2000

Low jitter design techniques for monolithic CMOS phase-locked and delay-locked systems

Lin Wu
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Low jitter design techniques for monolithic CMOS phase-locked and delay-locked systems

by

Lin Wu

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Electrical Engineering (Microelectronics)
Major Professor: William C. Black, Jr.

Iowa State University
Ames, Iowa
2000

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Lin Wu

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Committee Member

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

To my husband Huawen
and my parents Dake and Shulin.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Motivation

Phase-Locked Loops (PLLs) and Delay-Locked Loops (DLLs) find wide application in communication, digital I/O interface, disk drive electronics, RF engineering, etc. While the concept of phase locking has been in use for more than half a century, monolithic implementation of PLLs and DLLs to enable high speed and low cost system has become popular only during the past one or two decades. Today, there are still many interesting topics to deal with pertaining to monolithic implementations.

With the ever increasing demand for reducing the cost of integrated circuit, more and more circuits are now tending to be implemented with CMOS technology. In this situation, PLLs and DLLs are usually embedded in a large system, in which all circuit blocks share the same substrate. Noise coupling effects become the most critical factor that limits the performance of the PLL and DLL. How to design the PLL and DLL circuit to achieve low jitter performance in a rather noisy environment is the major task of designers in this area.

The motivation of this Ph.D. work is to explore system level as well as circuit level design techniques for low jitter PLL and DLL systems. Three prototype chips for different applications were implemented and tested to demonstrate the monolithic design technique for low jitter PLLs and DLLs.

The key contributions of this Ph.D. work are:

(1) An ultra low jitter precise delay multi-channel clock generator is implemented. As the first design to achieve real time on-chip continuous delay calibration, this chip also achieves the best single channel jitter performance thus far reported.

(2) A fibre channel Gigabit transceiver based on PLL are implemented with two test chips. Experimental results show that the jitter performance meets the required specification with low power dissipation.

(3) A rigorous analysis method for analyzing jitter properties of PLL and DLL systems is developed. Unlike traditional s-domain methods, this method is based on a z-domain model that can give a closed form statistical jitter expression in the time domain.

(4) A system level method to enhance the jitter performance of PLL is proposed. This method incorporates a cascade PLL-DLL structure to further filter the output jitter. The resulting system preserves the frequency synthesis capability of a PLL and can achieve the same level of low jitter performance as that of a DLL.

(5) A new nonlinear behavioral simulation environment for PLL and DLL systems is developed. This simulation is based on a novel model of the delay cell. This model not only simplifies the computation in previous work, but also enables more noise simulation functions that are not achieved in previous work.

(6) A fully differential current steering analog PLL for clock recovery to reduce noise coupling effects is proposed. As an alternative way to do clock recovery, this technique provides better noise immunity.

(7) A useful investigation of jitter measurement methods is conducted. This is a topic that is very important but has seldom been addressed in publications. From this discussion, the pros and cons of each method are made clear. This helps to choose the best test method for a specific application.

1.2 Dissertation Organization

In chapter 2, the underlying principles of phase locked loops and delay locked loops will be briefly introduced. The focus is on the important issues in monolithic PLL and DLL design. The goal is to provide a sufficient background knowledge for the material that follows.

Chapter 3 concentrates on the noise issues of PLL and DLL design and testing. A rigorous jitter analysis of the PLL and DLL will be conducted, which provides theoretical insight into the noise characteristic of both systems. Moreover, jitter measurement methods will be summarized, which is helpful to understand and to interpret the experimental results in the following chapters.

Based on the theoretical analysis in chapter 3, a new architecture to suppress jitter in frequency synthesis output is proposed in chapter 4. This new structure combines the advantage of both PLLs and DLLs and achieves better jitter performance.

In chapter 5, a new nonlinear behavioral system which models and simulates both PLL and DLL will be presented. It is not only used to validate the theoretical analysis, but also provides a tool for top-down optimization and bottom-up verification.

The application of PLL in fibre channel transceiver is explored in chapter 6. On the transmitter side, PLL is used as frequency multiplication clock generator and on the receiver side, it acts as clock recovery block. The testing results of the two prototype chips will be given.

In chapter 7, the delay calibrated low jitter PLL and DLL multi-phase clock generator will be given. The new architecture that enables the calibration will first be introduced. Delay calibration implementation is then discussed. Low jitter circuit design techniques will also be covered. Experimental results on the prototype chip will be presented at the end of this chapter that shows the effectiveness of the calibration scheme.

Finally the conclusion of this work will be presented in chapter 8.

CHAPTER 2 AN OVERVIEW OF PHASE-LOCKED LOOPS AND DELAY LOCKED LOOPS

PLLs have been playing an important role in electronic systems for more than half a century. With the rapid advancement of integrated-circuit (IC) technology in terms of speed and complexity, its application has been further expanded. There have been unceasing efforts on system level investigation and circuit implementation of PLLs for decades. A good introduction to phase locked loops can be found in the literature. As representative works, Gardner [1], Wolaver [2], Best [3] and Egan [4] et. al. are highly recommended.

The history of the DLL is relatively young compared with that of the PLL. Since it is similar to the phase locked loop in many ways and it is simpler, there is not much literature on the system level analysis of such a system. However, this does not prevent its wide application in modern monolithic systems. A rather specific introduction to DLLs can be found in Baker [5].

The goal of this chapter is to provide a brief introduction to phase locked and delay locked loops: their basic principles, their typical applications and their building blocks. Since the focus of this dissertation is the monolithic implementation of both systems, the discussion will emphasize the trend in this area.

2.1 Phase Locked Loop Fundamentals

2.1.1 Operating principles of PLL

A phase-locked loop is basically a circuit synchronizing the output signal from an oscillator with a reference clock or input signal in frequency as well as in phase. In the synchronized state (or often called *locked* state), the phase error between the oscillator's output signal and the input signal is zero, or remains constant.

The basic structure of a PLL is shown in Figure 2.1. It is usually composed of three blocks. Though many variations exist due to different applications and different implementation details, the functionality

of these blocks are similar. They are:

1. A voltage-controlled oscillator (VCO),
2. A phase detector (PD),
3. A loop filter (LF).

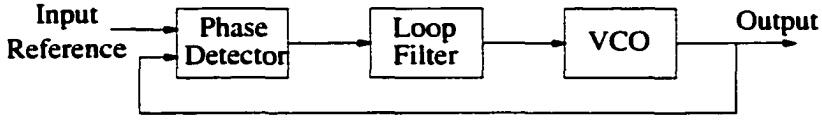


Figure 2.1 Basic PLL block diagram

The phase detector compares the phase of the input signal and the VCO signal to generate an output proportional to the phase difference. The output from the PD is converted into a control voltage by the loop filter. According to the control voltage, the VCO produces an output with corresponding frequency. Since phase is the integral of frequency over time, controlling the voltage over time eventually controls the output phase. Once in lock, the VCO output signal has the same phase and frequency as the input signal.

The theoretical analysis of PLLs remains an interesting topic due to its highly nonlinear behavior. However, a linear model approximates it well when lock is achieved. The analysis of the *lock* state based on control theory is well established. The linearized s-domain model for analyzing a PLL is shown is Figure 2.2.

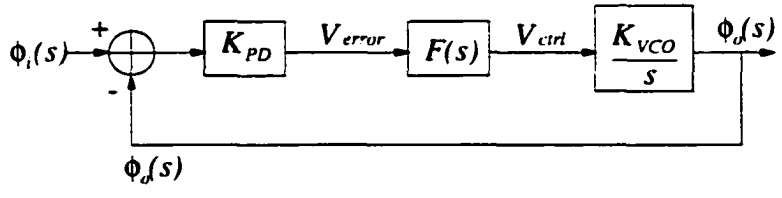


Figure 2.2 Linearized s-domain model of PLL

In this model the phase detector is modeled as a block with gain K_{PD}, which generally has units of Volts/radian. The expression to describe the function of the phase detector is:

$$V_{error}(s) = K_{PD}(\phi_i(s) - \phi_o(s)) \tag{2.1}$$

The model of the VCO is:

$$f_{out} = f_c + K_{VCO} \times V_{ctrl} \tag{2.2}$$

where f_c is the free running frequency of the VCO and K_{VCO} is the gain of the VCO. The output phase is the integral of frequency over time. In the s-domain, the output phase is described by:

$$\phi_o(s) = \frac{K_{vco}}{s} \cdot V_{ctrl}(s) \tag{2.3}$$

where the factor of $\frac{1}{s}$ is due to the integration.

Putting all of the above models together, the closed loop phase transfer function is:

$$H(s) = \frac{\phi_o(s)}{\phi_i(s)} = \frac{K_{PD} K_{VCO} F(s)}{s + K_{PD} K_{VCO} F(s)} \tag{2.4}$$

The exact nature of the transfer function in equation 2.4 is not clear until an expression for the loop filter is given. In practical implementations, the loop filter is usually a low pass filter that adds another pole or more poles into the system. By doing so, there is more flexibility in the design of the PLL and more trade-offs in its performance parameters. In its simplest form, the first order low pass filter can be expressed as

$$F(s) = \frac{K_{LPF}}{1 + \frac{s}{\omega_{LPF}}} \tag{2.5}$$

By substituting 2.5 into 2.4, the loop transfer function now becomes

$$H(s) = \frac{K}{\frac{s^2}{\omega_{LPF}} + s + K} \tag{2.6}$$

where $K = K_{PD} K_{VCO} K_{LPF}$ is called the *loop gain* and expressed in rad/s.

In order to understand the dynamic behavior of the PLL, Equation 2.6 can be converted into the familiar form used in control theory as follows, where ξ is the damping factor and ω_n is the natural frequency:

$$H(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{2.7}$$

where

$$\omega_n = \sqrt{\omega_{LPF} K} \quad , \quad \xi = \frac{1}{2}\sqrt{\frac{\omega_{LPF}}{K}} \tag{2.8}$$

Control theory shows that ξ is usually greater than 0.5 and preferably equal to $\frac{\sqrt{2}}{2}$ so as to provide an optimally flat frequency response. Therefore, K and ω_{LPF} cannot be chosen independently. As will be shown shortly, the charge-pump PLL overcomes this drawback.

The transfer function in Equation 2.7 is a second-order low pass filter. Since it is the transfer function of *phase* not voltage, it means that the high frequency variation on the input phase will be filtered out and does not appear in the output phase. In this sense, the PLL has a jitter suppression function on the input.

The generic PLL considered thus far is of second order. In principle, the low-pass filter can include more poles to achieve sharper cut-off characteristics. However, such systems are difficult to stabilize, especially when process and temperature variations are taken into account. Therefore, for monolithic implementations, a higher order (> 3) PLL is seldom used. On the other hand, in many cases the PLL inevitably has a third pole, for example, if a capacitor is added in parallel with the LPF output port to suppress high frequency variations on the control voltage. Thus, most practical PLLs are third order with the third pole being much farther from the the origin than the other two.

2.1.2 Charge-pump PLL

A popular trend in recent years is the use of a phase/frequency detector incorporating 3-state sequential logic. This kind of phase detector delivers two digital outputs *Up* and *Down*, which needs a charge pump to convert the logic levels into analog signals suitable for controlling the voltage-controlled oscillator. The combination of the sequential phase detector and the charge pump forms a new type of PLL — a *charge pump PLL* that has unique advantages over a conventional PLL: it expands the acquisition range to be only limited by the tuning range of the VCO. It also has several other merits that will be shown in this section.

A generic charge-pump PLL block diagram is shown in Figure 2.3. A charge pump is nothing more than a three-position, electronic switch that is controlled by the three states of the phase detector: *Up*, *Down*, and *N(nochange)*. When the switch is set in the *Up* or *Down* position, it delivers a *pump current* $\pm I_p$ to the loop filter. In the *N* position, both switches are open, thereby isolating the loop filter from the charge pump and the phase detector.

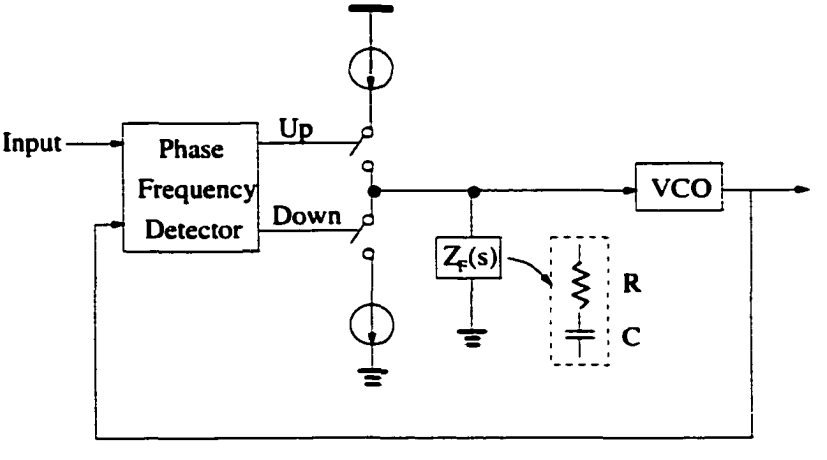


Figure 2.3 Charge-pump PLL

The inherent sampling operation makes a charge-pump PLL a discrete-time system. Exact analysis must take this characteristic into account. However, as shown by Gardner [6], if the loop bandwidth is much less than the input frequency, continuous time analysis based on the transfer function still provides an excellent approximation to the behavior of a charge-pump loop.

Suppose the phase error $\phi_i - \phi_o$ is ϕ_e, then the average current charging the loop filter will be $\frac{I_p \phi_e}{2\pi}$ and the average change in the control voltage of the VCO equals

$$V_{ctrl}(s) = \frac{I_p \phi_e(s)}{2\pi} Z_F(s) \tag{2.9}$$

For a locked loop, the VCO output phase is

$$\phi_o(s) = \frac{K_{VCO} V_{ctrl}(s)}{s} \tag{2.10}$$

Combining 2.9, 2.10 and $\phi_e(s) = \phi_i(s) - \phi_o(s)$, the loop transfer function is

$$H(s) = \frac{K_{VCO} I_p Z_F(s)}{2\pi s + K_{VCO} I_p Z_F(s)} \tag{2.11}$$

To obtain a stable system, the simplest loop filter will be a resistor R in series with a capacitor C, therefore $Z_F(s) = R + \frac{1}{sC}$. The transfer function becomes

$$H(s) = \frac{\frac{I_p K_{VCO}}{2\pi C}(RCs + 1)}{s^2 + \frac{I_p K_{VCO} R}{2\pi} s + \frac{I_p K_{VCO}}{2\pi C}} \tag{2.12}$$

where this system has a zero at $\omega_z = -\frac{1}{RC}$. By defining

$$\begin{aligned} \omega_n &= \sqrt{\frac{I_p K_{VCO}}{2\pi}} \\ \xi &= \frac{R}{2}\sqrt{\frac{I_p C K_{VCO}}{2\pi}} \\ K &= \frac{K_{VCO} I_p R}{2\pi} \\ \tau_z &= RC \end{aligned} \tag{2.13}$$

where ω_n is the *natural frequency*, ξ is the *damping factor*. K is the *loop gain* and τ_z is the time constant of the loop filter, equation 2.12 can therefore be expressed as

$$H(s) = \frac{\omega_n^2(\tau_z s + 1)}{s^2 + 2\xi\omega_n s + \omega_n^2} = \frac{Ks + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{2.14}$$

Equation 2.14 shows that a charge-pump PLL has similar behavior to a traditional PLL. Notice that in equation 2.13, ω_n is independent of R. This suggests that in charge-pump PLL, both ω_n and ξ can be optimized independently. This is one advantage over the conventional PLL discussed in section 2.1.1.

Another attribute of the charge-pump PLL is that the steady-state phase error is zero. Applying the final-value theorem, the static phase error is found to be

$$\phi_e = \frac{2\pi\Delta\omega}{K_{VCO} I_p Z_F(0)} \tag{2.15}$$

where $\Delta\omega$ is the frequency offset between the input signal and the free-running frequency of the VCO. Notice that when $Z_F(0) \to \infty$, $\phi_e = 0$. Therefore, whenever a charge-pump PLL is in lock, there is no phase error between the input and the output. This is usually a desired characteristic, which is not always true in a conventional PLL.

The inclusion of a resistor R in the loop filter stabilizes the loop. However, it introduces ripple on the control voltage. When the charge pump is turned on during each cycle, the pump current I_p is driven to the loop filter. Due to the resistor, an instantaneous voltage jump of $\Delta V = I_p R$ will happen on the control voltage. At the end of the charging/discharging interval, the pump current is turned off and a voltage jump of equal magnitude occurs in the opposite direction. The frequency of the VCO follows this ripple on the control voltage so there will be frequency excursion of $\Delta\omega = K_{VCO} I_p R$ for each pump cycle. This introduces jitter on the VCO output, which is undesired in many applications. Therefore additional filtering is usually required. This can be done by adding an additional capacitor C_2 in parallel with the RC network, which is shown in Figure 2.4. This modification introduces a third pole in the PLL, requiring further study of stability issues. Gardner provides a criteria for the stability of such a system [6]. As a rule of thumb, if the third pole is far from the dominant poles, it does not affect the loop stability. Therefore, C_2 is usually much smaller than C.

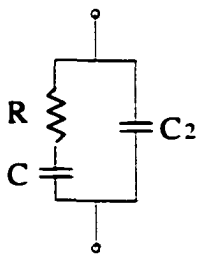


Figure 2.4 General form of charge pump loop filter

As the loop bandwidth becomes comparable with the input frequency, discrete-time analysis must be applied. Discrete-time analysis on a second order system can be found in Gardner [6]. For higher orders, the analysis will be very complicated.

In summary, a charge-pump PLL has significant advantages over a traditional PLL. Moreover, it is

especially suitable for monolithic implementations. Therefore nearly all contemporary monolithic PLLs are charge-pump type PLLs.

2.1.3 Typical applications

The applications of phase locked loops are so widespread that here we can only concentrate on the most important ones in monolithic systems.

2.1.3.1 Narrow band filter for noise and jitter suppression

In communication systems, the transmitted signal is corrupted by the channel and is obscured by noise by the time it reaches the receiving end. In order to achieve the required signal-to-noise ratio, the sideband noise around the carrier ω_c must be suppressed. This usually requires a narrow band filter with very high Q.

As discussed earlier, a PLL is basically a phase low pass filter that can suppress the noise in the input. Making the bandwidth of the PLL sufficiently small results in an equivalently high Q narrow band filter. In other words, the output from the PLL takes the average of the input frequency over a great many cycles. The variation on the input is averaged out. This is illustrated in Figure 2.5.

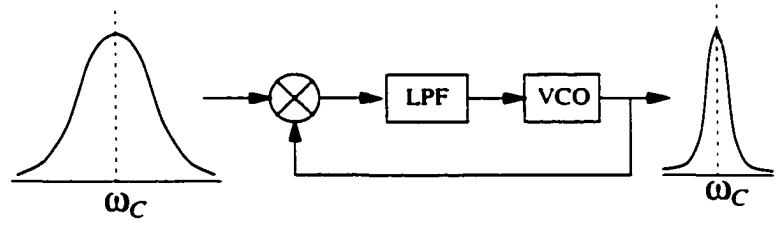


Figure 2.5 PLL as narrow band filter

2.1.3.2 De-skew clock generator

In most digital systems, all on-chip clocks are ultimately derived from a board level system clock. Due to buffer and interconnection delays, skew between the on-chip and off-chip clocks is inevitable. For many applications, this skew is not tolerable. In this situation, a PLL can be used to align the local on-chip clock with the original system clock. If a charge-pump PLL is used, this skew can in principle be completely canceled ($\theta_e = 0$). This is shown in Figure 2.6.

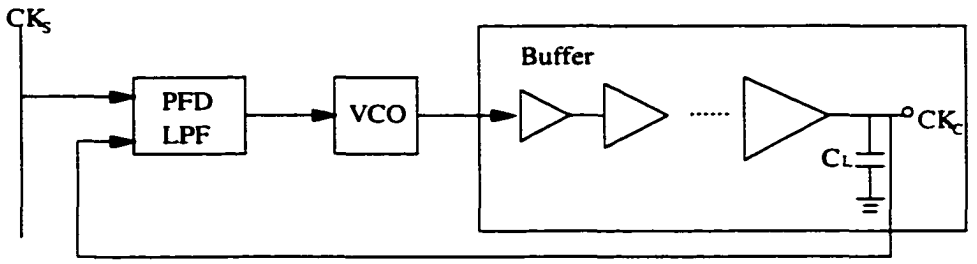


Figure 2.6 PLL as de-skewing clock generator

2.1.3.3 Frequency synthesis

If a $\div M$ frequency divider is added in the feedback path between the VCO and the phase detector, as shown in Figure 2.7, the VCO will output a signal whose frequency is exactly M times the input frequency. This frequency multiplication is very useful in many monolithic applications.

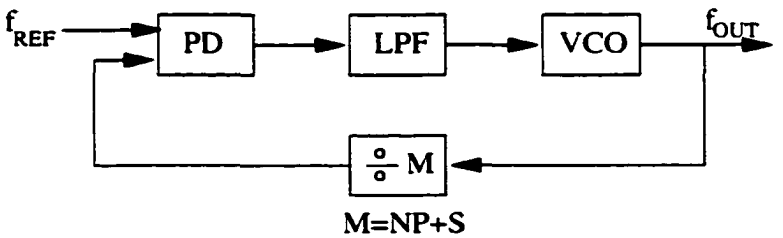


Figure 2.7 PLL with frequency multiplication

Due to the speed limitations of PC boards, it is extremely difficult and costly to provide a very clean high frequency reference clock to the chip. However, the speed of the on-chip system may require such a high frequency clock. The frequency multiplication mechanism of PLL allows a low speed clock to be input from the board and a high speed on-chip clock to be available for internal use.

This frequency synthesis function is also widely used in wireless transceivers that employ a local oscillator whose output frequency must be varied in small, precise steps. If the frequency divider is not fixed but can vary in small steps, the output frequency will also vary in a similar fashion.

2.1.3.4 Clock recovery

In digital communications, the transmitted data needs to be regenerated at the receiver end. Usually the clock associated with the data is not separately transmitted but is instead merged with the data into a continuous bit stream. Therefore the sampling clock needs to be extracted (recovered) from the

incoming bit stream. A PLL can perform this function.

Unlike the previous situations, the input to the PLL now is not a periodic signal but rather a sequence that depends upon both clock and data. This makes the PLL design for this specific application more challenging. The PLL not only needs to create the periodic clock signal which is synchronized to the data stream but also needs to maintain the frequency when data transitions are absent. The block diagram of such a clock recovery system is shown in Figure 2.8. As a major application of PLLs, a detailed discussion of this circuit will be carried out in Chapter 6.

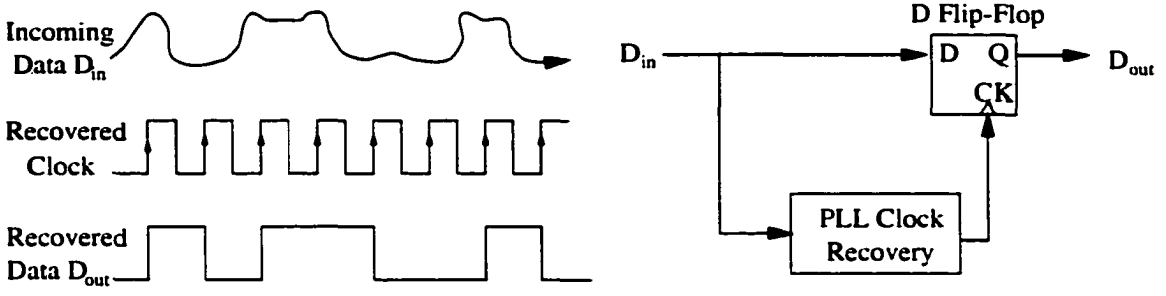


Figure 2.8 PLL based clock recovery

2.2 Delay Locked Loop Fundamentals

2.2.1 Operating principles of DLL

If the VCO in a PLL is replaced by a Voltage-Controlled-Delay-Line (VCDL), the feedback loop is then called a Delay Locked Loop. Its architecture is shown in Figure 2.9. It is similar to a PLL in many ways but still has significant differences.

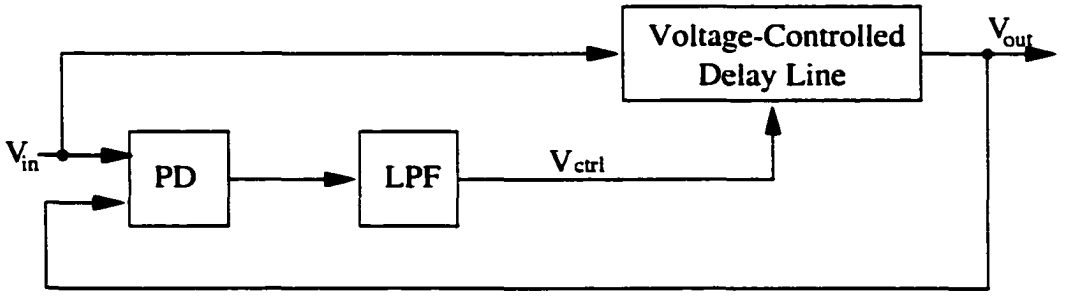


Figure 2.9 Delay locked loop block diagram

The reference input is not only fed into the phase detector, but is also fed into the delay line. The output from the delay line is aligned to the input by the feedback loop. The basic idea is that if a periodic signal is delayed by an integer number of cycles, the phase shift can be considered zero. When in lock, the total delay between the on chip clock and the reference clock is exactly nT (where T is the period) with no skew exists. In this case, the DLL does not create a new signal but only delays the reference clock. This is the basic difference between a DLL and a PLL. As will be discussed shortly, this difference gives DLLs some unique attributes.

The linearized s-domain model for analyzing DLLs is shown is Figure 2.10. Here the only difference from the PLL is that the VCO model is replaced by the VCDL model. There is no integration in the DLL but just a constant gain K_{VCDL}. There is also no stability problem in DLL. A simple capacitor C is enough to serve as the loop filter. The phase transfer function is:

$$\begin{aligned} H(s) = \frac{\phi_o(s)}{\phi_i(s)} &= \frac{K_{PD}K_{VCDL}F(s)}{1 + K_{PD}K_{VCDL}F(s)} \\ &= \frac{\frac{K_{PD}K_{VCDL}}{C}}{s + \frac{K_{PD}K_{VCDL}}{C}} \end{aligned} \tag{2.16}$$

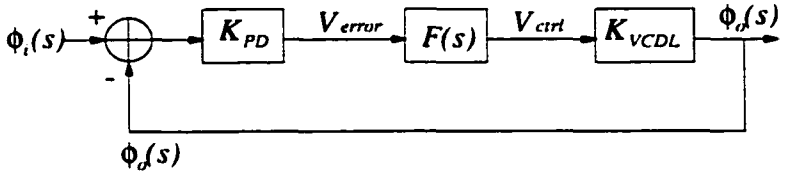


Figure 2.10 Linearized s-domain model of DLL

Two observations can be drawn from Equation 2.16: (1) DLL is a phase low pass filter. It can suppress the jitter on the input. (2) DLL is a first order system. It has relaxed trade-offs between gain, bandwidth and stability. It is only characterized by the loop bandwidth, which is

$$\omega_n = \frac{K_{PD}K_{VCDL}}{C} \tag{2.17}$$

A more significant advantage of DLL over PLL is that it generally has less jitter than a PLL. This is the major reason why it is widely used. With its input jitter suppression function, it is applied in our proposed structure that will be described in Chapter 4.

