle\phi_A(t)$ *or* $\triangle\phi_R(t)$ *is equal to* $|\triangle\phi(t)|$.

Proof: The phase quantities $\triangle\phi_A(t)$ and $\triangle\phi_R(t)$ are extracted by the *add* and *remove* functions $A(t)$ and $R(t)$, respectively. At any given time instant either $A(t)$ or $R(t)$ extracts a phase quantity to nullify the phase error. So, if $\triangle\phi_R(t)$ is equal to $|\triangle\phi(t)|$ and $\triangle\phi_A(t)$ is zero, then, from Equation 4.4, we get the output phase to be equal to $2\pi f_o t$, which is the phase of a error-free signal. Similarly, from Equation 4.4, we get the output to be equal to $2\pi f_o t$, if $\triangle\phi_A(t)$ is equal to $|\triangle\phi(t)|$ and $\triangle\phi_R(t)$ is equal to zero. In either case the phase error is reduced to zero and the error-free phase quantity is recovered. ■

4.1.3 Architecture of Jitter Reduction Circuit

The circuit used for jitter reduction is shown in Figure 4.8. The part of the circuit shown within the dotted lines behaves as an inverter. It also does the function of jitter detection and reduction with reference to D_{ref}. Here τ is the desired bit period for the clock signal. The buffer G is used to delay the signal $\overline{D_{out}}$ by $(\tau - \tau_{pd})$ *seconds*, before it reaches the input of the jitter reduction circuit. The period τ_{pd} is the propagation delay of the circuit from D_{in} to $\overline{D_{out}}$. The inputs to this circuit are the signals D_{in} and D_{ref}. Signal D_{in} is the input jittered signal and D_{ref} is the loop back signal of the circuit. A timed Boolean expression is obtained

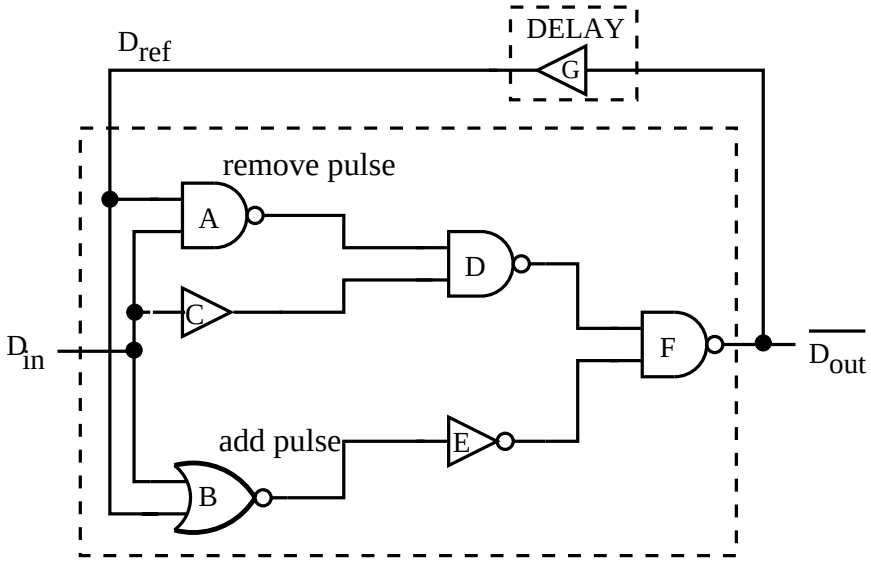


Figure 4.8: Jitter Reduction Circuit

for the jitter reduction circuit based on the properties of its three functions: (1) inversion, (2) jitter reduction and (3) reference signal generation.

4.1.3.1 Inversion

Assuming a zero-delay model, the timed Boolean expressions for the various gates in the jitter reduction circuit are given below:

$$\begin{aligned} A(t) &= \overline{D_{in}}(t) + \overline{D_{ref}}(t) \\ B(t) &= \overline{D_{in}}(t) \cdot \overline{D_{ref}}(t) \\ C(t) &= D_{in}(t) \\ D(t) &= \overline{A}(t) + \overline{C}(t) \\ E(t) &= \overline{B}(t) \\ F(t) &= \overline{D}(t) + \overline{E}(t) \end{aligned} \tag{4.7}$$

Therefore, the output signal $\overline{D_{out}(t)}$ is given as follows:

$$\overline{D_{out}}(t) = F(t) = ((\overline{D_{in}}(t) + \overline{D_{ref}}(t)) \cdot D_{in}(t)) + \tag{4.8}$$

$$(\overline{D_{in}}(t) \cdot \overline{D_{ref}}(t)) \tag{4.9}$$

On expanding the above equation and simplifying, we get:

$$\begin{aligned} \overline{D_{out}}(t) = F(t) &= \overline{D_{ref}}(t)(D_{in}(t) + \overline{D_{in}}(t)) \\ &= \overline{D_{ref}}(t) \end{aligned} \tag{4.10}$$

So, the above expression proves that the jitter reduction circuit behaves as an inverter, inverting the input signal D_{ref}.

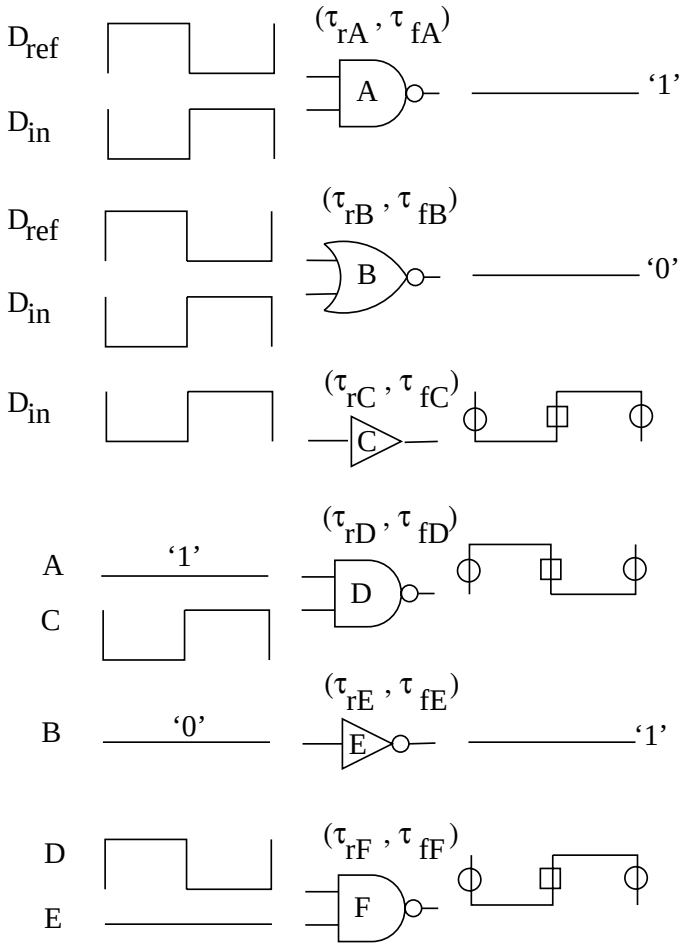


Figure 4.9: Gate Outputs – Non-Zero Delay Case

Now, let us expand the timed Boolean expression for a non-zero delay model. Consider the outputs of the various gates in the jitter reduction circuit as shown in Figure 4.9. Each gate is characterized by two delays, rising τ_r and falling τ_f, respectively. The timed Boolean expression for $\overline{D_{out}}(t)$ is split into $\overline{D_{out\_r}}(t)$, for

the rising transition and $\overline{D_{out\_f}}(t)$, for the falling transition, respectively. For the rising transition, $\overline{D_{out}}(t)$ depends on τ_{rF}, τ_{fD} and τ_{rC} delays as shown in Figure 4.9, marked by squares on the corresponding transitions. There are three different paths in the circuit: ADF, CDF and BEF. Depending on the path a particular signal traverses, the delay parameters of the gates lying on that path are included in that signals timing information. The variables X_A, X_B and X_E represent unknown delay parameters for the corresponding gates, i.e., it cannot be determined from the available information, whether to include the rising or falling delay parameter for that particular gate. The timed Boolean expression for the rising transition is then given as follows:

$$\begin{aligned}\overline{D_{out\_r}}(t) &= (\overline{D_{in}}(t-\tau_{rF}-\tau_{fD}-X_A)+ \\ &\quad \overline{D_{ref}}(t-\tau_{rF}-\tau_{fD}-X_A))\cdot \\ &\quad D_{in}(t-\tau_{rF}-\tau_{fD}-\tau_{rC}))+ \\ &\quad (\overline{D_{in}}(t-\tau_{rF}-X_E-X_B)\cdot \\ &\quad \overline{D_{ref}}(t-\tau_{rF}-X_E-X_B)) \qquad (4.11)\end{aligned}$$

Similarly, the timed Boolean expression for the falling transition can be obtained and is given as follows:

$$\begin{aligned}\overline{D_{out\_f}}(t) &= (\overline{D_{in}}(t-\tau_{fF}-\tau_{rD}-Y_A)+ \\ &\quad \overline{D_{ref}}(t-\tau_{fF}-\tau_{rD}-Y_A))\cdot \\ &\quad D_{in}(t-\tau_{fF}-\tau_{rD}-\tau_{fC}))+ \\ &\quad (\overline{D_{in}}(t-\tau_{fF}-Y_E-Y_B)\cdot \\ &\quad \overline{D_{ref}}(t-\tau_{fF}-Y_E-Y_B)) \qquad (4.12)\end{aligned}$$

The delays on which the falling transition depend are marked as circles in Figure 4.9 and the variables Y_A, Y_B and Y_E represent the unknown delay parameters for the gates A, B and E. To determine the unknown parameters, we will use the jitter reduction property, which is explained in the next section.

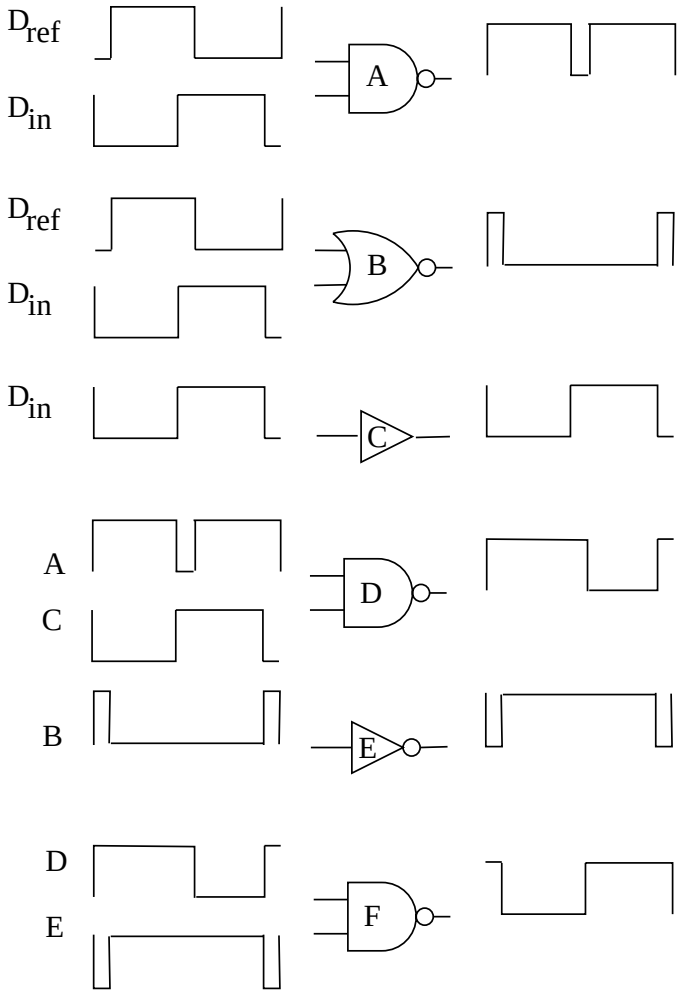


Figure 4.10: Case 1 – D_{in} Leading D_{ref}

4.1.3.2 Jitter Reduction

The jitter reduction property of the jitter reduction circuit is explained using two cases: (1) D_{in} leading D_{ref} and (2) D_{in} lagging D_{ref}, respectively. Consider the outputs of the various gates as shown in Figure 4.10 for Case 1.

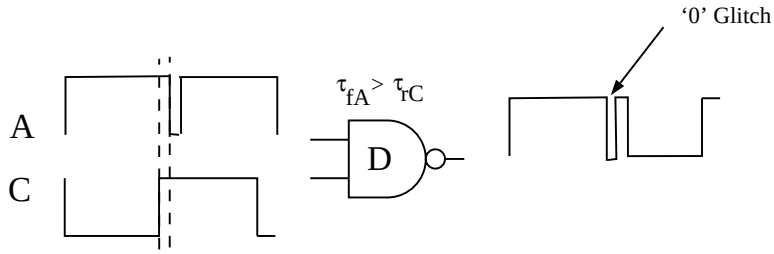


Figure 4.11: A Glitch at the Output of Gate D for Case 1

For gate D, if τ_{fA} is greater than τ_{rC}, we get a glitch as shown in Figure 4.11. Therefore, the constraint on delay parameters τ_{fA} and τ_{rC} is given as follows:

$$\tau_{fA} \leq \tau_{rC} \tag{4.13}$$

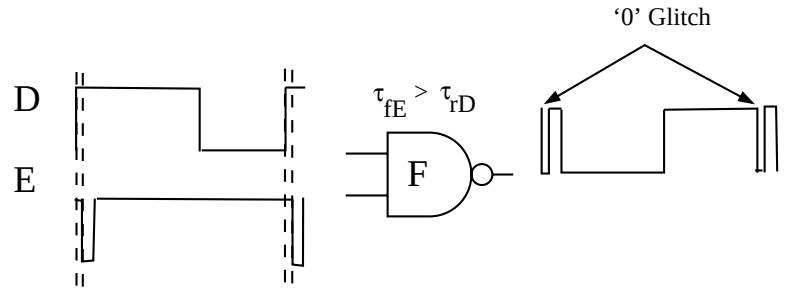


Figure 4.12: A Glitch at the Output of Gate F for Case 1

Similarly, for gate F, if τ_{fE} is greater than τ_{rD}, we get a glitch as shown in Figure 4.12. Therefore, the constraint on delay parameters τ_{fE} and τ_{rD} is given as follows:

$$\tau_{fE} \leq \tau_{rD} \tag{4.14}$$

Now consider the outputs of the various gates for Case 2 as shown in Figure 4.13. Similarly to Case 1, we can observe the outputs of gates D and F, respectively, as shown in Figures 4.14 and 4.15, to arrive at a timing constraint

on the delay parameters, τ_{rA}, τ_{fC}, τ_{rE} and τ_{fD}. The constraints are given as follows:

$$\tau_{rA} \geq \tau_{fC} \tag{4.15}$$

$$\tau_{rE} \geq \tau_{fD} \tag{4.16}$$

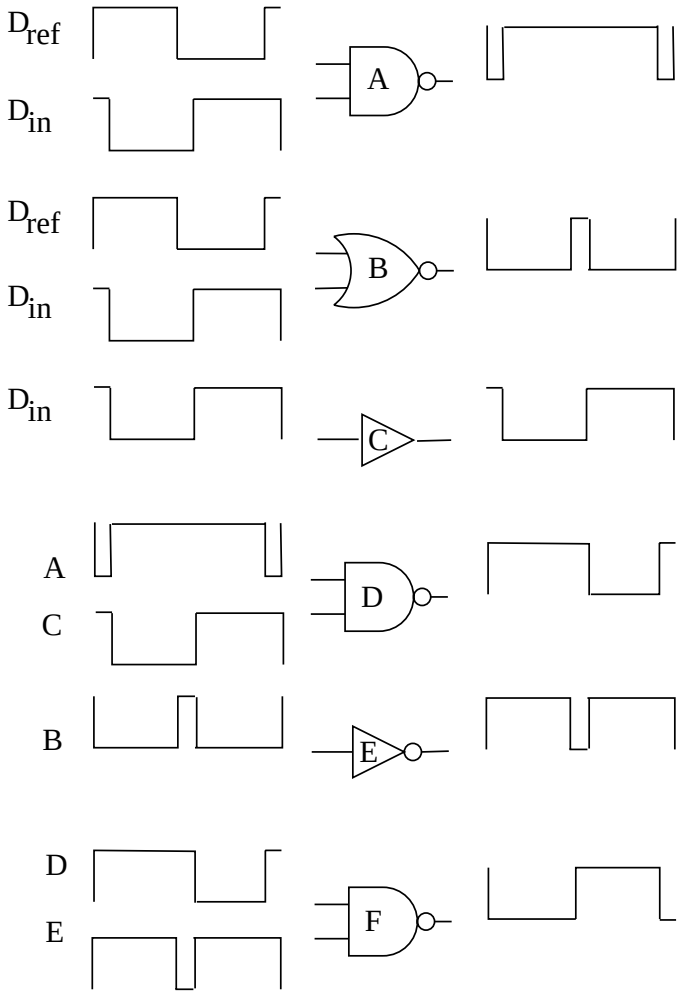


Figure 4.13: Case 2 – D_{in} Lagging D_{ref}

Choosing X_A, X_B and X_E: From the analysis of Case 1 and 2, we can choose the unknown delay parameters: X_A, X_B and X_E. From Equation 4.11, we see that the rising transition of $\overline{D_{out}}(t)$ depends on the delay parameters τ_{fD} and τ_{rC}, respectively. From Equations 4.16 and 4.13, we see that these delay parameters depend on τ_{rE} and τ_{fA}, respectively. To find a value for these unknown delay parameters X_A, X_B and X_E, we have to choose either a rising or falling delay for the parameters, and so we can use the knoweledge from the timed Boolean

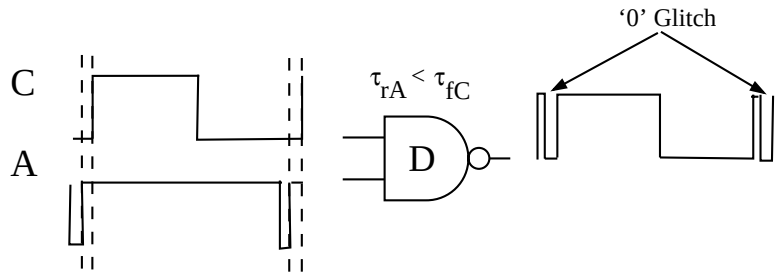


Figure 4.14: A Glitch at the Output of Gate D for Case 2

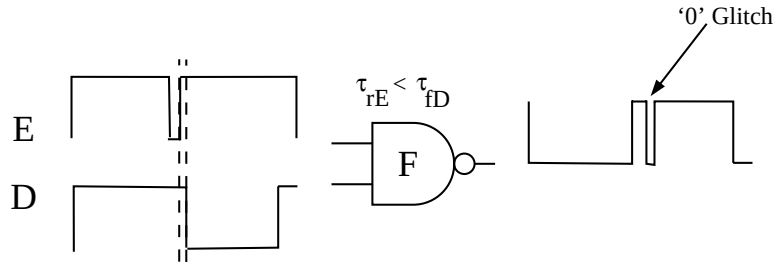


Figure 4.15: A Glitch at the Output of Gate F for Case 2

expressions and the timing constraints already obtained to pick a value for these parameters. From Equation 4.13, we see that $\tau_{fA} \leq \tau_{rC}$ and from Equation 4.16, we see that $\tau_{rE} \geq \tau_{fD}$ and therefore, we choose τ_{fA} for X_A and τ_{rE} for X_E, respectively as shown below:

$$X_A = \tau_{fA} \tag{4.17}$$

$$X_E = \tau_{rE} \tag{4.18}$$

$$X_B = \tau_{fB} \tag{4.19}$$

The delays in Equation 4.11 for D_{in} should be matched. This leads to the requirement that $t - \tau_{rF} - \tau_{fD} - X_A = t - \tau_{rF} - \tau_{fD} - \tau_{rC}$ or $X_A = \tau_{rC}$. Since $\tau_{fA} \leq \tau_{rC}$ and there is no constraint on τ_{rA}, we choose $X_A = \tau_{fA}$. Also, $t - \tau_{rF} - \tau_{fD} - \tau_{rC} = t - \tau_{rF} - X_E - X_B$. Since $\tau_{rE} \geq \tau_{fD}$, we choose $X_E = \tau_{rE}$, since we have no constraint on τ_{fE}. X_B is chosen as τ_{fB}, since gate E inverts the output of gate B, and we have already constrained gate E's delay as τ_{rE},

so $X_B = \tau_{fB}$ is chosen for consistency. These choices will make the path delays equal in the TBF of Equation 4.11 provided that $\tau_{fA} = \tau_{rC}$, $\tau_{rE} = \tau_{fD}$ and $\tau_{fB} = \tau_{rC}$. However these are merely sufficient conditions, as other choices might also equalize the path delays.