2.2.2 Typical applications

Since a delay line does not generate a signal, it is difficult to perform frequency multiplication. Therefore, it is mainly applied as clock and timing generator where frequency synthesis is not needed.

such as the clock generator for microprocessor and DRAM, etc.

Besides jitter suppression and de-skewing functions, DLLs can also provide precisely spaced timing edges even in the presence of temperature and process variations. This is very useful for many digital systems. In practice, mismatches between stages limits the edge delay accuracy. Chapter 7 will present a new circuit design that targets this delay mismatch problem.

2.3 Building Blocks

Though the phase lock loop is simple in principle, its monolithic implementation involves many subtleties and it continues to be an area of intense study. Innovations in the design of its building blocks have tremendously improved the speed, power dissipation and jitter performance of phase-locked systems. Among all these innovations, the majority have pertained to the design of the voltage controlled oscillator, the core part of the PLL. The phase detector also has many forms and has also been extensively studied. By contrast, there is much less variation on the implementation of loop filters. Almost all recent designs use passive forms. In the following sections, a brief discussion about VCOs and phase detectors will be carried out. The implementation of delay lines will also be considered.

2.3.1 Voltage-controlled oscillator

The design of oscillators itself is a very broad topic and a rather independent one. Since the focus of this dissertation is about the CMOS implementation, the discussion will be limited to this scenario.

The general design trends of CMOS VCOs can be summarized as follows: to achieve (1) higher speed, (2) lower jitter or phase noise, with less cost. Almost all recent publications on VCOs emphasized at least one of these two trends. Jitter and phase noise performance is especially more important in recent years. To understand how progress can be made in the design of VCOs to comply with these trends, two commonly used monolithic VCO topologies will be reviewed: (1) LC tank oscillator and (2) ring oscillator. Since the discussion on the topology is closely related to jitter and phase noise, the concept of jitter and phase noise needs to be briefly introduced first.

2.3.1.1 Jitter and phase noise

Both jitter and phase noise are manifestations of random variation in phase, which is always unwanted and is to be minimized. Jitter is the time domain characterization of this error, while phase noise is the frequency domain characterization. This is illustrated in Figure 2.11.

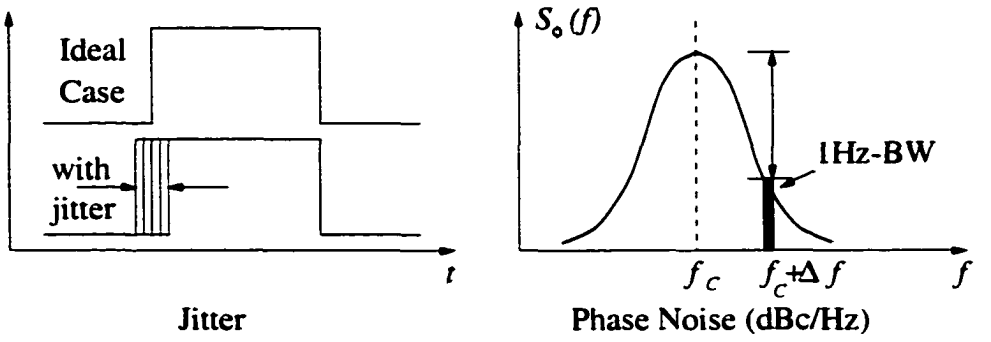


Figure 2.11 Jitter and phase noise

Jitter is a stochastic process. Usually *r.m.s.* value and *peak-to-peak* value are used to measure it. This random process is reflected in the frequency domain as a sideband around the carrier frequency. The phase noise is specified in dBc/Hz at a given offset, where dBc refers to the power level in dB relative to the carrier. Therefore, the phase noise of an oscillator at a given offset Δf is found from the ratio of the power in a 1-Hz bandwidth at the offset frequency $f_c + \Delta f$, to the total power of the carrier. For reasonable resolution bandwidth, as described by Robins [7], this can be approximated by the Y-distance between the center f_c and offset $f_c + \Delta f$ on the spectrum, which is also indicated in the figure.

Even though jitter and phase noise measure the same phenomenon — noise effect, their applications are different. For RF applications such as a wireless transceiver that has the tightest jitter requirements, phase noise is used to measure this performance. For applications with loose jitter requirement, such as digital communication, time domain jitter measurement is more frequently used.

2.3.1.2 *LC* tank oscillator

Though there are many variations of this type of oscillator, it usually includes an *LC* resonator and an active device. The configuration should form a positive feedback to generate oscillation. The most commonly used configuration is a Colpitts oscillator, which is shown in Figure 2.12.

The theoretical analysis of phase noise in *LC* oscillators has been ongoing since 1960's. If only tank loss is considered, the phase noise at a given offset $\Delta\omega$ of such an oscillator is

$$L(\Delta\omega) = 10log[\frac{\overline{v_n^2}/\Delta f}{\overline{v_c^2}}] = 10log[\frac{2kT}{P_c}(\frac{\omega_c}{2Q\Delta\omega})^2] \tag{2.18}$$

Equation 2.18 shows that the phase noise at a given offset improves as both the carrier power P_c and the tank quality factor Q increase. It also shows that the phase noise decreases linearly with Δf^{-2}.

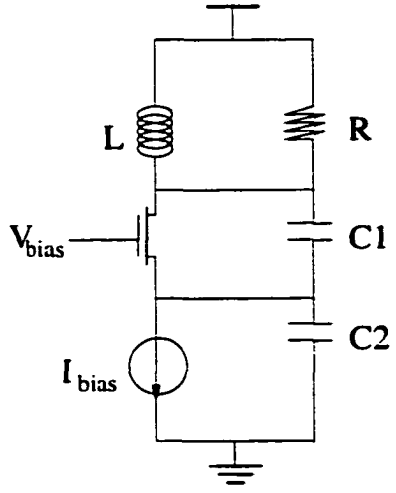


Figure 2.12 LC tank oscillator — Colpitts oscillator

In reality, besides tank loss, active device noise plays an important role. Leeson [8] modified the above equation by considering this effect:

$$L(\Delta\omega) = 10log\{\frac{2FkT}{P_c a}[1 + (\frac{\omega_c}{2Q\Delta\omega})^2](1 + \frac{\Delta\omega_{1/f^3}}{|\Delta\omega|})\} \tag{2.19}$$

The major modifications include introducing an empirical factor F to account for the increased noise in the $\frac{1}{\Delta\omega^2}$ region, adding unity inside the brackets to account for the noise floor, and a multiplicative factor (last parentheses) to provide a $\frac{1}{\Delta\omega^3}$ behavior at sufficiently small offset frequencies.

Besides enhancing the quality factor Q to lower the phase noise, Equation 2.19 also suggests that F must be lower as well, which has an ambiguous design implication. A more recent analysis on the design of LC tank oscillators is described by Hajimiri [9] and Lee [10].

The major challenge in the design of integrated LC oscillators is the implementation of inductors. It is usually implemented with a metal spiral inductor, which takes large area. The resulted LC tank quality factor Q is relatively low. Nonetheless, the phase noise performance of such an oscillator is still the best among monolithic oscillators. The fundamental reason is that it has a narrow band frequency-selective element, an LC resonator, to filter out the noise. Therefore this type of oscillator is mainly applied in RF applications that have the least jitter tolerance.

A fully differential circuit is usually used to enhance power supply rejection and improve noise immunity. Design issues of differential LC oscillators can be found in [11]. Fully monolithic LC tank oscillators can be found in many recent publications, such as from Nguyen [12], Razavi [13], Dauphinee [14], Dec [15], Kim[16], etc. Among them, [13], [15] and [16] are CMOS implementations.

2.3.1.3 Ring oscillator

Ring oscillators are more widely used in integrated systems than LC oscillators because they are easy to implement at rather low cost.

A ring oscillator topology is shown in Figure 2.13. Here, a cascade of M delay cells forms a ring structure. A total phase shift of 180° will make this structure oscillate, with a period of $2MT_d$, where T_d is the delay of a single stage. An obvious advantage of a ring oscillator over an LC oscillator is that it can provide multi-phase clocks: the clock can be output from any stage and clocks from different stages have certain defined relationships. It is worth noting that multiple loops can be used if necessary to increase the speed [17] [18].

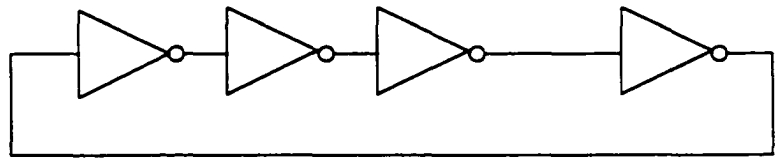


Figure 2.13 Ring oscillator

The delay cell in a ring oscillator can have various forms as long as it can provide adequate delay time. The most common form is a simple gain stage that can be found in two categories based on either (1) single-ended current starved inverter, or (2) fully differential current steering amplifier, which is shown in Figure 2.14.

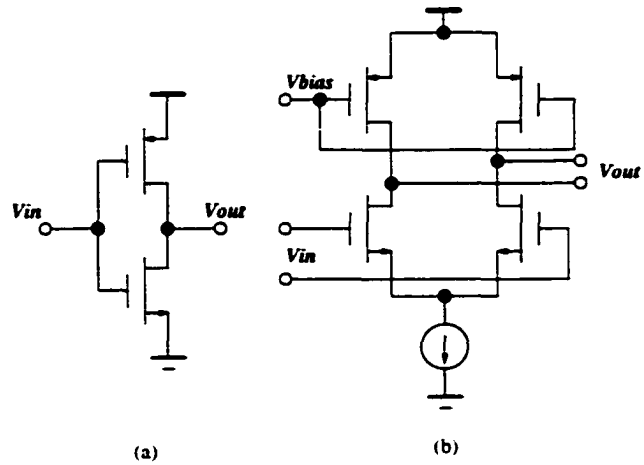


Figure 2.14 Two general forms of delay cell: (a) single-ended form (b) fully differential form

If a single ended form is used, the total number of stages should be odd to satisfy the oscillation criteria. On the other hand, if a differential pair is used, the number of stages needs not be odd; the total phase shift can be changed by 180° if the output signals of one of the stages are swapped. With an even number of stages, the oscillator can provide quadrature outputs, i.e., outputs that are 90° out of phase. This feature is very useful for many applications. Moreover, since a differential pair has good power supply rejection, which is important to jitter performance, it is almost always used. Nevertheless, for some applications with relaxed jitter requirements, such as clock generation for microprocessors, single-ended approaches can still be found [19].

In order to vary the oscillation frequency, either M or T_d needs to be tunable. The examples of changing the effective number of M can be found in [20] and [21]. However, this tuning method is seldom used. The second tuning method — changing T_d can be done by directly controlling the equivalent delay time constant of the delay cell. Two methods can be applied: (1) capacitive tuning, or (2) resistive tuning. Capacitive tuning changes the effective load capacitance, such as the *pn* junction capacitor. The drawback is the small tuning range. By contrast, resistive tuning can realize a large, relatively uniform frequency variation and lends itself to differential control. An illustration of resistor tuning is shown in Figure 2.15. This can be done by controlling the tail current or controlling the load impedance, separately or jointly. Resistive tuning is the dominant tuning method in contemporary designs.

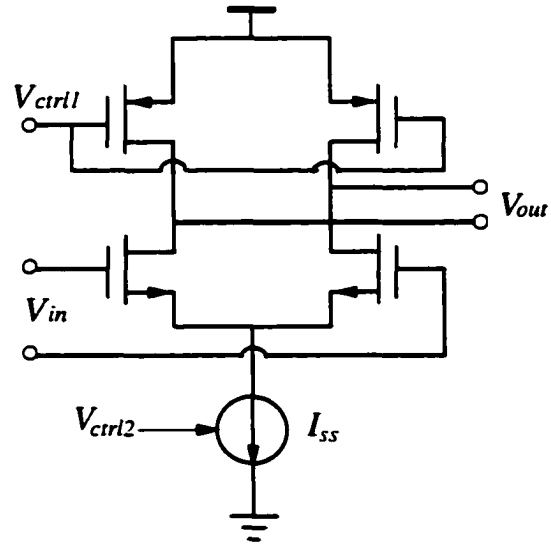


Figure 2.15 Resistive tuning

The above discussion on delay stages is also valid for delay lines used in DLLs. The only difference between a delay line and a ring oscillator is that the delay line does not form a loop.

An important issue in ring oscillator design is the minimum number of stages to attain reliable operation. Since oscillation occurs at a frequency where the total phase shift is zero and the loop gain is unity, decreasing the number of stages will make the phase shift and *DC* gain of each individual stage increase. In reality, a two stage ring oscillator usually does not operate reliably. Typically 3 or more stages are used.

The study of jitter in ring oscillator has only begun in recent years. Some initial results that provide useful design guidelines are presented by Weigandt [22] Razavi [23] Mcneil [24] and Hajimiri [25]. Though the form of the result of different analysis is somewhat different, the basic point is the same:

$$\textit{Jitter or phase noise} \propto \frac{1}{V_{swing}} \cdot \frac{1}{I_{SS}} \tag{2.20}$$

where V_{swing} is the voltage swing of the oscillator and I_{SS} is the total supply current. Equation 2.20 shows that increasing the power dissipation will help to improve the jitter performance. However, with a specific power dissipation, there are still other strategies that can improve the jitter performance. These are summarized as below:

(1) Within the delay cell, if it is working under switching mode, that is the NMOS and PMOS are not turned on all the time, less noise will be generated. This is easily understood because the noise of the active device exists only when the device is turned on. If it is turned off, no noise will contribute to the output. Therefore, adopting a fast-slewing delay cell performing full switching will improve the jitter performance [26].

(2) Increasing the gate voltage on the differential pair transistor and keeping the gain of the delay cell small (1.5-3) helps to reduce the jitter [22].

The jitter performance of ring oscillators is usually worse than that of *LC* oscillators. Therefore, they are generally not used for RF applications, although this might change in the near future. Some work has been reported which intends to use ring oscillators instead of *LC* oscillators to reduce cost [27]. So far, they are widely applied in digital communications, digital I/O interfaces, and so on. As a summary, the main features of *LC* and ring oscillators are compared in Table 2.1.

The phase locked and delay locked loops in this dissertation target serial data communication clock generators and clock recovery circuits, therefore ring oscillators are used for all designs.

Finally, there is another type of oscillator worthy of mention — relaxation oscillators. Since they are less popular than the above two types especially in the last five years, they are not discussed here.

Table 2.1 Comparison of LC oscillator and ring oscillator

	LC Oscillator	Ring Oscillator
speed	GHz range	Up to GHz range
phase noise or jitter	small	relatively big
tuning range	narrow	wide
multi-phase clocks	generally not a feature	inherent
hardware cost	relatively high	low
dominant application	RF	digital communication

Design examples on this type of oscillator can be found in [28] [29] [30]. Its jitter analysis was described by Abidi in [31].

2.3.2 Phase detector

The characteristics of the phase detector are very important to the dynamic performance of the whole loop. Discussed here are three general types of phase detectors. Notice that the phase detector for clock recovery is somewhat special, and will be further discussed in Chapter 6.

2.3.2.1 Gilbert multiplier phase detector

A Gilbert multiplier phase detector is shown in Figure 2.16. For small signal inputs to ports A and B, the average output will be

$$\overline{V_{out}(t)} = \frac{\alpha AB}{2}\cos\Delta\phi \tag{2.21}$$

when $\Delta\phi$ is around $\frac{\pi}{2}$, Equation 2.21 becomes

$$\overline{V_{out}(t)} = \frac{\alpha AB}{2}(\frac{\pi}{2} - \Delta\phi) \tag{2.22}$$

yielding $K_{PD} = -\alpha AB/2$. Notice that when $\omega_A \neq \omega_B$, the average output is zero. Therefore it cannot be used as a frequency detector.

If full switching occurs in a Gilbert multiplier, it becomes an XOR gate. Unlike the Gilbert multiplier, an XOR gate extends the linear range for $0 <| \Delta\phi |< 180°$, which is shown in Figure 2.17.

Notice that in both detectors, the output depends on the duty cycle of the inputs. In many cases, this is a drawback. Therefore, a phase detector sensitive only to signal edges is developed.

2.3.2.2 RS latch phase detector

This phase detector employs an edge-triggered RS latch, which is shown in Figure 2.18. A rising edge of A drives Q to "1" and that of B drives Q to "0". Thus, the differential output goes high or low

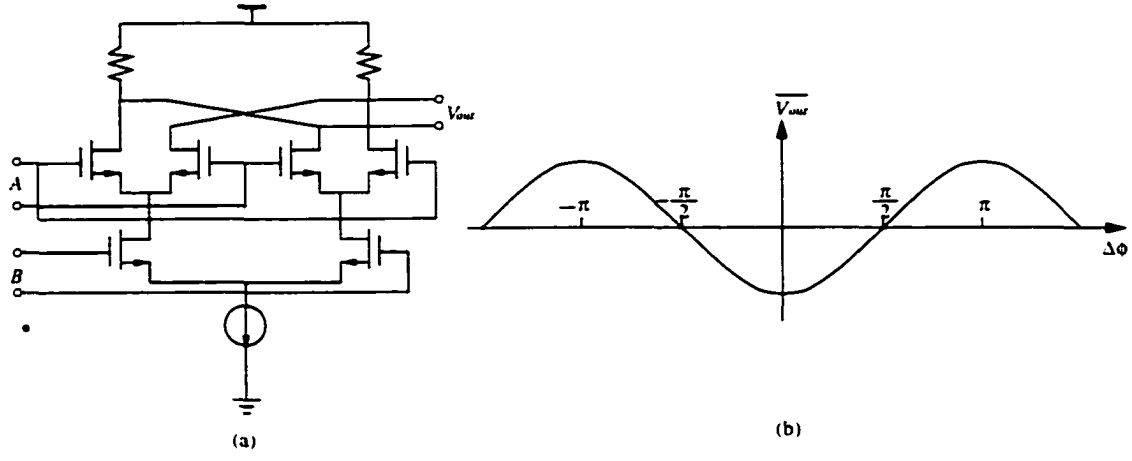


Figure 2.16 Gilbert multiplier phase detector (a) Gilbert cell; (b) Gilbert phase detector characteristic

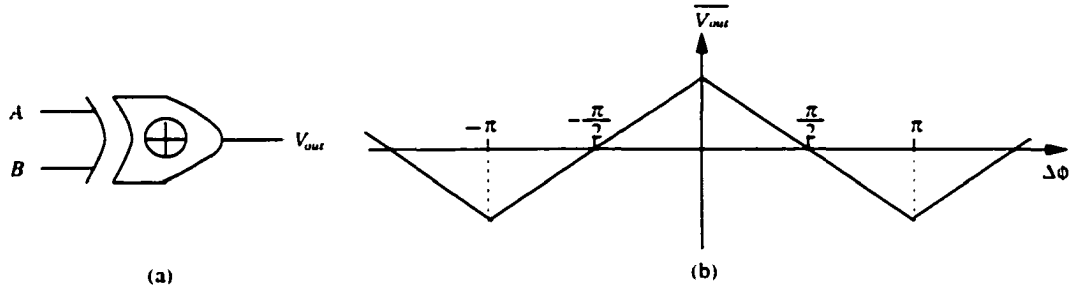


Figure 2.17 Characteristic of XOR phase detector

between a rising edge at one input and a rising edge at the other. It is not sensitive to the signal duty cycle. It is still not a frequency detector.

2.3.2.3 Phase-frequency detector

This type of detector is extremely useful because it significantly increases the capture range and lock speed of PLLs. It is usually employed with a charge pump at its outputs to convert the three-state logic levels into the control voltage. PLLs with this type of phase detector are called charge-pump PLLs, which were analyzed in section 2.1.2.

This phase detector is composed of sequential logic which generates two outputs: Up and $Down$. If the frequency of B is less than that of A, then the PFD produces positive pulses at Up, while $Down$ remains at zero. This output combination indicates that the frequency of B should be increased. In the situation if frequency of A is less, the outputs are reversed. If the frequency of A and B are the

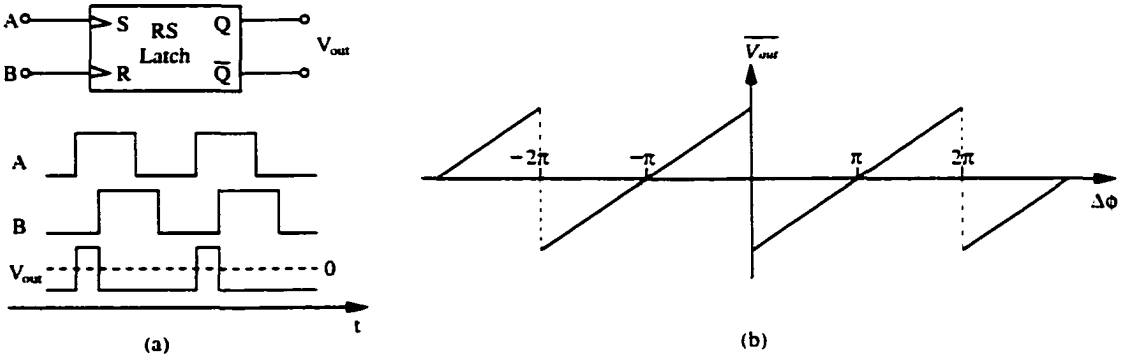


Figure 2.18 Edge-triggered *RS* latch phase detector (a) operation; (b) RS latch phase detector characteristic

same, the circuit generates pulses at either *Up* or *Down*, indicating the phase difference. Notice that *Up* and *Down* will never be "1" at the same time. This PFD can be implemented with two *D*-flipflops and one *NAND* gate as shown in Figure 2.19(c). This type of phase-frequency detector is the most popular one in modern PLL design [32].

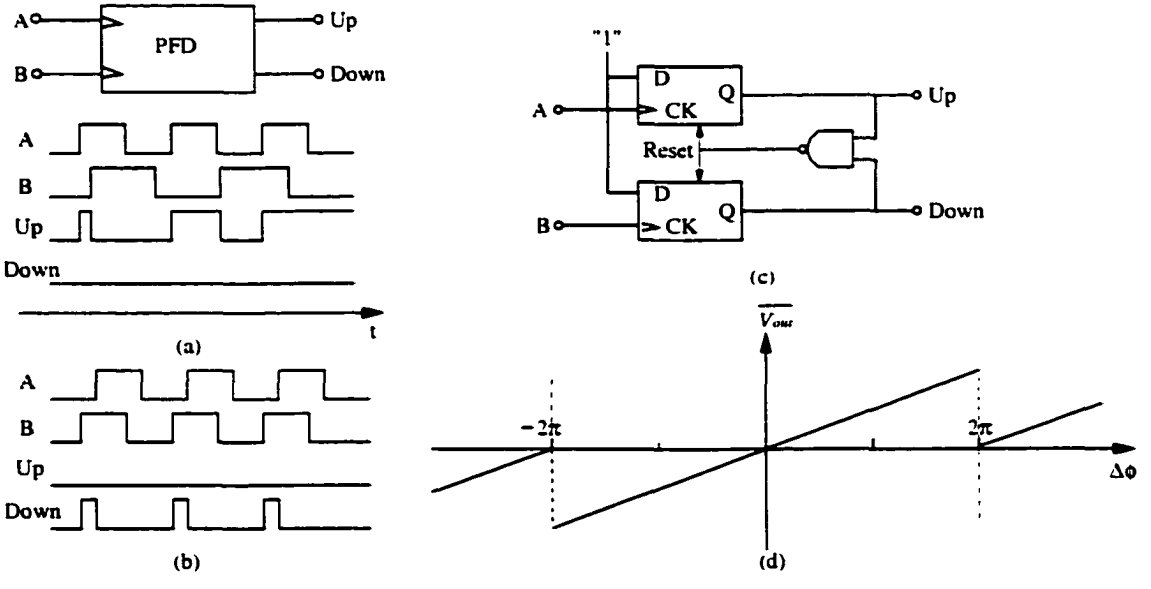


Figure 2.19 Sequential phase frequency detector (a) $f_A > f_B$: (b) $f_A = f_B$ and A lags B: (c) PFD implementation: (d) PFD characteristic

CHAPTER 3 JITTER IN PHASE-LOCKED AND DELAY LOCKED LOOPS

Noise is inevitable in monolithic systems and it is usually the limiting factor of the system performance, especially in communication systems. Any type of noise generated by any block in PLL and DLL or any coupling noise injected into the loop will definitely decrease the timing accuracy the PLL and DLL can achieve. This issue becomes especially important when PLL or DLL is embedded in a rather big system where the environment turns out to be noisy. In those *system-on-a-chip* applications, low noise (or low jitter) performance is the dominant issue in integrated PLL and DLL design.

In this chapter, based on the introduction in Chapter 2, the noise analysis of PLL and DLL systems will be carried out. The analysis is composed of two parts: frequency domain analysis and jitter analysis based on a discrete time model as well as statistical methods. The goal is to provide theoretical insight into this issue and to give out design implications.

In practice, another important issue of designing low noise PLL and DLL is measuring jitter appropriately. Therefore, at the end of this chapter, jitter measurement will be discussed and the features of different methods will be compared.

3.1 Noise Property of PLL and DLL

3.1.1 Introduction

3.1.1.1 Noise sources in integrated circuit

There are many noise mechanisms in integrated circuits. Generally speaking, noise can be divided into *inherent noise* and *coupling noise*.

Inherent noise refers to the noise always accompanying the devices: active (transistors) or passive (resistors). It is random in nature and can never be eliminated. Some examples of inherent noise are thermal, shot, and flicker noise. Detailed introduction into each of these noise mechanism can be found in any textbook on analog circuit design, such as in [33], [34], and [35]. Inherent noise can be reduced by

proper circuit design, such as choosing the appropriate topology or increasing the power consumption, but is rather irrelevant to circuit wiring and layout.

Coupling noise is the result of unwanted interaction between different parts of a circuit or the circuit and the outside world. This type of noise may or may not appear as random signals. And it is significantly affected by the layout pattern. It can be significantly reduced by careful layout and correct decoupling. Typical examples of such noise are power supply and substrate coupling noise, which is an important topic still undergoing much study. Some research has been done by Herzel [36] and Heydari [37] on their effect on PLLs.

3.1.1.2 Noise analysis method

An example of a noise waveform is shown in Figure 3.1. Since it is a stochastic process, measuring it must use a statistical method. The basics of random process can be found in [38] and [39].

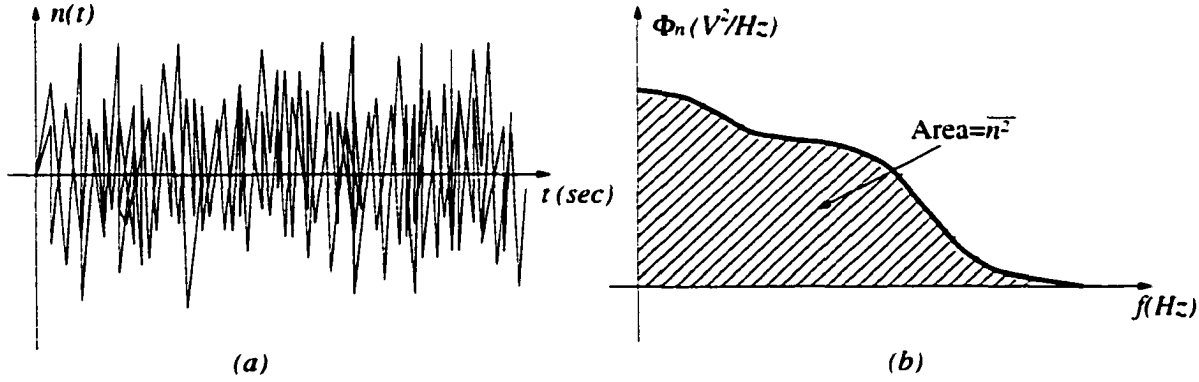


Figure 3.1 An example of noise. (a) noise as a function of time; (b) power spectral density of noise

A measure of the noise signal strength is the mean-square value $\overline{n^2}$, also called its "power". This is defined by

$$\overline{n^2} \equiv \lim_{t\to\infty} \int_0^t n^2(\tau)d\tau \tag{3.1}$$

The root-mean-square(rms) value is defined as

$$n_{rms} \equiv \sqrt{\overline{n^2}} \tag{3.2}$$

The distribution of the power in the frequency domain is given by the *power spectral density* $\Phi_n(f)$ for the variable $n(t)$. An example of power spectral density is shown in Figure 3.1(b). It gives for each

frequency the mean-square value within a 1-Hz band centered on that frequency. Therefore, the total mean-square value is the area under the curve:

$$\overline{n^2} = \int_0^\infty \Phi_n(f)df \tag{3.3}$$

Equation 3.3 is a convenient tool for finding time averages from frequency domain information.

If the random noise is passing through a linear time-invariant system, the output will be shaped by the system, which is

$$\Phi_{on}(f) = \Phi_n(f)|\ H(f)\ |^2 \tag{3.4}$$

where $H(f)$ is the frequency response of the linear system.

3.1.2 Noise property of PLL

If any building block of a PLL or the input signal exhibit noise, then the output signal will also suffer from noise. The noise will cause the output phase uncertainty—phase noise or jitter. In general, all the loop components, including the phase detector, the loop filter, and the VCO may contribute noise. A general discussion on how the spectrum of a given noise source is shaped as it propagates to output can be found in [40]. The loop response to VCO noise is especially important. Consequently, the discussion here will focus on the loop response to VCO phase noise as well as the phase noise on the input signal.

3.1.2.1 PLL response to VCO noise

The noise of a VCO has been discussed in section 2.3.1. A natural question is that if a noisy VCO is inside the PLL, what is the loop function in dealing with this type of noise? In order to answer this, the loop transfer function corresponding to the noise of VCO should first be developed. The model used to analyze this behavior is shown in Figure 3.2.

When considering the VCO noise, suppose the input is noiseless, i.e. the *excess* phase of the input is zero ($\Phi_{in} = 0$). The transfer function on the VCO noise is

$$(0 - \phi_o(s))K_{PD}F(s)\frac{K_{VCO}}{s} + \phi_{nvco}(s) = \phi_o(s) \tag{3.5}$$

$$\Longrightarrow H_{nvco}(s) = \frac{\phi_o(s)}{\phi_{nvco}(s)} = 1 + \frac{K_{PD}F(s)K_{VCO}}{s} \tag{3.6}$$

For a simple low-pass loop filter, the above equation becomes

$$H_{nvco}(s) = \frac{\phi_o(s)}{\phi_{nvco}(s)} = \frac{s(s+\omega_{LPF})}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{3.7}$$

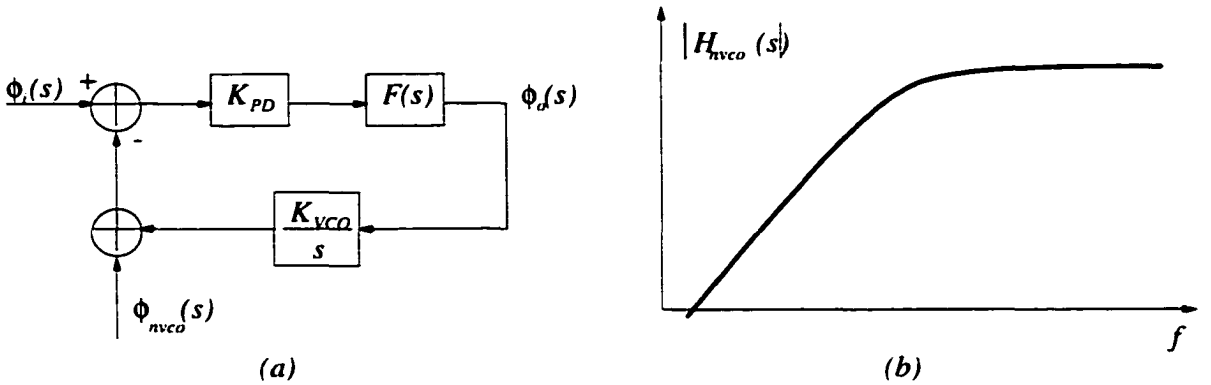


Figure 3.2 Noise transfer function of a PLL from VCO to output (a) model; (b) frequency response

where ω_n and ξ have the same meaning as that in equation 2.8.

This transfer function has the same poles as those of equation 2.7. It also has two zeros $\omega_{z1} = 0$ and $\omega_{z2} = -\omega_{LPF}$, which makes it a *phase high pass* filter, which is shown in Figure 3.2(b). This means that phase noise with slow variation on the VCO can be corrected by the feedback loop and does not reflect into the output. By contrast, if the phase noise on the VCO is at high frequencies, the feedback loop does not have enough time to correct it and the loop becomes transparent to this noise. It directly appears at the output. Therefore, in order to suppress the jitter on the VCO, the loop should be made as fast as possible, i.e. the higher the loop bandwidth, the better the loop can correct the VCO phase error.

3.1.2.2 PLL response to input noise

If there is noise on the input signal, the PLL transfer function on this noise will have the same form as that of Equation 2.7, which is

$$H_n(s) = \frac{\phi_o(s)}{\phi_{ni}(s)} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{3.8}$$

The model and frequency response is shown in Figure 3.3. The response is a *phase low pass* filter, which means that when the noise on the input varies slowly, the PLL output follows it as per its nature. If the rate of the input variation is increased, the PLL will eventually fail to track the input. Therefore, the output does not change with high frequency input jitter. This is an attribute of a PLL as a narrow band filter. The lower the loop bandwidth, the better the jitter suppression on the input.

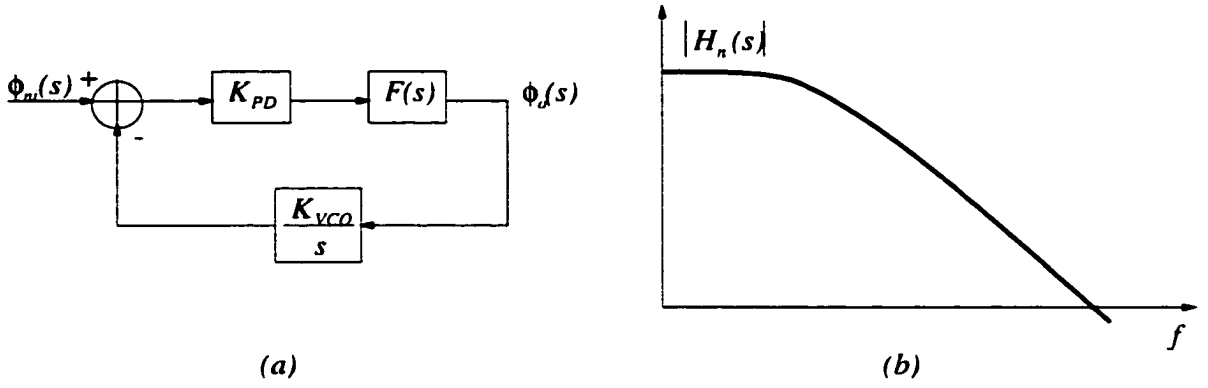


Figure 3.3 Noise transfer function of a PLL from input to output (a) model; (b) frequency response

3.1.2.3 Summary

Combining the above two results, the noise spectral density of the output corresponding to the input noise and VCO noise is

$$\Phi_o(f) = \Phi_{ni}(f)|\ H_n(f)\ |^2 + \Phi_{nvco}(f)|\ H_{nvco}\ |^2 \tag{3.9}$$

where $H_n(f)$ has a low pass response and $H_{nvco}(f)$ has a high pass response. These two characteristics are in conflict with each other. Therefore, a trade off must be made between the two by considering which one is dominant. In applications where the input has negligible noise(e.g. PLL as clock generator or frequency synthesizer, the input is derived from a crystal oscillator), the loop bandwidth should be maximized to reduce both the VCO phase noise and the lock time. On the other hand, in applications where the input is rather noisy (e.g. in clock recovery), the loop bandwidth should be minimized. However, this may make the lock time longer that is undesirable for many applications. As a solution, an additional loop for aided acquisition can be used. This is widely adopted in commercial products from Vitesse [41] AMCC [42] and AMD [43]. In this dissertation, the trade-off on the loop bandwidth in the above two situations where a PLL is used as a low jitter on-chip clock generator and for clock recovery purposes will be shown with real chip design examples.

3.1.3 Noise property of DLL

A similar analysis of the DLL can be carried out by substituting $\frac{K_{VCO}}{s}$ with K_{VCDL} and using $\frac{1}{sC}$ for the loop filter. The resulting equations are first order low-pass and high-pass filter for the input noise and the delay-line noise respectively. However, the case for DLL is somewhat different.

First of all, unlike PLLs, there is no oscillator in a DLL. Even though the delay line still exhibits noise, the noise flushes away at the end of the delay line in the same cycle and will not affect the next cycle. Therefore, there is no need to suppress this noise. Intuitively, a DLL can be seen to have better jitter performance.

Secondly, since a DLL is usually used as a clock generator, the input typically comes from a rather stable source such as a crystal oscillator. As such, it is usually not necessary to lower the loop bandwidth to suppress the input jitter. It is still preferred that the loop bandwidth be high to reduce the lock-in time.

3.2 Jitter Analysis of PLL and DLL

Based on the generic discussion of the noise property of PLLs and DLLs, a rigorous jitter analysis of both systems with discrete-time models will be developed.

3.2.1 Introduction

There are different ways of categorizing jitter. Before pursuing rigorous analysis of jitter in PLL and DLL systems, two definitions commonly used to characterize jitter in short and long term time instants will be introduced first.

3.2.1.1 Cycle-to-cycle jitter

The cycle-to-cycle jitter is defined as the deviation between periods of two adjacent cycles, i.e. variation of period N relative to period $N-1$. For a clock with a nominal period of T_0, random fluctuations in the phase cause a timing error Δt_N to accompany each period of oscillation. For white noise disturbances, the timing error has Gaussian distribution with zero mean, with a variance that is denoted by $\overline{\Delta t_{VCO}{}^2}$. In this case the timing error between one cycle of oscillation and the next is uncorrelated. Cycle-to-cycle jitter measures the short term stability of a periodic signal. Usually it is the smallest jitter that can be specified.

3.2.1.2 Accumulated jitter

The total error variance at time t with respect to an ideal time base is called accumulated jitter, which is denoted by $\overline{\Delta t_{tot}^2}$. It represents the accumulated timing error from time zero to time t. In an oscillator, a perturbation in the phase during one period of oscillation changes the starting point of the next. For Gaussian noise, the cycle-to-cycle jitter adds up and the total error variance grows

linearly as time goes by. The accumulated jitter tends to increase towards infinity for a long enough time. Accumulated jitter measurement indicates the long term stability of a periodic signal.

The illustration of the differences between cycle-to-cycle jitter and accumulated jitter is shown in Figure 3.4.

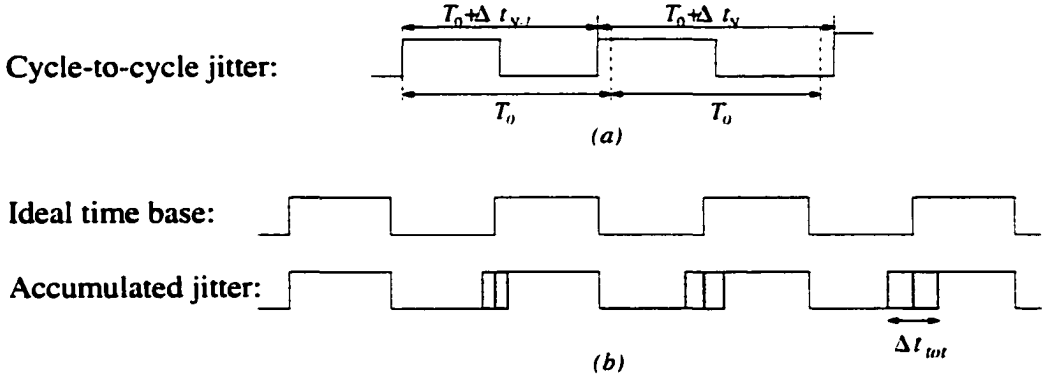


Figure 3.4 Illustration of cycle-to-cycle jitter and accumulated jitter

3.2.2 Jitter analysis of PLL systems

As stated above, the jitter in a free-running oscillator keeps accumulating, which is shown with a dashed line in Figure 3.5. If, however, this oscillator is put into a phase locked loop, the accumulated jitter will not keep increasing but stay at an equilibrium value after a certain time, which is shown with a solid line in Figure 3.5. The time it takes to reach the steady state value is the reciprocal of the PLL loop bandwidth.

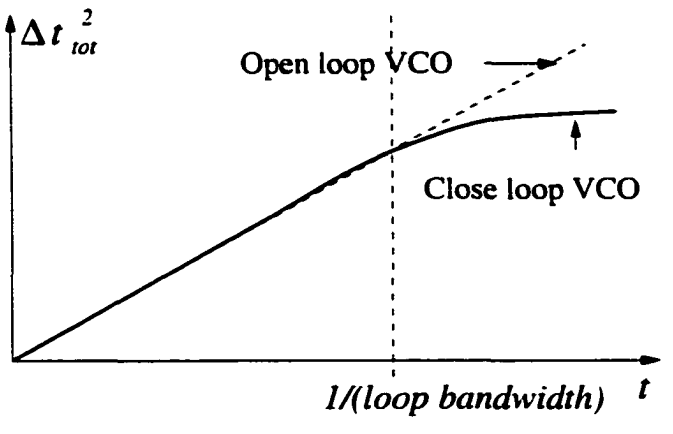


Figure 3.5 Open loop and close loop accumulated output jitter

This can be simply explained as follows: since the basic function of a PLL is to synchronize the VCO output to the reference input, if there is any timing error in the VCO, it tends to correct it with the feedback control. However, there is always a response time in this action. When t is small, the PLL does not have enough time to correct the VCO error, thus the jitter accumulates in the same manner as that in the free running case. When the time reaches the reciprocal of the loop bandwidth, the PLL is able to correct the error. For each new error event added, there is a past event's contribution being corrected. Consequently the accumulated jitter no longer increases but stays at a steady value. The steady value can be expressed with

$$\overline{\Delta t_{tot-pll}^2} = \alpha^2 \overline{\Delta t_{VCO}^2} \tag{3.10}$$

where α represents the jitter accumulation factor. Qualitatively, the higher the loop bandwidth, the smaller the jitter accumulation factor. A rigorous analysis of PLLs will give the quantitative expression between α and PLL loop parameters.

Due to the fact that most modern PLLs are charge pump based, the system is inherently a discrete time system. Unlike traditional s-domain analysis methods, z-domain analysis better fits this system. Some work on discrete-time modeling of PLLs was carried out by Hein [44], Kovacs [45] and Kim [46]. By treating the VCO jitter as a noise event happening at time interval $t = nT$, z-domain analysis and statistical methods provide the closed form expression of the jitter.

By looking at the noise transfer function Equation 3.6 again,

$$H_{nvco}(s) = \frac{\phi_o(s)}{\phi_{nvco}(s)} = 1 + \frac{K_{PD} F(s) K_{VCO}}{s} \tag{3.11}$$

The z-domain expression for F(s)/s will first be developed:

$$\begin{aligned} F(s) &= R(1 + \frac{1}{RCs}) \\ F'(s) &= \frac{F(s)}{s} = R(\frac{1}{s} + \frac{1}{RCs^2}) \\ \Longrightarrow F'(z) &= R(\frac{1}{1-z^{-1}} + \frac{1}{RC}\frac{Tz^{-1}}{(1-z^{-1})^2}) \\ &= R\frac{1 + (TRC-1)z^{-1}}{(1-z^{-1})^2} \end{aligned} \tag{3.12}$$

Therefore,

$$\begin{aligned} \phi_o(z) &= \frac{1}{1 + K_{PD}K_{VCO}R(\frac{1+(T/RC-1)z^{-1}}{(1-z^{-1})^2})} \times \phi_{nvco}(z) \\ &= \frac{(1-z^{-1})^2 \phi_{nvco}(z)}{(1 + K_{PD}K_{VCO}R) + (\frac{K_{PD}K_{VCO}RT}{RC} - K_{PD}K_{VCO}R - 2)z^{-1} + z^{-2}} \end{aligned} \tag{3.13}$$

Usually, $\frac{K_{PD}K_{VCO}RT}{RC} \ll K_{PD}K_{VCO}R \ll 1$. By ignoring $\frac{K_{PD}K_{VCO}RT}{RC}$ and let $\epsilon = K_{PD}K_{VCO}R$, the above equation becomes

$$\phi_o(z) = \frac{(1-z^{-1})^2\phi_{nvco}(z)}{(1+\epsilon)+(-\epsilon-2)z^{-1}+z^{-2}} \tag{3.14}$$

For the VCO jitter, suppose at the nth cycle there is a jitter Δt_n added, then the phase variation is $2\pi\Delta t_n/T$. Consider the fact that this jitter effect exists ever after, the unit step function best describes it. Therefore $\phi_{nvco}(nT) = \frac{2\pi\Delta t_n}{T}u(nT)$. The z-domain expression is $\phi_{nvco}(z) = \frac{2\pi\Delta t_n}{T}\cdot\frac{1}{1-z^{-1}}$. By substituting $\phi_{nvco}(z)$ into Eqn. 3.14, we get the z-domain output jitter expression as:

$$\begin{aligned}\phi_o(z) &= \frac{\frac{2\pi}{T}\Delta t_n(1-z^{-1})}{(1+\epsilon)-(\epsilon+2)z^{-1}+z^{-2}} \\ &= \frac{2\pi\Delta t_n}{T}\frac{1}{(1+\epsilon)-z^{-1}}\end{aligned} \tag{3.15}$$

The corresponding time domain expression is

$$\phi_o(nT) = \frac{2\pi\Delta t_n}{T}(\frac{1}{1+\epsilon})(\frac{1}{1+\epsilon})^n u(nT) \tag{3.16}$$

The accumulated jitter at time nT is

$$\phi_{tot}(nT) = \sum_{k=-\infty}^{n}\frac{2\pi\Delta t_k}{T}(\frac{1}{1+\epsilon})^{n-k}\frac{1}{1+\epsilon} \tag{3.17}$$

The accumulated jitter $\phi_{tot}(nT)$ is a random variable. In order to get its *rms* value, the expectation of $\phi_{tot}(nT)$ is

$$E[\phi_{tot}^2(nT)] = E[(\sum_{k=-\infty}^{n}\frac{2\pi\Delta t_k}{T}(\frac{1}{1+\epsilon})^{n-k}\frac{1}{1+\epsilon})(\sum_{l=-\infty}^{n}\frac{2\pi\Delta t_l}{T}(\frac{1}{1+\epsilon})^{n-l}\frac{1}{1+\epsilon})] \tag{3.18}$$

When $k \neq l$, $E[\Delta t_k \Delta t_l] = 0$ because they are uncorrelated. When $k = l$, $E[(\Delta t_k)^2] = \overline{\Delta t_{VCO}{}^2}$, i.e. the VCO *rms* cycle-to-cycle jitter. Equation 3.18 can thus be further simplified into

$$\begin{aligned}E[\phi_{tot}^2(nT)] &= (\frac{2\pi}{T})^2\overline{\Delta t_{VCO}{}^2}(\sum_{k=0}^{\infty}(\frac{1}{1+\epsilon})^{2k})(\frac{1}{1+\epsilon})^2 \\ &= (\frac{2\pi}{T})^2\overline{\Delta t_{VCO}{}^2}\frac{1}{1-(\frac{1}{1+\epsilon})^2}(\frac{1}{1+\epsilon})^2 \\ &= (\frac{2\pi}{T})^2\overline{\Delta t_{VCO}{}^2}\frac{1}{2\epsilon+\epsilon^2}\end{aligned} \tag{3.19}$$

Since $\epsilon \ll 1$, $\frac{1}{2\epsilon+\epsilon^2} \approx \frac{1}{2\epsilon}$, therefore

$$E[\phi_{tot}^2(nT)] \approx \frac{1}{2\epsilon}\cdot\frac{2\pi}{T}\overline{\Delta t_{VCO}^2} \tag{3.20}$$

The *rms* accumulated output jitter is

$$\sqrt{E[\phi_{tot}^2(nT)]} = \sqrt{\frac{1}{2\epsilon}}\frac{2\pi}{T}\sqrt{\overline{\Delta t_{VCO}^2}} \tag{3.21}$$

Equation 3.21 is the important result of this analysis. It shows quantitatively the jitter accumulation factor of a PLL is

$$\alpha = \sqrt{\frac{1}{2\epsilon}} = \sqrt{\frac{1}{2K_{PD}K_{VCO}R}} = \sqrt{\frac{1}{2\omega_n^2 RC}} = \sqrt{\frac{1}{2RC}}\cdot\frac{1}{\omega_n} \tag{3.22}$$

where ω_n is loop bandwidth, R and C is the loop filter resistor and capacitor.

Equation 3.22 demonstrates that the higher the loop bandwidth, the smaller the jitter accumulation factor. Therefore, in applications where VCO jitter is a dominant factor, high loop bandwidth is desired. This result is consistent with the qualitative discussion in section 3.1.2.

3.2.3 Jitter analysis of DLL systems

Similar analysis can be carried out for a DLL. The quantitative result will provide support that there is no jitter accumulation effect in DLLs.

The DLL model with VCDL noise is shown in Figure 3.6. The noise transfer function of a DLL on the delay line noise is

$$H_n(s) = \frac{\phi_o(s)}{\phi_n(s)} = \frac{1}{1 + K_{PD}K_{VCDL}F(s)} \tag{3.23}$$

Since the loop filter is just a capacitor, $F(s) = 1/sC$. $F(z) = \frac{1}{C}\cdot\frac{1}{1-z^{-1}}$, therefore

$$\phi_o(z) = \frac{1}{1 + K_{PD}K_{VCDL}\frac{1}{C}\frac{1}{1-z^{-1}}}\phi_n(z) = \frac{1 - z^{-1}}{(1 + \frac{K_{PD}K_{VCDL}}{C}) - z^{-1}}\phi_o(z) \tag{3.24}$$

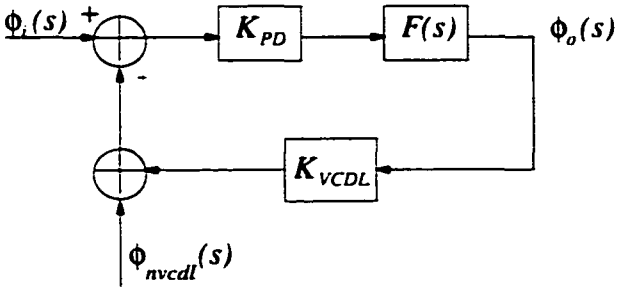


Figure 3.6 DLL noise model

For the VCDL jitter, suppose at the nth cycle there is a jitter Δt_n added, then the phase variation is $2\pi\Delta t_n/T$. Unlike a VCO, this jitter event no longer exists after one clock cycle. Impulse function δ best describes this. Therefore $\phi_n(nT) = \frac{2\pi\Delta t_n}{T}\delta(nT)$, the z-domain expression is $\phi_n(z) = \frac{2\pi\Delta t_n}{T}$. By substituting $\phi_n(z)$ into equation 3.24, the z-domain output jitter expression is:

$$\phi_o(z) = \frac{2\pi\Delta t_n}{T}\cdot\frac{1-z^{-1}}{1+\frac{K_{PD}K_{VCDL}}{C}-z^{-1}} \tag{3.25}$$

Usually $K_{PD}K_{VCDL}/C \ll 1$, let $\epsilon = K_{PD}K_{VCDL}/C$, then

$$\phi_o(z) = \frac{2\pi\Delta t_n}{T}\frac{1-z^{-1}}{1+\epsilon-z^{-1}} \tag{3.26}$$

The time domain expression is

$$\phi_o(nT) = \frac{2\pi\Delta t_n}{T}[\delta(nT) + (\frac{1}{1+\epsilon}-1)(\frac{1}{1+\epsilon})^n u(nT)] \tag{3.27}$$

The accumulated jitter at time nT is

$$\phi_{tot}(nT) = \frac{2\pi\Delta t_n}{T} + \sum_{k=-\infty}^{n}\frac{2\pi\Delta t_k}{T}(\frac{1}{1+\epsilon}-1)(\frac{1}{1+\epsilon})^{n-k} \tag{3.28}$$

The expectation value of the accumulated jitter is

$$\begin{aligned} E[\phi_{tot}^2 nT] &= (\frac{2\pi}{T})^2 E[\Delta t_n^2] + (\frac{2\pi}{T})^2(\frac{1}{1+\epsilon}-1)^2 E[\Delta t_n^2]\sum_{k=0}^{\infty}(\frac{1}{1+\epsilon})^{2k} + \\ &\quad 2(\frac{2\pi}{T})^2 E[\Delta t_n^2](\frac{1}{1+\epsilon}-1) \\ &= (\frac{2\pi}{T})^2\overline{\Delta t_n^2}[1+(\frac{-\epsilon}{1+\epsilon})^2\cdot\frac{1}{1-(\frac{1}{1+\epsilon})^2}+2(\frac{1}{1+\epsilon}-1)] \\ &= (\frac{2\pi}{T})^2\overline{\Delta t_n^2}\frac{2-\epsilon^2}{2+3\epsilon+\epsilon^2} \end{aligned} \tag{3.29}$$

Since $\epsilon \ll 1$, $\frac{2-\epsilon^2}{2+3\epsilon+\epsilon^2} \approx \frac{2}{2} = 1$, therefore the *rms* accumulated jitter of a DLL is

$$\sqrt{E[\phi_{tot}^2(nT)}} \approx 1 \times \frac{2\pi}{T}\sqrt{\overline{\Delta t_n^2}} \tag{3.30}$$

Equation 3.30 clearly shows that the jitter accumulation factor of a DLL is 1. There is no accumulation effect in DLLs. Therefore the jitter performance of DLLs is usually better than that of PLLs in this sense.