Choosing Y_A, Y_B and Y_E: Similarly, From Equation 4.12, we see that the falling transition of $\overline{D_{out}}(t)$ depends on the delay parameters τ_{rD} and τ_{fC}, respectively. From Equation 4.15, we see that $\tau_{rA} \geq \tau_{fC}$ and from Equation 4.14, we see that $\tau_{fE} \leq \tau_{rD}$ and therefore, we choose τ_{rA} for Y_A and τ_{fE} for Y_E, respectively as shown below:

$$Y_A = \tau_{rA} \tag{4.20}$$

$$Y_E = \tau_{fE} \tag{4.21}$$

$$Y_B = \tau_{rB} \tag{4.22}$$

The delays in Equation 4.12 for D_{in} should be matched. This leads to the requirement that $t - \tau_{fF} - \tau_{rD} - Y_A = t - \tau_{fF} - \tau_{rD} - \tau_{fC}$ or $Y_A = \tau_{fC}$. Since $\tau_{rA} \geq \tau_{fC}$ and there is no constraint on τ_{fA} we choose $Y_A = \tau_{rA}$. Also, $t - \tau_{fF} - \tau_{rD} - \tau_{fC} = t - \tau_{fF} - Y_E - Y_B$. Since $\tau_{fE} \leq \tau_{rD}$, we choose $Y_E = \tau_{fE}$, since we have no constraint on τ_{rE}. Y_B is chosen as τ_{rB}, since gate E inverts the output of gate B, and we have already constrained gate E's delay ass τ_{fE}, so $Y_B = \tau_{rB}$ is chosen for consistency. These choices will make the path delays equal in the TBF of Equation 4.12 if $\tau_{rA} = \tau_{fC}$, $\tau_{rB} = \tau_{fC}$ and $\tau_{rD} = \tau_{fE}$. However these are merely sufficient conditions, as other choices might equalize the path delays.

Substituting the values of X_A, X_B, X_E, Y_A, Y_B and Y_E in Equations 4.11 and 4.12, we get the final timed Boolean expressions for the rising and the falling transition as follows:

$$\begin{aligned} \overline{D_{out\_r}}(t) \;=\; & (\overline{D_{in}}(t - \tau_{rF} - \tau_{fD} - \tau_{fA}) + \\ & \overline{D_{ref}}(t - \tau_{rF} - \tau_{fD} - \tau_{fA})) \cdot \\ & D_{in}(t - \tau_{rF} - \tau_{fD} - \tau_{rC})) + \end{aligned}$$

$$(\overline{D_{in}}(t-\tau_{rF}-\tau_{rE}-\tau_{fB})\cdot$$
$$\overline{D_{ref}}(t-\tau_{rF}-\tau_{rE}-\tau_{fB})) \quad (4.23)$$

$$\begin{aligned}\overline{D_{out\_f}}(t) &= (\overline{D_{in}}(t-\tau_{fF}-\tau_{rD}-\tau_{rA})+\\ &\quad \overline{D_{ref}}(t-\tau_{fF}-\tau_{rD}-\tau_{rA}))\cdot\\ &\quad D_{in}(t-\tau_{fF}-\tau_{rD}-\tau_{fC}))+\\ &\quad (\overline{D_{in}}(t-\tau_{fF}-\tau_{fE}-\tau_{rB})\cdot\\ &\quad \overline{D_{ref}}(t-\tau_{fF}-\tau_{fE}-\tau_{rB})) \quad (4.24)\end{aligned}$$

From Equation 4.23, we see that to avoid glitches during a rising transition of the output signal is that the three path delays must be equal as shown below:

$$\begin{aligned}\tau_{rF}+\tau_{fD}+\tau_{fA} &= \tau_{rF}+\tau_{fD}+\tau_{rC}=\\ &\quad \tau_{rF}+\tau_{rE}+\tau_{fB} \quad (4.25)\end{aligned}$$

or $\tau_{fA}=\tau_{rC}$ and $\tau_{fD}+\tau_{rC}=\tau_{rE}+\tau_{fB}$. Similarly, from Equation 4.24, we see that to avoid glitches during a falling transition of the output signal is that the three path delays must be equal as shown below:

$$\begin{aligned}\tau_{fF}+\tau_{rD}+\tau_{rA} &= \tau_{fF}+\tau_{rD}+\tau_{fC}= \quad (4.26)\\ &\quad \tau_{fF}+\tau_{fE}+\tau_{rB}\end{aligned}$$

or $\tau_{rA}=\tau_{fC}$ and $\tau_{rD}+\tau_{fC}=\tau_{fE}+\tau_{rB}$. To achieve the above constraints is very difficult in any CMOS technology process, because it is very difficult to match the asymmetric delays of NAND and NOR gates lying in these paths in order to achieve identical path delays because of process variations.

Theorem 4.2: *The necessary condition to avoid glitches at the output signal* $\overline{D_{out}}$*, is that the three path delays must be equal and also* $\tau_{fA}\leq\tau_{rC}$*,* $\tau_{fE}\leq\tau_{rD}$*,* $\tau_{rA}\geq\tau_{fC}$ *and* $\tau_{rE}\geq\tau_{fD}$*.*

Proof: If the three path delays are equal, then the following constraints are met:

$$\begin{aligned} \tau_{rF} + \tau_{fD} + \tau_{fA} &= \tau_{rF} + \tau_{fD} + \tau_{rC} = \\ & \tau_{rF} + \tau_{rE} + \tau_{fB} \end{aligned}$$

or $\tau_{fA} = \tau_{rC}$ and $\tau_{fD} + \tau_{rC} = \tau_{rE} + \tau_{fB}$. Also,

$$\begin{aligned} \tau_{fF} + \tau_{rD} + \tau_{rA} &= \tau_{fF} + \tau_{rD} + \tau_{fC} = \\ & \tau_{fF} + \tau_{fE} + \tau_{rB} \end{aligned}$$

or $\tau_{rA} = \tau_{fC}$ and $\tau_{rD} + \tau_{fC} = \tau_{fE} + \tau_{rB}$. From Equations 4.25 and 4.26 , we see that these are necessary conditions and also if $\tau_{fA} \leq \tau_{rC}$, $\tau_{fE} \leq \tau_{rD}$, $\tau_{rA} \geq \tau_{fC}$ and $\tau_{rE} \geq \tau_{fD}$, we satisfy the conditions to avoid glitches at the outputs of gates D and F as given in Equations 4.13, 4.14, 4.15 and 4.16 and hence the output $\overline{D_{out}}$ will have no glitches. ■

4.1.3.3 Reference Signal Generation

To generate a reference signal of equal pulse width τ, the following constraint must be satisfied to avoid DCD jitter in D_{ref}, which will otherwise be transferred to the output signal $\overline{D_{out}}(t)$:

$$\tau_{rDELAY} = \tau - \tau_{\overline{D_{out\_r}}} \tag{4.27}$$

$$\tau_{fDELAY} = \tau - \tau_{\overline{D_{out\_f}}} \tag{4.28}$$

4.1.4 Optimized Jitter Reduction Circuit

In this section a new optimized jitter reduction circuit is proposed, where the problem of matching the three path delays is removed. The new jitter reduction circuit shown in Figure 4.16 has a composite gate (transistors $5 - 14$) called $COMP$ and a modified inverter (transistors $1 - 4$) called INV and the circuit has only one path from the inverter to the composite gate and so, the problem of matching path delays does not exist.

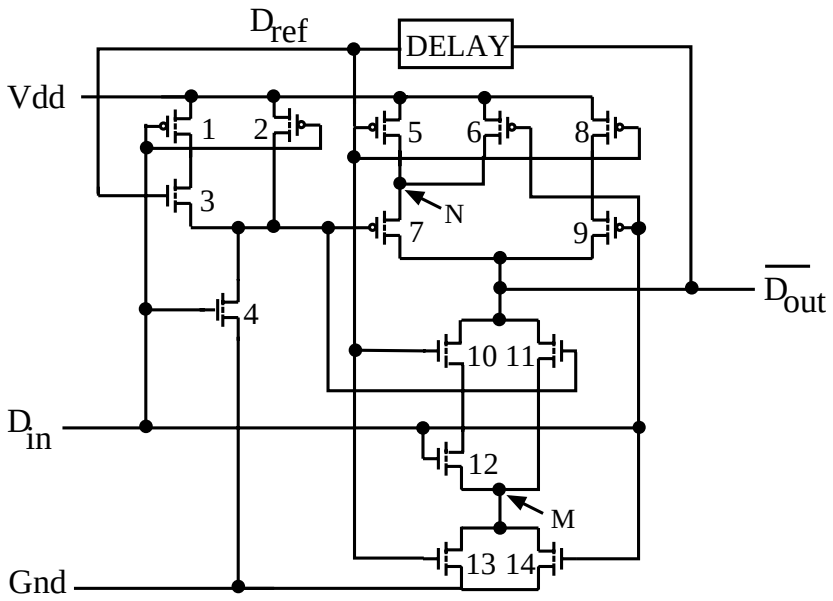


Figure 4.16: Optimized Jitter Reduction Circuit

The timed Boolean expressions, similar to Equations 4.11 and 4.12 in the previous section for the rising and the falling transitions, is given below:

$$\begin{aligned}
\overline{D_{out_r}}(t) &= (\overline{D_{in}}(t - \tau_{rCOMP}) + \\
&\quad \overline{D_{ref}}(t - \tau_{rCOMP})) \cdot \\
&\quad D_{in}(t - \tau_{rCOMP} - \tau_{fINV})) + \\
&\quad (\overline{D_{in}}(t - \tau_{rCOMP}) \cdot \\
&\quad \overline{D_{ref}}(t - \tau_{rCOMP})) \qquad (4.29)
\end{aligned}$$

$$\begin{aligned}
\overline{D_{out_f}}(t) &= (\overline{D_{in}}(t - \tau_{fCOMP}) + \\
&\quad \overline{D_{ref}}(t - \tau_{fCOMP})) \cdot \\
&\quad D_{in}(t - \tau_{fCOMP} - \tau_{rINV})) + \\
&\quad (\overline{D_{in}}(t - \tau_{fCOMP}) \cdot \\
&\quad \overline{D_{ref}}(t - \tau_{fCOMP})) \qquad (4.30)
\end{aligned}$$

where T_{rCOMP}, T_{fCOMP}, T_{rINV} and T_{fINV} are the rising and falling propagation

delays of the composite gate and the modified inverter, respectively. To find the constraints on the inverter delay (T_{rINV} and T_{fINV}), let us look into the jitter reduction process of the new optimized circuit. First, to derive the constraints for the falling delay T_{fINV}, we look at the p-tree of the composite gate, since the falling transition of the inverter affects the rising delay of the composite gate. Consider the case shown in Figure 4.17 where D_{in} is leading D_{ref}, where N is

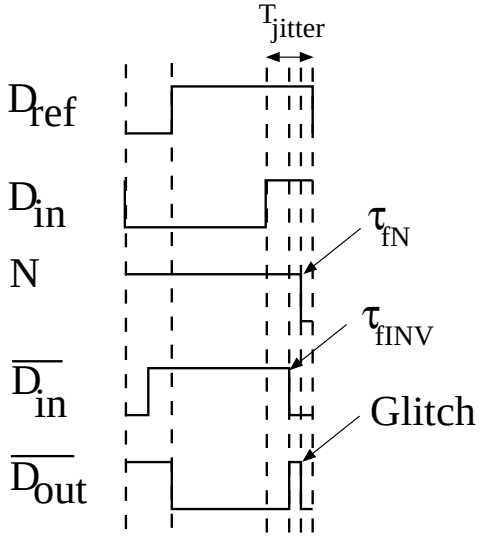


Figure 4.17: Optimized Jitter Reduction Circuit – D_{in} is leading D_{ref}

the nodal voltage at the drain ends of transistors 5 and 6, T_{jitter} is the timing jitter present between the reference signal D_{ref} and the input signal D_{in} and τ_{fN} is the falling propagation delay of the node N. From the above figure, we see that the node N remains in logic '1' state as long as either D_{in} or D_{ref} is '0.' At the moment both of them go to '1' when $\overline{D_{in}}$ is '0,' there exists a path between the nodes N and $\overline{D_{out}}$. At this instant $\overline{D_{out}}$ is already at logic '0.' The node N voltage, which is at logic '1' at this point, is pulled down to '0' as a result of the charge transfer between the nodes N and $\overline{D_{out}}$ and a glitch appears in the output node $\overline{D_{out}}$ for the period the node N is '1.' The key observation from the above discussion is that the output node $\overline{D_{out}}$ is vulnerable during the period T_{jitter}, since both D_{in} and D_{ref} are '1' and the output node is exposed to the node N.

The only way to prevent this is to remove the path existing between the node N and the output node through transistor 7, driven by $\overline{D_{in}}$, by increasing the falling propagation delay of the inverter, so that the signal $\overline{D_{in}}$ will remain at '1' for the period T_{jitter} and there will be no path between the nodes N and $\overline{D_{out}}$ and there will be no glitch in the output. Therefore, the constraint on τ_{fINV} is given below:

$$\tau_{fINV} \geq T_{jitter} \tag{4.31}$$

Theorem 4.3: *If the falling inverter delay τ_{fINV} is greater than or equal to the timing jitter T_{jitter}, then there will be no glitches at the output $\overline{D_{out}}$.*

Proof: If the falling inverter delay τ_{fINV} is greater than or equal to the timing jitter T_{jitter}, then from Equation 4.31, the constraint for avoiding glitches is met and hence the output $\overline{D_{out}}$ will have no glitches. ■

Now as shown in Figure 4.18, where the inverter falling delay is greater than T_{jitter}, we see that the glitch is removed at the output $\overline{D_{out}}$, but the pulse width of logic '0' is increased as a result. This introduces a DCD jitter of width $\tau_{fINV} - T_{jitter}$.

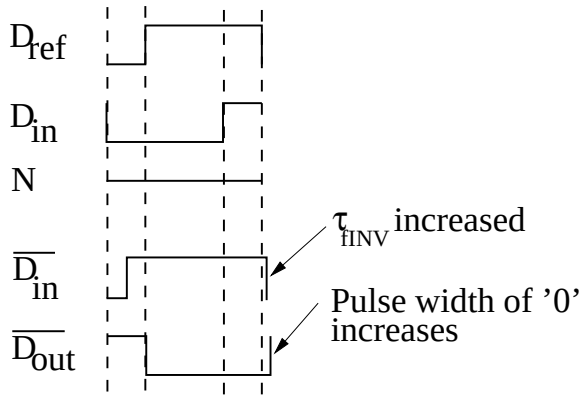


Figure 4.18: Removal of the Glitch by Increasing the Inverter Falling Delay

Similarly, to derive the constraint on the rising delay of the inverter τ_{rINV}, let us look at the n-tree of the composite gate. Consider Figure 4.19, where M is

the nodal voltage at the drain end of transistors 11 and 12 and τ_{rM} is the rising propagation delay of the node M.

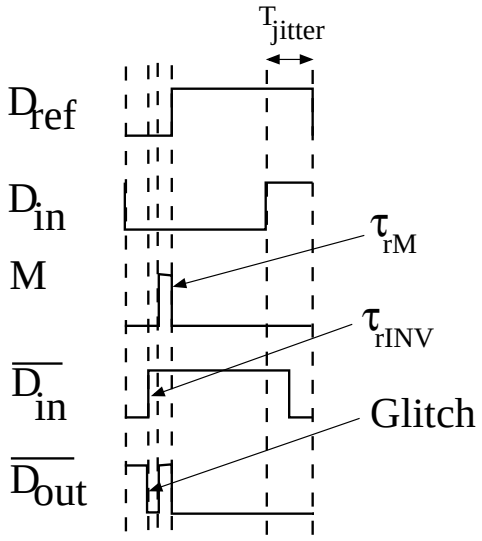


Figure 4.19: Glitch Occurring at the Output $\overline{D_{out}}$ Due to the Nodal Voltage M

From the above figure, we see that when D_{in}, D_{ref} and $\overline{D_{in}}$ are '0,' the output $\overline{D_{out}}$ is cut-off from the n-tree and is pulled to '1' by the p-tree of the composite gate. But when $\overline{D_{in}}$ goes to '1' after the propagation delay τ_{rINV}, there exists a path between the nodes M and $\overline{D_{out}}$. Since the node M is at logic '0' and $\overline{D_{out}}$ is logic '1,' charge transfer occurs between the nodes M and $\overline{D_{out}}$ and M goes to logic '1' after the delay τ_{rM}. For the period when the node M is at logic '0' and $\overline{D_{in}}$ is '1,' there exists a path between the nodes M and $\overline{D_{out}}$ through transistor 11 and the output is pulled down to '0,' thereby causing a glitch in the output. The output goes back to '1' as soon as the node M is charged to '1.' The key observation is that the output $\overline{D_{out}}$ is affected during the period T_{jitter}, since both D_{in} and D_{ref} are '0' and $\overline{D_{in}}$ is '1' in this period, creating a path between the nodes M and $\overline{D_{out}}$. To overcome this problem, the rising delay of the inverter has to be increased, so that $\overline{D_{in}}$ is zero during the period T_{jitter}. Therefore, the constraint on τ_{rINV} is given below:

$$\tau_{rINV} \geq T_{jitter} \tag{4.32}$$

Theorem 4.4: *If the rising inverter delay τ_{rINV} is greater than or equal to the timing jitter T_{jitter}, then there will be no glitches at the output $\overline{D_{out}}$.*

Proof: If the rising inverter delay τ_{rINV} is greater than or equal to the timing jitter T_{jitter}, then from Equation 4.32, the constraint for avoiding glitches is met and hence the output $\overline{D_{out}}$ will have no glitches. ■

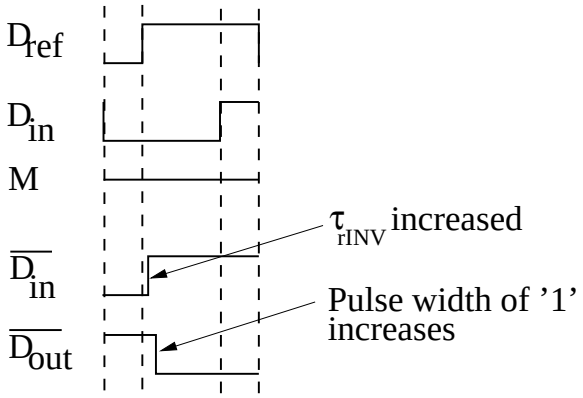


Figure 4.20: Removal of the Glitch by Increasing the Inverter Rising Delay

Now as shown in Figure 4.20, where the inverter rising delay is greater than T_{jitter}, we see that the glitch is removed at the output $\overline{D_{out}}$, but the pulse width of logic '1' is increased as a result. This introduces a DCD jitter of width $\tau_{rINV} - T_{jitter}$. Similar constraints for both τ_{rINV} and τ_{fINV} can be obtained from the case where D_{in} is lagging D_{ref}.

From the above discussion we derive the following conditions for increasing the rising and the falling delays of the modified inverter. The rising delay has to be increased only when D_{in} and D_{ref} are logic '0' and the falling delay has to be increased only when D_{in} and D_{ref} are logic '1,' respectively. Based on the above conditions a inverter design is proposed as shown in Figure 4.21. For the rising delay, when D_{ref} is '0,' the inverter chooses the path through the transistor $I2$, which gives the longer delay and when D_{ref} is '1' the parallel combination of delays through the transistors $I1, I3$ and $I2$ gives the shorter delay. Similarly, for the falling delay, when D_{ref} is '1,' the path through the transistor $I5$ gives the

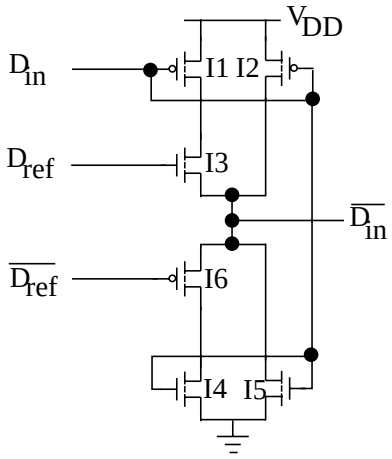


Figure 4.21: New Inverter Design

longer delay and when D_{ref} is '1,' the parallel combination of the paths through the transistors $I4, I6$ and $I5$ gives the shorter delay.