3.3 Jitter Measurement

To accurately measure the jitter of a signal is an important issue in modern IC testing, especially when the jitter is in the pico-second range. However, unlike circuit design, this topic is less frequently

addressed in publications. To better understand the testing results in this dissertation, it is necessary to briefly discuss this issue here. The goal is to emphasize significant issues that cannot be neglected as well as to provide a background knowledge on how jitter measurement can be done.

Jitter can be measured directly or indirectly. The choice of methods usually depends on the application. As stated in Chapter 2, jitter and phase noise are two concepts characterizing the same phenomenon in different domains: time domain and frequency domain. The measurement methods can therefore be divided into these two domains.

3.3.1 Frequency domain measurement

This type of measurement quantifies the phase noise from the spectrum of the output signal. The equipment is a spectrum analyzer or is based on a spectrum analyzer. Note that in order to get reliable results, a high quality spectrum analyzer should be used that has a low enough noise floor.

As discussed in chapter 2, the high frequency phase variation causes the signal spectrum to spread around the center frequency. For reasonable resolution bandwidth, the phase noise is measured by the difference in the heights of the spectrum at the center f_c and at a given frequency offset $f_c + \Delta f$, which was shown in Figure 2.11. It is usually characterized by how many dB the spectrum is lower than the center at a given offset, e.g. "-100dBc/Hz at offset of 100KHz".

The relationship between phase noise and cycle-to-cycle jitter [47] [48] is:

$$\sigma_{T_0}^2 = \frac{\Delta f^2}{f_c^3} . S_\phi(\Delta f) \tag{3.31}$$

where $\sigma_{T_0}^2$ is the variance of the cycle, i.e. cycle-to-cycle jitter and $S_\phi(\Delta f)$ is the phase noise at a given frequency offset ΔfHz. With this equation, the cycle-to-cycle jitter can be obtained as long as the phase noise has been measured. However, this equation is based on the assumption that the only noise source is white noise. As discussed in Chapter 2, Equation 2.19, when the offset is at the low or high ends, the noise source is generally dominated by noise sources other than white noise. Therefore, when applying this equation, care must be taken such that the measured phase noise to be used should be in the region where white noise dominates. This region is characterized by $S_\phi(\Delta f)$ begin linearly proportional to $1/\Delta f^2$ on a log-log scale.

Spectrum analysis is easy to do and can be used to obtain rather accurate phase noise measurement. However, phase noise measurement does not directly address the issue of accumulated phase error. The spectrum only tells what frequencies, other than f_c, the signal visits but not how it visits. This matters because if the signal lingers at a certain frequency f_m too long from its center frequency, it

can accumulate a big phase error. In contrast, if the signal deviates quickly back and forth about its center frequency, it may also visit the same frequency f_m, but each time is so brief that it accumulates almost no phase error. These are two quite different situations, although the spectrums can be the same. Therefore, in order to get the time domain jitter information, direct methods in the time domain should be applied.

3.3.2 Time domain measurement

Since jitter is a statistical characteristic of a signal, *peak-to-peak* and *rms* values are usually used to characterize it when measuring it directly.

High speed sampling oscilloscopes are commonly used to measure time jitters. For this thesis, an alternative equipment, $Wavecrest^{TM}$ DTS-2075 time interval analyzer, was also available. Regardless of what equipment is used, and supposing that the signal jitter and equipment noise are uncorrelated, one general formula showing the relationship between the measured result and the real signal jitter is [49]:

$$signal\ jitter = \sqrt{measured\ jitter^2 - equipment\ noise^2} \tag{3.32}$$

3.3.2.1 Time domain measurement with high speed sampling oscilloscope: communication analyzer

The principle of this equipment is to capture the waveform and do statistics on the waveform. High input bandwidth (larger than 20GHz) is required to guarantee genuine waveform capturing. It obtains the jitter histogram by sampling the waveform for a rather long time. Like all sampling oscilloscopes, it needs a trigger source. The quality of the trigger source greatly affects the result. Based on the trigger used, there are two approaches that can be adopted with this equipment.

If a rather clean reference clock synchronized to the signal to be measured is available, it can be used as the ideal time base. By using it as the triggering source, the relative phase relationship of the signal and this base (i.e. accumulated jitter) is directly measured. In this case, the jitter of this reference clock should also be deducted according to Equation 3.32 when calculating the real signal jitter.

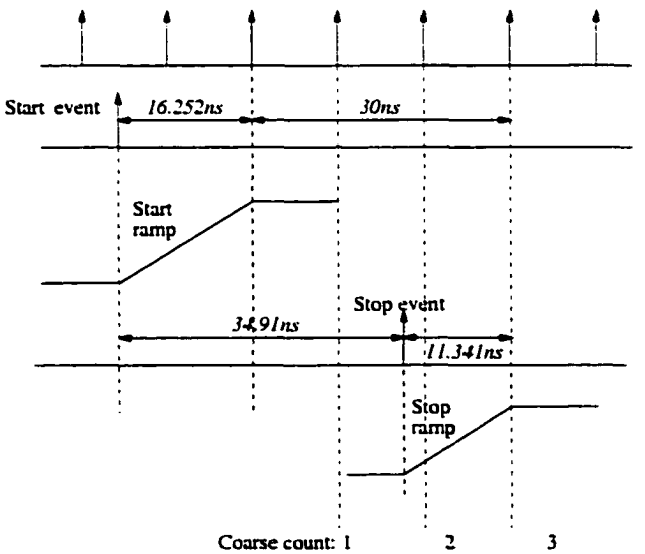
This testing set up brings an extra advantage when testing PLL circuit. If the reference clock input to the PLL also acts as the triggering source, it can tell whether or not the PLL is locked. If a stable output waveform cannot be obtained, it will show that the PLL is not synchronizing to the input, therefore it is not locked.

If, however, this stable reference clock does not exist, a method called *differential phase measurement* can be used. In this method, the jittery signal is not compared with an ideal reference but with a delayed version of itself. In this case, the triggering source used is the signal itself, or it is self-triggered. At a large enough delay, the delayed waveform is uncorrelated with the original. The resulting jitter is twice the actual jitter, thus the term *differential* measurement. This measurement requires an oscilloscope with an accurate delayed time base sweep feature. The delay cycle should be set to be in the hundreds, thousands, or tens of thousands. The resulting waveform can be used to construct the jitter histogram.

The application of a communication analyzer in direct jitter measurement is widely accepted in industry as well as in academia. Therefore, by using this equipment, it is possible to compare our work with the achievements in prior art. The communication analyzer used in this work is HP83480A. Its input bandwidth is 20GHz. Its noise floor is roughly $1ps$ [49].

3.3.2.2 Time domain measurement with $Wavecrest^{TM}$ DTS-2075

Unlike the communication analyzer, the principle of DTS-2075 is different. It is not based on capturing the waveform. Instead, it directly measures time intervals based on an accurate internal time base. This method can be simply explained in Figure 3.7:



Start ramp begins at start event and ends at the second time base edge after start event.
Stop ramp begins at stop event and ends at the second time base edge after stop event
Height of each ramp is digitized using a 14-bit ADC.
Coarse counter counts the number of time base periods between ends of start and stop event.

Figure 3.7 DTS-2075 time interval measurement illustration

The time interval between the start and stop event is calculated as:

$$N_{coarsecount} \times 10ns + T_{startramp} - T_{stopramp} = T_{measuredinterval} \tag{3.33}$$

Example: $3 \times 10.000ns + 16.252ns - 11.341ns = 34.911ns$.

Compared with a communication analyzer, DTS-2075 does not require external triggering. Therefore, the result does not rely on the quality of the triggering source. It directly gives out the statistical result in a rather short testing time. Another advantage of this equipment over a communication analyzer is that it can directly measure the delay of two signals. This feature is important to the testing in Chapter 7. However, it has some disadvantages. First, it cannot measure the accumulated jitter. Second, it cannot determine whether the phase lock loop is working. Third, the input bandwidth is not as high as that of a communication analyzer. Fourth its noise floor is a little bit higher. After calibration, the noise floor is at $3.4ps$ [50].

Since the instruments have both advantages and disadvantages, they are used jointly in this dissertation to make measurement more accurate.

As a summary of this section, different jitter measurement methods are compared in Table 3.1.

Table 3.1 Comparison of jitter measurement methods

Method	Frequency domain	Time domain	
Basic equipment	spectrum analyzer	oscilloscope	DTS-2075
Pros	accurate phase noise measurement	direct accumulated jitter and cycle-to-cycle jitter measurement	direct time interval measurement, direct delay measurement of two signals
Cons	indirect	quality of triggering source is important	low input bandwidth, higher noise floor
Application	RF	data communication	

3.4 Summary

Low noise performance is the major issue in contemporary monolithic PLL and DLL design. In this chapter, noise properties of both PLL and DLL systems are addressed qualitatively and quantitatively. Rigorous jitter analysis based on discrete-time model as well as statistical method is carried out of both systems. The goal is to provide theoretical knowledge on how to achieve low jitter performance.

At the end of this chapter, jitter measurement methods are discussed with their advantages and disadvantages. The goal is to understand the potential and limitations of each method, which is

important for a successful design, and to explain the testing results in later chapters.

CHAPTER 4 CASCADED PLL-DLL FOR FREQUENCY SYNTHESIS WITH JITTER SUPPRESSION

Based on the discussion in previous chapters, the merits as well as the drawbacks of a PLL and DLL are clear. In brief, with an oscillator in the loop, the PLL can *create* an output frequency other than the reference frequency, which is the reason why the PLL is widely used as frequency synthesizer and clock recovery circuit. With only a delay line, this is not possible for a DLL. On the other hand, also due to the difference between an oscillator and a delay line, the jitter performance of a PLL is usually worse than that of a DLL.

In this chapter, an architecture that combines the advantages of both PLL and DLL is proposed to achieve better jitter performance. This architecture is able to perform frequency synthesis with a jitter performance comparable with that of a DLL.

In the following sections, this new architecture will first be introduced. Theoretical analysis of the noise performance of such a system will be carried out to provide the design guideline of such system.

4.1 Architecture

Let's start by discussing two ideas to improve the performance of a PLL and a DLL.

The first idea is to use DLL to do frequency synthesis with its inherent low noise performance. Some work already has been reported on this topic. See [55] [56] [57]. The basic concept is to make use of the uniformly distributed multi-phase delay line outputs to construct a periodical signal whose frequency is an integer multiple of the reference frequency. Strictly speaking, this is not frequency synthesis but only frequency multiplication. Due to the stage mismatch, the constructed signal will have a fixed duty cycle variation, which introduces distortions in the signal spectrum. Therefore, this type of DLL can only be used in certain applications where the distortion is not important or can be filtered out. In conclusion, a delay line cannot perform the same as an oscillator. Further detail about this idea can be found in [55].

As an alternative to the above idea, another question to be asked is: can the PLL (which is good at frequency synthesis) achieve the same level of jitter performance as the DLL does?

By reviewing the noise property of both systems as shown in Chapter 3, the noise transfer function of the PLL is:

$$\phi_{onpll} = \frac{2\pi\Delta t_n}{T} \cdot \frac{1}{1+\epsilon} \cdot \frac{1}{1-\frac{1}{1+\epsilon}z^{-1}} \tag{4.1}$$

while the noise transfer function of the DLL is:

$$\phi_{ondll} = \frac{2\pi\Delta t_n}{T} \cdot \frac{1}{1+\epsilon} \cdot \frac{1-z^{-1}}{1-\frac{1}{1+\epsilon}z^{-1}} \tag{4.2}$$

By comparing both equations,

$$\phi_{ondll} = \phi_{onpll} \cdot (1-z^{-1}) \tag{4.3}$$

If a new PLL can be created with the same noise transfer function as that of the DLL (Equation 4.2), its noise performance will be the same as the DLL. It is clear this new PLL must have the following relationship with the traditional PLL:

$$\phi_{onpllnew} = \phi_{onpll}(1-z^{-1}) \tag{4.4}$$

Equation 4.4 indicates that the new PLL should perform $\phi_{onpll}(nT) - \phi_{onpll}((n-1)T)$ in the time domain. Since the parameter ϕ effectively represents the excessive phase, i.e. jitter, $\phi_{onpll}(nT) - \phi_{onpll}((n-1)T)$ implies that the jitter accumulation effect needs to be removed, which is impossible for a conventional PLL. Therefore, new methods need to be explored to improve the jitter performance of a PLL.

If the PLL itself is kept untouched but its output is passed through a time averaging circuit, the output jitter can be reduced. The question lies in how to implement such a time averaging circuit. Besides performing the averaging function, the extra noise introduced by the additional circuit should be as small as possible.

Recall that a DLL is a phase low pass filter, in the time domain it acts exactly like a time averaging circuit. The degree of the averaging function depends on its bandwidth. Moreover, a DLL only adds a small amount of extra noise to the output since it generates low level noise by itself. Therefore a DLL can be used to filter the PLL output noise. Since the PLL has a high-pass transfer function for VCO noise, and the DLL has a low pass transfer function for input noise and no jitter accumulation, the PLL and DLL can be cascaded to lower the overall jitter accumulation factor. Figure 4.1 illustrates the proposed architecture.

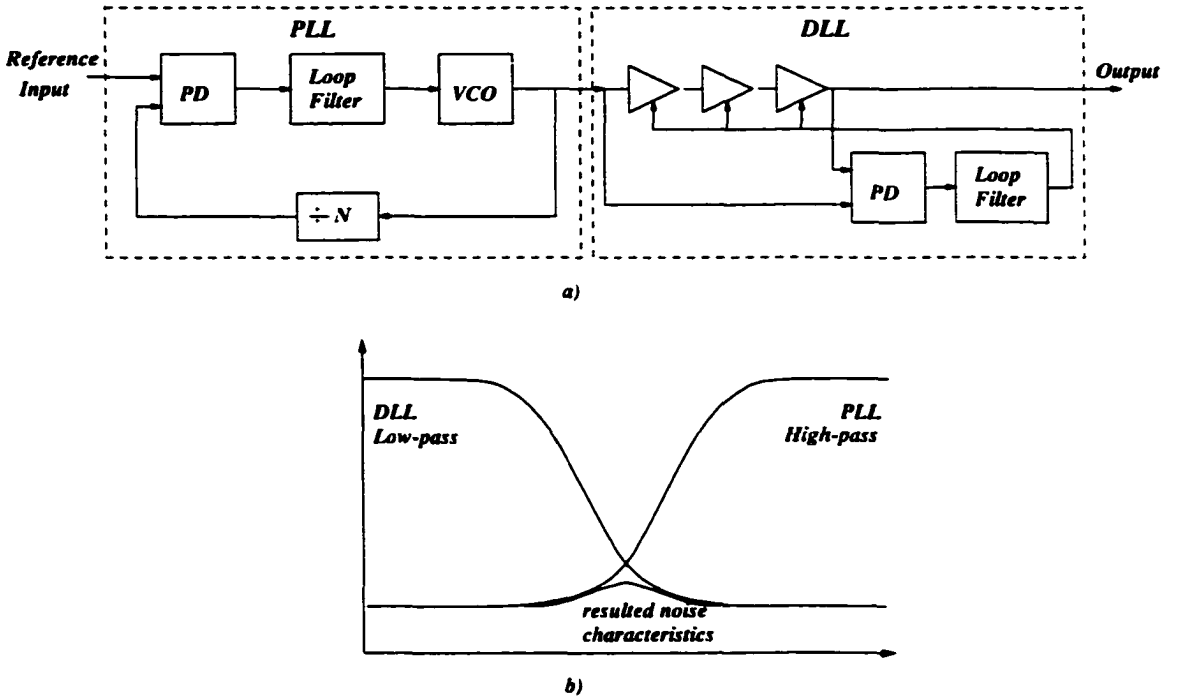


Figure 4.1 Cascaded PLL/DLL with jitter suppression (a) Block diagram: (b) Jitter suppression principle

The noisy output from the PLL is the input to the DLL, where its jitter can be filtered out. Figure 4.1(b) is the noise transfer function that demonstrates qualitatively why this system has a jitter suppression mechanism. Intuitively, when the DLL bandwidth is lower than the bandwidth of the PLL, this system can suppress the PLL output noise. The lower the DLL bandwidth, the better the jitter reduction effect. In real applications, proper choice of the bandwidth of the PLL and the DLL can be made according to the system requirements. The final output noise is the filtered PLL noise plus the delay line noise.

4.2 Noise Suppression Analysis

4.2.1 Analysis

The model to analyze the proposed architecture is shown as follows in Figure 4.2. In order to get the noise transfer function, let the input jitter $\phi_i = 0$, then

$$\begin{cases} -\phi_{pll} K_{PDpll} K_{VCO} \frac{Z_{Fpll}(s)}{s} \frac{1}{N} + \phi_{nVCO} = \phi_{pll} \\ (\phi_{pll} - \phi_{out}) K_{PDdll} K_{VCD} Z_{Fdll}(s) + \phi_{nvcdl} = \phi_{out} \end{cases}$$

From the above two equations, the final output

$$\phi_{out} = \frac{K_{PDdll} K_{VCDL} Z_{Fdll}(s) \phi_{nVCO} + (1 + \frac{K_{PDpll} K_{VCO}}{N} \frac{Z_{Fpll}(s)}{s}) \phi_{nVCDL}}{(1 + \frac{K_{PDpll} K_{VCO}}{N} \frac{Z_{Fpll}(s)}{s})(1 + K_{PDdll} K_{VCDL} Z_{Fdll}(s))} \tag{4.5}$$

Therefore,

$$\frac{\phi_{out}}{\phi_{nVCO}} \Big|_{\phi_{nVCDL}=0} = \frac{K_{PDdll} K_{VCDL} Z_{Fdll}(s)}{(1 + \frac{K_{PDpll} K_{VCO}}{N} \frac{Z_{Fpll}(s)}{s})(1 + K_{PDdll} K_{VCDL} Z_{Fdll}(s))} \tag{4.6}$$

$$= H_{plln}(s) H_{dlln}(s) \tag{4.7}$$

$$\frac{\phi_{out}}{\phi_{nVCDL}} \Big|_{\phi_{nVCO}=0} = \frac{1}{1 + K_{PDdll} K_{VCDL} Z_{Fdll}(s)} \tag{4.8}$$

Equation 4.8 shows that for DLL delay line noise, this structure behaves the same as a general DLL. It does not have accumulation effect. Therefore, it only adds a small portion of noise on the final output. What needs to be analyzed is equation 4.6.

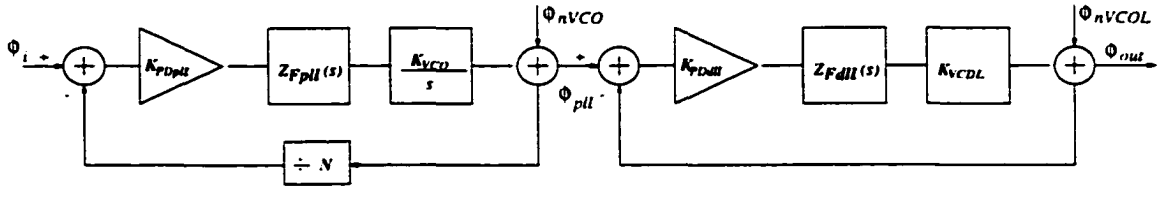


Figure 4.2 Cascaded PLL/DLL noise analysis model

The z-domain model on Equation 4.7 is

$$\frac{\phi_{out}(z)}{\phi_{nVCO}(z)} \Big|_{\phi_{nvcdl}=0} = H_{plln}(z) H_{dlln}(z) \tag{4.9}$$

where

$$H_{plln}(z) = \frac{(1 - z^{-1})^2}{(1 + \frac{K_{PDpll} K_{VCO} R}{N}) + [\frac{K_{PDpll} K_{VCO} R}{N}(\frac{T}{RC} - 1) - 2] z^{-1} + z^{-2}} \tag{4.10}$$

$$H_{dlln}(z) = \frac{1 - z^{-1}}{(1 - z^{-1}) + \frac{K_{PDdll} K_{VCDL}}{C}} \tag{4.11}$$

$$\tag{4.12}$$

Since usually $\frac{T}{RC} \ll 1$, neglect this item in Equation 4.10 and let $K_{pll} = \frac{K_{PDpll} K_{VCO} R}{N}$ (it is PLL loop gain and is also usually $\ll 1$) . Equation 4.10 becomes

$$H_{plln}(z) = \frac{(1 - z^{-1})^2}{(1 + K_{pll}) + (-K_{pll} - 2) z^{-1} + z^{-2}} \tag{4.13}$$

In Equation 4.11, let the DLL loop gain $K_{dll} = \frac{K_{PDdll}K_{VCDL}}{C}$, it becomes

$$H_{dlln}(z) = \frac{K_{dll}}{(1+z^{-1}) + K_{dll}} \tag{4.14}$$

Therefore,

$$\begin{aligned}
\phi_{out}(z) &= \frac{(1-z^{-1})^2}{(1+K_{pll}) + (-K_{pll}-2)z^{-1} + z^{-2}} \cdot \frac{K_{dll}}{(1+z^{-1}) + K_{dll}} \cdot \frac{2\pi\Delta t_n}{T} \cdot \frac{1}{1-z^{-1}} \\
&= \frac{2\pi\Delta t_n}{T} \cdot \frac{1}{(1+K_{pll}) - z^{-1}} \cdot \frac{K_{dll}}{(1+K_{dll}) - z^{-1}} \\
&= \frac{2\pi\Delta t_n}{T} \cdot \frac{K_{dll}}{K_{dll} - K_{pll}} [\frac{1}{1+K_{pll}} \cdot \frac{1}{1 - \frac{1}{1+K_{pll}} z^{-1}} - \frac{1}{1+K_{dll}} \cdot \frac{1}{1 - \frac{1}{1+K_{dll}} z^{-1}}]
\end{aligned} \tag{4.15}$$

The corresponding discrete time domain expression is

$$\phi_{out}(nT) = \frac{2\pi\Delta t_n}{T} \cdot \frac{K_{dll}}{K_{dll} - K_{pll}} [\frac{1}{1+K_{pll}} \cdot (\frac{1}{1+K_{pll}})^n u(nT) - \frac{1}{1+K_{dll}} \cdot (\frac{1}{1+K_{dll}})^n u(nT)] \tag{4.16}$$

The accumulated jitter

$$\begin{aligned}
\phi_{tot}(nT) &= \sum_{k=-\infty}^{n} \phi_{out}(kT) \\
&= \frac{2\pi}{T} \frac{K_{dll}}{K_{dll} - K_{pll}} \sum_{k=0}^{\infty} \Delta t_k [\frac{1}{1+K_{pll}} (\frac{1}{1+K_{pll}})^k - \frac{1}{1+K_{dll}} (\frac{1}{1+K_{dll}})^k]
\end{aligned} \tag{4.17}$$

Finally, the statistical expectation value of the accumulated jitter is

$$\begin{aligned}
E[\phi_{tot}^2(nT)] &= (\frac{K_{dll}}{K_{dll} - K_{pll}})^2 (\frac{2\pi}{T})^2 \overline{\Delta t_n^2} \\
&\cdot [\frac{1}{(1+K_{pll})^2 - 1} - \frac{2}{(1+K_{pll})(1+K_{dll}) - 1} + \frac{1}{(1+K_{dll})^2 - 1}]
\end{aligned} \tag{4.18}$$

Therefore, the jitter accumulation factor of the cascade system is

$$\alpha^2 = (\frac{K_{dll}}{K_{dll} - K_{pll}})^2 [\frac{1}{(1+K_{pll})^2 - 1} - \frac{2}{(1+K_{pll})(1+K_{dll}) - 1} + \frac{1}{(1+K_{dll})^2 - 1}] \tag{4.19}$$

The above equation is the important result of the jitter performance of the proposed system. It shows clearly that the jitter accumulation factor is a function of both loop gains of the PLL and DLL. By appropriately choosing the loop gain of both sub-systems, the jitter accumulation factor can be reduced. Shown in Figure 4.3 is the plot of the accumulation factor α as a function of K_{pll} and K_{dll}.

There are five curves in Figure 4.3, corresponding to five different K_{pll} values respectively. Several observations can be made from this figure:

Observation 1:

When K_{pll} is increasing from 10^{-5} to 10^{-1}, the jitter accumulation factor α decreases. That is, the bigger the PLL loop gain K_{pll}, the smaller the jitter accumulation factor, the better the jitter performance. This is consistent with that of a stand alone PLL .

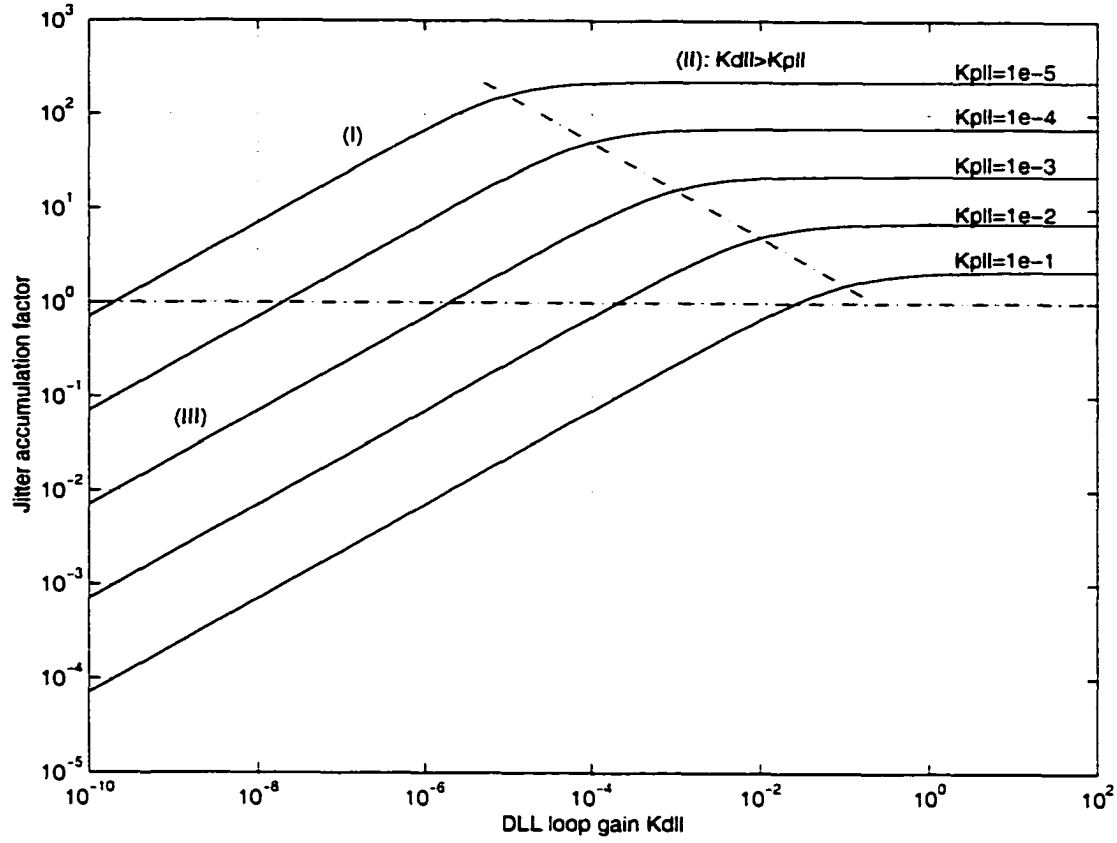


Figure 4.3 Jitter accumulation factor α as a function of K_{pll} and K_{dll}

Observation 2:

In region (II), when the DLL loop gain is bigger than the loop gain of the PLL, the jitter suppression function of the DLL on the PLL is not effective. Therefore, the accumulation factor α remains the same as that of a PLL. If the proposed system is used, region (II) should be avoided.