The longer delays for both the rising and the falling transitions of the inverter can be obtained by changing the W/L ratios of the transistors $I2$ and $I5$, respectively. The delays are changed by changing the length L of both the transistors for a fixed value of the width W. The value by which the length L has to be changed depends on the parameter T_{jitter}, since it has been shown that the rising and the falling delays have to be greater than T_{jitter} to avoid a glitch at the output. But T_{jitter} is a stochastic parameter and the only way to determine the optimum length L for the transistors $I2$ and $I5$, respectively, is by testing the modified inverter with a input signal D_{in} with jitter and by varying the length L. The width W of all of the transistors in the modified inverter was fixed at $80nm$ and the inverter was tested for various input jitter types. The optimized inverter design in shown in Figure 4.22. The optimized inverter circuit has two delays for the rising transition and only one delay for the falling transition. This is because the rising transition required an explicitly longer delay, but for the falling transition the delay obtained was sufficient enough to both avoid the glitches and also produce only small amount of DCD jitter. The expression for the longer rising

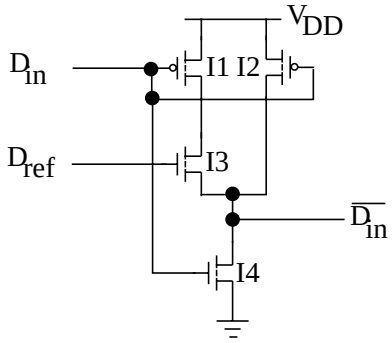


Figure 4.22: Optimized Inverter Design

delay when D_{ref} is '0' is given as follows:

$$\tau_{rINV} = R_{I2} \times (C_{I2} + C_{I3} + C_{I5}) \tag{4.33}$$

where R_{I2} is the resistance due to the transistor $I2$, and C_{I2}, C_{I3} and C_{I5} are the drain capacitances of the transistors $I2$, $I3$ and $I5$. The shorter delay for the rising transition of the inverter is given as follows:

$$\tau_{rINV} = (R_{I2} \| (R_{I1} + R_{I3})) \times (C_{I2} + C_{I3} + C_{I5}) \tag{4.34}$$

The expression for the falling transition delay is given as follows:

$$\tau_{rINV} = R_{I5} \times (C_{I2} + C_{I3} + C_{I5}) \tag{4.35}$$

The delays obtained for the rising and the falling transitions as stated previously were sufficient enough to avoid the glitches at the output and also to provide only a small amount of DCD jitter. The results of testing the Tx jitter reducer with the modified inverter and its effect on the output jitter are discussed in the next section.

In the modified circuit, transistors $5-7$ of the p-tree and and transistors $10-12$ of the n-tree perform the *remove* operation. Transistors 8 and 9 of the p-tree and transistors 13 and 14 of the n-tree do the *add* operation. The delay element is made of cascaded buffers in order to get the desired delay value. The delay is increased or decreased by changing the length L of each p and n transistor of each buffer. Apart from varying the delays in the inverter, the lengths of various

other transistors in the composite gate were changed to see if they affect the performance of the jitter reduction process and it was determined that the effect is negligible. The W/L ratios of all the transistors in the the jitter reduction circuit are shown in Table 4.1.

Table 4.1: W/L Ratios of All the Transistors in the Optimized Jitter Reduction Circuit

Transistor No.	W/L Ratio
1	$80nm/80nm$
2	$80nm/700nm$
3	$80nm/80nm$
4	$80nm/80nm$
5	$80nm/80nm$
6	$80nm/80nm$
7	$80nm/80nm$
8	$80nm/80nm$
9	$80nm/80nm$
10	$80nm/80nm$
11	$80nm/80nm$
12	$80nm/80nm$
13	$80nm/80nm$
14	$80nm/80nm$

Theorem 4.3: *If the inverter delays τ_{rINV} and τ_{fINV} are equal to the timing jitter T_{jitter}, then there will be no glitches at the output $\overline{D_{out}}$.*

Proof: If the inverter delays τ_{rINV} and τ_{fINV} are equal to the timing jitter T_{jitter}, then the constraints given in Equations 4.31 and 4.32 are met and, therefore, there will be no glitches at the output $\overline{D_{out}}$. ■

4.2 Results for Transmit Side Jitter Reducer

The Tx jitter reducer circuit is designed in the $70nm$ Berkley Predictive process using CADENCE^{TM} tools and simulated using the SPECTRE^{TM} analog simulator. The circuit corrects a clock signal of frequency $1\,GHz$. The propagation delay τ_{pd} of the jitter reduction circuit is $118.3\,ps$ for rising and $184.4\,ps$ for falling transitions. The delay element is designed using two buffers and their transistor lengths were chosen, such that the propagation delay (delay element) is $381.3\,ps$ for rising and $344.7\,ps$ for falling transitions, compensating for the asymmetric delays introduced by the jitter reducer circuit, so that the signal $\overline{D_{out}}$ will have a pulse width of $499.6\,ps$ for logic LOW and $529.1\,ps$ for logic HIGH, almost equal to $500ps$ each for logic LOW and HIGH. As explained in Section 6.5, due to constraints on the falling and the rising delays of the inverter, the inverter was modified to avoid glitches, but as a result there will be DCD jitter in the output. But from Figure 4.23, we see that when the input jitter is 0, the RMS DCD jitter produced by the jitter reducer circuit at the output is around $9ps$, which is quite below the industry cut-off of $35.71ps$.

4.2.1 Testing the Tx Jitter Reducer with Input Jitter

Table 4.2: Input Jitter Types for Tx Jitter Reducer

Case #	Jitter Type	PJ ($RMS\,ps$)	RJ ($RMS\,ps$)
1	PJ ($60\,MHz$)	vary	-
2	PJ ($600\,MHz$)	vary	-
3	RJ	-	vary
4	PJ ($60\,MHz$), RJ	18.05	vary
5	PJ ($60\,MHz$), RJ	44.37	vary
6	PJ ($60\,MHz$), RJ	vary	10.47
7	PJ ($60\,MHz$), RJ	vary	18.23

Table 4.2 shows the types of input jitter for which the Tx jitter reducer circuit is tested. Column 2 shows the jitter type, columns 3 and 4 show whether periodic

or random jitter is varied or a fixed value is chosen. The jitter of the input signal with period $1000ps$ is varied by changing the jitter amplitude. For example to introduce periodic jitter, the original time instants of a signal are determined and are modified as follows:

$$t_{PJ} = t_{original} + A_{PJ} \sin(2\pi F_{PJ} t_{original}) \tag{4.36}$$

where t_{PJ} is the modified time instant of the signal with periodic jitter, $t_{original}$ is the original time instant of a signal, A_{PJ} is the periodic jitter amplitude and F_{PJ} is the frequency of the periodic jitter. To vary the periodic jitter of a particular frequency F_{PJ}, the amplitude A_{PJ} is varied by choosing values in multiples of $10ps$. Likewise, a random jitter is modeled as follows:

$$t_{RJ} = t_{original} + A_{RJ}(2 \times rand(1, N) - 1) \tag{4.37}$$

where t_{RJ} is the modified time instant of signal with periodic jitter, A_{RJ} is the random jitter amplitude and N is the number of time instants. The random jitter is varied by changing the random jitter amplitude A_{RJ} in multiples of $10ps$.

MATLAB is used to generate the input signals with various jitters and also to compute the RMS jitter in the output signal. The jitter reducer circuit is simulated with these input signals using the SPECTRE analog simulator. VERILOG modules are used to capture the time instants at which the signals transition and this information is taken to MATLAB to compute the RMS jitter in the signals using a MATLAB script.

4.2.1.1 Analysis

First, an output RMS jitter of 35.71 ps is chosen as a cut-off, as it gives a BER of 10^{-12}, and this value is the one σ standard deviation of the jitter distribution for a good SERDES circuit [24]. The period of the clock signal is $1000ps$. For all of the cases, the output RMS jitter is compared to the input RMS jitter and analyzed as shown below:

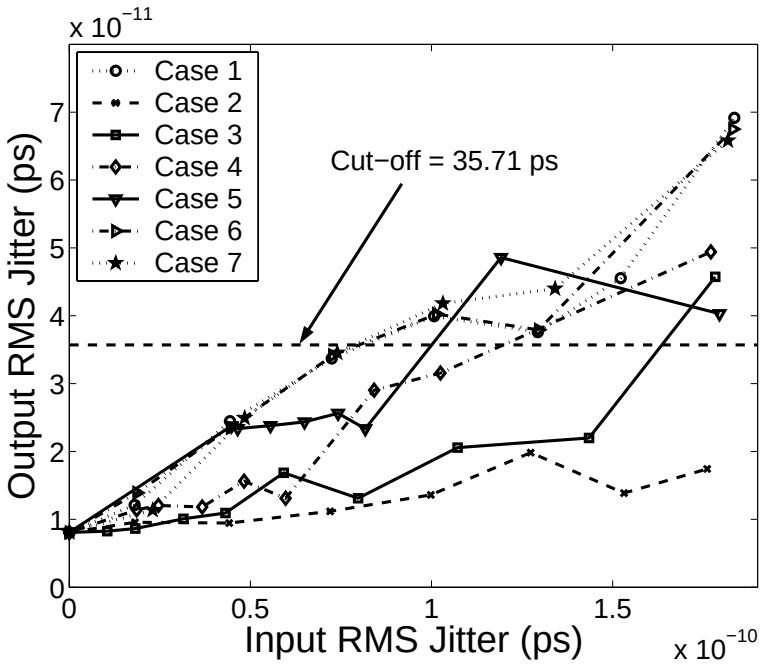


Figure 4.23: Jitter Transfer Function of Tx Jitter Reducer for Various Input Jitter Types

Case 1:

For Case 1, the input jitter is a periodic jitter of $60MHz$, which is a slow frequency jitter. In this case the jitter reduction hardware is tested by varying the jitter amplitude, thereby changing the input RMS jitter values. As seen from Figure 4.23, the output RMS jitter is much reduced and it crosses the cut-off for input RMS values greater than $75ps$. The corresponding output RMS value when the cut-off is crossed is $36ps$.

Case 2:

For Case 2, the input jitter is a periodic jitter of $600MHz$, which is a high frequency jitter. Since the jitter amplitude switches fast between high and low values, it does not affect the jitter performance of the jitter reducer, as evident from the plot shown above. This is the best case for the jitter reducer and all of its output RMS jitter values are below the cut-off.

Case 3:

In Case 3 a random input jitter is used. For a random jitter the output will have a non-linear relationship with the input, as seen in the plot. The jitter reducer is tested for various input RMS random jitter values. The cut-off is crossed for a value of $160ps$, which is a very high value, implying that even though the distribution is random, the jitter reducer is able to reduce the input random jitter drastically.

Case 4:

For Case 4, a combination of periodic jitter of $60MHz$ and random jitter is used. The PJ is fixed with a RMS value of $18.05ps$ and the RJ is varied. The output RMS jitter crosses the cut-off for a input RMS value of $120ps$, which is far greater than the cut-off of $35.71ps$.

Case 5:

Case 5 is similar to the previous case, except that the periodic jitter RMS value is increased to $44.37ps$. As expected, the output jitter performance is jitter degraded and the cut-off is crossed for a input RMS value around $100ps$, which is still greater than the cut-off of $35.71ps$, implying that for most of the input RMS jitter values the output RMS jitter is not greater than the industry standard and therefore the corresponding BER will be less than or equal to 10^{-12}.

Case 6:

For Case 6 a combination of periodic jitter and random jitter is chosen with the random jitter fixed at a RMS value of $10.47ps$. The cut-off is crossed for a input RMS value around $75ps$.

Case 7:

Case 7 is same as the previous case, except that the random jitter RMS value is changed to $18.23ps$. Even though the RMS value of the input random jitter

was increased, it did not degrade the output RMS jitter proportionally and this behavior can be attributed to the random jitter's characteristics. The output RMS jitter crosses the cut-off for an input RMS jitter value of $75ps$ as in the previous case. The Tx jitter reducer, on average, reduced input RMS jitter over seven cases by 62.44%.

4.2.2 Comparison with Adaptive PLL Technique

Table 4.3: Peak-to-Peak Jitter Reduction by the New and Adaptive PLL Techniques [39]

Method	Peak-to-Peak Jitter				%
	Case #	A_{RJ}	Before (ps)	After (ps)	Reduction
Adaptive PLL of Xia *et al.* [39]	1	-	29.2	17.5	40.06
New Jitter Reduction Technique	1	0	21.5	16.5	23.26
	2	2	**29.5**	**17.6**	**40.34**
	3	10	37.2	21.7	41.67
	4	30	51.8	28.6	44.79
	5	40	78.5	32.0	59.24
	6	50	102.4	37.1	63.77
Average			53.48	25.58	45.51

The performance of the Tx jitter reduction circuit is compared with the adaptive PLL technique proposed by Xia *et al* [39]. They have designed a PLL to generate a clock of frequency $500MHz$ using IBM 180 nm CMOS technology. Table 4.3 shows the results of random peak-to-peak jitter reduction for both of the methods. For our experiment, we took six cases of the output signal $\overline{D_{out}}$ with different peak-to-peak random jitter values as shown in column 4. The random jitter in the output signal is due to a combination of the random jitter in the input signal D_{in} and also the random jitter present in the jitter reduction circuit. The peak-to-peak random jitter value in the output signal is varied by varying the amplitude of the random jitter in the input signal D_{in} and the quantity A_{RJ} in Equation 4.37 is used to vary the random jitter in the input signal and column

3 gives the different values of the quantity A_{RJ}. In case 1 of our experiment, the A_{RJ} value used is zero, implying that the peak-to-peak random jitter value obtained is only due to the random jitter in the jitter reduction circuit. Columns 5 and 6 show the reduction in peak-to-peak jitter obtained using both of the methods and their corresponding percentage reduction. For the comparison, the result from the adaptive PLL technique is compared with the result shown in boldface for the new technique. We see that the results are comparable, but the difference is that the new technique is applied on a PLL that generates a clock of frequency 1 GHz and the technique uses only 12 transistors as opposed to the adaptive PLL technique, which uses huge on-chip hardware involving two counters and a comparator and is used for reducing jitter in a clock signal of frequency $500MHz$, a slower signal than the $1GHz$ signal we use. The average peak-to-peak jitter reduction obtained using our technique is 45.51%.

4.2.3 Phase Delay Introduced by the Jitter Reducer Circuit – Compensation

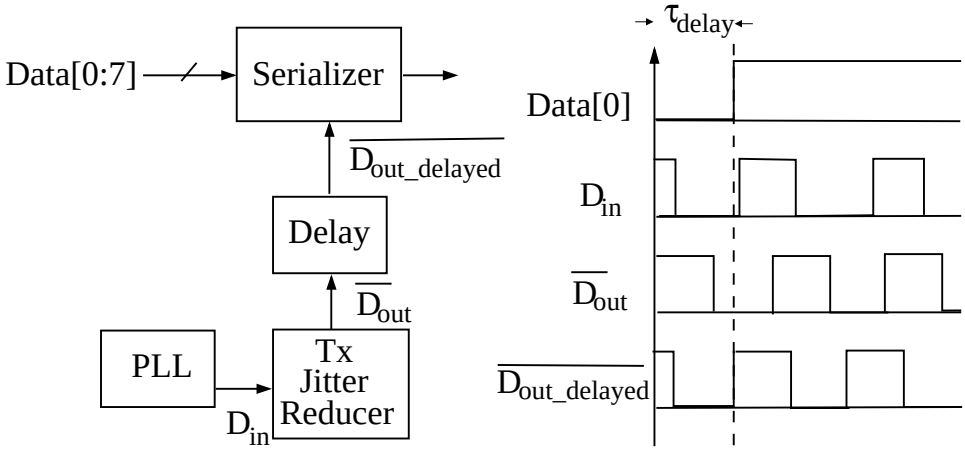


Figure 4.24: Tx Jitter Reducer Phase Delay Compensation

The jitter reducer circuit output signal $\overline{D_{out}}$ has a phase delay of $\tau_{delay}\, seconds$ with reference to the rising edge of the input $Data[0]$ of the serializer as shown

in the timing diagram of Figure 4.24. The signal $\overline{D_{out}}$ is phase aligned with the serializer input through a delay element that introduces a delay of τ_{delay} *seconds* to $\overline{D_{out}}$, thereby generating the new phase aligned signal $\overline{D_{out\_delayed}}$. From the timing diagram in Figure 4.24, we see that the Tx jitter reducer reduces the jitter in the input jittered signal D_{in} and the jitter reduced signal $\overline{D_{out}}$ is out of phase with the signal $Data[0]$ by τ_{delay} *seconds*. Finally, after the signal $\overline{D_{out}}$ is delayed by the delay element, the delayed signal $\overline{D_{out\_delayed}}$ is in phase with the signal $Data[0]$.

4.3 Summary

The proposed transmit side jitter reduction technique effectively reduced the input RMS jitter over seven input jitter cases, on average, by 62.44% and peak-to-peak random jitter, on average by 45.51%. The proposed technique uses an external hardware to reduce jitter rather than modifying the internal PLL circuity. The methodology is based on a simple mathematical approach where the amount of jitter present is determined on-chip and is reduced. The jitter reduced signal is looped-back to the input of the jitter reducer to compute the error again. The hardware proposed is also simple to design as it is made of only 14 transistors.

Chapter 5

SERDES with Transmit and Receive Side Jitter Reducers

In this chapter a jitter reduction technique is proposed for reducing the jitter in the receive side of the high-speed SERDES circuits. The SERDES circuit is then integrated with the transmit side and the receive side jitter reducers and its performance is analyzed.

5.1 Jitter Reduction Technique for the Receive Side Clock and Data Recovery Circuits

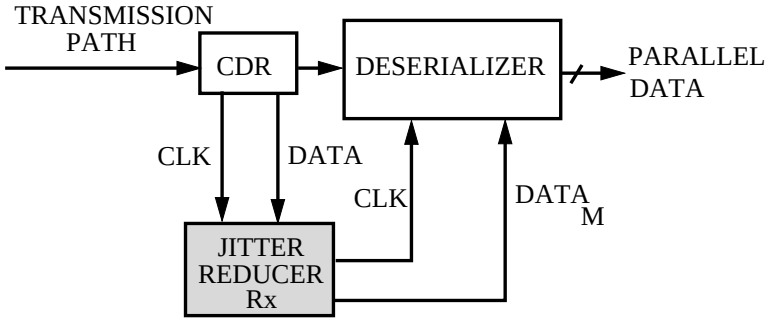


Figure 5.1: Receive Side Jitter Reducer in the SERDES Circuit

Figure 5.1 shows where the jitter reducer is present in the receive side of the SERDES circuit. The jitter reducer takes CLK and $DATA$ as input and produces at its output the modified data signal $DATA_M$, which along with the original CLK signal is given to the de-serializer.

5.1.1 Problem with the Clock and Data Recovery Circuit

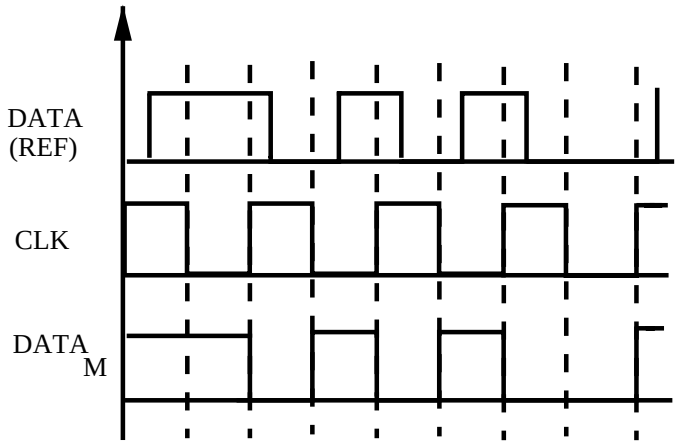


Figure 5.2: Timing Waveform for Receive Side Jitter Reduction

At the receive side of the SERDES, the timing jitter between the recovered clock and data from the CDR circuit results in erroneous data being latched by the flip-flop (a bit-error). The solution is to reduce this jitter between the recovered clock and the data. Figure 5.2 shows the modified data signal $DATA_M$ that is aligned with the clock signal after the process of jitter reduction.

5.1.2 Process of Jitter Reduction and Jitter Reduction Circuit

In this section a receive side jitter reducer circuit is proposed and implemented. This circuit is based on the circuit used for the transmit side jitter reduction but with some modification and the reason for the modification is because of the nature of the reference signal used in the case of the receive side jitter reducer, which is the received serial data signal. In the case of the Tx jitter reducer, the reference signal is generated from the propagation delays of the jitter reducer circuit and the delay element, respectively, and so its signal transitions do not depend on the input signal. As a result the output signal has less DCD jitter. But in the case of the Rx jitter reducer, the reference signal used is one of the input

jittered signals, which is the incoming serial data and as a result the output has relatively more DCD jitter. So an explicit technique is required to improve the jitter reducer circuit. Figure 5.3 shows the receive side jitter reduction circuit. The process of jitter reduction is similar to that of the transmit side as explained in Section 4.1.1, except that, instead of using a time delayed signal as a reference, the data signal from the CDR circuit is used.

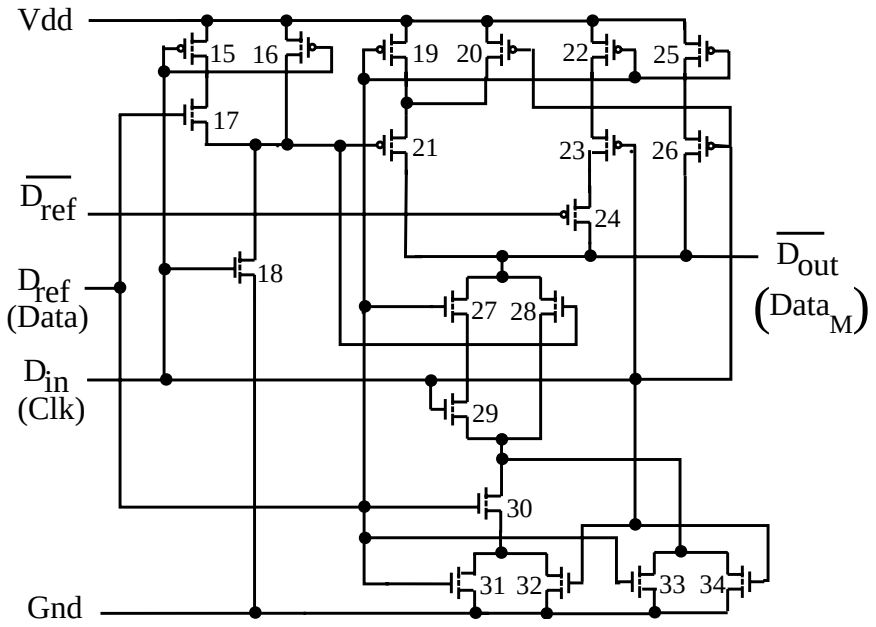


Figure 5.3: Receive Side Jitter Reduction Circuit

The timed Boolean expression for the rising and the falling transitions of the output is given as follows:

$$\begin{aligned}
\overline{D_{out_r}}(t) &= (\overline{D_{in}}(t - \tau_{rCOMP}) + \\
&\quad \overline{D_{ref}}(t - \tau_{rCOMP})) \cdot \\
&\quad D_{in}(t - \tau_{rCOMP} - \tau_{fINV})) + \\
&\quad (\overline{D_{in}}(t - \tau_{rCOMP}) \cdot \\
&\quad \overline{D_{ref}}(t - \tau_{rCOMP})) \qquad (5.1)
\end{aligned}$$

$$\overline{D_{out\_f}}(t) = (\overline{D_{in}}(t - \tau_{fCOMP}) +$$

$$\overline{D_{ref}}(t - \tau_{fCOMP})) \cdot$$
$$D_{in}(t - \tau_{fCOMP} - \tau_{rINV})) +$$
$$(\overline{D_{in}}(t - \tau_{fCOMP}) \cdot$$
$$\overline{D_{ref}}(t - \tau_{fCOMP})) \quad (5.2)$$

Apart from controlling the inverter delays as explained in Section 4.1.4, the rising (τ_{rCOMP}) and the falling (τ_{fCOMP}) delays of the composite gate are also controlled to reduce the DCD jitter in the output. From the experiments conducted, it was observed that the reason for the increased DCD jitter in the output was that under certain input conditions, when the falling delay of the composite gate τ_{fCOMP} was greater than the rising delay τ_{rCOMP}, it increased the DCD jitter as shown in Figure 5.4. To reduce the DCD jitter in the output, the rising and

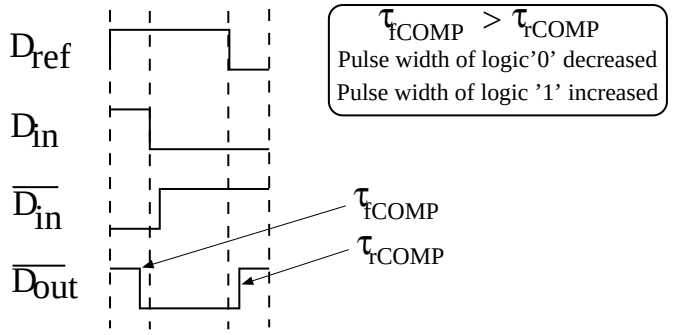


Figure 5.4: Receive Side Jitter Reduction Circuit – A Case for DCD Jitter

falling delays of the Rx jitter reducer have to be modified under certain input conditions. From the above figure, we see that when D_{ref} and D_{in} are logic '0,' the falling delay has to be reduced and when D_{ref} and D_{in} are logic '1,' the rising delay has to be increased to reduce the DCD jitter. This can be accomplished by modifying the p-tree and n-tree, respectively, by providing two different delays under the input conditions stated above.

Theorem 5.1: *If the falling delay τ_{fCOMP} is decreased when D_{ref} and D_{in} are logic '0' and if the rising delay τ_{rCOMP} is increased when D_{ref} and D_{in} are logic*

'1,' then the DCD jitter at the output $\overline{D_{out}}$ will be reduced.

Proof: From Figure 5.4, we see that if the falling delay τ_{fCOMP} is decreased when D_{ref} and D_{in} are logic '0' and if the rising delay τ_{rCOMP} is increased when D_{ref} and D_{in} are logic '1,' then the pulse width of logic '0' will be increased and the pulse width of logic '1' will be reduced, thereby reducing the DCD jitter. ■

From Figure 5.3, we see that the part of the circuit that does the process of *add* in the p-tree and the n-tree, respectively is modified to provide two different delays for both the rising and the falling cases. For the p-tree, when D_{ref} and D_{in} are '0,' the rising delay is longer and is primarily because of transistors 25 and 26, since all other paths in the p-tree are cut off and the expression is given as follows:

$$\tau_{rCOMP} = (R_{25} + R_{26}) \times (C_{21} + C_{24} + C_{26} + C_{27} + C_{28}) \tag{5.3}$$

For the n-tree, when D_ref and D_{in} are logic '1,' the parallel combination of the paths through the transistors $30, 31, 32$ and $33, 34$ gives the shorter falling delay and the expression is given as follows:

$$\begin{aligned} \tau_{fCOMP} &= (R_{27} + R_{29}) + ((R_{30} + (R_{31} \| R_{32})) \| (R_{33} \| R_{34})) && (5.4) \\ &\quad \times (C_{21} + C_{24} + C_{26} + C_{27} + C_{28}) && (5.5) \end{aligned}$$

As explained in Section 4.1.4, the W/L ratios of the various transistors were determined based on the performance of the Rx jitter reducer for various input jitter types. From the experiments conducted, as discussed in the next sections, the optimal values of the various transistors are show in Table 5.1. The p-transistors $22 - 26$ perform the *add* operation and transistors $19 - 21$ perform the *remove* operation, respectively, for the p-tree. The n-transistors $31 - 34$ perform the *add* operation and transistors $27 - 28$ perform the *remove* operation, respectively, for the n-tree. Transistors $15 - 18$ constitute an inverter.

Table 5.1: W/L Ratios of All the Transistors in the Receive Side Jitter Reduction Circuit

Transistor No.	W/L Ratio
15	$80nm/80nm$
16	$80nm/1.2\mu m$
17	$80nm/80nm$
18	$80nm/80nm$
19	$80nm/80nm$
20	$80nm/80nm$
21	$80nm/80nm$
22	$80nm/80nm$
23	$80nm/80nm$
24	$80nm/80nm$
25	$80nm/1.2\mu m$
26	$80nm/1.2\mu m$
27	$80nm/80nm$
28	$80nm/80nm$
29	$80nm/80nm$
30	$80nm/80nm$
31	$80nm/80nm$
32	$80nm/80nm$
33	$80nm/400nm$
34	$80nm/400nm$

Table 5.2: Input Jitter Types for Rx Jitter Reducer

#	Jitter Type	RMS PJ (ps)	RMS RJ (ps)	RMS DCD (ps)
1	PJ (60 MHz) - D_{in}	vary	-	-
2	PJ (60 MHz) - D_{ref}	vary	-	-
3	RJ - D_{in}, D_{ref}	-	vary	
4	PJ (600 MHz) - D_{in} (300 MHz) - D_{ref}	vary D_{in}	-	-
5	PJ (600 MHz) - D_{in}, (300 MHz) - D_{ref}	vary D_{in} & D_{ref}	-	-
6	DCD - D_{in}	-	-	vary
7	DCD - D_{ref}	-	-	vary
8	PJ (60 MHz), RJ, DCD - D_{in} PJ (60 MHz), RJ - D_{ref}	43.77 -D_{in} 42.44 -D_{ref}	5.15 -D_{in} 9.27 -D_{ref}	vary -
9	PJ (60 MHz), RJ - D_{in} PJ (60 MHz), RJ, DCD -D_{ref}	43.77 -D_{in} 42.44 -D_{ref}	5.15 -D_{in} 9.27 -D_{ref}	vary -

5.2 Results for Receive Side Jitter Reducer

The circuit was designed in a 70 nm Berkeley Predictive process using CADENCE^{TM} tools and simulated using the SPECTRE^{TM} analog simulator. Table 5.2 shows the various jitter types for the D_{in} and D_{ref} signals with which the circuit was tested. In the table below, we have used the notations D_{in} and D_{ref} to represent the inputs for which the jitter conditions are set. Column 2 shows the type of jitter present either in the D_{in} (clock) or the D_{ref} (data) signal and columns $3-5$ show whether the jitter amplitude is varied or a fixed value is used.

5.2.1 Analysis

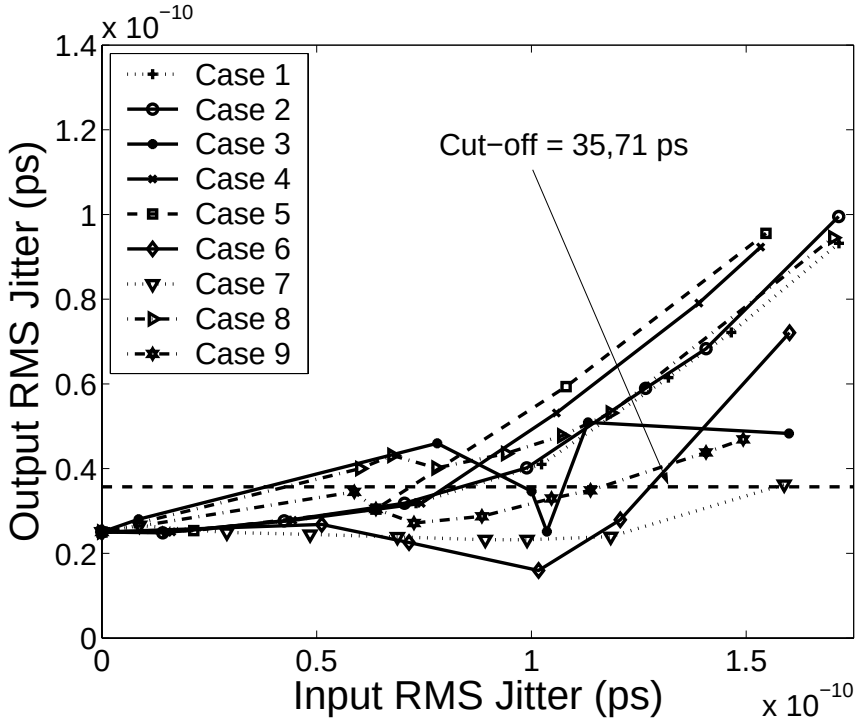


Figure 5.5: Jitter Transfer Function of the Rx Jitter Reduction Circuit

Figure 5.5 shows the plot of output *vs.* input RMS jitter. As in the case of the transmit side, a cut-off of 35.71ps was chosen to analyze the performance of the receive side jitter reducer with respect to the industry standard. The period

of the clock signal is $1000ps$. The analysis of all the cases is given below:

Case 1:

In Case 1 a periodic jitter of $60MHz$ is given to D_{in}, which is the clock signal and no jitter is present in the data signal (D_{ref}). The output RMS jitter crosses the cut-off for input RMS jitter values greater than $80ps$. The output RMS jitter increases monotonically as the input RMS jitter increases, which in turn is varied by changing the jitter amplitude of the input periodic jitter.

Case 2:

In Case 2 the periodic jitter is added in the data signal and there is no jitter in the clock signal. The output RMS jitter performance is almost similar to the previous case, implying that regardless of where the jitter is present the jitter reducer is able to reduce it.

Case 3:

In Case 3, random jitter is introduced both in the clock and the data signal. The output RMS jitter performance is the least of all the cases. The output RMS jitter is crossed at around $40ps$. Also, the output RMS jitter does not show a linear relationship with the input RMS random jitter.

Case 4:

In Case 4 a periodic jitter of $600MHz$ is given to the clock signal and a periodic jitter of $300MHz$ is given to the data signal to find the effect of the presence of periodic jitter in both the clock and the data signal at the same time. For this case, the periodic jitter is fixed for the data signal and is varied for the clock signal by varying the jitter amplitude. From the plot we see that the output RMS jitter crosses the cut-off at around $75ps$, which is a good value. The result demonstrates that even in the presence of periodic jitter in both inputs, the jitter reducer is able to reduce it and align the data signal with the clock signal.

Case 5:

For Case 5 the input jitter conditions are same as the previous case, except that the periodic jitter amplitude is varied for both inputs. The output RMS jitter is crossed at around $75ps$, the same as in Case 4.

Case 6:

In Case 6, duty cycle distortion jitter is introduced into the clock signal and the output RMS jitter is determined for various DCD jitter values. The output jitter performance is very good, with the cut-off crossed for input values greater than $130ps$. The jitter reducer reduces the jitter drastically, if the input jitter present is of the DCD jitter type.

Case 7:

In case 7 the DCD jitter is present in the data signal. From the plot we see that this is the best case, with the output RMS jitter values crossing the cut-off for input RMS values greater than $160ps$.

Case 8:

In Case 8 a combination of periodic jitter, random jitter and DCD jitter is introduced in the clock signal, with the periodic and random jitter fixed with RMS values of $43.77ps$ and $5.15ps$, respectively, and varying the DCD jitter. Similarly, for the data signal, a combination of only periodic and random jitter is introduced, where both the periodic and random jitter are fixed at RMS values of $42.44ps$ and $9.27ps$, respectively. The output RMS jitter performance is almost similar to Case 3, where only RJ was used. So, the jitter performance in this case can be attributed to the random jitter characteristics. But still the output RMS jitter values are less than the corresponding input RMS jitter values.

Case 9:

Case 9 is similar to the previous case, except that the DCD jitter is present in the data signal and not the clock signal. The output RMS jitter is considerably better than the previous case with the output RMS jitter crossing the cut-off for input RMS jitter values greater than $110ps$.

5.3 Jitter Performance of SERDES with Tx and Rx Jitter Reducers

In this section, the performance of the high-speed SERDES circuit in the presence of transmit and receive jitter reducers is analyzed. Table 5.3 shows the three configurations for the SERDES Circuit.

Table 5.3: Three Cases of SERDES Jitter Reducers

Case #	Tx Jitter Reducer	Rx Jitter Reducer
1	no	no
2	no	yes
3	yes	yes

In Case 1 there are no jitter reducers present. From this we can deduce the original jitter performance of the SERDES circuit. In Case 2, the receive side jitter reducer is inserted. In this case the effect of the receive side jitter reducer in improving the BER performance of the SERDES can be analyzed. Finally, in Case 3, both the transmit side and the receive side jitter reducers are inserted. From this case, we can deduce the combined effect of both jitter reducers in improving the BER performance of the SERDES circuit.

5.3.1 BER Analysis

The SERDES circuit in all the three cases is tested by introducing a periodic jitter of $60MHz$ in the input clock signal signal at the transmit side of the SERDES circuit. Figure 5.6 shows the plot of the output RMS jitter against the input RMS jitter. The output RMS jitter is measured at the output of the clock

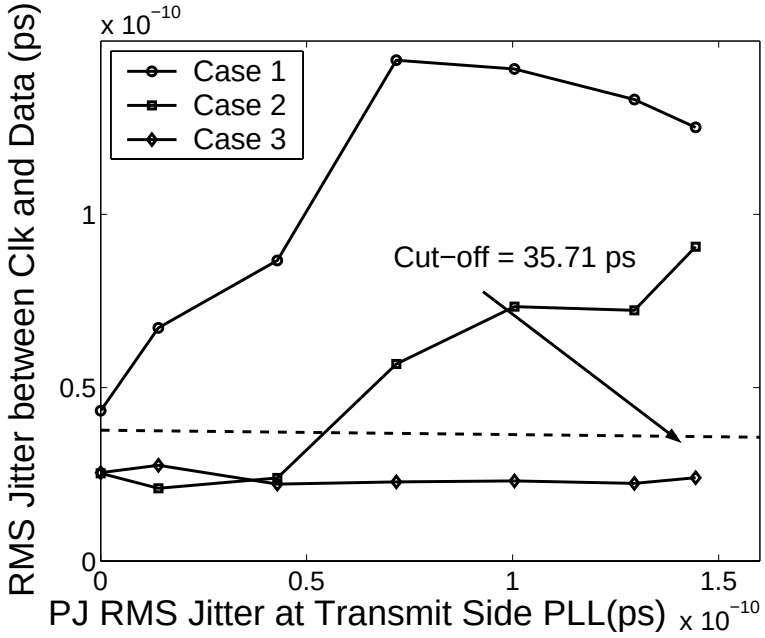


Figure 5.6: Total Jitter Transfer Function with the Tx and Rx Jitter Reducers

and data recovery circuit in Case 1 and at the output of the receive side jitter reducer in Cases 2 and 3. The input periodic jitter is varied by varying the jitter amplitude. The output RMS jitter performance is analyzed in all three cases as given below:

Case 1 – No Jitter Reducers:

In this case, the output RMS jitter values are almost the same as the input RMS jitter values. Since there is no jitter reducer at the transmit side, the jitter from the clock is transferred to the data signal, and when such a jittered data signal is received by the CDR circuit, the clock recovered from the incoming jittered data signal is also jittered. Figure 5.6 clearly shows this behavior, where the output RMS jitter values are as bad as the input RMS jitter values. This outcome clearly demonstrates the need for the transmit side and the receive side jitter reducers. In terms of the BER performance, for a particular case where the input RMS jitter is $71.77ps$, the BER is computed probabilistically using the

equations given in the work by Hong *et al.* [14] is 8.3×10^{-2}, which is a highly degraded BER performance.

Case 2 – Receive Side Jitter Reducer Present:
In this case, the receive side jitter reducer is inserted to reduce the jitter between the clock and the data signal at the output of the CDR circuit. As expected, the output RMS jitter values are considerably lower than the corresponding input RMS jitter values, but they are not reduced drastically. The primary reason is that the incoming serial data has considerable levels of jitter present in it, which can be reduced. The BER computed for the particular case is reduced to 1.09×10^{-5}, which is three orders of improvement.

Case 3 – Transmit and Receive Side Jitter Reducers Present:
Finally, in this case both jitter reducers are present. The SERDES circuit jitter performance is considerably improved with the output RMS jitter values remaining below the cut-off for all of the input RMS jitter values. The main reason other than the jitter reduction by the receive side jitter reducer is that the jitter in the incoming serial data is already reduced by the transmit side jitter reducer, and when such a jitter reduced signal is presented to the receive side jitter reducer, the complexity of its jitter reduction process is greatly reduced. The greatest improvement in BER performance is achieved for this case, where the BER is further improved to 6.44×10^{-20}.

5.4 Summary

The benefits of the transmit side and receive side jitters are clearly demonstrated in this chapter, where it is shown than the BER can be improved drastically from 8.3×10^{-2} to 6.44×10^{-20}. Apart from this, the receive side jitter reducer is shown to reduce output RMS jitter, on average, by 35.88% for various input RMS jitter conditions.

Chapter 6
Circuit Design Issues and Testing

In this chapter, the Tx and Rx jitter reducers are analyzed for their performance under *process variations* (PV). The effect of transistor width sizing in controlling the output response variations in the presence of process variation is analyzed. The jitter reducer circuit is designed in a layout and the parasitics are extracted and their effect on the output jitter performance is determined. In the latter part of the chapter a testing scheme for performing the various tests for the SERDES circuit is proposed and implemented.

6.1 Monte Carlo Analysis and Process Variation Parameters

Monte Carlo analysis, a feature of SPECTRE^{TM}, is used for the process variation experiment. The following parameters are varied for both the nMOS and pMOS transistors of the Berkeley Predictive process: $t_{ox},\ c_j,\ c_{jsw},\ \mu_0,\ v_{tho}$ and p_{clm}, where t_{ox} is the gate oxide thickness, c_j is the junction thickness, c_{jsw} is the junction side wall capacitance, μ_0 is the low-field surface mobility and p_{clm} (used only in SPECTRE models, whereas in SPICE models λ is used) is the channel length modulation. All of the parameters have Gaussian distributions with their nominal values as mean and a standard deviation of 20% for $c_j,\ c_{jsw}$ and p_{clm} and 10% for $t_{ox},\ v_{tho}$ and μ_0.

6.2 Conditions on the Timing Delays Due to Process Variations

From Section 4.1.4 we observed that to reduce glitches in the output the inverter delays should be greater than T_{jitter} and on increasing the inverter delays, we get DCD jitter. To see the effect of process variations on the inverter delays and their effect on causing glitches and DCD jitter in the output, we consider two parameters, Δ_r and Δ_f, respectively, where Δ_r and Δ_f are the quantities by which either the rising or the falling inverter delay is either increased or decreased under process variations. Let J_{DCD} be the allowable DCD jitter, which is equal to $35.71ps$, the industry cut-off. The constraint on the rising and the falling delays to avoid glitches under process variations at the output is as follows:

$$(\tau_{rINV} - \Delta_r) > T_{jitter} \tag{6.1}$$

$$(\tau_{fINV} - \Delta_f) > T_{jitter} \tag{6.2}$$

The reason for the above conditions is that under process variations, the inverter delays should not be reduced below the T_{jitter} value to avoid glitches (refer to Section 4.1.4 for details). The conditions for avoiding DCD jitter for both the rising and the falling inverter delays are as follows:

$$(\tau_{rINV} + \Delta_r) - T_{jitter} \leq J_{DCD} \tag{6.3}$$

$$(\tau_{fINV} + \Delta_f) - T_{jitter} \leq J_{DCD} \tag{6.4}$$

The reason for the above conditions is that if the inverter delays are increased above the T_{jitter} value, they result in DCD jitter and the maximum allowed DCD jitter is $35.71ps$, so that under the case where there is no jitter in the input signals, the RMS DCD jitter produced internally from the jitter reducer circuit will be less than $35.71ps$. The effect of process variation on the jitter reducer circuit performance can be controlled by changing the widths W of all the transistors, both in the transmit and the receive side jitter reducer. The process of determining this optimal transistor width is explained in the next section.