Observation 3:

In region (III), the jitter accumulation factor is $\alpha < 1$, which is not possible in reality. The lower limit of the accumulated jitter is set by the cycle-to-cycle jitter, which is usually the smallest jitter that is specified for a periodical signal and therefore cannot be further reduced. Thus the minimum value of α is 1. In summary, though theoretical analysis can reach region (III), it is not valid in reality. Therefore, it is meaningless to further reduce the DLL bandwidth after α reaches 1.

Observation 4:

In region (I), the DLL is active in filtering the PLL output noise. The lower the DLL loop gain, the better the final output jitter. With a PLL loop gain of K_{pll}, the loop gain of the DLL that makes the

cascaded system work in region (I) is:

$$\frac{-3K_{pll}^3 - 8K_{pll}^2 - 3K_{pll} + 2 - \sqrt{K_{pll}^6 + 8K_{pll}^5 + 26K_{pll}^4 + 28K_{pll}^3 - 7K_{pll}^2 - 12K_{pll} + 4}}{2(K_{pll}^3 + 3K_{pll}^2 + K_{pll} - 1)} < K_{dll} < K_{pll} \qquad (4.20)$$

The above expression is obtained by setting the jitter accumulation factor=1 (i.e. Eqn. 4.19=1) and solving for K_{dll}.

A plot showing $\alpha(K_{pll}, K_{dll})$ in 3D form is presented in Figure 4.4. The plane in the middle is where $\alpha = 1$.

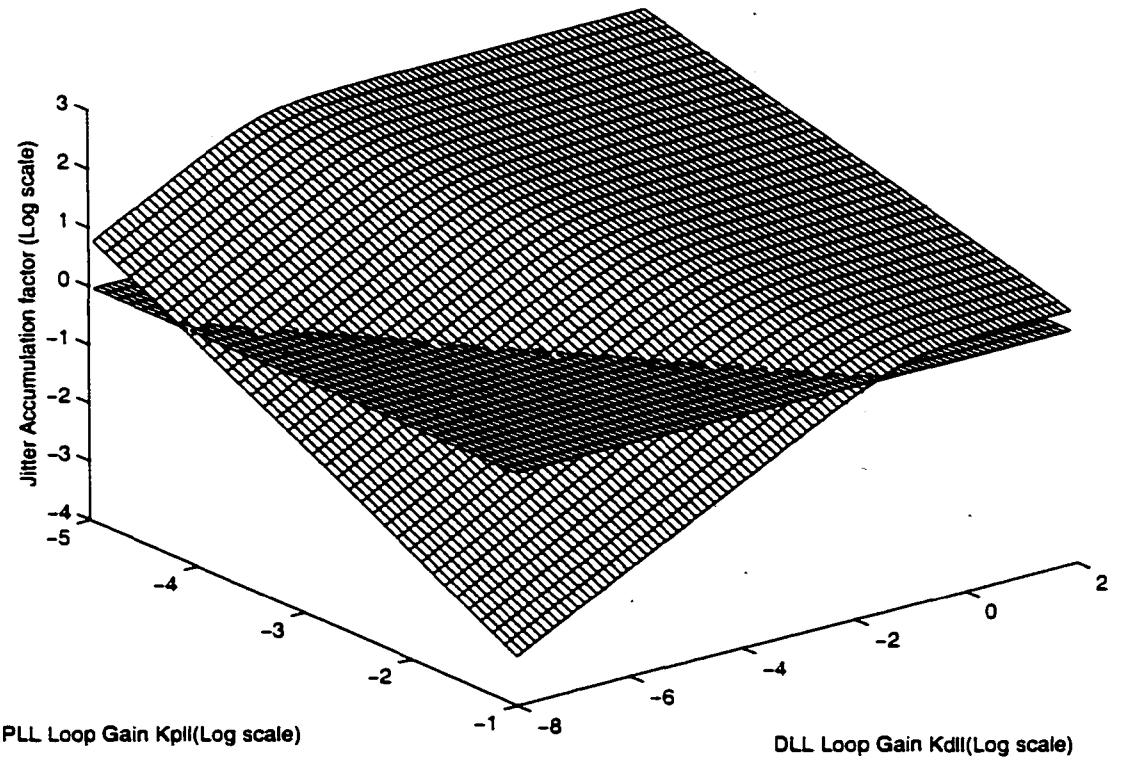


Figure 4.4 3D plot of jitter accumulation factor α as a function of K_{pll} and K_{dll}

4.2.2 Design implication

If the cascade PLL/DLL system is used for frequency multiplication or frequency synthesis, proper design strategy needs to be followed to achieve good overall performance.

Design implication 1:

In this type of application, the reference input to the PLL is coming from a low noise signal generator or crystal. Its jitter is rather small. The jitter of the oscillator inside the PLL is the dominant jitter source. Therefore, K_{pll} should be chosen to be as big as possible to suppress the oscillator jitter, without jeopardizing the loop stability.

Design implication 2:

After K_{pll} is chosen, select the DLL loop gain K_{dll} such that it satisfies Eqn. 4.20 to make the system working in region (I). Within this region, the smaller the K_{dll}, the better the jitter suppression function. However, too small a K_{dll} will make the dynamic tracking time very long. Therefore, a trade-off needs to be made between the jitter suppression and the dynamic performance.

The above analysis is focused on reducing the jitter accumulation factor, which is targeting the oscillator noise. It is worth noting that for the PLL input reference noise, the DLL still performs the jitter filtering function. Therefore this structure can still reduce the output noise. Hence, this structure can suppress any noise appearing in the output of the PLL.

4.3 Summary

A structure which combines the merit of both a PLL and DLL is proposed. This cascade PLL/DLL structure is able to do frequency synthesis with jitter performance comparable with that of a DLL. The final output jitter is

$$\sigma_{out} = \sqrt{\sigma_{sys}^2 + \sigma_{dll}^2} \tag{4.21}$$

where σ_{out} is the total final output jitter, σ_{sys} is the PLL output jitter filtered by this structure (which has been analyzed), and σ_{dll} is the additional jitter introduced by the DLL block. Ideally, suppose that accumulation factor is reduced to one and suppose that cycle-to-cycle jitter of the oscillator and the delay line are at the same level, then the best output jitter would be:

$$\sigma_{out} = \sqrt{2} \cdot \textit{cycle-to-cycle jitter} \approx 1.4 \cdot \textit{cycle-to-cycle jitter} \tag{4.22}$$

CHAPTER 5 NONLINEAR BEHAVIORAL MODELING AND JITTER SIMULATION OF PLL/DLL SYSTEMS

In the top-down design flow of a high performance PLL or DLL, in addition to the system design techniques discussed in the previous chapters, effective simulation tools are also in demand, especially those for transient noise simulation.

This chapter presents a new method for modeling VCO and Voltage Controlled Delay Line (VCDL) circuits that allows inclusion of device noise and supply coupling effects with simplified numerical computation [51]. The resulted PLL and DLL behavioral simulation enables an accurate prediction of system performance during both locked and unlocked conditions with a great reduction in CPU computation time over transistor level simulators. Simulation results are presented and compared with theoretical predictions as well as measurement results that demonstrate the effectiveness of this scheme. The simulation technique discussed in this chapter is part of the low jitter design techniques that compose this dissertation [51].

5.1 Introduction

As previously mentioned, fully monolithic PLLs and DLLs are widely used in data communication, digital I/O interfaces and RF applications. In many of these circuits, the jitter behavior of the PLL and DLL is the performance-limiting factor in the system. Hence, it is important to understand the sources of this jitter and predict the performance of the system. Since noise coupling effects may dominate actual jitter performance and significantly affect the lock-in behavior, it is imperative that these influences be properly modeled in PLL and DLL simulations.

Traditional transistor level simulators, such as SPICE, require significant CPU time to simulate the PLL and DLL transient behavior. Furthermore, time-domain noise simulation in these simulators is very difficult to implement and intolerably slow. Behavioral level simulation thus becomes an attractive way to simulate PLL/DLL systems with a substantial reduction in CPU time. Nonetheless, in order to uncover potential design problems, the behavioral model should reflect the real circuit behavior as

much as possible, including realistic nonlinear effects, noise coupling, and so on during both tracking and locking conditions. As representative works in this area, [52] and [53] proposed a behavioral simulation technique for PLL and DLL systems. However, jitter simulations with this model can only be carried out based on the assumption that the PLL or DLL is in-lock, particularly in regards to DLLs.

In this work, a new behavioral simulator for PLLs as well as DLLs that overcomes the shortcomings in the prior art was developed. This simulator allows system acquisition behavior as well as in-lock jitter performance to be evaluated in the presence of noise. Both simulators can be used in the early design stages as well as in the final design verification by applying appropriate approximations at different stages. Section 5.2 introduces the new modeling technique of VCO and VCDL. PLL and DLL behavioral models for noise analysis are discussed in section 5.3. In section 5.4, simulation results are presented and compared with theoretical predictions as well as measurement results. The summary of this modeling of work is given in section 5.5.

5.2 Nonlinear Behavioral Modeling For VCO And VCDL

To a large extent, good behavioral simulation of PLL and DLL circuits relies on a good behavioral model of the VCO and VCDL. The well-known mathematical model for a VCO is:

$$\phi(t) = \int_{-\infty}^{t} f(\tau)d\tau, \ \ where\ f(t) = g(V_{ctrl}(t)) \tag{5.1}$$

This equation shows that the phase $\phi(t)$ is the integration of frequency $f(t)$ over time, and the frequency f is controlled by a control voltage V_{ctrl}. The difficulty of modeling the VCO in this method lies in the fact that the output phase cannot follow the instantaneous change of the control voltage due to the integration. In [52], a somewhat complicated numerical procedure was established to model this lag in the VCO output. Moreover, other protocols were needed to synchronize the analog simulator and digital simulator when applying that VCO model in a PLL simulation [53]. Nonetheless, the VCDL behavioral model is not discussed in this approach.

Unlike use of the traditional mathematical relationship described above, a different method that directly constructs the output waveform is adopted. In the following discussion, an RC model for a single delay cell is first established. Based upon this model, both VCO and VCDL models can be easily derived. The consequent VCDL model makes the thorough behavioral simulation of a DLL system possible, which was not achieved in the prior art.

5.2.1 Delay cell model

A typical fully differential delay cell is shown in Figure 5.1 (a). The physical mechanism which determines the delay time in a delay cell is due to the non-zero charging and discharging time of various capacitors within the circuit across a certain voltage swing.

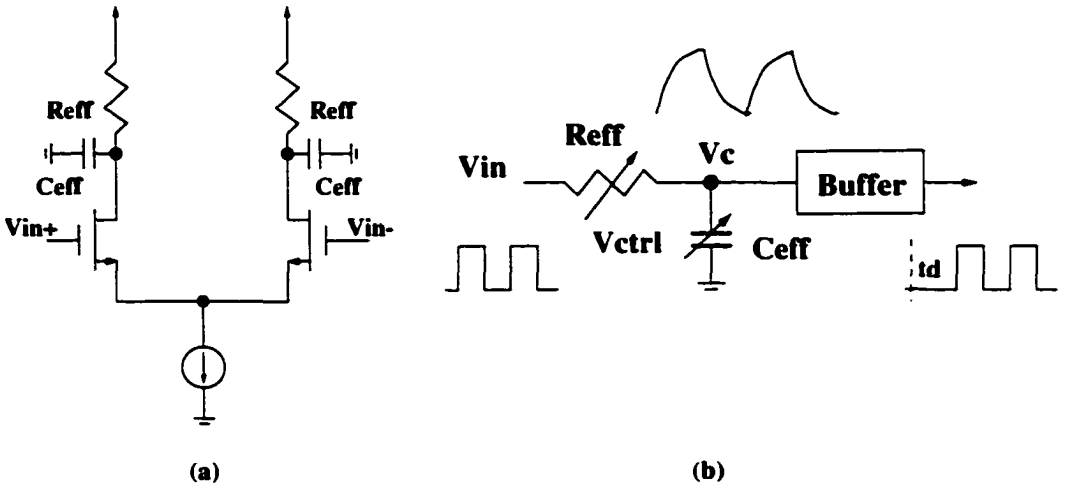


Figure 5.1 (a) Delay cell; (b) Delay cell model

The simplest way to model such a mechanism is to assume an RC network where charging and discharging time is determined by passive component values. By assigning R_{eff} to be the equivalent load resistance of the cell and C_{eff} to be the total capacitance at the output node, the charging and discharging time is then defined by the time constant τ, with:

$$\tau = \frac{V_{SW}}{slew\ rate} = \frac{V_{SW}}{I_{ss}/C_{eff}} = R_{eff}C_{eff} \tag{5.2}$$

where V_{SW} is the voltage swing at the output node, and the equivalent resistance $R_{eff} = \frac{V_{SW}}{I_{ss}}$. Usually, the delay is adjusted by changing V_{SW} or tail current I_{ss}. From this observation, a very straight forward model for a delay cell is shown in Figure 5.1(b), where the delay cell is represented by a simple RC circuit. If the output is a "0"and "1" logic clock signal, a buffer can be added to reshape V_c with an appropriate nonlinear characteristic. The static function between the control voltage V_{ctrl} and the cell delay t_d can be modeled as a linear relationship:

$$t_d = R_{eff}C_{eff} = KV_{ctrl}^{-1} \tag{5.3}$$

during early design stages. After transistor level circuit has been designed, a more accurate relationship can be extracted from SPICE simulation.

With this model, the equation that describes the delay cell becomes Eqn. 5.4, where the state variable is now V_c,

$$R_{eff}(V_{ctrl}(t))C_{eff}(V_{ctrl}(t)) \times \frac{dV_c(t)}{dt} + V_c(t) = V_{in}(t) \tag{5.4}$$

The advantage of using V_c instead of f is that the difficulty in modeling the lag of output phase with respect to instantaneous control voltage $V_{ctrl}(t)$ change is avoided. Here the effect of $V_{ctrl}(t)$ is reflected into the waveform of $V_c(t)$ immediately. The variation of $V_c(t)$ eventually causes the variation of the frequency f and therefore the variation of output phase $\phi(t)$. Solving equation 5.4 is much easier than solving equation 5.1.

The idea behind equation 5.4 is important for the following reasons: (a) It changes the traditional concept used in modeling VCO. (b) It treats the modeling of VCO and VCDL as one problem. (c) Since the delay can be evaluated directly from the input-output waveform relationship, there is no requirement that the system must be in-lock. Therefore, it expands the scope of simulating PLL and DLL systems. Both acquisition and in-lock performance can be simulated, which is impossible in the previous approaches. Behavioral simulation based on equation 5.4 is similar to SPICE simulation in the sense that the output time domain waveform is constructed, but with greatly reduced CPU time. Since parameters R_{eff} and C_{eff} have very clear physical meaning, the extraction is straightforward.

5.2.2 Application in VCO modeling

The VCO is modeled in Figure 5.2 by connecting the delay cell model in Fig.5.1(b) into a ring structure. Logic level outputs may be obtained by reshaping V_{ci} with a buffer. This VCO model inherits the merit of the RC delay cell model discussed in the previous section.

However, unlike a delay line, an oscillator has no external input. In order to construct the waveform, some oscillation mechanism needs to be introduced into this behavioral model. This is realized by defining every delay cell into a two-state machine:

State 1: charging *State 2: discharging*

The transition between these two states occurs when V_{ci} reaches either the high voltage limit V_H or the low voltage limit V_L. That is, when V_{ci} charges, it keeps rising until V_H, and then switches to the discharge state until V_{ci} reaches V_L. Since the total charging and discharging time defines the period, the choice of V_H and V_L cannot be arbitrary. Assuming that there is no mismatch, it must satisfy the following two conditions: (1) The delay of every single stage is $\tau_{eff} = R_{eff}C_{eff}$; (2) The total time of charging and discharging is $M\tau_{eff}$.

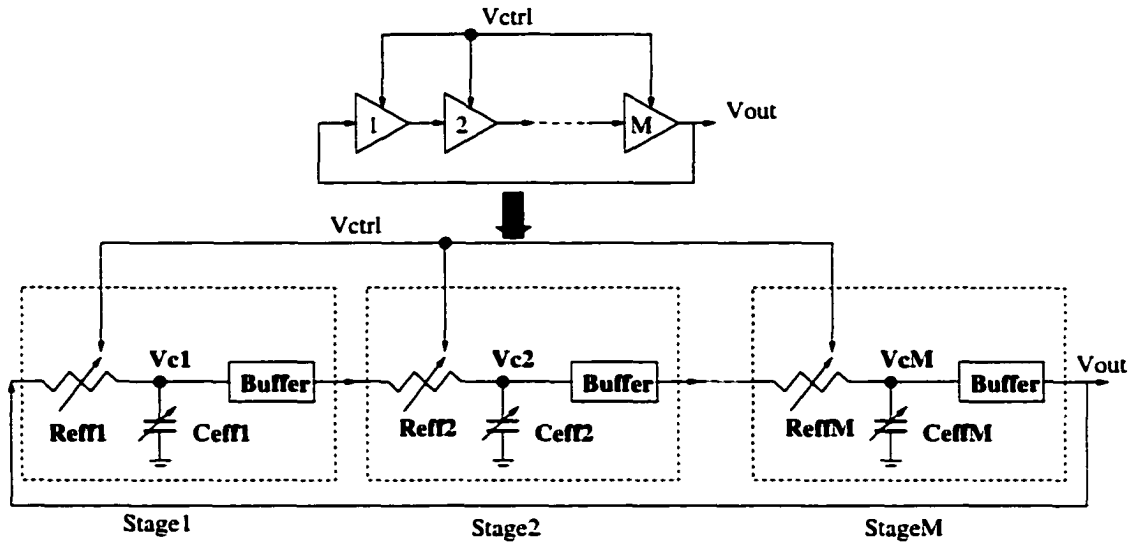


Figure 5.2 Square wave ring oscillator VCO model

As an example, suppose the output has a 50% duty cycle, the power supply is VDD and the stage number is M, then

$$V_H = \frac{e^{M/2} - 1}{e^{M/2} - e^{-M/2}} VDD \tag{5.5}$$

$$V_L = \frac{1 - e^{-M/2}}{e^{M/2} - e^{-M/2}} VDD \tag{5.6}$$

An additional advantage of this VCO model is that simulation of mismatch effects in multi-phase clock generators is possible. Besides this, even if at first glance the proposed model is purely for a ring oscillator based VCO, it can also be applied to other types of VCO circuits by assuming $M = 1$ and using the appropriate relationship of $\tau_{eff} \propto V_{ctrl}^{-1}$.

5.2.3 Application in modeling VCDL

VCDL behavioral modeling has seldom been addressed in the literature. The model in [53] can only simulate the noise behavior when the DLL system is in-lock and cannot simulate the DLL acquisition behavior, especially with the inclusion of noise effects. As the role of behavioral simulation is to uncover the potential design problems, the inclusion of lock-in and other dynamic loop behavior is highly desirable. In this work, the VCDL model can be easily established by configuring the delay cell model in the same flavor as the real delay line circuit as shown in Figure 5.3.

The state equation to describe each stage is:

$$R_{eff}(i)C_{eff}(i) \times \frac{dV_{ci}(t)}{dt} + V_{ci}(t) = V_{i-1}(t) \tag{5.7}$$

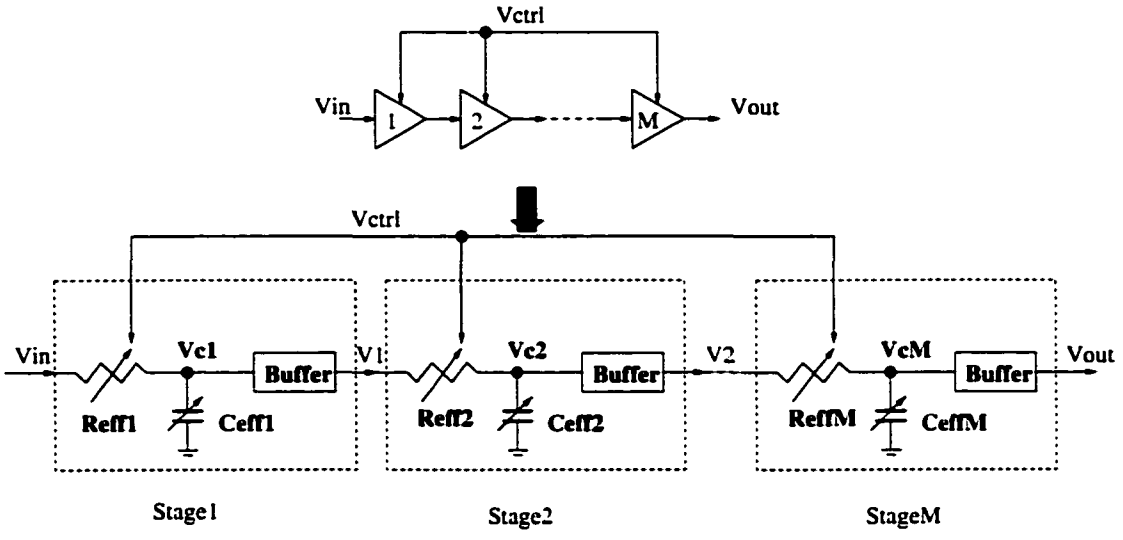


Figure 5.3 VCDL model

where $R_{eff}(i)$ and $C_{eff}(i)$ are stage i's equivalent resistance and capacitance respectively. V_{ci} is the *ith* voltage on $C_{eff}(i)$ and V_{i-1} is the input to stage i, which is the output of stage $i-1$. With this RC model, multi-phase clock outputs V_1, V_2, $\cdots$, V_{out} can be computed sequentially. In this sense, multi-phase clock channel mismatch effect can also be simulated, if necessary.

5.3 PLL And DLL System Simulators

With the VCO and VCDL model available, the phase detector and the charge pump loop filter models need to be set up for PLL/DLL system simulation.

The behavioral model for the phase-frequency detector is a three-state state machine described in [2]. It is shown as follows in Figure 5.4. The three states are: the VCO or VCDL output V (1) is *early*, (2) is $in-lock$ or (3) is *late* with respect to the reference clock R. Whenever there is a rising edge in R, it moves the state upwards toward *late* state and whenever there is rising edge of V it moves the state downwards to *early* state. The behavioral implementation of such state machine is easy. The outputs from the phase detector Up and *Down* are used to control the charge pump.

The loop filter is modeled with simple differential equations. This is shown in Figure 5.5, where (a) is for PLL and (b) is for DLL.

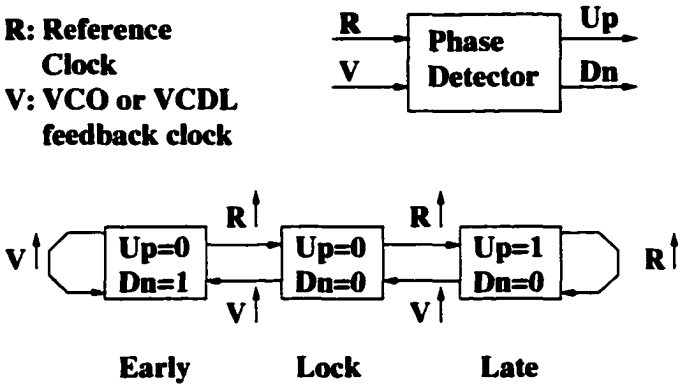


Figure 5.4 Three-state phase-frequency detector model

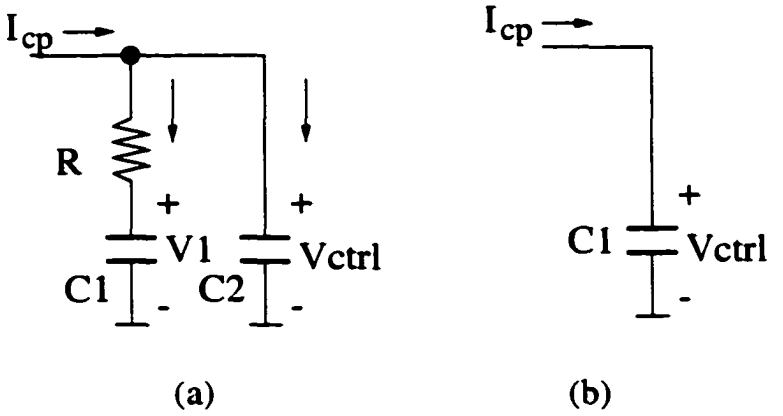


Figure 5.5 Loop filter (a) PLL; (b) DLL

For a PLL, the differential equations to describe the loop filter are:

$$\begin{cases} RC_1 \frac{dV_1}{dt} + V_1 = V_{ctrl} \\ C_1 \frac{dV_1}{dt} + C_2 \frac{dV_{ctrl}}{dt} = I_{cp} \end{cases}$$

where I_{cp} is the pump current controlled by the 3-state *Up Down* output combination from the phase detector.

For a DLL, since only one capacitor fulfills the function, the differential equation is rather simple:

$$C_1 \frac{dV_{ctrl}}{dt} = I_{cp} \tag{5.8}$$

The numerical method to solve the above equations is well established.

Both behavioral simulators were implemented using *C* language. By reviewing the system setup as a whole, the voltage waveform is the only remaining unsolved variable. There is no interface needed between blocks. The developed simulator is compact.

As an illustration of the effectiveness of the behavioral simulator, a comparison is made between SPICE and our behavioral simulation results in Figure 5.6. This is a simulation of the PLL acquisition process without noise. Both waveforms are the control voltage on the oscillator, which is the direct indicator of the acquisition process. Both simulations were done on an HP-B180 workstation. The delay cell t_d and its control voltage static relationship in this behavioral simulation is extracted from the transistor level that includes real circuit nonlinear effects. On the upper half is the SPICE simulation result. It takes as many as 2610 seconds to get this result. On the lower half is the simulation result from this behavioral simulator. It only takes 84 seconds to get the result. This figure shows that with an accurately extracted model, the behavioral simulator can predict the system behavior with the same level of accuracy as that of SPICE with greatly reduced CPU time. Comparison of DLL simulations with both simulators reaches a similar conclusion.