6.3 Optimal Transistor Width

First, the transistor width W for which the jitter performance of the jitter reducers are optimal under process variations is determined. For this experiment the Tx jitter reducer is used and is tested with an input signal with no jitter in it. The transistor widths are varied and the output RMS jitter under process variations is determined.

Table 6.1: Optimal Transistor Width

$W\ (nm)$	Output RMS RJ (ps)
100	39.08
160	39.14
200	38.63
300	37.96
500	37.13
700	46.09
1000	70.81

Table 6.1 shows the various values of W considered and the corresponding output RMS random jitter. The output RMS jitter decreases initially, but for widths over 700 nm it increases. A width of value 200 nm is chosen, because the corresponding output RMS jitter is closer to the cut-off and also it does not increase the overall chip area of the jitter reducers.

6.4 Tx and Rx Jitter Reducers under Process Variations

The Tx and Rx jitter reducers are tested for their jitter performance under process variations. For the Tx jitter reducer, to simplify the process of jitter measurement, instead of the loop back signal, an external reference signal is provided. It is tested with a periodic jitter of 60 MHz on the input signal D_{in}. In the case of the Rx jitter reducer, it is tested with a periodic jitter of 60 MHz on the D_{in} (clock) signal.

Figure 6.1 shows the performance of the jitter reducers. The solid lines and broken lines are for the Tx and Rx jitter reducers, respectively. For the Tx

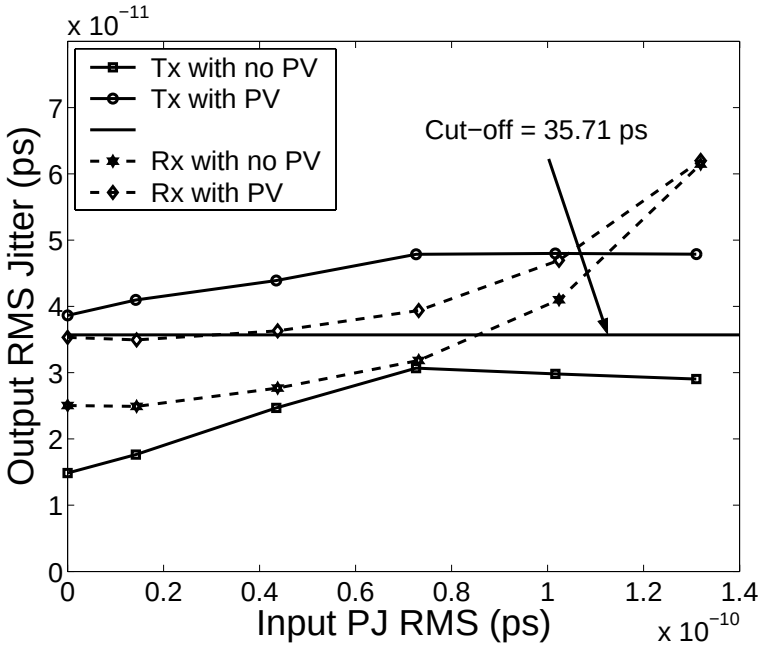


Figure 6.1: Jitter Transfer Function of the Tx and Rx Jitter Reducers under Process Variations

jitter reducer the output RMS jitter under process variations is higher compared to the output RMS jitter without process variations, which can be improved with good circuit design and for the Rx jitter reducer, the output RMS jitter is initially higher but approaches the output RMS jitter without process variations for higher input RMS jitter values. But for both the Tx and Rx jitter reducers, even under process variations the output RMS jitter is considerably lower than the corresponding input RMS jitter. In the case of the Tx jitter reducer an input RMS jitter of $72.59\,ps$ is reduced to $41.73\,ps$ and for the Rx jitter reducer an input RMS jitter of $73.14\,ps$ is reduced to $39.38\,ps$, which are a 42.41% and a 46.16% reduction, respectively.

6.5 Layout and Parasitics

The Tx jitter reducer circuit was designed using the Layout Editor of CADENCE^{TM} and parasitic capacitances were extracted. The propagation delay of the delay element was adjusted to account for the extra delay in the jitter

reducer circuit due to the parasitics.

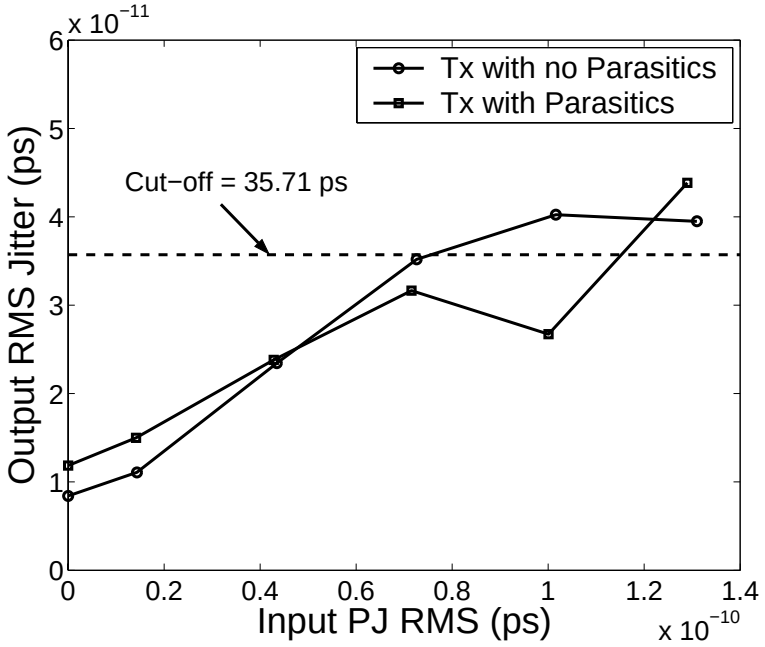


Figure 6.2: Jitter Transfer Function of the Tx Jitter Reducer with Parasitic Capacitances

The extracted circuit was tested for its jitter performance with a periodic jitter of 60 MHz. Figure 6.2 shows the performance of the extracted circuit, and the output RMS jitter is almost the same as the jitter reducer circuit with no parasitics. This was primarily due to the modified delay of the delay element, which accounted for the extra delay. Therefore, the delay constraint that the propagation delay of the delay element and the jitter reducer circuit should be equal to τ (the bit period of logic HIGH and LOW) is not affected. As far as testing the jitter reducer for input jitter frequencies close to a $1GHz$, we infer from Case 2 of the analysis of testing the Tx jitter reducer in Section 4.2.1.1 that the higher the input jitter frequency, the less will be its effect on the output RMS jitter, since the jitter switches between high and low values very fast and the Tx jitter reducer is very efficient in reducing the high frequency jitter (see Figure 4.23). Figures 6.3 and 6.6 show the layouts of the Rx and Tx jitter reducers in a $70nm$ Berkeley Predictive process, respectively. Figures 6.4 and 6.5 show the result of periodic and DCD jitter reduction in D_{in} by the Tx jitter reducer and Figure 6.7 shows the result of DCD jitter reduction by the Rx jitter reducer in

D_{ref}.

6.6 Pole-Zero Analysis

Addition of an external circuit block as a load to an existing block tends to change the magnitude and phase response of the latter, if there is a change in the location of poles and zeros of the existing block. In our case, we wanted to determine whether there was any change in the magnitude and phase behavior of the the PLL circuit both at the transmit and the receive side as a result of adding the jitter reducers at their respective outputs. This was done using the *pole-zero* (PZ) analysis feature of SPECTRE^{TM}. Tables 6.2 and 6.3 show the values of the poles and zeros of the PLL circuit at the transmit side without and with the Tx jitter reducer.

Table 6.2: Transmit Side PLL without Tx Jitter Reducer – Values of Poles and Zeros

#	Poles (Hz)	
	Real	Imaginary
1	-6.91146e+08	0.00000e+00
2	2.89281e+09	± 2.25773e+09
3	-4.33070e+09	0.00000e+00
4	-1.62474e+09	± 4.12059e+09
5	-6.51648e+09	0.00000e+00
#	Zeros (Hz)	
	Real	Imaginary
1	6.82379e+08	0.00000e+00
2	2.07588e+08	± 7.51593e+08
3	-9.60646e+08	± 1.15217e+09
4	-1.60208e+09	0.00000e+00
5	-6.17364e+09	0.00000e+00
6	-9.10407e+09	0.00000e+00

On comparing the two tables, we observe there are no changes in the locations of the poles and zeros, except for the last zero, which is $-9.10407e+09(Hz)$ in the case of the PLL without the jitter reducer and is $-9.10240e+09(Hz)$ in

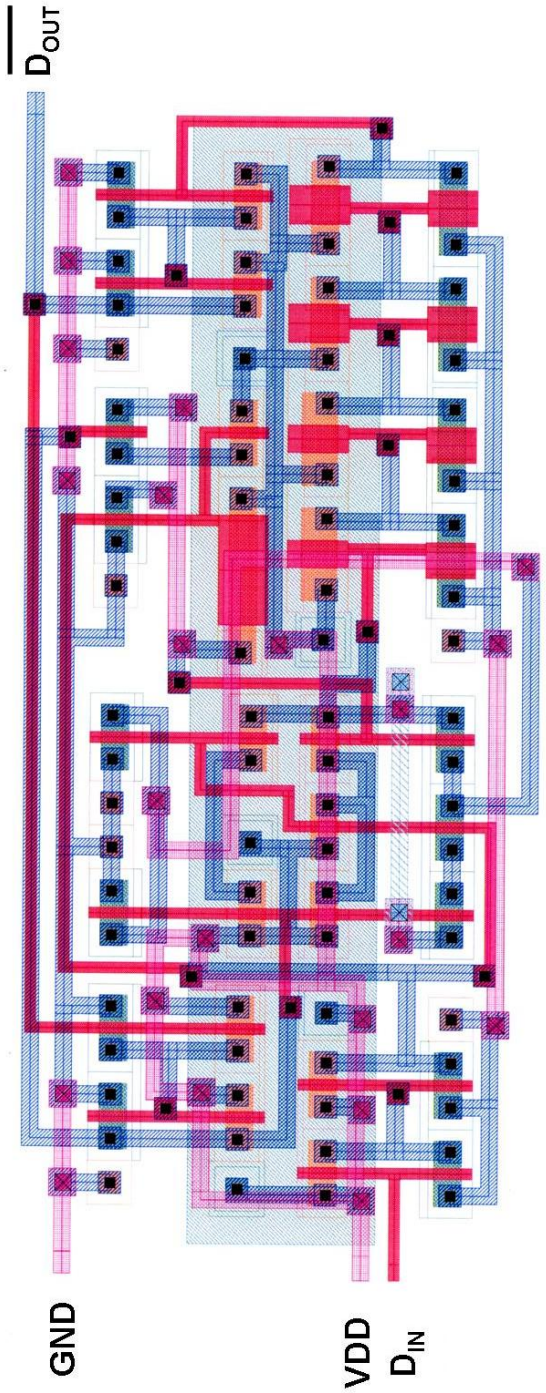


Figure 6.3: Tx Jitter Reducer Layout

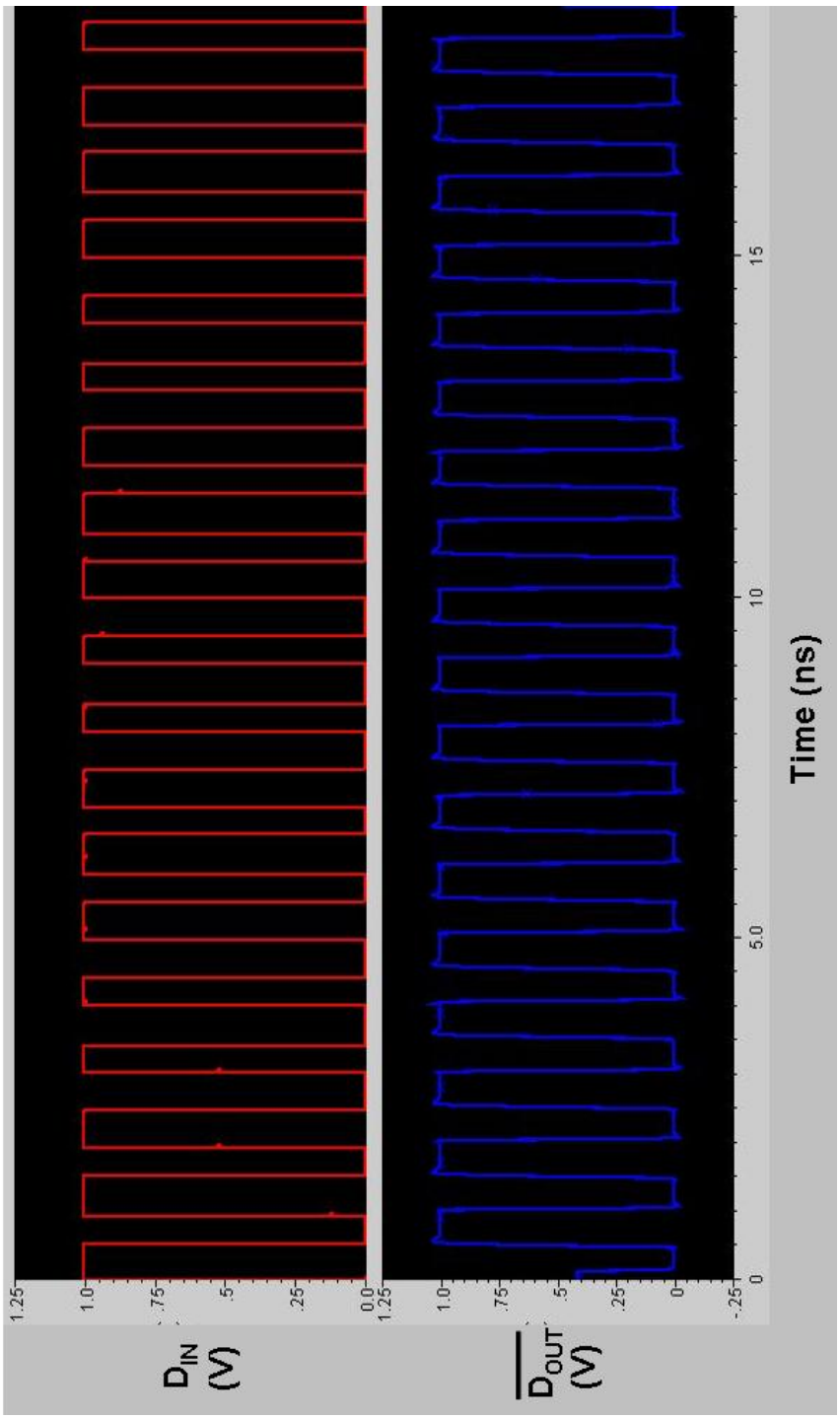


Figure 6.4: Periodic Jitter Reduction by Tx Jitter Reducer

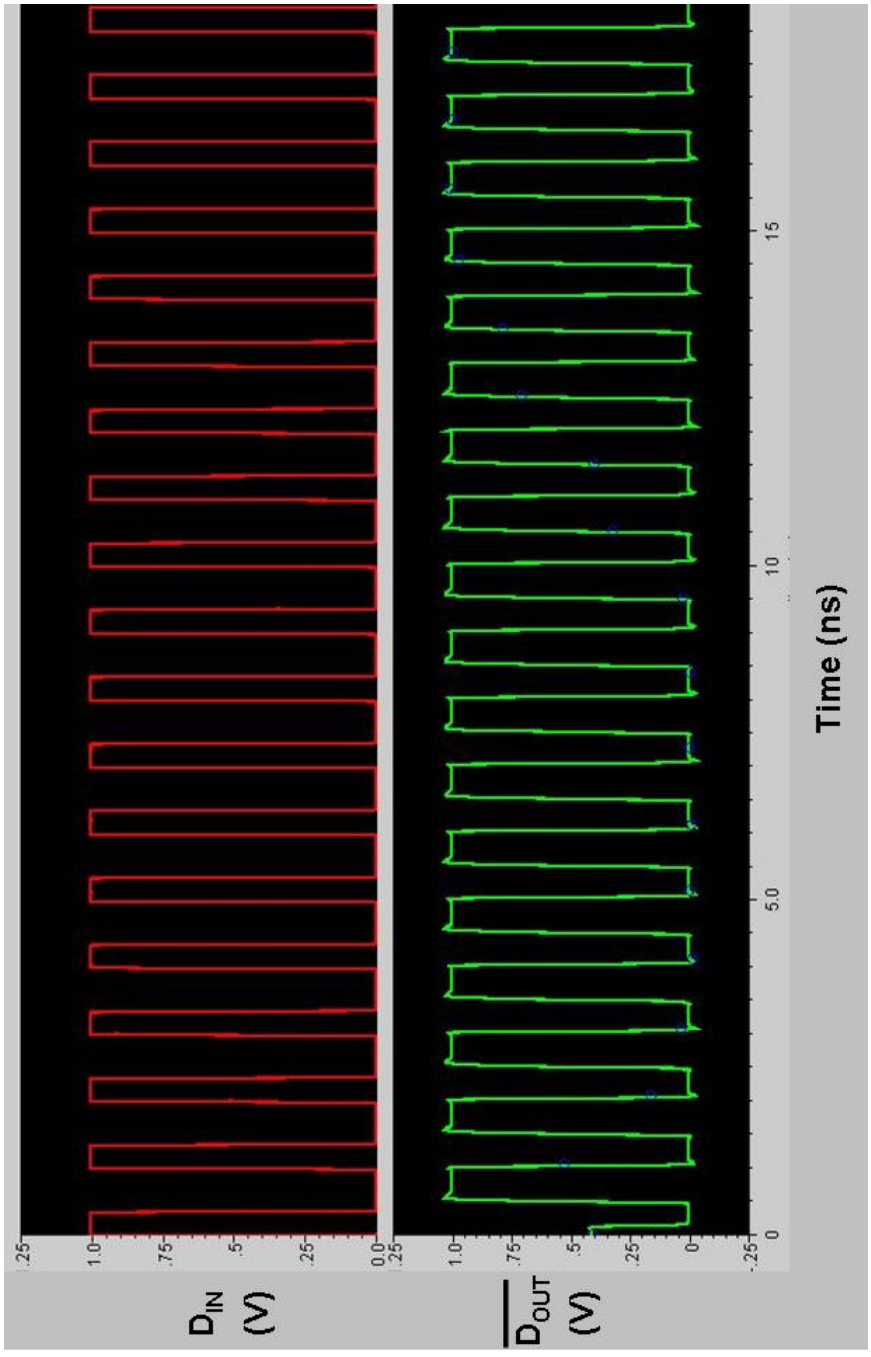


Figure 6.5: DCD Jitter Reduction by Tx Jitter Reducer

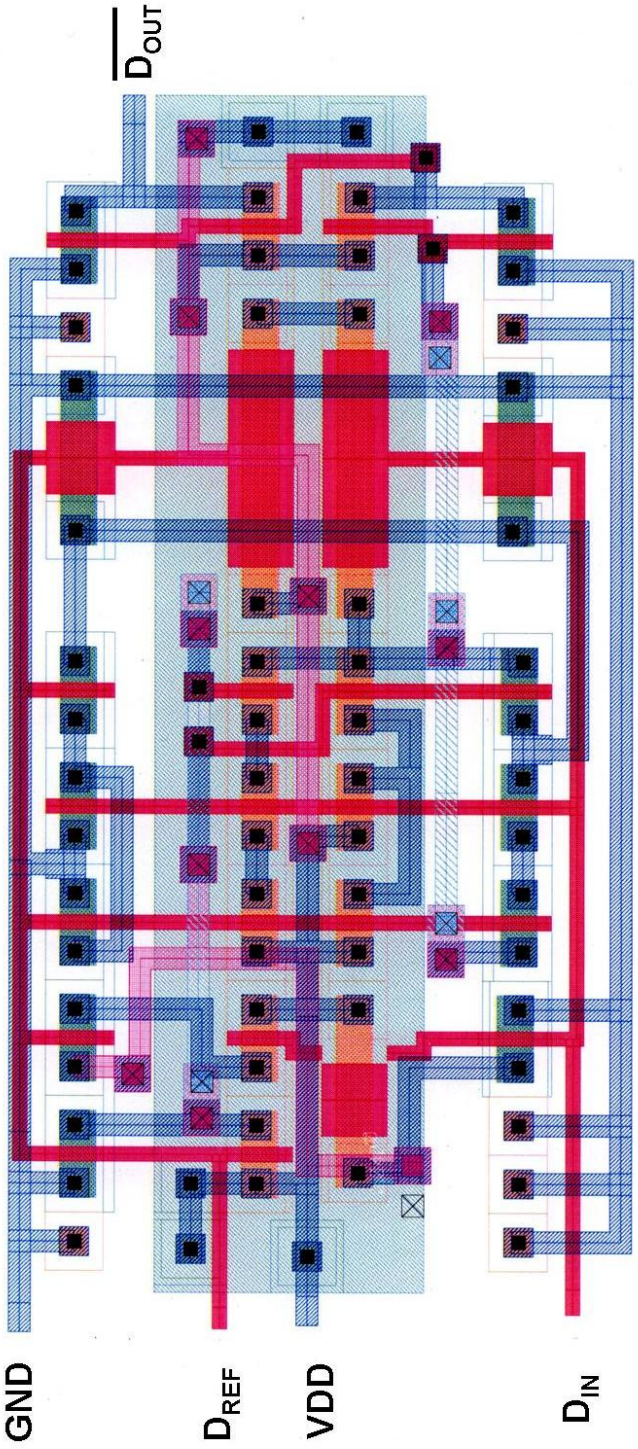


Figure 6.6: Rx Jitter Reducer Layout

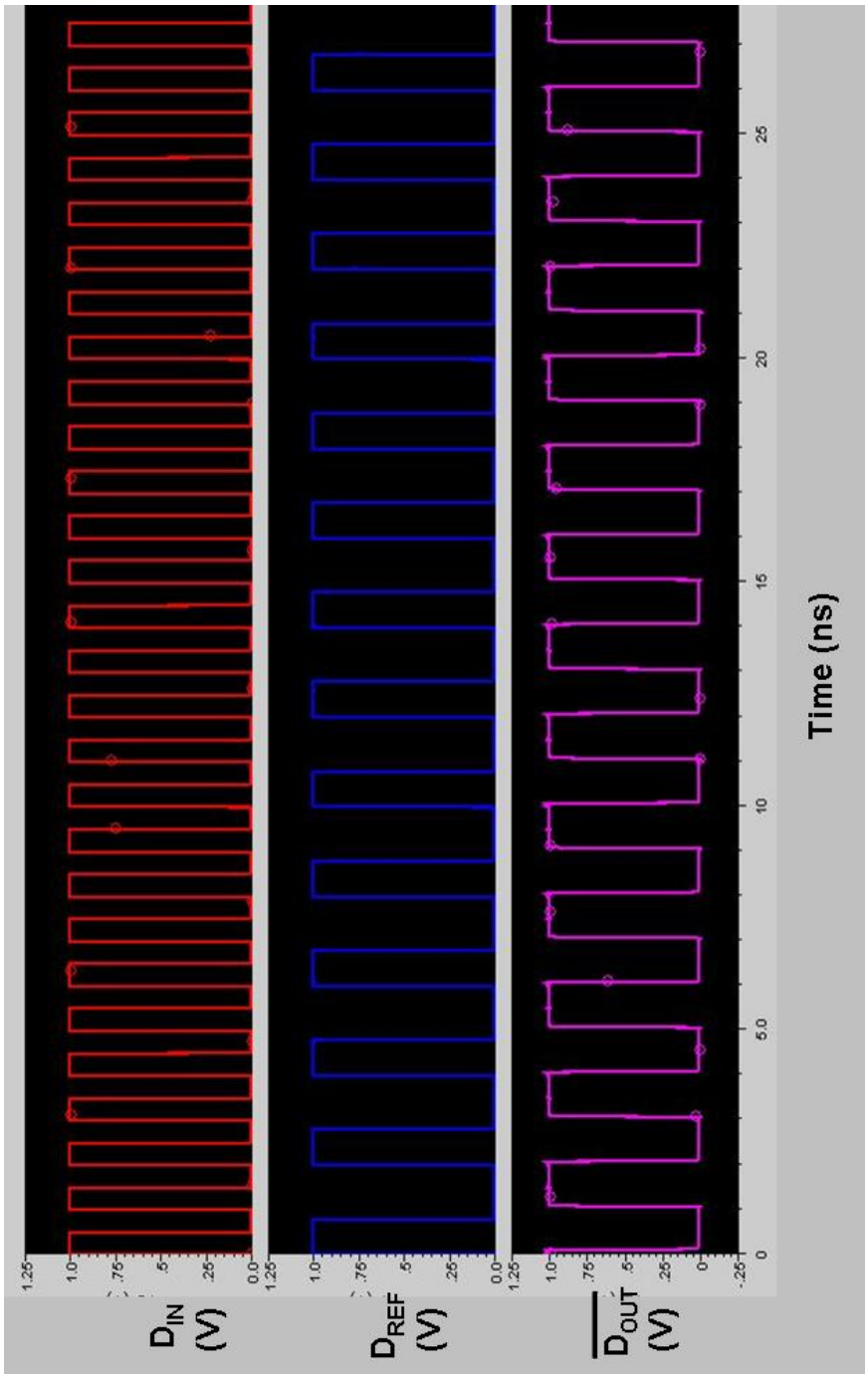


Figure 6.7: DCD Jitter Reduction by Rx Jitter Reducer

Table 6.3: Transmit Side PLL with Tx Jitter Reducer – Values of Poles and Zeros

#	Poles (Hz)	
	Real	Imaginary
1	-6.91146e+08	0.00000e+00
2	2.89281e+09	± 2.25773e+09
3	-4.33070e+09	0.00000e+00
4	-1.62474e+09	± 4.12059e+09
5	-6.51648e+09	0.00000e+00
#	Zeros (Hz)	
	Real	Imaginary
1	6.82379e+08	0.00000e+00
2	2.07588e+08	± 7.51593e+08
3	-9.60646e+08	± 1.15217e+09
4	-1.60208e+09	0.00000e+00
5	-6.17364e+09	0.00000e+00
6	**-9.10240e+09**	0.00000e+00

the case of the PLL with the jitter reducer. As a result a minute change in the magnitude and phase behavior is expected and the above conclusion is evident from Figures 6.8 and 6.9, which show the magnitude and phase response over a frequency range for the transmit side PLL with and without the jitter reducer. The gain is around $160dB$ in the case without the jitter reducer and it reduces it around $156dB$, which is negligible since gain is already huge and as far as the phase behavior is concerned there is a change of few degrees over the frequency range, which effect on the performance of the PLL is negligible. As a result, we have established clearly that the addition of the jitter reducer at the output of the PLL circuit at the transmit side does not affect its performance.

Tables 6.4 and 6.5 show the values of the poles and zeros of the PLL circuit at the receive side with and without the Rx jitter reducer. On comparing the two tables, we see that after adding the Rx jitter reducer there are three new poles and two new zeros created and these are high frequency poles and zeros over the $10GHz$ range. Therefore, in the frequency range of operation, which is around $1GHz$, there should not be any change in the magnitude and the phase behavior

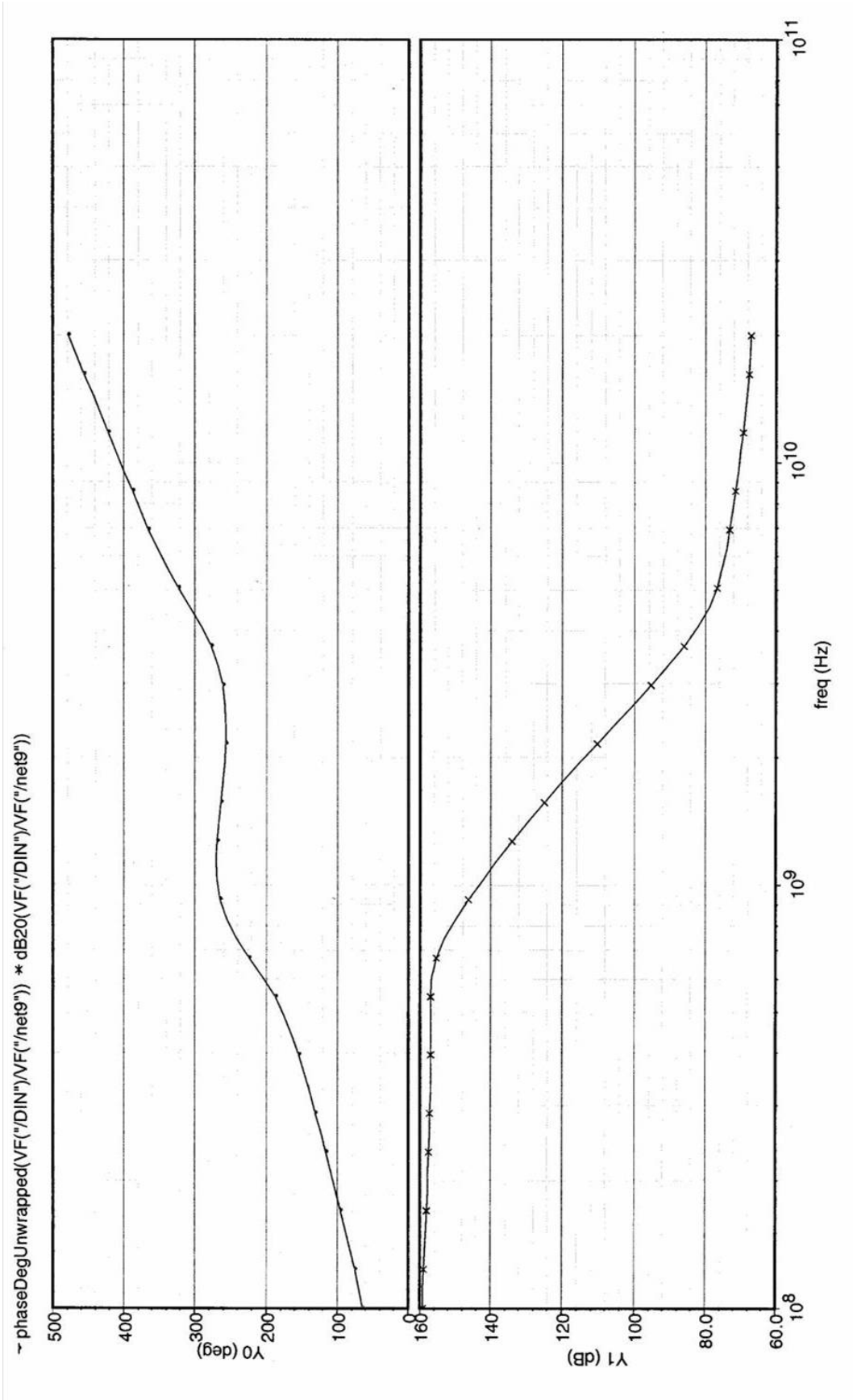


Figure 6.8: Magnitude and Phase Response of the Transmit Side PLL without the Tx Jitter Reducer

Figure 6.9: Magnitude and Phase Response of the Transmit Side PLL with the Tx Jitter Reducer

of the PLL circuit at the receive side. This conclusion is verified my obtaining the magnitude and the phase response of the PLL circuit with and without the Rx jitter reducer. From Figures 6.10 and 6.11, we see that there is no change in the magnitude and the phase behavior of the PLL circuit around $1GHz$ and, therefore, the effect of the the RX jitter reducer on the magnitude and the phase behavior of the PLL circuit is negligible. The key observation from the pole-zero analysis is that both the transmit and the receive side jitter reducers have the same effect as a normal digital load on their respective PLL circuits.

Table 6.4: Receive Side PLL without Rx Jitter Reducer – Values of Poles and Zeros

#	Poles (Hz)	
	Real	Imaginary
1	-6.91146e+08	0.00000e+00
2	2.89281e+09	±2.25773e+09
3	-4.33070e+09	0.00000e+00
4	-1.62474e+09	±4.12059e+09
5	-6.51648e+09	0.00000e+00
6	-1.02391e+10	0.00000e+00
7	-1.77032e+10	0.00000e+00
8	-7.73675e+10	0.00000e+00
9	-8.20301e+10	0.00000e+00
#	Zeros (Hz)	
	Real	Imaginary
1	1.37060e+08	0.00000e+00
2	6.82383e+08	0.00000e+00
3	2.07590e+08	±7.51594e+08
4	-9.60646e+08	±1.15217e+09
5	-1.60208e+09	0.00000e+00
6	-6.17364e+09	0.00000e+00
7	9.17732e+09	0.00000e+00
8	-8.00192e+10	0.00000e+00
9	-1.21810e+11	0.00000e+00
10	1.26214e+11	0.00000e+00

Table 6.5: Receive Side PLL with Rx Jitter Reducer – Values of Poles and Zeros

#	Poles (Hz)	
	Real	Imaginary
1	-6.91146e+08	0.00000e+00
2	2.89281e+09	±2.25773e+09
3	-4.33070e+09	0.00000e+00
4	-1.62474e+09	±4.12059e+09
5	-6.51647e+09	0.00000e+00
6	-1.02351e+10	0.00000e+00
7	**-1.42503e+10**	0.00000e+00
8	**-1.49923e+10**	0.00000e+00
9	-1.76849e+10	0.00000e+00
10	**-5.24770e+10**	0.00000e+00
11	-7.73645e+10	0.00000e+00
12	-8.20288e+10	0.00000e+00
#	Zeros (Hz)	
	Real	Imaginary
1	1.37058e+08	0.00000e+00
2	6.82384e+08	0.00000e+00
3	2.07590e+08	±7.51595e+08
4	-9.60646e+08	±1.15217e+09
5	-1.60208e+09	0.00000e+00
6	-6.17364e+09	0.00000e+00
7	9.17732e+09	0.00000e+00
8	**-1.42745e+10**	0.00000e+00
9	**-1.52220e+10**	0.00000e+00
10	-8.00192e+10	0.00000e+00
11	-1.21810e+11	0.00000e+00
12	1.26214e+11	0.00000e+00

Figure 6.10: Magnitude and Phase Response of the Receive Side PLL without the Rx Jitter Reducer

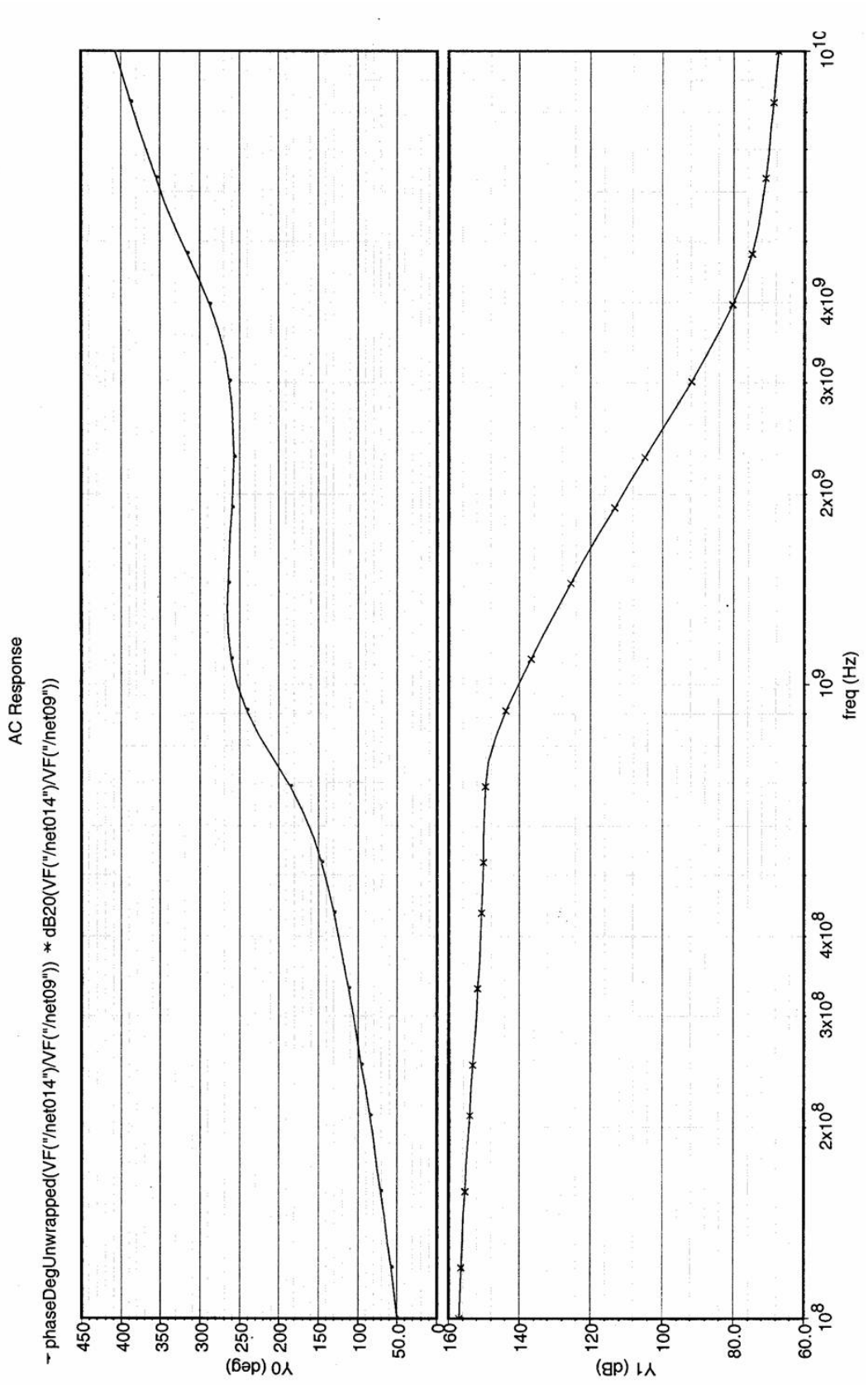


Figure 6.11: Magnitude and Phase Response of the Receive Side PLL with the Rx Jitter Reducer

6.7 Phase Noise Analysis

In this section, the phase noise present at the output of the PLLs and the jitter reducers both at the transmit and the receive side is estimated. The phase noise is calculated at frequencies offset from the center frequency of the oscillator in the PLL circuit. First, the output noise is calculated at the offset frequencies, then the phase noise is calculated using the equation [29] below:

$$PN(f) = 10 \times \log\left(\frac{N_{out}(f)^2}{V^2/2}\right) \tag{6.5}$$

where $N_{out}(f)$ is the total output noise present at the output of the PLL and the jitter reducer circuits and V is the magnitude of the center frequency. The phase noise equation shown above computes the ratio of the noise power at an offset frequency to the power in the center frequency term. The output noise computed, mainly consists of the thermal and flicker noise values of the transistors in the respective circuits. As again for the range of the offset frequencies, the noise contributed by the various circuit elements is determined and the total noise $N_{out}(f)$ is computed for the range of offset frequencies. For our experiment, the offset frequencies range from $1KHz$ to $100MHz$. The *pnoise* analysis feature of the CADENCE^{TM} tool is used to compute the phase noise. Figures 6.12, 6.13, 6.14 and 6.15 show the phase noise at the outputs of the PLL and the jitter reduction circuits at the transmit and the receive side, respectively.

In the case of the transmit side, the phase noise at the output of the PLL circuit remains at $-103dB$ for the offset frequencies from $1KHz$ to $1MHz$ and then starts reducing further and reaching a value of $-126dB$ for the offset frequency of $100MHz$. The phase noise at the output of the transmit side jitter reducer drops to $-124dB$ for the offset frequencies from $1KHz$ to $10MHz$ and further reduces to $-132dB$ for the offset frequency of $100MHz$. Similarly, in the case of the receive side, the phase noise at the output of the PLL circuit is the same as in the case of the transmit side, whereas the phase noise at the output of the receive side jitter reducer reduces considerably and ranges between $-146.135dB$ and $-146.18dB$ for the offset frequencies from $1KHz$ to $100MHz$. From the

above discussion, it is evident that the jitter reducer circuits at the transmit and the receive side reduce the phase noise.

6.8 Test Architecture for SERDES

A test architecture is shown in Figure 6.16, where the test hardware for testing the jitter reducers is integrated with the current *bit-error rate test* (BERT) scheme for the SERDES [31]. The pins TCK_{IN} and $TDATA_{IN}$ are used to supply the test clock and data signals, respectively, and the test response can be tapped using the test pins TCK_{OUT} and $TDATA_{OUT}$, respectively. To reduce the pin count, some of the functional input and output pins can be used in the test mode to supply the test inputs and receive the test responses accordingly. In our proposed testing scheme there are three phases. First, the jitter reducers are tested for their catastrophic transistor stuck-open and stuck-closed faults, then the receiver jitter tolerance test is performed and finally, either a deterministic or probabilistic BER test is performed.