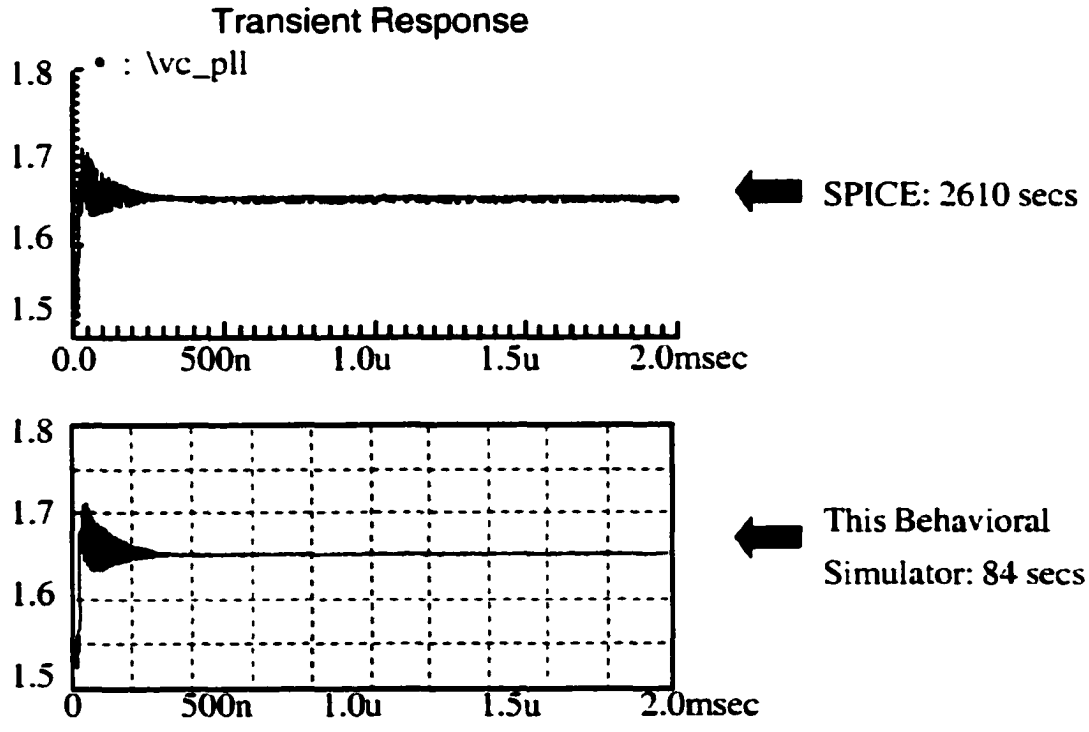


Figure 5.6 Comparison of SPICE simulator and the proposed simulator

Besides enhancing the efficiency of the simulation, a more powerful function of behavioral simulation is the capability of doing jitter analysis, which can be very difficult using SPICE. In order to do this, appropriate noise mechanisms need to be added into the simulator. There are two major noise sources in monolithic PLL and DLL systems:

(1) Device noise: It consists mainly of thermal noise, shot noise and 1/f noise. Since 1/f noise is usually filtered by the loop, it is not considered here. Device noise can be added as additive white Gaussian noise into R_{eff} or C_{eff} in the delay cell model shown in Figure 5.1(b).

(2) Substrate and power supply coupling noise: This type of noise is the dominant noise source when a PLL or DLL is used as a clock generator for a high-speed digital system. This type of noise is quite different from device noise in that it is generally not white and the noise at different circuit nodes is highly correlated [25]. Detailed modeling of such noise is a rather important topic and is beyond the scope of this work. However, it can be added into the delay cell model with random sequence of the desired correlation.

Also, to minimize simulator convergence noise, appropriate simulation time steps must be chosen along with careful implementation of the numerical algorithm [54].

5.4 Jitter Simulation

The goal of this section is to demonstrate the performance of the proposed behavioral simulator in noise analysis. Simulation results are compared with theoretical predictions as well as measurement results whenever possible. All the simulations are done for an operating frequency of $100MHz$, i.e. period is $10ns$.

5.4.1 Open loop VCO jitter

As the first example, the simulated open loop VCO jitter characteristics are shown in Figure 5.7. Noise is added by changing the delay of each cell randomly and is assumed to be normally distributed with $\sigma = 0.25\% \times \overline{t_d}$, where $\overline{t_d}$ is the open loop average unit cell delay. In this specific case, we set $\overline{t_d} = 1/4 \times 1/100MHz = 2.5nsec$. With the above noise source, the r.m.s. value of the cycle-to-cycle jitter is 2.23psec, which is 0.223% of the oscillation period. The accumulated jitter r.m.s. value keeps rising as the measurement interval ΔT increases. The slope of the curve in Figure 5.7 is caused by the well known jitter accumulation within the open loop VCO and should be minimized in the design. This result is consistent with the measurement result in [24] and the discussion in chapter 3.

5.4.2 In-lock jitter evaluation

The in-lock jitter performance is the major concern of PLL and DLL systems. Figure 5.8 is a summary of the new behavioral simulation results versus theoretical prediction. Both the PLL and DLL are working as clock generators where the reference clock can be considered noiseless. Again, the

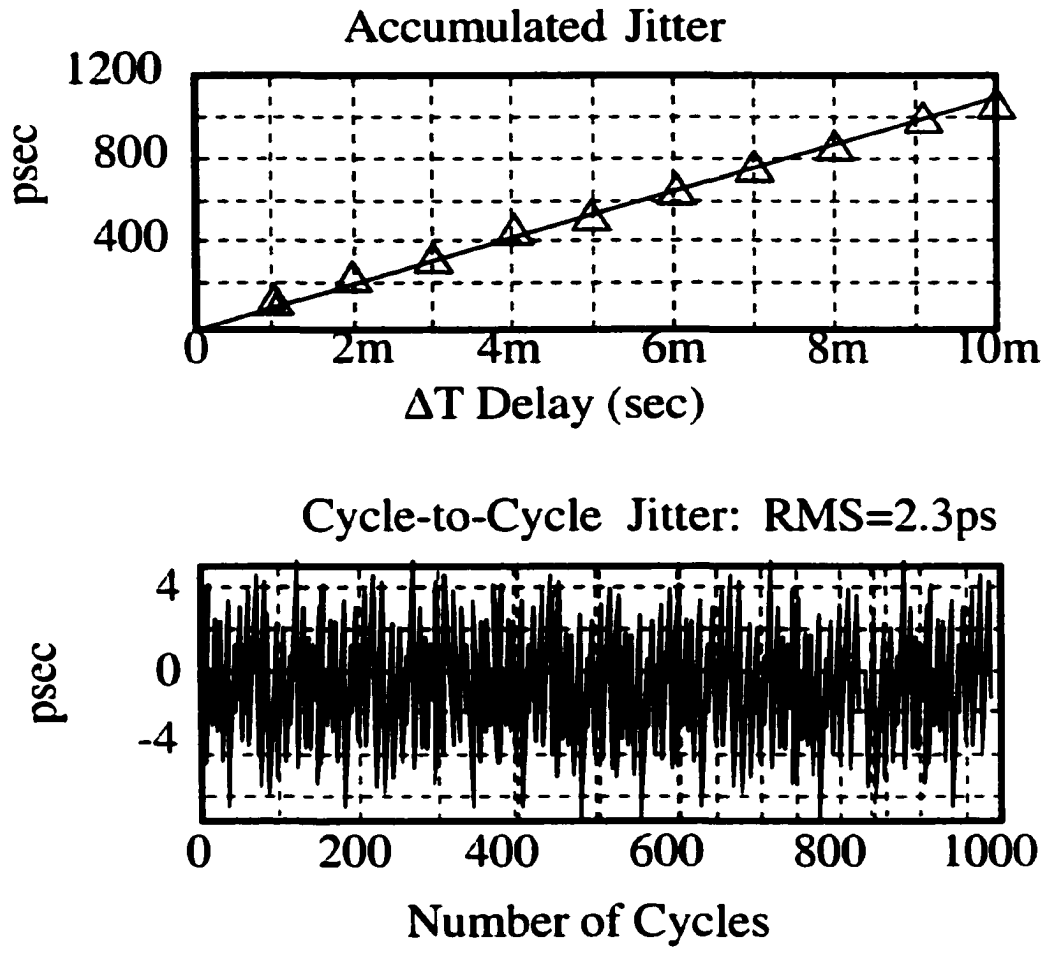


Figure 5.7 Open loop VCO jitter

frequency is $100MHz$. The r.m.s jitter of the VCO as well as the VCDL is assumed to be 4.9psec. From this figure, the PLL jitter accumulation effect is obvious: the lower the loop bandwidth, the larger the accumulated jitter. By contrast, there is no such accumulation effect in DLLs. The above simulation results validate the theoretical analysis results in chapter 3. Note that for the PLL, there is a deviation between simulation result and theoretical prediction at high loop bandwidth. This is because at high loop bandwidth, the assumption used to do the theoretical analysis in chapter 3 no longer holds. Thus, the theoretical prediction is not valid. Under this situation, the PLL accumulated jitter will approach the limit set by the VCO cycle-to-cycle r.m.s jitter instead i.e., the jitter accumulation factor reaches 1. Simulation reflects this trend correctly.

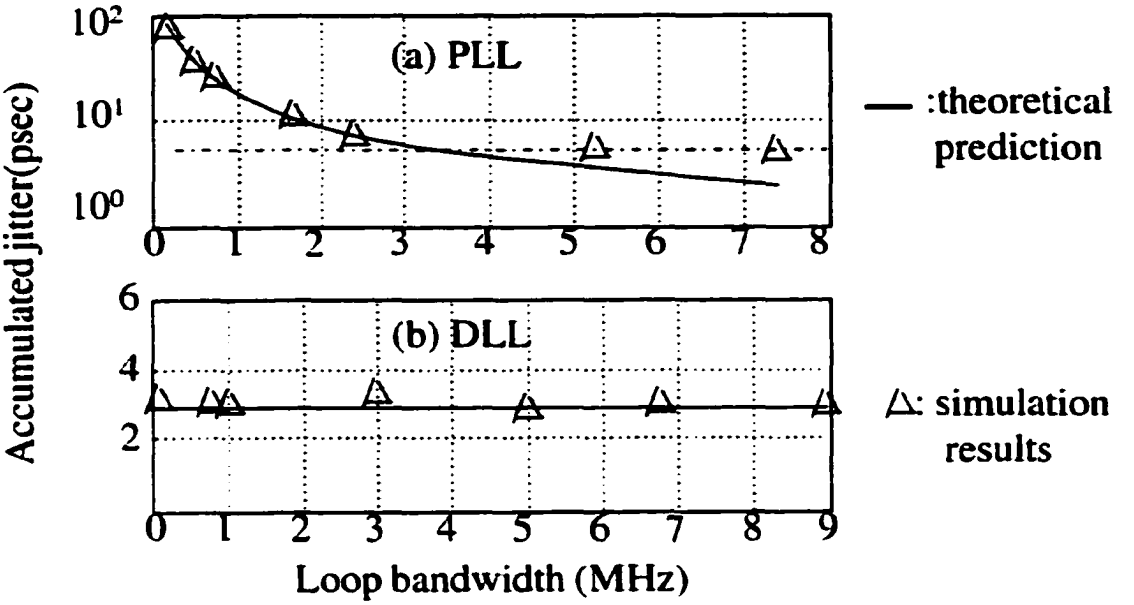


Figure 5.8 PLL and DLL in-lock jitter comparison

5.4.3 PLL and DLL Dynamic performance

Dynamic acquisition performance with the presence of noise is another important PLL/DLL characteristic. However, former behavioral simulators cannot handle this situation. In this work, both PLL and DLL circuits can be simulated along with the coupled noise. Shown in Figure 5.9 and Figure 5.10 are examples of behavioral simulation results of a PLL and DLL respectively. Both the PLL and DLL are tracking in to a $100MHz$ reference clock.

Figure 5.9 shows the simulation results of a PLL with four different levels of noise: (1) no noise; (2) $\sigma = 1\%T = 100ps$ noise; (3) $\sigma = 3\%T = 300ps$ noise; and (4) $\sigma = 5\%T = 500ps$ noise. It is clear that the dynamic performance deteriorates as the noise increases. When noise is over a certain limit, the PLL can no longer lock to the reference clock. Result (4) shows this case. A similar situation happens in a DLL, which is shown in Figure 5.10.

With this simulation capability, the upper limit of the noise that does not make the PLL or DLL lose lock can be identified.

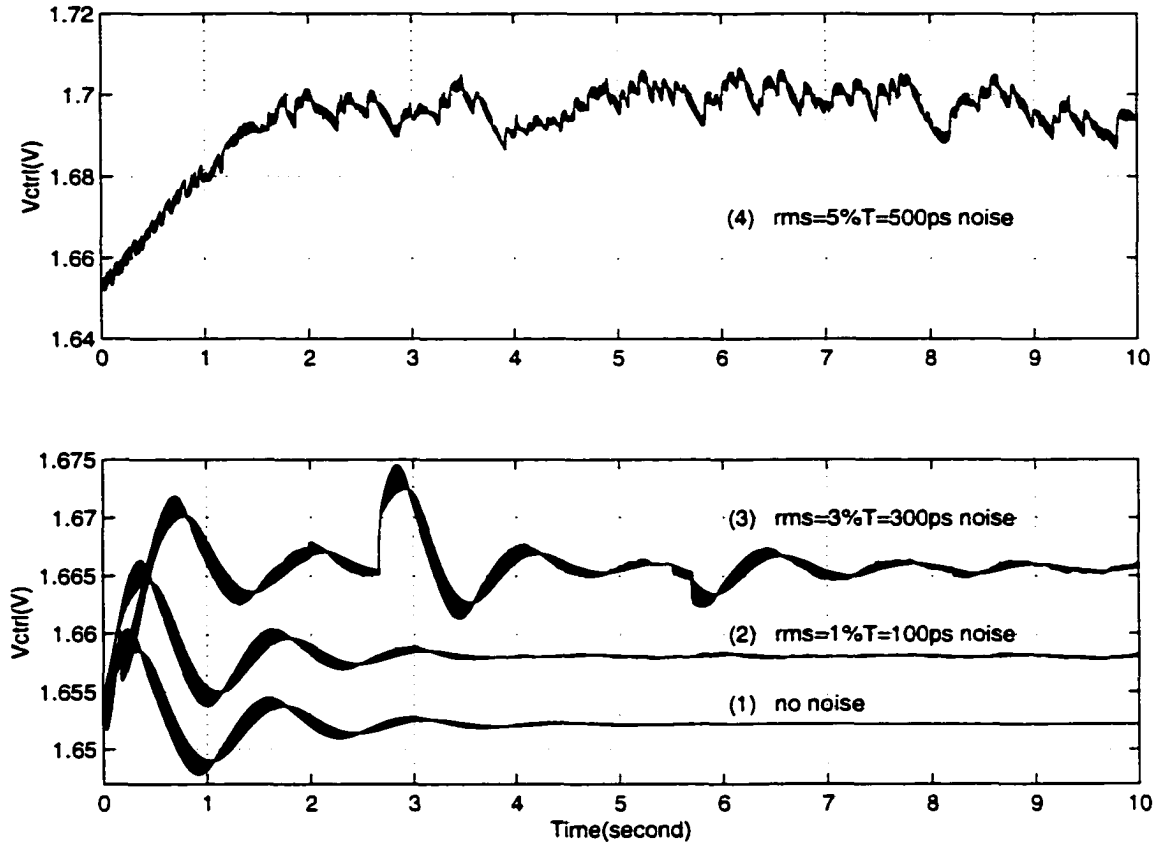


Figure 5.9 Behavioral simulation of PLL track-in process with noise

5.5 Summary

New VCO and VCDL behavioral models have been demonstrated based upon a newly proposed delay cell model. The consequent numerical algorithm for behavioral simulation is simplified. Comparisons between our simulation results and theoretical prediction as well as measurement results show the effectiveness of these new models in noise analysis with greatly reduced CPU time. Besides for the PLL model, the first published thorough behavioral simulation environment for DLL systems was also proposed. Both behavioral simulators are not only useful in top-down design, but also in bottom-up verification. This simulation method is a useful technique for low jitter system design.

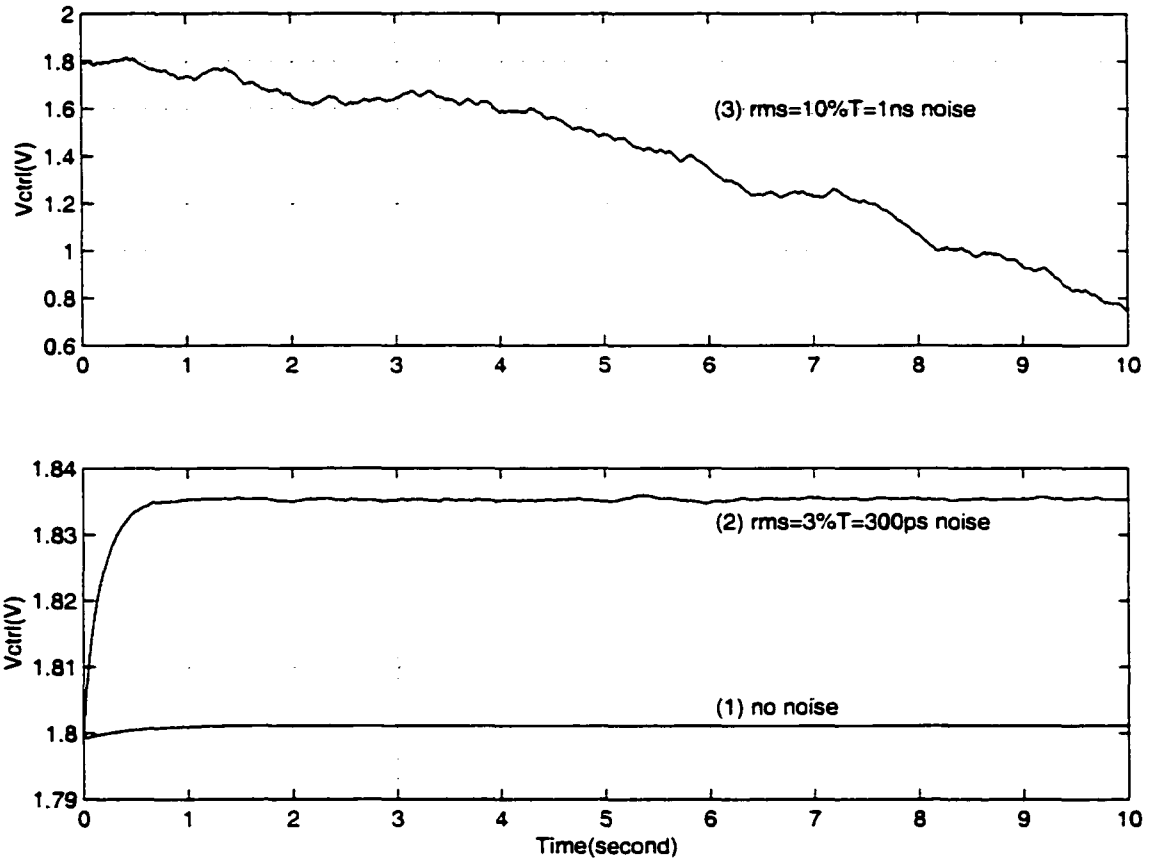


Figure 5.10 Behavioral simulation of DLL track-in process with noise

CHAPTER 6 APPLICATION IN GIGABIT FIBRE CHANNEL TRANSCEIVER FOR SERIAL DATA COMMUNICATION

In recent years, the speed of serial data communication has been enhanced to Gbits/sec range by taking advantage of high-speed media such as fibre channel or fast Ethernet (1000BaseT). Several years ago, GaAs and bipolar technologies were used for such high-speed applications. However, the advent of mature deep sub-micron CMOS technology makes it possible to expand the application of CMOS technology into the GHz domain with its inherent advantage of low cost. The primary motivation for this work is to implement the fibre channel transceiver with sub-micron CMOS technology. The major issue in this application is the design of two high performance PLLs. On the transmitter side, it is used as a clock generator to implement $\times 10$ frequency multiplication. On the receiver side, it acts as clock recovery circuit that provides the clock for the whole system based on the incoming data. The speed and jitter requirements pose a challenge on CMOS technology, especially the CMOS clock recovery circuit. Consequently, the emphasis of this work was to explore high-speed CMOS techniques for clock recovery.

In the following sections, an overview of the fibre channel transceiver is first briefly covered. The function and importance of PLLs in this application is then explained. The focus of this chapter is the detailed implementation of such a clock generator and clock recovery circuit. For clock recovery, two circuit designs are proposed. Experimental results from the two prototype chips will be presented at the end of this chapter.

6.1 Gigabit Fibre Channel Transceiver Overview

6.1.1 Function description

The fibre channel transceiver is the physical layer interface for performing high-speed serial data transmission over a fibre optic interface conforming to the requirements of ANSI X3T11 Fibre Channel specification. The chip runs at 1.25 or 1.0625Gbit/s data rates with an associated 10-bit data word.

The functional block diagram of a fibre channel transceiver is shown in Figure 6.1 [41]. The chip performs parallel-to-serial conversion on the transmitter side. The PLL is used to synthesize the high speed clock from a low-speed reference. This high speed clock converts the 10-bit 125 or 106.25Mbyte/s parallel data into 1.25 or 1.0625Gbit/s serial data; the serialized data is then passed through a line driver and transmitted over the fibre channel. On the receiver side, the 1.25 or 1.0625Gbit/s serial data is converted back into 10-bit parallel data word at 125 or 106.25 Mbyte/s rate. Since the clock information is embedded in the transmitted data stream, a clock recovery circuit is needed at the receiver side to recover the clock and to synchronize the receiver to the input data. Usually this clock recovery is based on a PLL.

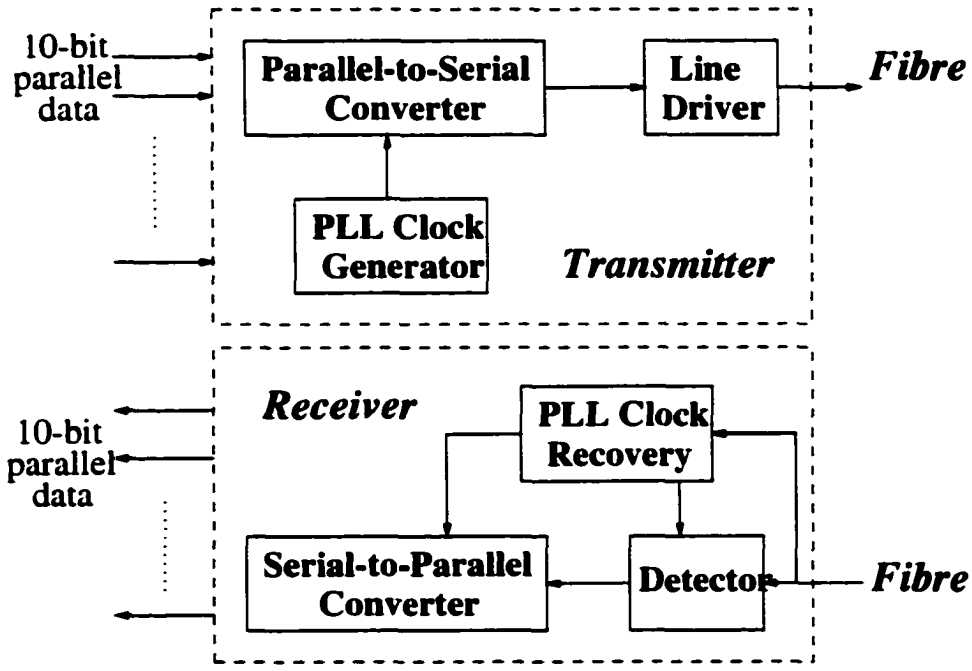


Figure 6.1 Gigabit fibre channel transceiver

The specification of the Bit-Error-Rate (BER) of this type of transceiver is 10^{-12}. In order to achieve this, the jitter specification on the transmitted signal is r.m.s jitter $< 20ps$ and peak-to-peak jitter $< 100ps$. With this transmitted signal, a robust clock recovery circuit on the receiver side can recover the data with 10^{-12} BER. Usually, the parallel-to-serial converter on the transmitter side and the serial-to-parallel converter on the receiver side are shift registers, which are easy to implement. The major challenge of this design is the implementation of these two PLLs, especially the clock recovery PLL.

6.1.2 System implementation

There are many ways to implement the transceiver. In this approach, a 5-channel time-interleaved structure is used on the transmitter side to relax the speed requirement on the inner circuit and, therefore, to reduce the power dissipation. At the output of each channel, a high speed MUX is used to recombine all the outputs into serial data. This approach can be shown in Figure 6.2.

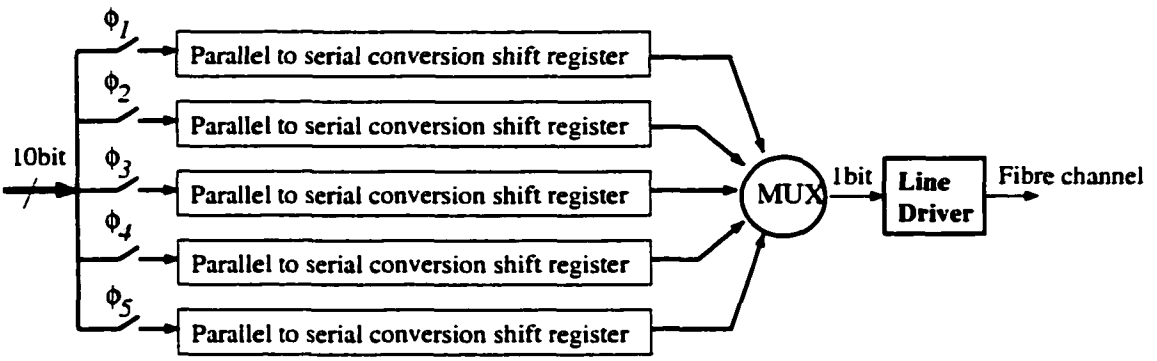


Figure 6.2 Proposed parallel transmitter structure

The clock generator needs to produce the above interleaved phases $\phi 1$, $\phi 2$, $\phi 3$, $\phi 4$, $\phi 5$. Since there are a total of 5 channels working at the same time, each of them is working at one-fifth of the full speed. Because the final speed is 1.0625Gbit/s, and the reference clock is $106.25MHz$, $\phi 1$, $\phi 2$, $\phi 3$, $\phi 4$, $\phi 5$ should be at $212.5MHz$, which have their relationship illustrated in Figure 6.3.