6.8.1 Testing the Tx and Rx Jitter Reducers

The Tx and Rx jitter reducers are tested for analog catastrophic faults. Here, these are transistor stuck-open (closed) faults in each of the 14 (Tx) and 20 (Rx) transistors, respectively. Jitter reducers are driven with input signals containing periodic jitter. Some of these faults affect the propagation delay of the signals inside the jitter reducers, thereby affecting the jitter reduction. In the white paper on jitter [24], it is given that for an industry BER cut-off of 10^{-12}, the following condition should be met:

$$7\sigma = \frac{T_B}{2} \tag{6.6}$$

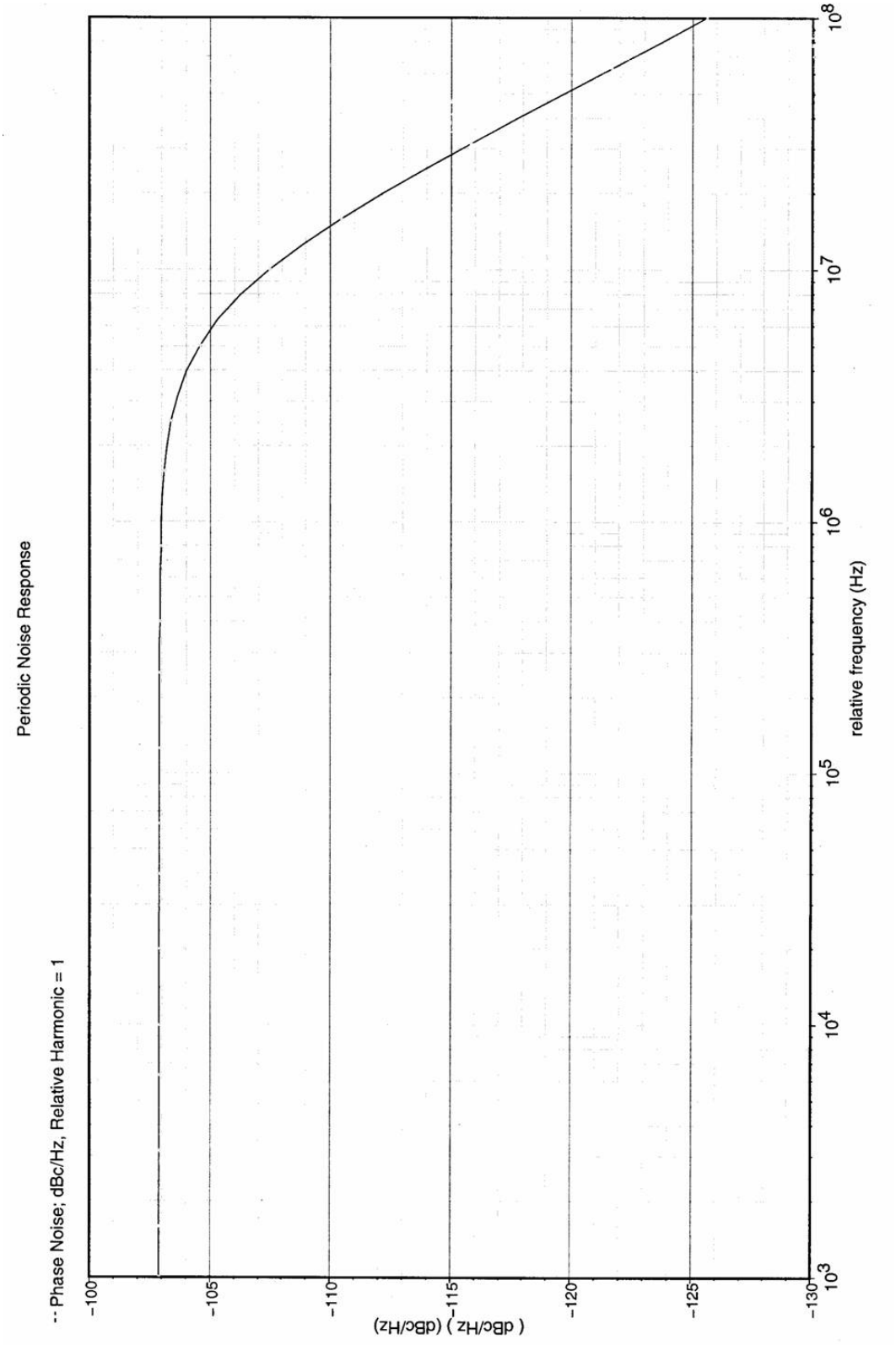


Figure 6.12: Phase Noise at the Output of the PLL Circuit at the Transmit Side

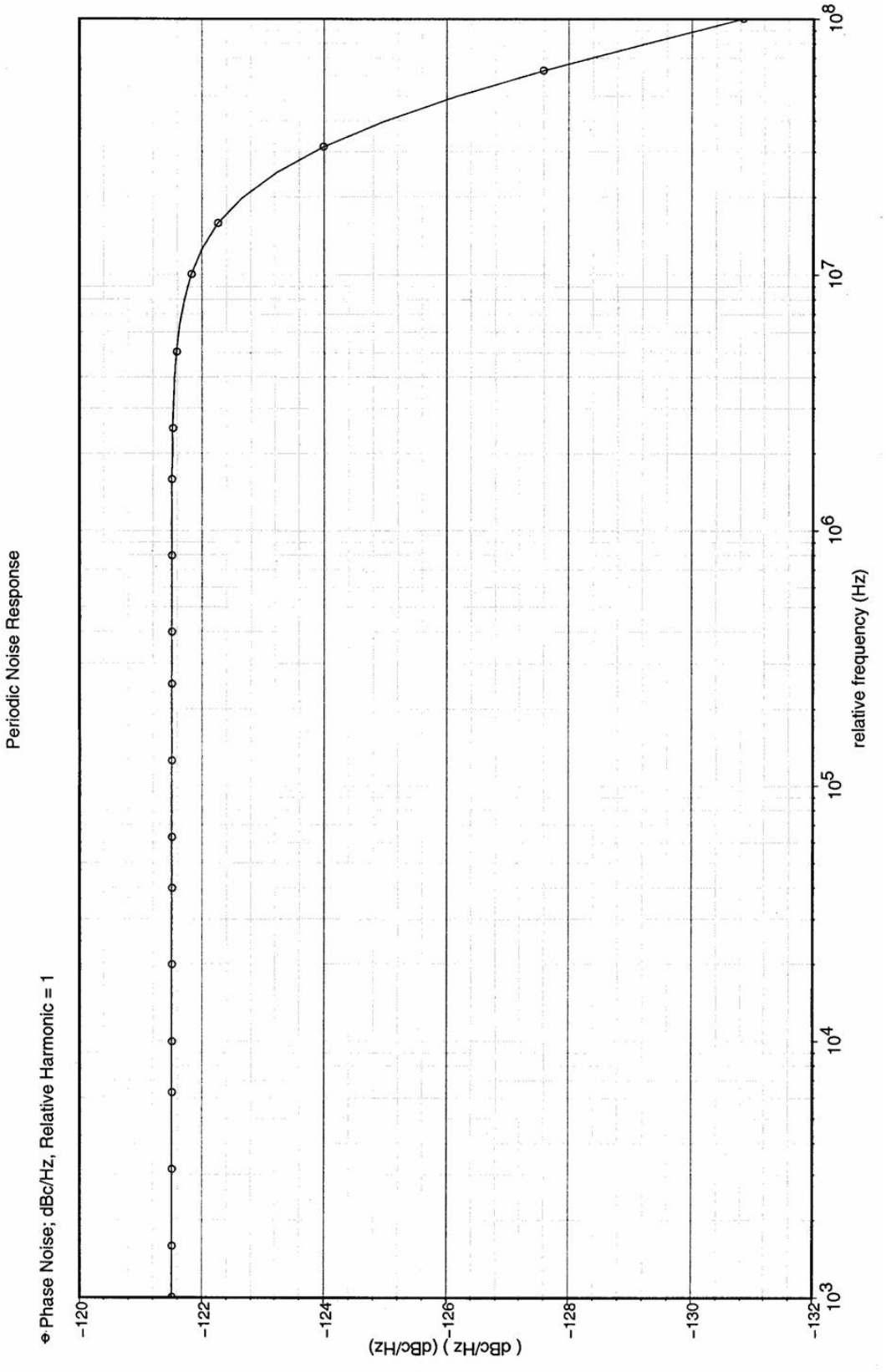


Figure 6.13: Phase Noise at the Output of the Jitter Reduction Circuit at the Transmit Side

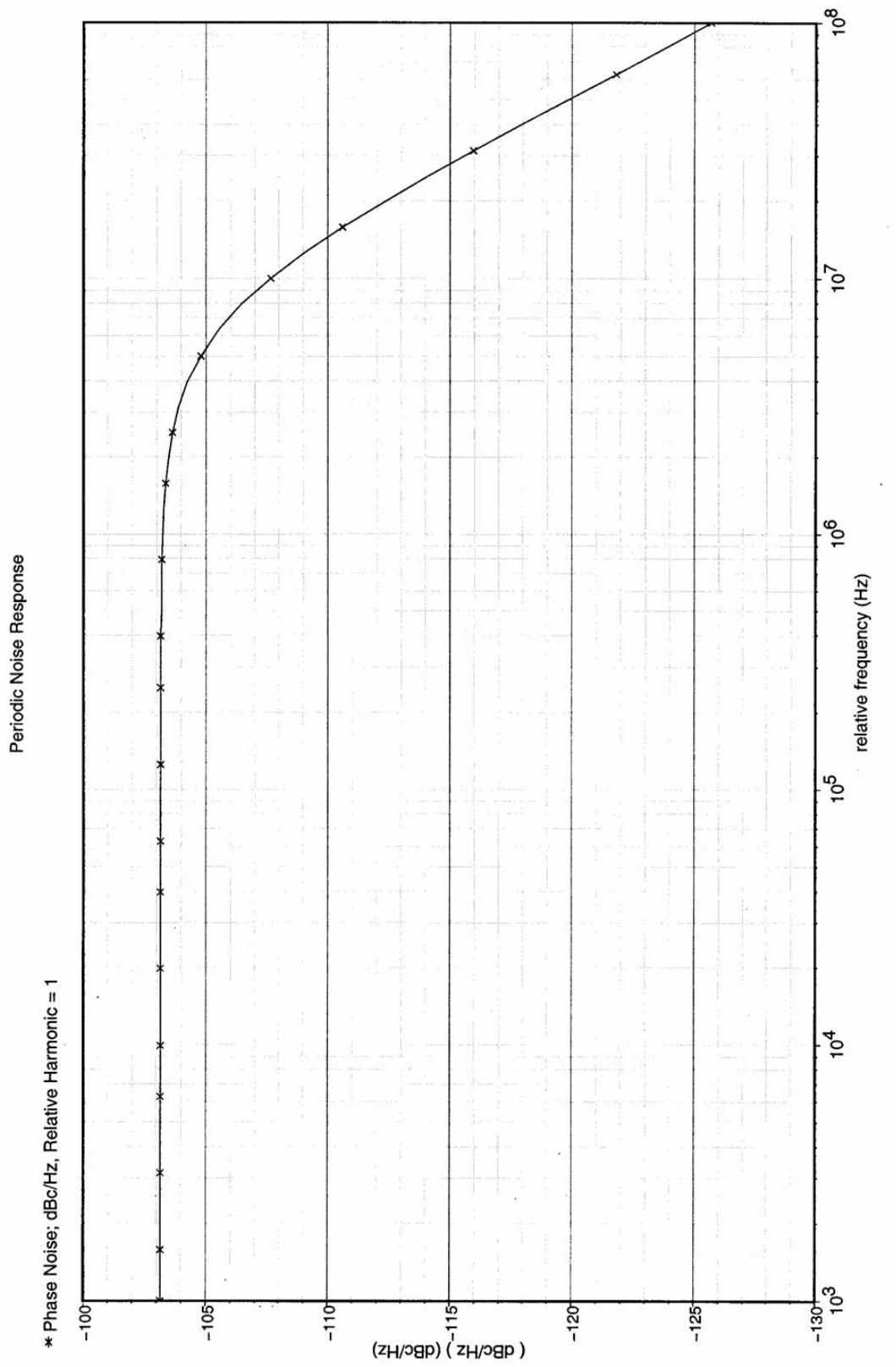


Figure 6.14: Phase Noise at the Output of the PLL Circuit at the Receive Side

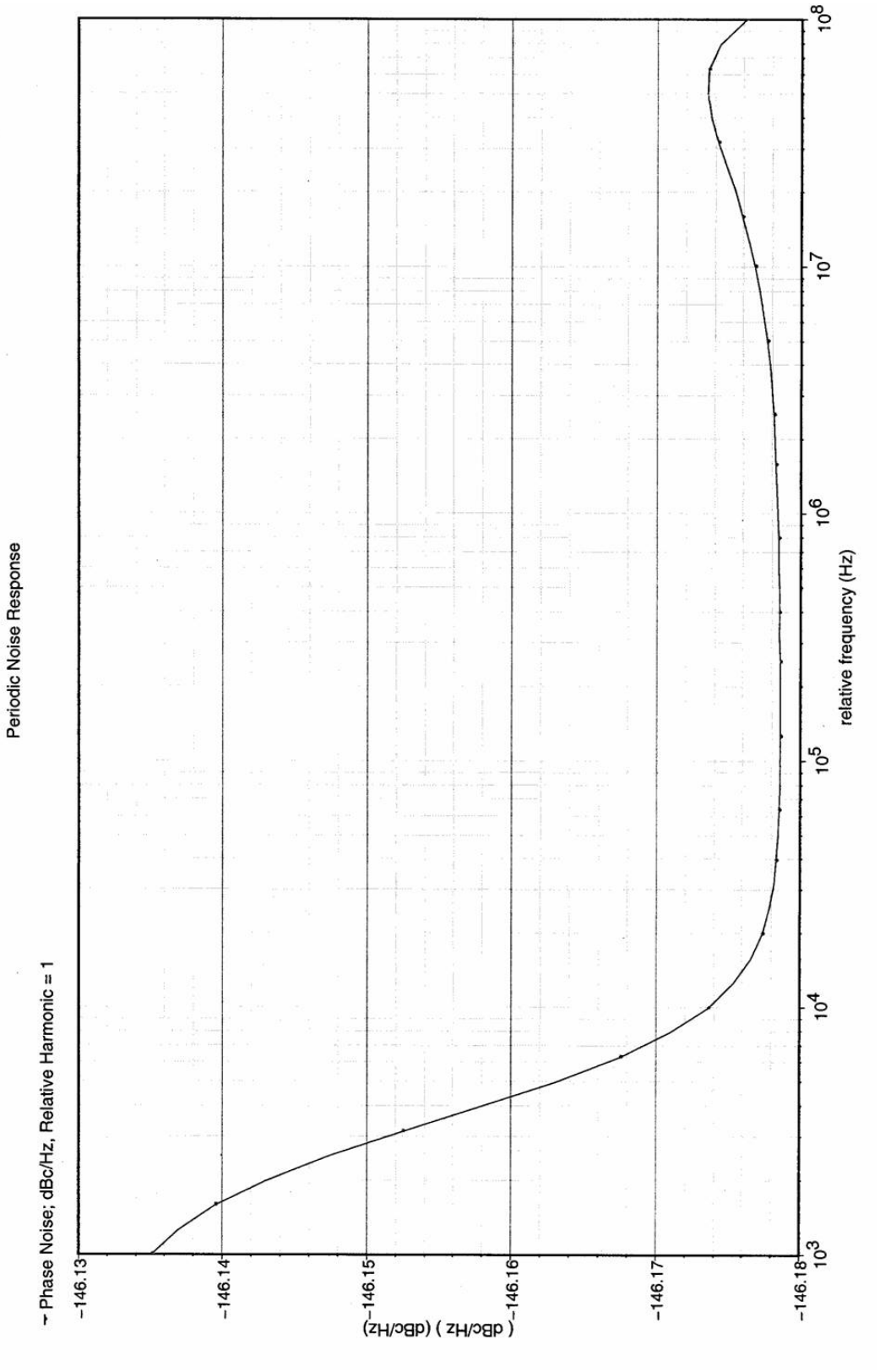


Figure 6.15: Phase Noise at the Output of the Jitter Reduction Circuit at the Receive Side

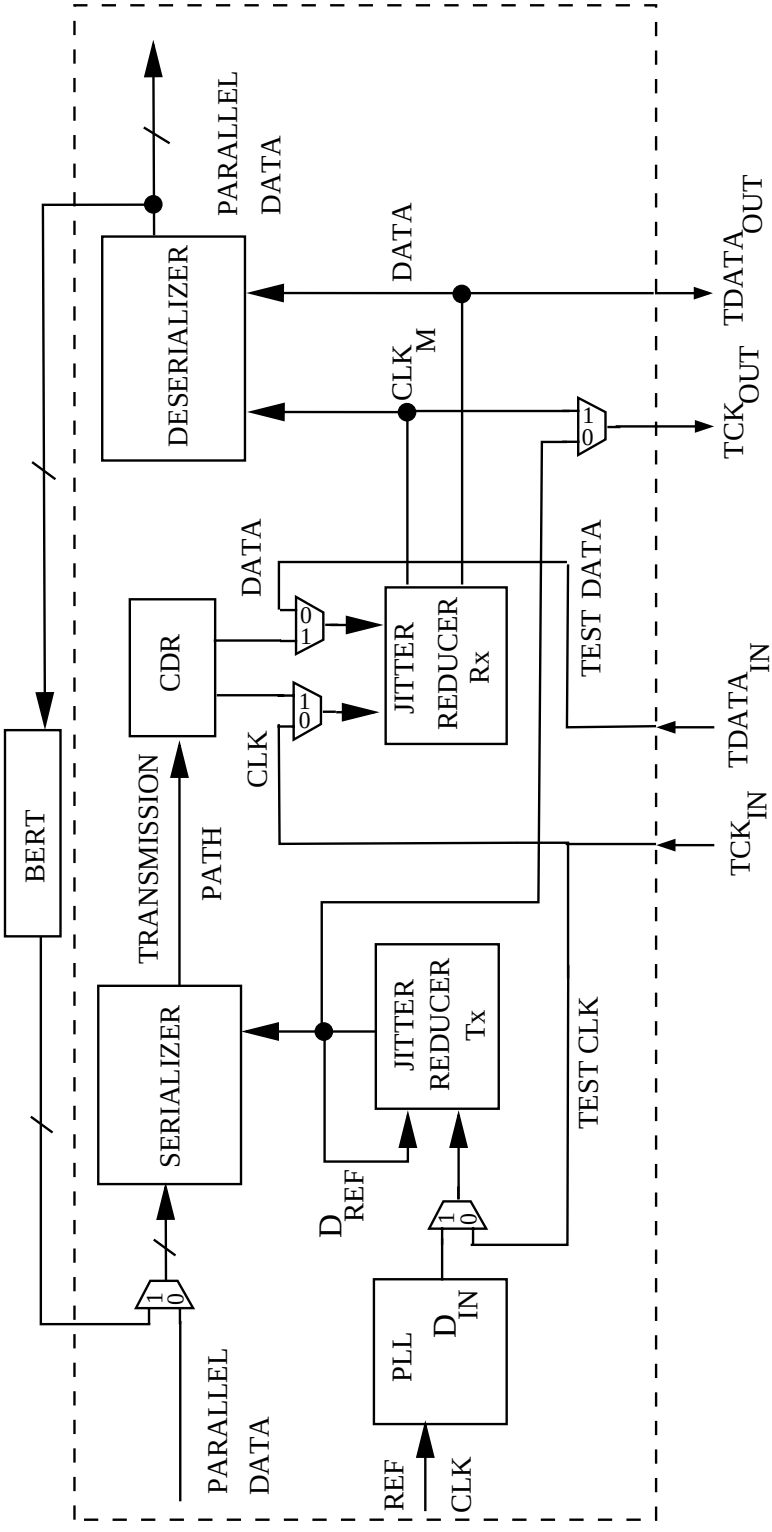


Figure 6.16: Testing the SERDES

where T_B is the unit bit period. In our case the unit bit period is $500ps$. Therefore, the ideal σ of the output jitter distribution should be:

$$\sigma = \frac{500ps}{2 \times 7} \tag{6.7}$$

$$\sigma = 35.71ps \tag{6.8}$$

For the purpose of testing the jitter reducer circuits for transistor faults, we have taken an input RMS jitter of $60ps$ for Tx and Rx jitter reducers, respectively, and the corresponding output RMS jitter for this input is less than $35.71ps$ for both jitter reducers. So, the following condition is used for transistor fault testing:

$$\text{Output RMS Jitter}_{\text{transistor fault}} > 35.71ps$$

The input pattern stimulus is a single-clock period digital signal of frequency $1GHz$, but the fault is excited and propagated using analog effects. If part of the jitter reduction circuit delay is changed by the fault, it affects the output RMS jitter or it may even cause a catastrophic failure, where the output is permanently stuck at V_{DD} or GND. So, a jitter measurement using an external ATE during this test will observe the fault effect.

Table 6.6: Transistor Fault Testing for the Tx and Rx Jitter Reducers

Jitter Reducer	Stuck-open	Stuck-closed
Tx	3,4,5,6,7,8 9,10,12,13	1,2,3,4,5,7,9,10 11,12,13,14
Rx	16,17,18,19,20,21 24,28,29,30,34	16,18,21,28

Table 6.6 shows the list of transistors in Figures 4.16 and 5.3 for which the analog catastrophic stuck-open (closed) faults are detected. Stuck-open (closed) faults for the rest of the transistors could not be detected, because the jitter performance is not affected by these faults enough to fall below the jitter test threshold, but these faults can be detected by setting a cut-off less than $35.71ps$, i.e., around $22ps$. The transistor faults that were detected are not redundant

because when they are either stuck-open or shorted, they affect the output of the jitter reducer hardware, but only modestly.

In our case we use the jitter analysis testing, which is, in retrospect, delay fault testing, where the delay in the propagation of signals introduces timing jitter and when this timing jitter results in a output RMS timing jitter greater than a certain cut-off, we capture the fault effect and this is the essence of our transistor fault testing.

6.8.2 Receiver Jitter Tolerance Test

The receiver jitter tolerance test is the test where the receiver jitter transfer function, a plot of output RMS jitter against the input RMS jitter, is determined. As explained in Section 5.2 this can be obtained by testing the receive side jitter reducer, which is the final block in our proposed SERDES architecture before the clock and data signals are given to the de-serializer. From the proposed SERDES test architecture shown in Figure 6.16 the jitter tolerance test can be conducted in two ways. In the first case we can make the multiplexers accept the clock and data input from the test pins TCK_{IN} and $TDATA_{IN}$, respectively, instead of from the CDR circuit, so that we can introduce different types of jitter in the input signals. In the second case we can make the transmit side jitter reducer accept input from the TCK_{IN} test pin and we can drive the CDR circuit with the data signal from the transmit side. In both cases the output responses of the receive jitter reducer can be tapped and taken to the output test pins TCK_{out} and $TDATA_{OUT}$, respectively, and from there to the external ATE to determine the receiver jitter tolerance.

6.8.3 Probabilistic BER Test

The BER can be computed either deterministically or probabilistically. For the deterministic test the traditional BERT can be used, which has an on-chip pattern generator and a response analyzer. For the probabilistic approach the output

signals from the receive side jitter reducer can be taken to the external ATE, where the output jitter distribution can be computed. Once the standard deviation of the jitter distribution is known, the BER can be computed.

6.9 Summary

In this chapter various circuit design issues such as the process variations, optimal transistor widths, and parasitics and their effect on the performance of the transmit side and the receive side jitter reducers are analyzed. It is demonstrated clearly that the performance of the proposed jitter reducer circuits is not affected by these circuit design issues. Finally, an effective testing scheme is proposed, where apart from showing how to test the jitter reducers, it is also shown how to conduct the receiver jitter tolerance and the BER test using the proposed test scheme.

Chapter 7
Conclusion and Future Work

7.1 Conclusion

The essence of the new jitter reduction methodology proposed in this work is to improve the yield of the high-speed SERDES circuits. The only way the yield can be improved is by improving the BER performance of the SERDES, thereby reducing the likelihood that a bit-error will occur when the total number of bits transmitted is less than or equal to 10^{12}, which is the industry standard. The goal is achieved primarily by proposing and implementing jitter reducers for both the transmit and the receive side. The hardware proposed is designed efficiently so that the area overhead is not increased considerably.

For the transmit side, the proposed jitter reducer primarily reduced the jitter present in the clock signal generated by the phase-locked loop. The PLL clock signal has various types of jitters, such as the periodic and the random jitter. The efficiency of the proposed hardware is analyzed in terms of its output RMS jitter performance. The jitter reducer is tested for various types of jitter conditions and the corresponding output RMS jitter values are determined. An industry standard cut-off of $35.71ps$ was chosen and for each of the input jitter conditions, the input RMS jitter value for which this cut-off was crossed is determined. From the results produced it can be inferred that the transmit side jitter reducer effectively reduces the input jitter. The input RMS jitter is reduced, on average, by 62.44%. To compare the peformance of the jitter reducer with prior art, the peak-to-peak jitter reduced by the jitter reducer is compared with the adaptive PLL peak-to-peak jitter reduction technique proposed by Xia *et al.* [39]. The peak-to-peak

jitter is reduced from 29.5ps to 17.8ps, which is 40.35% reduction as compared to 40.06% reduction obtained by the adaptive PLL technique. Our technique was implemented on a 1GHz clock signal in the 70nm Berkeley Predictive model process, whereas the prior work was implemented on a 500MHz clock signal in an IBM 180nm process.

In the case of the receive side of the SERDES circuit, the jitter reduction technique proposed for the transmit side was used with certain modifications. For the jitter reduction there are two essential steps, first, the jitter error estimation and next the jitter reduction. For the transmit side, the input jittered clock signal and the looped-back signal were used for determining error and for the receive side, the recovered clock and the incoming serial data are used. As in the case of the transmit side, the receive side jitter reducer is also tested for various input jitter conditions in both the clock and the data signal. The receive side jitter reducer reduces the input RMS jitter, on average, by 35.88%.

Next, the ultimate advantage in having these jitter reducers is determined by analyzing the performance of the SERDES circuit under three conditions. In each of these conditions, the output RMS jitter and the probabilistic BER are computed. As clearly demonstrated in the results, having both the transmit and the receive side jitter reducers not only gives a better output RMS jitter performance but also improves the BER from 8.3×10^{-2} to 6.44×10^{-20}, which is a order of 10 improvement.

The major issue when designing circuits is how their performance is affected under process variations. The experiments done on the transmit and the receive side jitter reducers under process variations show their effects can be controlled by properly sizing the transistors. The effect of parasitics on the jitter performance is also analyzed and the results clearly show that their effect can be minimized with proper circuit design.

Finally, the proposed test scheme integrating the testing of the jitter reducers with the current testing scheme of the SERDES circuit shows how to test the jitter reducers for their stuck-at faults and also how to perform the receiver jitter

tolerance and BER test. The test scheme has three phases, where first, the stuck-at fault test is performed, followed by the jitter tolerance test and the BER test.

The merits of this work are as follows:

- The yield can be improved by simply adding these jitter reducers at the output of the PLL circuit in the transmit side and at the output of the CDR circuit in the receive side.
- The internal circuity of the PLL and CDR circuits need not be modified to reduce jitter.
- The proposed hardware is made of only 14 and 20 transistors for the transmit and the receive side jitter reducers, respectively.

The limitation of this work is that jitter reducers introduce extra delay in the signal, which is not present originally. The above-mentioned limitation can be overcome by adding extra buffers that add extra delay so that these signals are synchronized with the original signals, thereby removing the phase skews. The problems due to process variations in the buffer circuits are relatively easier to control, since in most cases they are made of simple inverter gates, whose variation in delays due to process variations can be controlled by proper transistor sizing. As explained in Section 6.2, the width W of the transistors in the buffer circuits can be changed as long as the constraints given in Equations 6.1, 6.2, 6.3 and 6.4 are met, in order to reduce the problems due to process variations.

7.1.1 Applications

The proposed jitter reduction circuits can be used to reduce timing jitter in the clock signals used in microprocessors. A particular scenario where these circuits can be used are the clock trees, where the jitter reducers can be used to reduce timing jitter present in the clock signals from different branches. Also to avoid synchronization problems, the timing jitter between the clock signal of a flip-flop

and the incoming data signal can be reduced using the technique proposed in Chapter 5.

Similarly, in the case of wireless transceivers, where the problem is simiar to the one in SERDES circuits, the jitter reducer circuits can be used to reduce the timing jitter in the clock signals generated by the PLL and in the case of mircowave *radio frequency* (RF) circuits, the timing jitter in the clock signals generated by the *local oscillator* (LO) circuits can be reduced using our jitter reduction circuits. The jitter reducers can be specifically used to reduce timing jitter in the clock signals used in the baseband processors, so that an incorrect bit is not latched by the flip-flops, and hence, reducing the bit-errors.

7.2 Future Work

In this section, we propose to modify the jitter reduction hardware, so that it can be used in jitter testing.

7.2.1 Jitter Testing

The current state of the art test methodology for testing SERDES circuits is a BERT test scheme. The main problem with the BERT scheme is that 10^{12} bits have to be transmitted using the on-chip pattern generator to the response analyzer to determine if bit errors have occurred, which is a time consuming process. The alternate approach is to compute the BER probabilistically from the output RMS jitter distribution. The accuracy of the BER computed depends on how accurately the jitter distribution is determined. The only way to estimate the jitter distribution is to determine the timing jitter in the output signal by taking it to the external ATE. The quality of the signal is degraded as it is transmitted through the interconnect wires, thereby distorting the signal by at least 5%. The timing jitter distribution estimated using such a distorted signal will therefore be inaccurate.

To overcome this problem, new hardware can be added to the receive side jitter reduction circuit. The new hardware will take the output signal and the ideal reference signal from the ATE to compute the timing jitter on-chip and transmit the measured jitter to the external ATE. The transmitted jitter information still suffers from the distortion effects of the interconnect wires, which can be eliminated to a certain extent by amplifying the jitter information using a high gain common source CMOS amplifier. Once the signal is received at the ATE, we can determine the original jitter information by converting it back to its original magnitude, and for this we have to first estimate the amplifier gain.

The accuracy of the above technique depends on the following:

- How much of the signal distortion is removed by the amplification.
- How accurately the original jitter information can be recovered after it has be received in the ATE.

The amplifier gain that is required to offset the distortion effects can be estimated by performing a *signal to noise and distortion ratio* (SNDR) analysis. Once the quality of the jitter information satisfies the constraints, we can use it to estimate the BER of the SERDES probabilistically.

Appendix A
User's Guide

A.1 Circuit Design

The SERDES circuit designed for this work consists of the following subcircuits:

- Five stage current controlled ring oscillator.
- Type 1 phase-locked loop with XOR phase detector, current mirror and oscillator.
- Transmit side jitter reduction circuit.
- Parallel to serial shift register (Serializer).
- Clock and data recovery circuit.
- Receive side jitter reduction circuit.
- Serial to parallel shift register (Deserializer).

All the circuits are designed in the $70nm$ Berkeley Predictive process and are in the design library $BP70nm$. The path to the library is given below:

$$/caip/u21/hariven/BP70nm \tag{A.1}$$

A.2 Circuit Testing

The circuit testing involves estimating the timing jitter, and for this we need to capture the time instants at which the the signal rises and falls. The signal

transition were captured using the verilog modules $cross1$ and $cross2$. The verilog routines and their circuit models are in the library $bmslib$. The path to the library is given below:

$$/caip/u21/hariven/my_libs/bmslib \tag{A.2}$$

The MATLAB routines for generating the various input jitter types and also to estimate the output jitter distribution are in the respective simulation directories of the subcircuits. For example, the routines for testing the transmit side jitter reduction circuit are in the directory give below:

$$/caip/u21/hariven/cadence/simulation/JRCT/spectre/schematic/netlist \tag{A.3}$$

The above mentioned directory contains the SPECTRE netlist of the circuit along with the MATLAB routines to generate jitter and also to estimate the output RMS jitter.

References

[1] E. H. Suckow. How to Apply SERDES Performance to your Design, http://www.us.design-reuse.com/articles/article4761.html, Jan. 2003.

[2] Wikipedia. 8B/10B Encoding, http://en.wikipedia.org/wiki/8B10B, Apr. 2007.

[3] Wikipedia. Timing Skew, http://en.wikipedia.org/wiki/Timing_skew, Apr. 2007.

[4] Actel Corporation. Simultaneous Switching Noise and Signal Integrity. Application Note, Mountain View, CA, June 2006.

[5] Agilent Technologies. Measuring Jitter in Digital Systems, Application note 1448-1, June 2003.

[6] A. Athavale and C. Christensen. *High-Speed Serial I/O Made Simple: A Designer's Guide with FPGA Applications*. Xilinx Connectivity Solutions, San Jose, CA, 2005.

[7] W. Dalal and D. Rosenthal. Measuring Jitter of High Speed Data Channels Using Undersampling Techniques. In *Proc. of the Int'l. Test Conf.*, pages 814–818, Oct. 1998.

[8] A. Dou and J. A. Abraham. On-Chip Jitter Measurement Using a Dual-Channel Undersampling Time Digitizer. In *Proc. of the Int'l. Mixed Signal Test Workshop*, pages 206–211, June 2006.

[9] Genesys Logic America, Inc. Multi-Gigabit SerDes: The Cornerstone of High Speed Serial Interconnects. Multi-Gigabit SerDes White Paper, Sept. 2003.

[10] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer. *Analysis and Design of Analog Integrated Circuits*. John Wiley & Sons, Inc., USA, 2000.

[11] A. Hajimiri and T. Lee. A General Theory of Phase Noise in Electrical Oscillators. *IEEE Journal of Solid-State Circuits*, 33(2):179–194, Feb. 1998.

[12] G. Hetherington and R. Simpson. Circular BIST Testing the Digital Logic within a High Speed Serdes. In *Proc. of the Int'l. Test Conf.*, pages 1221–1228, Oct. 2003.

[13] P. Heydari and M. Pedram. Analysis of Jitter due to Power-Supply Noise in Phase-Locked Loops. In *Proc. of the Custom Integrated Circuits Conf.*, pages 443–446, May 2000.

[14] D. Hong, C. K. Ong, and K. T. Cheng. BER Estimation for Serial Links Based on Jitter Spectrum and Clock and Recovery Characteristics. In *Proc. of the Int'l. Test Conf.*, pages 1138–1147, Oct. 2004.

[15] C. Hur, Y. Choi, H. Choi, and T. Kwon. A Low Jitter Phase-Locked Loop Based on a New Adaptive Bandwidth Controller. In *Proc. of the IEEE Asia-Pacific Conf. on Circuits and Systems*, pages 421 – 424, Dec. 2004.

[16] D. A. Johns and K. Martin. *Analog Integrated Circuit Design*. John Wiley & Sons, Inc., New York, USA, 1997.

[17] A. Krasniewski and S.Pilarski. Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits. *IEEE Trans. on Computer-Aided Design*, 8(1):46–55, Jan. 1989.

[18] A. Kuo, T. Farahmand, N. Ou, S. Tabatabaei, and A. Ivanaov. Jitter Models and Measurement Methods for High-Speed Serial Interconnects. In *Proc. of the Int'l. Test Conf.*, pages 1295–1302, Oct. 2004.

[19] W. K. C. Lam and R. K. Brayton. *Timed Boolean Functions: A Unified Formalism for Exact Timing Analysis*. Kluwer Academic Publishers, Norwell, Mass., 1994.

[20] B. Laquai and Y. Cai. Testing Gigabit Multilane SerDes Interfaces with Passive Jitter Injection Filters. In *Proc. of the Int'l. Test Conf.*, pages 297–304, Oct. 2001.

[21] Lattice Semiconductor Corporation. Serdes Jitter. Technical Note TN1084, Hillsboro, OR, May 2006.

[22] M. P. Li and J. B. Wilstrup. On the Accuracy of Jitter Separation from Bit Error Function. In *Proc. of the Int'l. Test Conf.*, pages 710–716, Oct. 2002.

[23] M. Mansuri, A. Hadiashar, and C.-K. K. Yang. Methodology for On-Chip Adaptive Jitter Minimization in Phase-Locked Loops. *IEEE Trans. on Circuits and Systems*, 50(11):870–878, Nov. 2003.

[24] Maxim Dallas Semiconductor. An Introduction to Jitter in Communication Systems. Application Note 1916, Dallas, TX, March 2003.

[25] E. J. McCluskey, R. H. Wilcox, and W. C. Mann. Transients in Combinational Logic Circuits. In *Redundancy Techniques for Computing Systems*, pages 9–46, 1962.

[26] C. K. Ong, D. Hong, K. T. Cheng, and L. C. Wang. Jitter Spectral Extraction for Multi-Gigahertz Signal. In *Proc. of the Asia and South Pacific Design Automation Conference*, pages 298–303, Jan. 2004.

[27] C. K. Ong, D. Hong, K. T. Cheng, and L. C. Wang. Random Jitter Extraction Technique in a Multi-Gigahertz Signal. In *Proc. of the Design Automation and Test in Europe Conf.*, pages 286–291, Feb. 2004.

[28] N. Ou, T. Farahmand, A. Kuo, S. Tabatabaei, and A. Ivanov. Jitter Models for the Design and Test of Gbps-Speed Serial Interconnects. *IEEE Design & Test of Computers*, 21(4):302–313, July 2004.

[29] B. Razavi. A Study of Phase Noise in CMOS Oscillators. *IEEE Journal of Solid-State Circuits*, 31(3):331–343, Mar. 1996.

[30] B. Razavi. *Design of Analog CMOS Integrated Circuits*. Tata McGraw-Hill Publishing Company Limited, New Delhi, India, 2002.

[31] S. Sunter, A. Roy, and J. F. Cote. An Automated, Complete, Structural Test Solution for SERDES. In *Proc. of the Int'l. Test Conf.*, pages 95–104, Oct. 2004.

[32] K. Taylor, B. Nelson, A. Chong, H. Nguyen, H. Lin, M. Soma, H. Haggag, J. Huard, and J. Braatz. Experimental Results for High-Speed Jitter Measurement Technique. In *Proc. of the Int'l. Test Conf.*, pages 85–94, Oct. 2004.

[33] A. Telba, J. M. Noras, M. Abou El Ela, and B. AlMashary. Jitter Minimzation in Digital Transmission using Dual Phase Locked Loops. In *Proc. of the Int'l. Conf. on Microelectronics*, pages 270 – 273, Dec. 2005.

[34] A. Telba, J. M. Noras, M. Abou El Ela, and B. Al-Mashary. Simulation Technique for Noise and Timing Jitter in Phase Locked Loop. In *Proc. of the Int'l. Conf. on Microelectronics*, pages 501 – 504, Dec. 2004.

[35] W. E. Thain and J. A. Connelly Jr. Simulating Phase Noise in Phase-Locked Loops with a Circuit Simulator. In *Proc. of the Int'l. Symp. on Circuits and Systems*, volume 3, pages 1760–1763, Apr. 1995.

[36] TranSwitch Corporation. DART De-jitter PLL Design and Analysis, AN-531, DART Device TXC-02030-AN2, Shelton, CT, Apr. 2000.

[37] Troisi Design Limited. Jitter effects on Analog to Digital and Digital to Analog Converters. Application note, Westford, MA, 2000.

[38] S. D. Vamvakos, C. Werner, and B. Nikolic. Phase-Locked Loop Architecture for Adaptive Jitter Optimization. In *Proc. of the Int'l. Symp. on Circuits and Systems*, volume 4, pages 161 – 164, May 2004.

[39] T. Xia, S. Wyatt, and R. Ho. Automated Calibration of Phase Locked Loop with On-Chip Jitter Test. In *Proc. of the North Atlantic Test Workshop*, pages 140–147, May 2006.

[40] T. Yamaguchi, M. Soma, and M. Ishida. Extraction of Peak-to-Peak and RMS Sinusoidal Jitter Using an Analytic Signal Method. In *Proc. of the VLSI Test Symp.*, pages 394–402, May 2000.

[41] T. J. Yamaguchi, M. Soma, D. Halter, and R. Raina. A Method for Measuring the Cycle-to-Cycle Periodic Jitter of High-Frequency Clock Signals. In *Proc. of the VLSI Test Symp.*, pages 102–110, May 2001.

[42] T. J. Yamaguchi, M. Soma, and M. Ishida. A New Method for Testing Jitter Tolerance of SerDes Devices Using Sinusoidal Jitter. In *Proc. of the Int'l. Test Conf.*, pages 717–725, Oct. 2002.

[43] T. J. Yamaguchi, M. Soma, J. Nissen, D. Halter, R. Raina, and M. Ishida. Testing Clock Distribution Circuits Using an Analytic Signal Method. In *Proc. of the Int'l. Test Conf.*, pages 323–330, Oct. 2001.

Vita

Hari Vijay Venkatanarayanan

Academic Degrees

Ph.D in Computer Engineering, Rutgers University
Advisor: Prof. M. L. Bushnell
GPA: 3.87 / 4.00

M.S. in Computer Engineering, Rutgers University
Sept. 2001 - Oct. 2004
Advisor: Prof. M. L. Bushnell

B. E in Electronics and Communications Engineering, Madras University, Chennai, India. Sept. 1997 - May 2001
GPA: 3.60 / 4.00

Professional Experience

2007	Ph.D., Rutgers University, New Brunswick, New Jersey Majored in Computer Engineering
2005-07	Teaching Assistant, ECE Dept., Rutgers University, New Brunswick, New Jersey
2004	M.S., Rutgers University, New Brunswick, New Jersey Majored in Computer Engineering
2002-05	Research Assistant, CAIP Center, Rutgers University, New Brunswick, New Jersey
2001-02	Teaching Assistant, Math. Dept., Rutgers University, New Brunswick, New Jersey

Publications

1. H. V. Venkatanarayanan and M. L. Bushnell, "Improving the Bit-Error Rate Performance of the Serializer-Deserializer (SERDES) Circuits with

Jitter Reduction Hardware," accepted in the *Proc. of the North Atlantic Test Workshop*, May 2007

2. H. V. Venkatanarayanan and M. L. Bushnell, "A New Jitter Reduction Technique for High-Speed Serializer-Deserializer (SerDes) Circuits," submitted to the *Int'l. Test Conf.*, 2007

3. R. Sethuram, O. Khan and H. V. Venkatanarayanan and M. L. Bushnell,"Power Reduction using a Neural Network Branch Predictor," *Proc, of the Int'l. Conf. on VLSI Design*, pp. 161-168, Jan. 2007

4. H. V. Venkatanarayanan and M. L. Bushnell, "An Area Efficient Mixed-Signal Test Architecture for Systems-on-a-Chip," *Proc. of the Int'l. Conf. on VLSI Design*, pp. 161-168, Jan. 2006

5. "An Area Efficient Test Architecture for Mixed-Signal SoC," Masters thesis, Dept. of Elec. and Comp. Eng., Rutgers University, Oct 2004