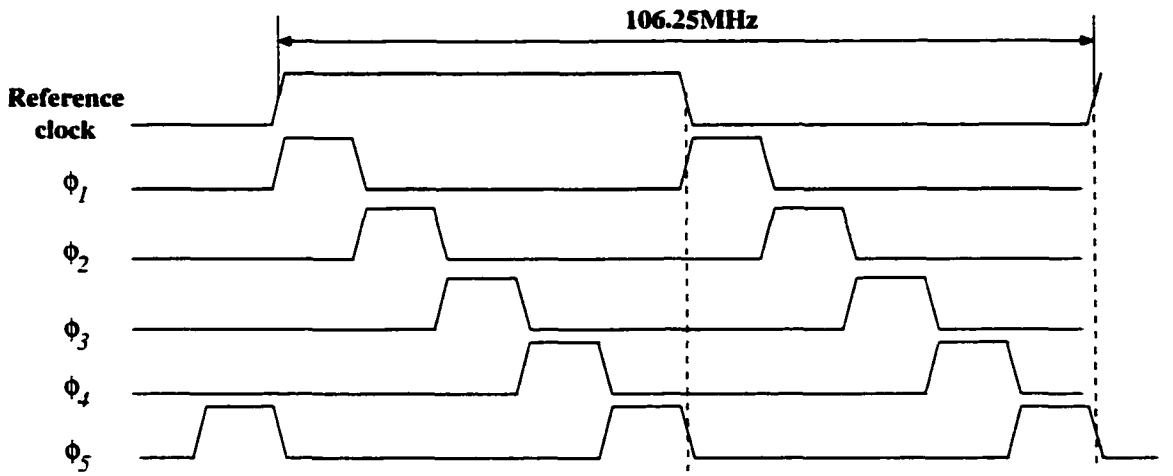


Figure 6.3 Illustration of 5-phase time interleaved clocks

6.2 PLL Based Clock Generator Design

The function of the transmitter PLL is to generate the above five phases at $212.5MHz$. Since the reference input is at $106.25MHz$, the PLL needs to implement $\times 2$ (multiply by two) frequency synthesis. Based on the above proposed transmitter structure, the block diagram of the PLL clock generator is shown in Figure 6.4.

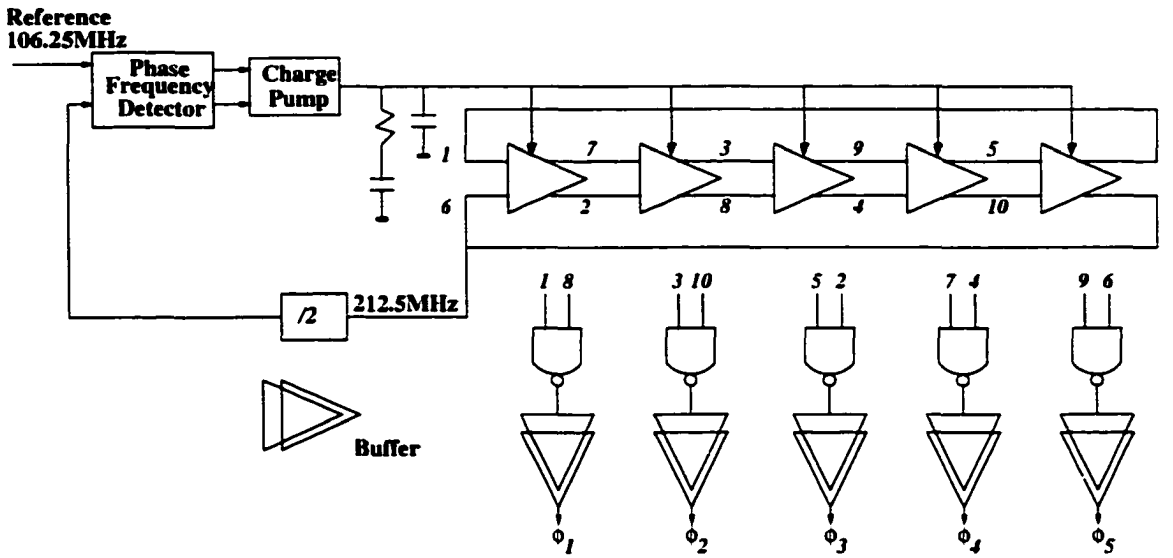


Figure 6.4 Transmitter PLL clock generator

In order to generate these 5 uniformly-distributed phases, a five-stage ring oscillator was used as the VCO. The delay cell is a fully differential current steering cell that has better power supply rejection performance [17]. In order to get the required pulse, NAND gates are added at the outputs of the VCO. After the NAND gate, buffers are added to enhance the driving capability of the clock pulses. Notice that a $\div 2$ divider is added in the feedback path to realize the $\times 2$ frequency multiplication. Once in lock, the output will be at $2 \times 106.25MHz = 212.5MHz$.

Since the frequency of this design has been set, specifications such as track-in range or lock range are not the concern. The only requirement is that the tuning range of the VCO is big enough to cover temperature variation as well as process corners. The basic design issue here is the choice of loop bandwidth to optimize the jitter performance. As discussed in Chapter 3, for a PLL acting as a clock generator, the loop bandwidth should be as high as possible. Usually the stability requirement sets the upper limit of loop bandwidth to be at least one order of magnitude lower than the working frequency. The operation frequency of this PLL is $106.25MHz$, the loop bandwidth cannot be over $10MHz$ to

guarantee stability. Considering the possible effect of process corners, it is set to be at $5MHz$. Testing results on this design will be given in Section 6.4.

6.3 Gigabit/second Clock Recovery Design

The most economic way to solve the synchronization problem between the data and the clock in fibre channel transceiver is to embed the clock information into the data stream and not to transmit it separately. This requires a clock recovery circuit at the receiving end to recover the clock information and to resynchronize the incoming data with it.

What makes a clock recovery PLL more difficult than a PLL for other applications is that normally a PLL tracks a *periodic* output with a *periodic* input, while a clock recovery PLL should extract a *periodic* clock signal from $non-periodic$ random data. Even when the data is absent, the frequency should not drift. To make the situation even worse, the data being transmitted in fibre channel is 8B/10B encoded NRZ data. The spectrum of NRZ signal lacks the frequency component at the bit rate that is exactly the frequency for the clock recovery circuit to recover. This frequency component needs to be created by the circuit described here.

Clock recovery in Gigabit/second range implementing with CMOS technology is a challenging topic. With deep sub-micron technology, speed is no longer the bottleneck. The major concern is to achieve the required jitter performance in a rather noisy environment. Two designs were proposed in this work, the first one being implemented with silicon. Experimental results will be presented to demonstrate its performance. Targeting a reduction of the noise coupling effects from the substrate as well as power supply, an analog PLL based on current steering technique will be presented in Section 6.3.2 as an alternative way of performing clock recovery.

6.3.1 Approach one

As mentioned earlier, most clock recovery circuits are based on PLL. Figure 6.5 shows the block diagram of a PLL based clock recovery circuit [58]. It is composed of: (1) Phase detector: detecting the phase difference between the incoming data and the recovered clock. It is the key part to fulfilling the clock recovery function; (2) Charge pump and loop filter: transferring high frequency phase difference into the low frequency control signal V_{ctrl} for the VCO; (3) Voltage-Controlled-Oscillator: generating a clock that is aligned to the incoming data. The recovered clock from VCO is used to resample the incoming data and act as the receiver system clock.

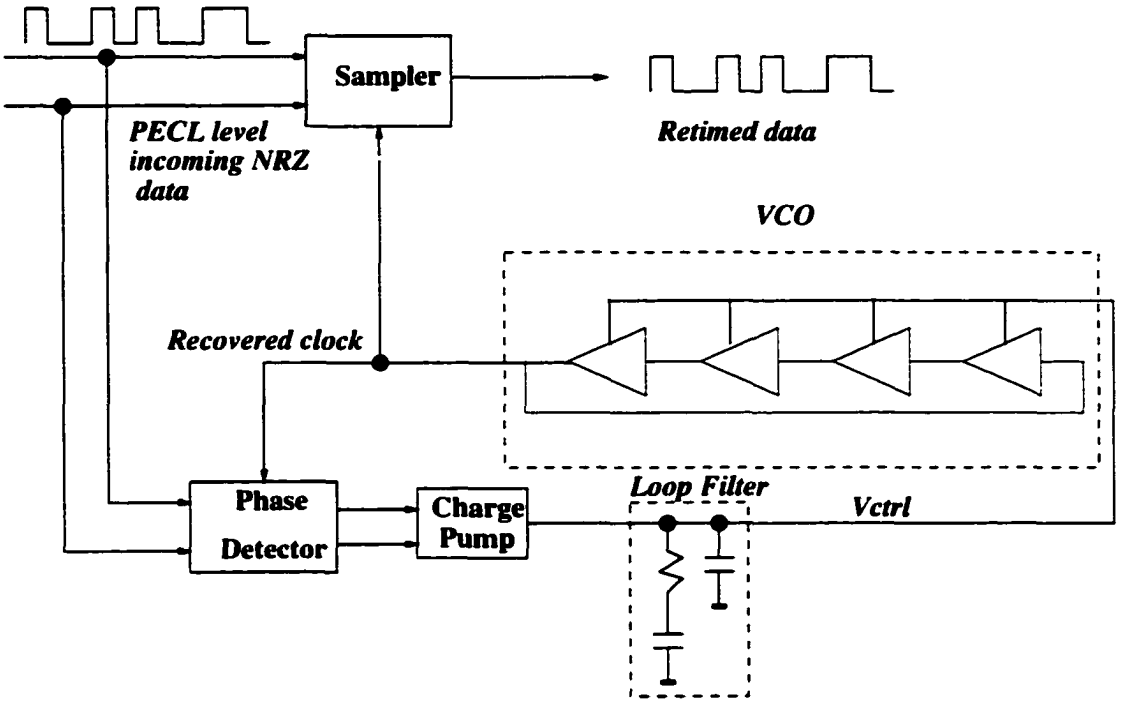


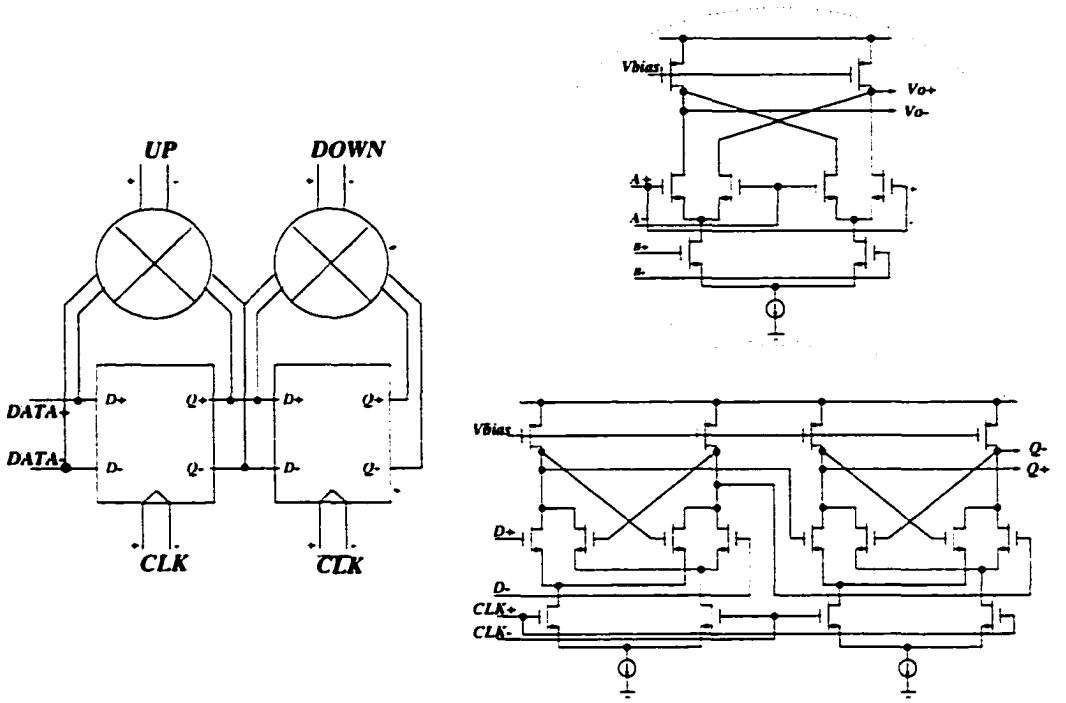
Figure 6.5 Clock recovery system architecture

Much effort was undertaken to develop high performance CMOS PLL building blocks. A fully differential high-speed phase detector was designed, especially for NRZ clock recovery. A four-stage ring oscillator that has a tuning range sufficient to cover process corners and a high-speed charge pump with on chip loop filter was also developed.

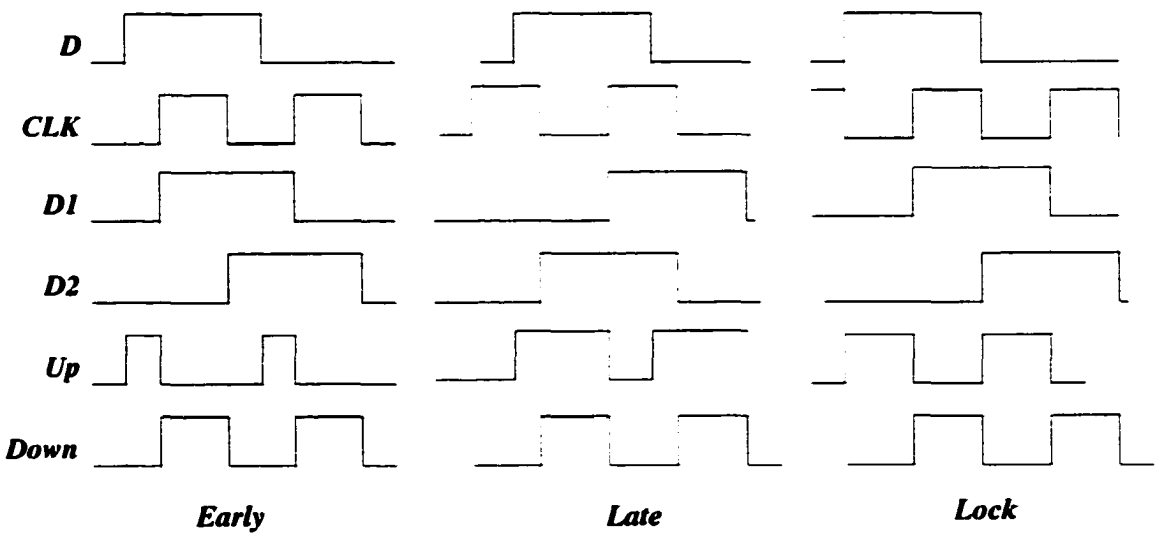
6.3.1.1 Clock recovery building blocks

A fully differential phase detector for NRZ clock/data recovery The structure of the phase detector determines the function of the PLL as a clock recovery circuit. As mentioned before, the clock frequency needs to be created from NRZ data. Figure 6.6(a) shows the phase detector used here [2] [60]. It consists of two D flip-flops and two Gilbert multipliers.

The incoming data is first sampled by CLK to get $D1$ which is aligned to CLK. $D1$ is further delayed by $\frac{1}{2}$bit time to get $D2$. The "Up" signal is produced by $D \times D1$ while the "Down" signal is $D1 \times D2$. Since $D2$ is always the $\frac{1}{2}$bit delayed value of $D1$, the "Down" pulse width remains constant. This is a reference. The pulse width of "Up" varies at the "early", "late" or "lock" situation which is shown in Figure 6.6(b). Note that whenever in "lock", the frequency of "Up" and "Down" signals are 2 times the highest data transition frequency, which is exactly at $2 \times 625MHz = 1.25GHz$. This creates



a) Circuit



b) Timing

Figure 6.6 Fully differential phase detector for clock recovery

the required clock component. If there is no transition in the incoming data, there is no output pulse to the charge pump, hence there is no change in the VCO control voltage and the VCO frequency is maintained. This is especially important for clock recovery.

Both D flip-flops and the multipliers use a fully differential structure, which turns out to be appropriate for this application: (1) By working at reduced voltage swing, it has the potential of operating at very high frequency; (2) It reduces current spikes on the power supply, creating less power-supply induced jitter; (3) It has the ability to accept PECL level differential input data RX+ and RX- directly, no interface circuit needed; (4) The D flip-flop which produces $D1$ also implements the resample function between incoming data and its clock. Thus the sampler in Figure 6.5 is already combined into this phase detector itself. No additional sampler circuit is needed.

Charge pump and loop filter The function of the charge pump and loop filter is to transfer the high frequency "Up" and "Down" pulses into the VCO analog control signal V_{ctrl}. The circuit is shown in Figure 6.7.

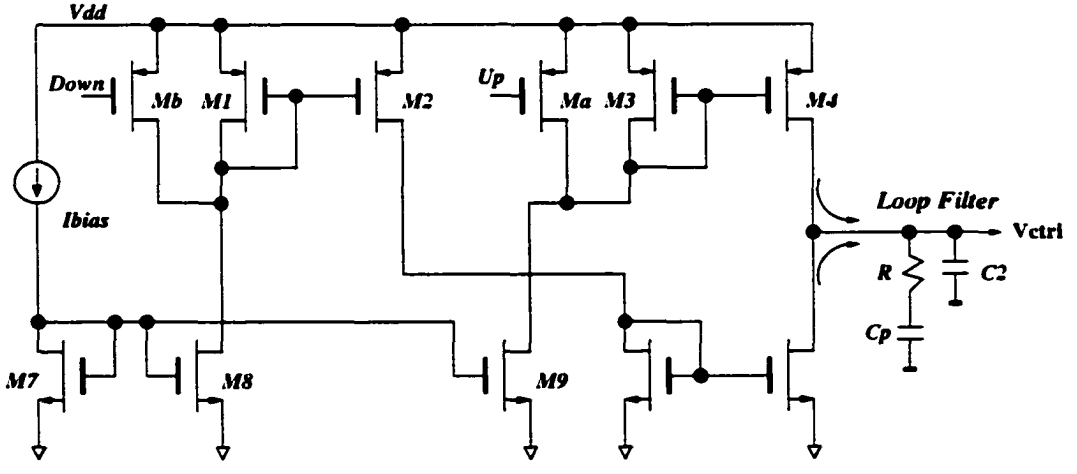


Figure 6.7 Charge pump and loop filter

When there are no "Up" and "Down" signals, there is no current output to the loop filter. When "Up" is high, M_a absorbs less current and there is more current flow in M_3, and consequently M_4. This will result in net current output to the loop filter that will increase the frequency of the VCO. Similarly, when "Down" is high this decreases the VCO frequency. By working at appropriate voltage swings, this structure avoids the fully on/off of M_a and M_b, thus it can work at high speed.

Voltage-Controlled-Oscillator(VCO) A ring oscillator structure was chosen because it can be easily integrated. The core circuit is a 4-stage ring oscillator. At the input there is a bias generator that transfers V_{ctrl} into two bias voltages to adjust the VCO frequency. At the output, it needs a buffer to enhance the driving capability, since the recovered clock will be used as the system clock of the whole receiver. The whole VCO is shown in Figure 6.8.

The fully differential delay cell is shown Figure 6.9.

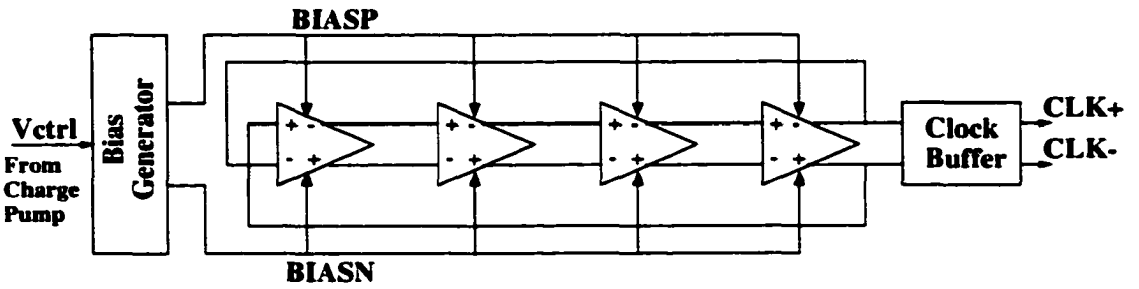


Figure 6.8 VCO block diagram

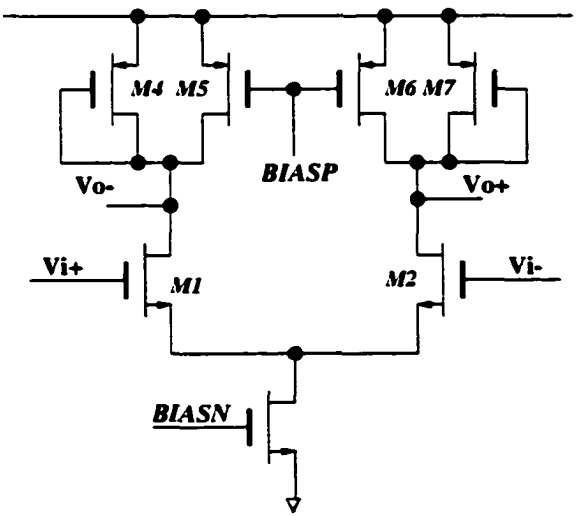


Figure 6.9 Fully differential delay cell

This delay cell has a rather wide tuning range to cover all process corners [59]. The frequency can be changed by adjusting $BIASP$ and $BIASN$ simultaneously. Furthermore, through careful placement to generate $BIASN$ and $BIASP$ appropriately, both the tuning range and the power dissipation of the VCO can be optimized.

The VCO simulated performance is summarized in Table 6.1. It can be seen that this VCO has very big tuning range to cover all process corners and temperature variations from $0°C$ to $100°C$.

Loop dynamics As mentioned in Chapter 3, for a PLL to be used as clock recovery circuit, the jitter suppression on the input is more important than the jitter suppression on the inner VCO noise. Thus loop bandwidth should be set low. However, this specific application also has a tracking time specification, which sets the lower limit on the loop bandwidth. The final setting is the trade-off between these two. The whole loop bandwidth $\omega_n \approx 2MHz$ and the damping factor $\xi \approx 0.7$.

Table 6.1 Simulated VCO Performance

	Fast corner, T=$0°C$	Normal, T=$65°C$	Slow corner, T=$100°C$
Tuning range	100M~2.2G	60M~1.9G	50M~1.7G
Power	240mW	210mW	190mW
Rise/Fall time	100ps	120ps	150ps

6.3.1.2 Simulation results

This PLL clock/data recovery circuit has been fabricated along with the whole transceiver in $HP0.35\mu m$ single-poly, triple-metal CMOS technology. Simulation results show that this structure works well for 1.25GHz clock/data recovery with all process corners and when temperature varies from $0°C$ to $100°C$.

To illustrate, the worst case simulation results are shown in Figure 6.10, 6.11. Worst case means that both PMOS and NMOS transistors are at slow corners and temperature is at $100°C$ with power supply down to 3V. Figure 6.10 shows the waveforms of the incoming PECL data and the recovered clock. It can be seen that when in lock, the rising edge of the recovered clock is aligned to the center of incoming data, which turns out to be the best sampling time. Figure 6.11 shows the input PECL level data and the resampled data $D1$ and $D2$ from the phase detector.

6.3.1.3 Summary

In this section, the first PLL clock/data recovery circuit for 1.25Gbits/sec fibre channel transceiver was introduced. The designed high speed CMOS PLL building blocks include a fully differential phase detector, current mirror charge pump and a 4-stage fully differential VCO. Experimental results will be presented in Section 6.4 to show the real chip performance.

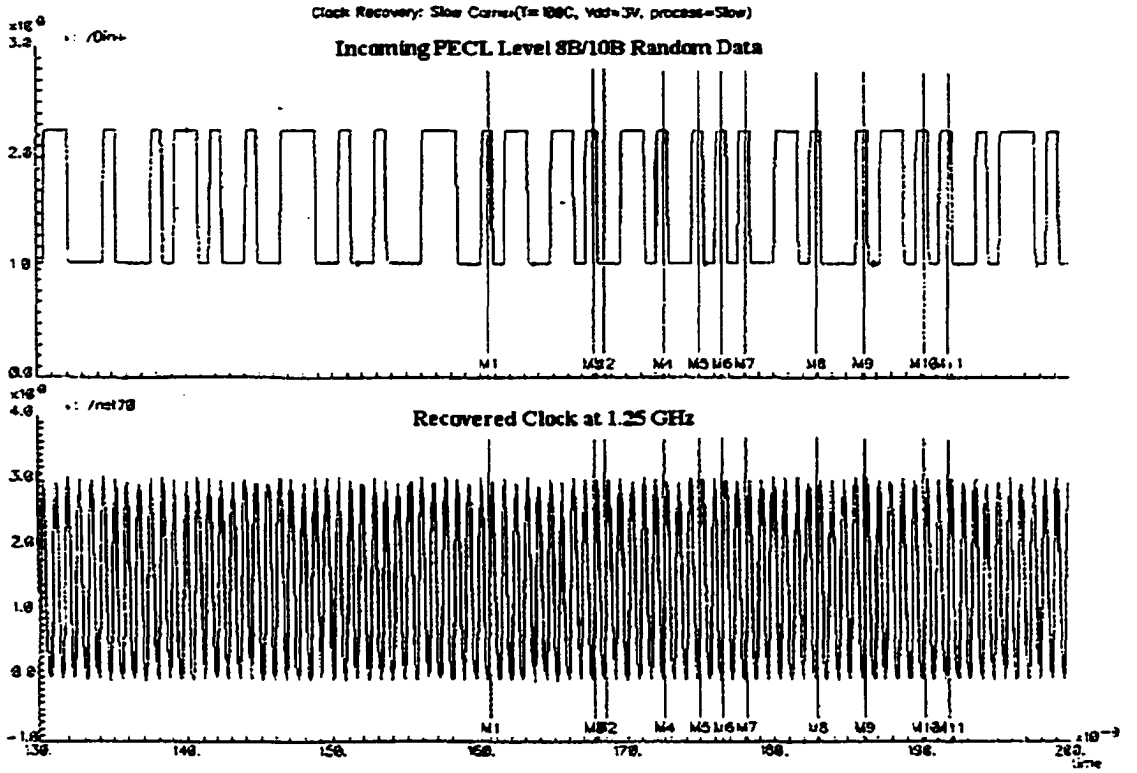


Figure 6.10 The worst case simulation results—input data and recovered clock in lock

6.3.2 Approach 2

Conventional CMOS PLL clock recovery circuits working around GHz range typically suffers from significant power supply coupling noise. This noise deteriorates the jitter of the PLL and degrades the system Bit-Error-Rate (BER). The research in [60] describes an analog approach that applies a fully differential current steering technique throughout the whole PLL system in order to reject supply coupled noise. Moreover, since this circuit works at reduced voltage swing, it not only has the potential of operating at very high frequency but also dissipates less power at high frequencies than conventional CMOS logic. Simulation results show that this clock recovery circuit can work at 1.25GHz with power dissipation of less than 100mW. All the simulations are based on $HP0.5\mu m$ CMOS single-poly triple-metal technology.

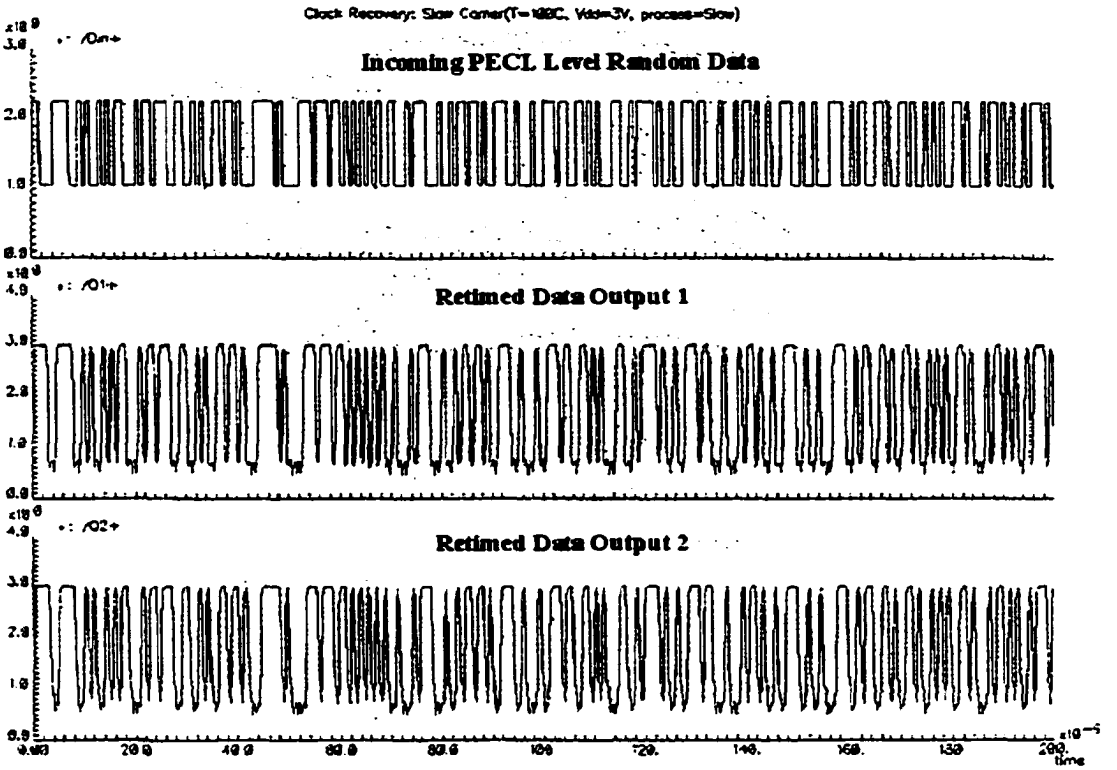


Figure 6.11 The worst case simulation results—input data and recovered data D1, D2

6.3.2.1 Fully differential current steering technique

Power supply coupling noise can be a major obstacle when implementing very high speed CMOS communication circuits. Current spikes produced during logic transitions can cause high frequency variations of the power supply and ground lines of several hundred mV or more. Worse yet, because transistors share the same substrate, substrate coupled noise is all but impossible to isolate. Numerous techniques are commonly employed to minimize the effect of power supply noise. These include separation of analog and digital power supplies, low package lead inductance, guard rings around digital circuitry, etc. However, these methods can only suppress the power supply coupled noise to a certain degree. Since the fundamental reason for supply noise generation is the variation of current during logic transitions, a more efficient way to minimize supply noise injection is to design logic circuits that maintain a constant supply current. This significantly reduces the current spikes in power supply and ground lines. The use of differential techniques will further improve performance by making supply

noise a common-mode signal, which can be rejected to some degree.

Current steering techniques have been previously proposed to reduce the amplitude of current spikes on the supply [17] [61] [62]. As an illustration, a fully differential current steering structure is shown in Figure 6.12(a). Each branch of the differential pair can be treated as a single ended current-mode inverter shown in Figure 6.12(b).

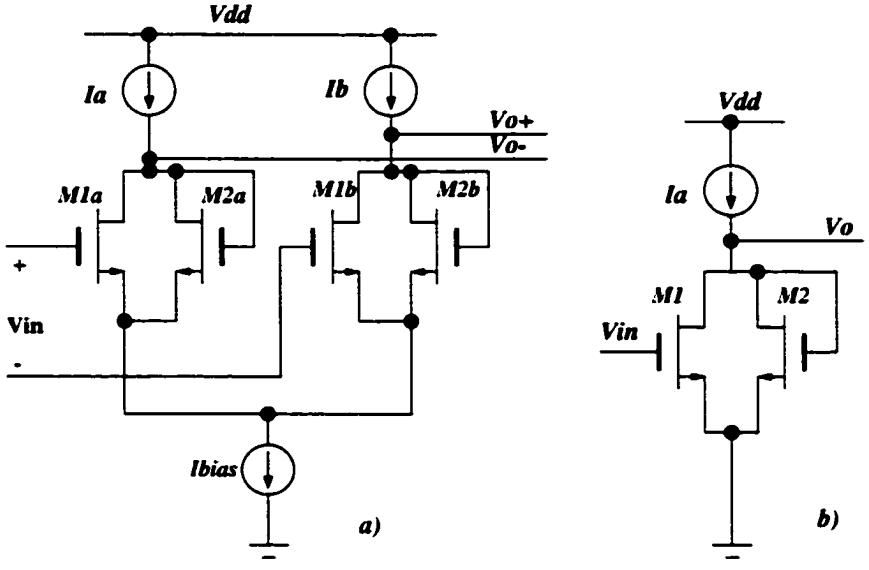


Figure 6.12 Current steering technique (a) fully differential;(b)single ended

The circuit shown in Figure 6.12 significantly reduces supply current spikes because the average supply current remains constant; the current has merely been steered down a different path depending upon the differential input voltage. The bias current source can be implemented with PMOS transistor in an N-well and be isolated from the noisy substrate. Consequently, very small switching noise is generated. If any switching noise exists, the fully differential structure can reject it at the output. Therefore, by using the fully differential current steering technique, it both reduces the generation of power supply noise, and significantly rejects noise that does exist.

6.3.2.2 Circuit design

This technique, shown in Figure 6.13, employs a fully differential structure throughout the PLL. All signal paths are differential. The following several figures show the phase detector, charge pump and VCO delay cell in differential forms respectively.

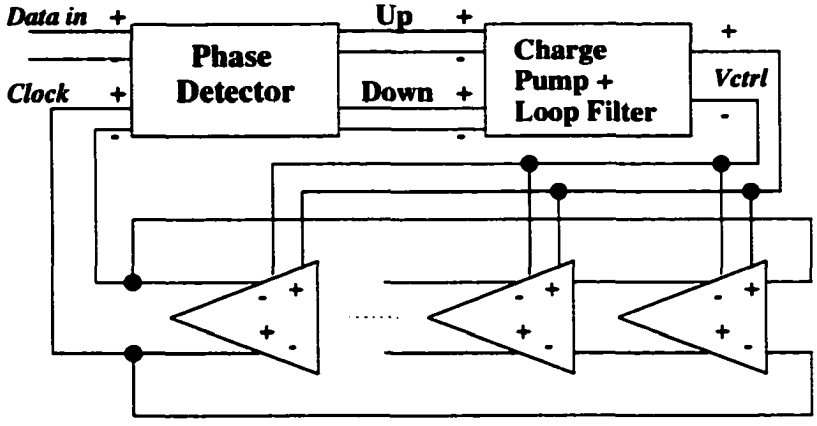


Figure 6.13 Fully differential system diagram

A. Fully differential phase detector and charge pump The phase detector, shown in Figure 6.14, uses the same circuit as that discussed in 6.3.1.1. This fully differential phase detector turns out to be a good solution for this application for the following four reasons: (1) By working at reduced voltage swing, it has the potential of operating at very high speed; (2) There are reduced current

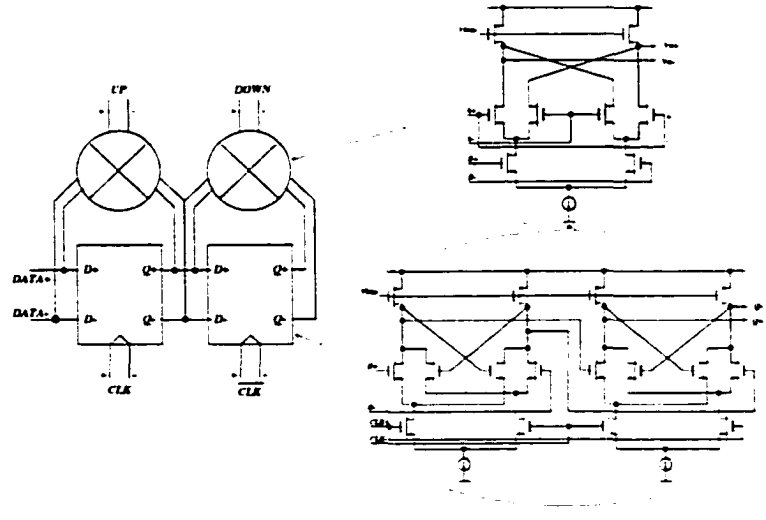
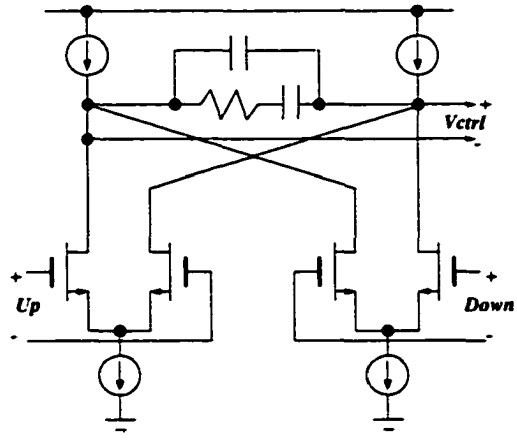


Figure 6.14 Fully differential phase detector

spikes on the power supply and the ground, creating less power-supply induced jitter; (3) It has the ability to accept PECL level differential input data RX+ and RX- directly, no interface circuit needed; (4) The D flip-flop, which produces $D1$, also implements the resample function between incoming data and its clock. Thus, the sampler is already combined into this phase detector itself. No additional

sampler circuit is needed.

Corresponding to the structure of the PD, the charge pump used here is also fully differential, as shown in Figure 6.15. In order to achieve good jitter performance, the values of the current sources are chosen to be 20uA. The loop filter is: C1=30pF, R=5K, C2=3p.



Up	Down	Vctrl
0	0	no change
0	1	ramp down
1	0	ramp up
1	1	no change

Figure 6.15 Fully differential charge pump

B. Voltage-Controlled-Oscillator Several critical parameters of the PLL, such as speed, timing jitter, spectral purity and power dissipation, strongly depend on the performance of the VCO. Ring oscillators usually have the advantages of small die size and simple structure, and are more suitable for integration on a standard process. The VCO implemented in this design uses a differential structure not only in the signal path but also in the control voltage path to suppress the power supply coupled noise. Current steering techniques are again applied to reduce the generation of switching noise. The whole VCO is a 3-stage ring oscillator using the delay cell shown in Figure 6.16.

By doing s-plane pole-zero analysis, the relationship of oscillation frequency and the differential control voltage V_{ctrlD} is approximated as follows:

$$\omega = \frac{\sin(\pi 5)}{C_L}\sqrt{\mu_n C_{OX}(\frac{W}{L}_1)(I_{BP} + \frac{1}{2}\sqrt{\mu_n C_{OX}(\frac{W}{L}_{C1})\sqrt{I_{SS}}V_{ctrlD}})} \tag{6.1}$$

where C_L is the total load capacitance at the output nodes.

From the above equation, when $V_{ctrlD} = 0$, the center frequency is given by:

$$\omega_0 = \frac{\sin(\pi 5)}{C_L}\sqrt{\mu_n C_{OX}(\frac{W}{L}_1)I_{BP}} \tag{6.2}$$

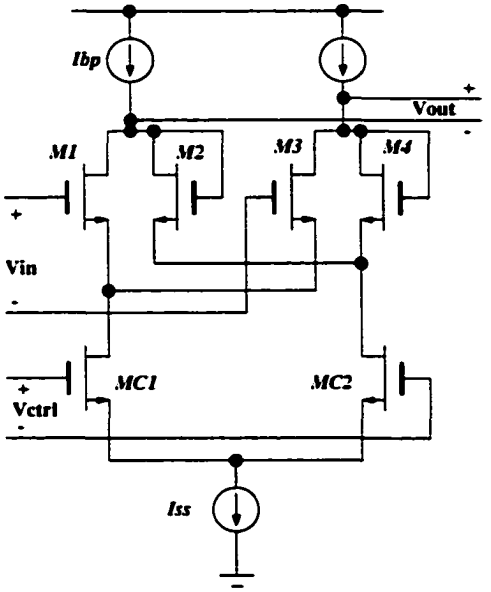


Figure 6.16 Fully differential VCO delay cell

The center frequency ω_0 can be set by choosing the appropriate value for I_{BP}. The variation of frequency is controlled by the differential control voltage V_{ctrlD}. The total bias current I_{SS} determines the gain of the VCO.

6.3.2.3 Simulation results

This PLL clock recovery circuit has been simulated with HSPICE using $HP0.5\mu m$ CMOS single-poly triple-metal process and a 1GHz package model. The power supply is 3.3V. Since in real fibre channel gigabit applications the incoming data is fully differential, a fully differential 8B/10B encoded data stream generated by a C program is used as the input data sequence. In order to do a comparison with a conventional single-ended approach, a design similar to that in [63] is simulated and its noise performance is set to be the reference (it is defined to be 100%).

Figure 6.17 shows the recovered clock spectrum. In Figure 6.18(a), the total supply current of proposed structure is shown. As a comparison, Figure 6.18(b) shows the total supply current of the single-ended counterpart.

From Figure 6.18, it is clearly shown that the current spikes generated from the fully differential structure are around 10mA, while in the single-ended approach, current spikes are as large as 70mA. Moreover, the power dissipation of this structure is about 100mW, which is only 70% of the single-end digital PLL clock recovery counterpart.

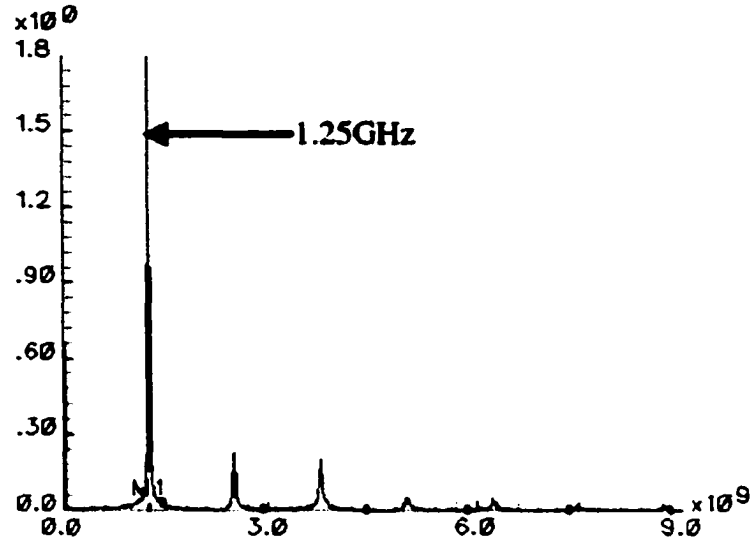


Figure 6.17 Recovered clock spectrum

6.3.2.4 Summary

As the second approach, a fully differential analog current steering technique is proposed to implement a Gigabit PLL clock recovery circuit. High noise immunity phase detector and VCO circuits improve jitter performance of the clock recovery circuit at GHz speed, particularly when operating in a noisy power supply environment. Simulation results show that the noise of this structure is about 60% that of traditional single-ended approaches. Moreover, since working at a reduced swing, the power dissipation is also smaller. This analog design method can be applied to other high speed integrated systems with minor modifications.

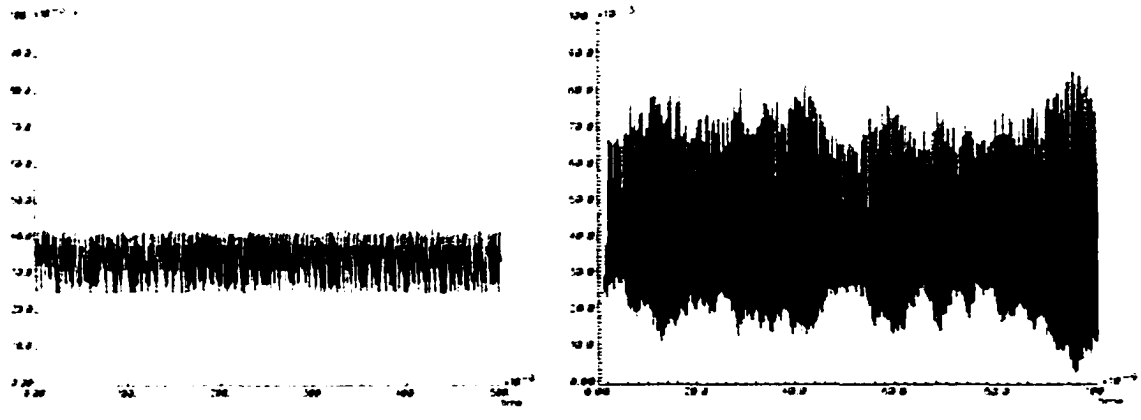


Figure 6.18 Comparison of power supply current of two approaches

6.4 Experimental Results

6.4.1 Experimental results of PLL clock generator

The transmitter PLL discussed in Section 6.2 is implemented with $HP0.5\mu m$ CMOS technology. The chip die photo is shown in Figure 6.19. As mentioned in Section 6.2, this PLL is a multi-phase clock generator with $\times 2$ frequency synthesis.

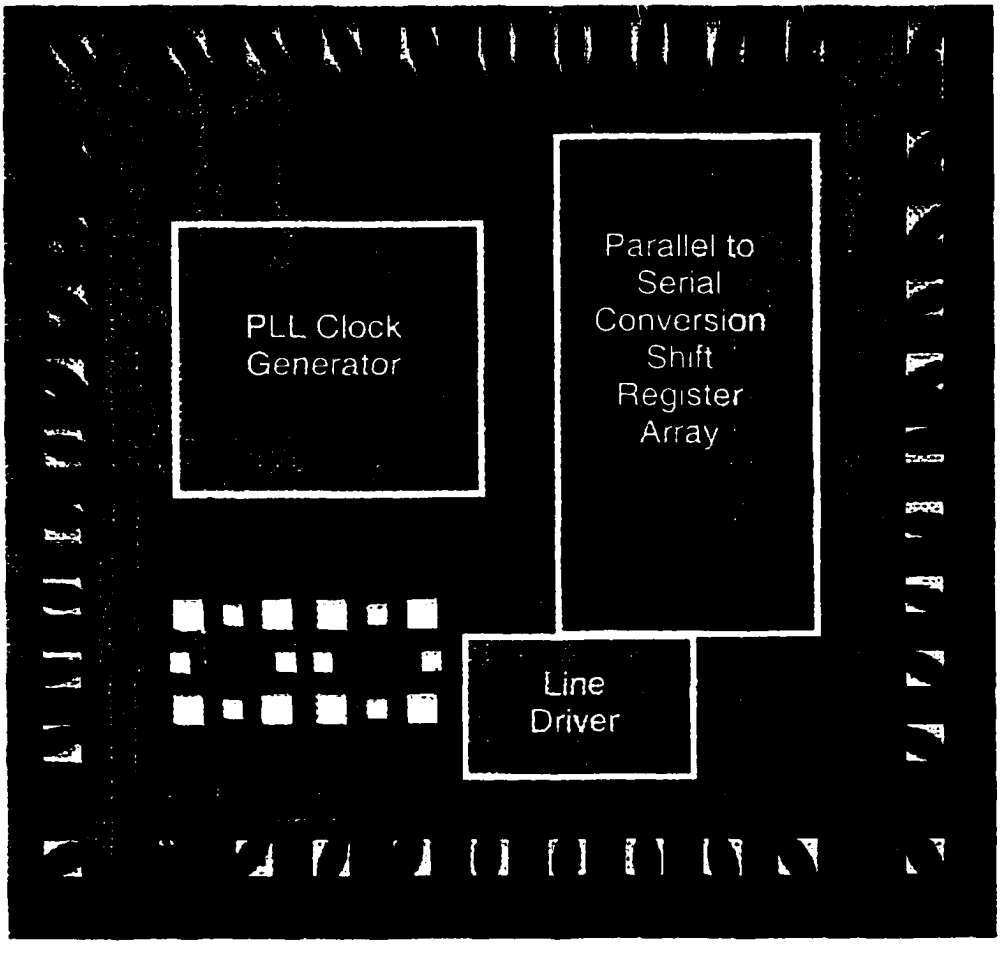


Figure 6.19 Die photo of transmitter

To observe the PLL outputs, three pins were used to output clock phases $\phi 1$, $\phi 3$, and $\phi 5$. The input reference comes from a signal generator or a crystal of $106.25MHz$. The outputs should be at $212.5MHz$. Shown in Figure 6.20 are the output waveforms corresponding to the input. This result was obtained with Tektronics TDS784D four channel sampling oscilloscope. Its input bandwidth is 1GHz.

From this result, it is clear that the PLL functions as expected. Three phase alternating output

clocks $\phi 1$, $\phi 3$, and $\phi 5$ at $212.5MHz$ were observed. Notice that the rising and falling edges of the pulses are not sharp. This is due to the package that slows down the edges.

Further investigation on the output frequency stability was performed by Wavecrest^{TM} time-interval analyzer DTS-2075. The result is shown in Figure 6.21. The performance of this PLL is summarized in Table 6.2.

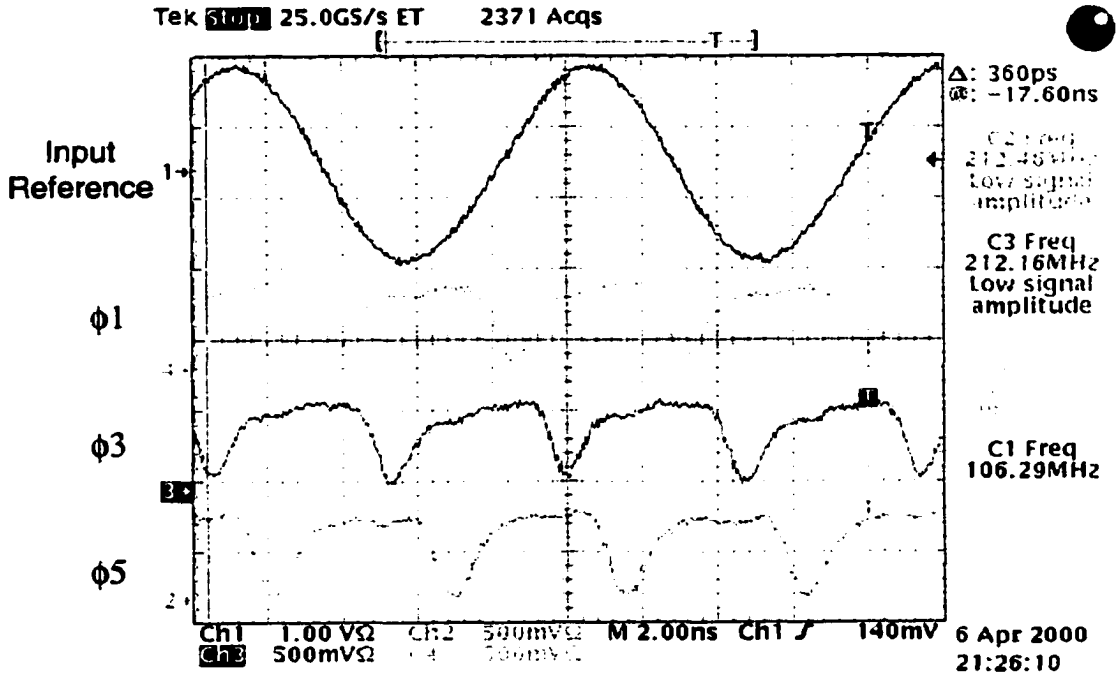


Figure 6.20 Input and output waveforms of transmitter PLL clock generator

Table 6.2 Transmitter PLL die summary

Fabrication process	HP0.5μm CMOS
Power supply	3.3V
Active area	$0.65 \times 0.8mm^2$
Operation frequency	$2 \times 106.25MHz = 212.5MHz$
Frequency stability	$\sigma = 14KHz$
Power dissipation	Core: 60mW, With buffer: 150mW

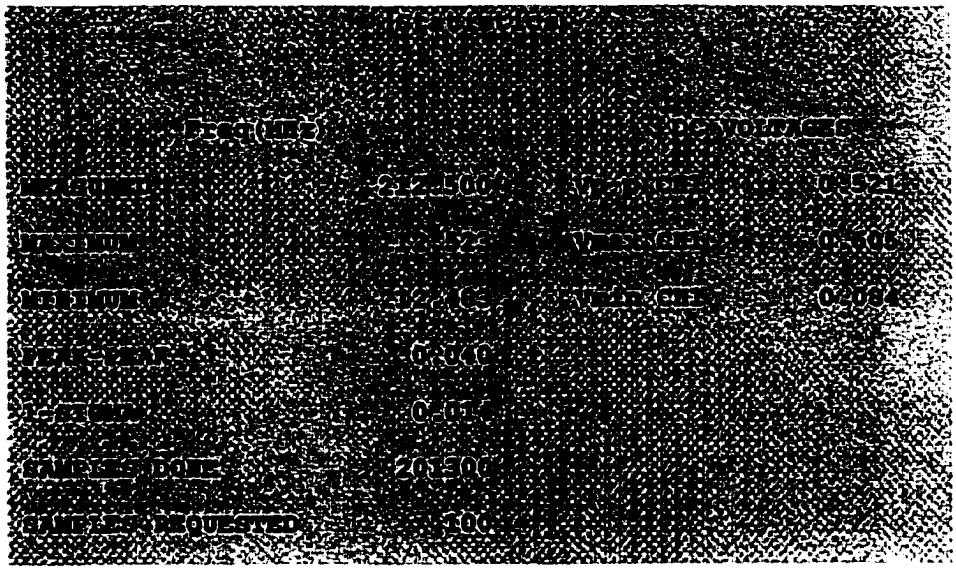


Figure 6.21 PLL output frequency stability

6.4.2 Experimental results of PLL clock recovery circuit

The clock recovery PLL discussed in Section 6.3.1 is implemented with HP0.35μm CMOS technology. The chip die photo is shown in Figure 6.22. This clock recovery circuit works at 1.25GHz.

To directly observe the recovered clock at 1.25GHz off-chip is extremely difficult because: (1) Under such high speed, it is very hard for the IC package and PCB to maintain the signal quality due to inevitable noise coupling, reflection, etc. (2) There needs to be a strong on-chip driver to drive the off-chip cable and package under such high speed. This will complicate the design, which is basically not necessary. Therefore, a $\div 20$ frequency divider is usually added at the output of the recovered clock. The off-chip clock is at $62.5MHz(1.25GHz \div 20)$. This speed greatly relaxes the testing.

Show in Figure 6.23 is the recovered clock waveform. It is captured by HP83480A communication analyzer, which has very high input bandwidth. The frequency is exactly at $62.5MHz$.

To investigate the recovered clock jitter performance, the output is fed into the $Wavecrest^{TM}$ DTS-2075. It can perform several types of statistical analysis on the input. Two of these results are shown in Figure 6.24 and Figure 6.25.

Figure 6.24 shows the jitter histogram of the recovered clock period. Ideally, $62.5MHz$ corresponds to a period of 16,000ps. It can be seen that the jitter distribution is very close to a normal distribution. This indicates that the primary noise has the characteristic of white Gaussian noise. For the latest 100 samples, the period mean value is 16,000.5ps. The r.m.s. jitter is 19.9ps and the peak-to-peak jitter

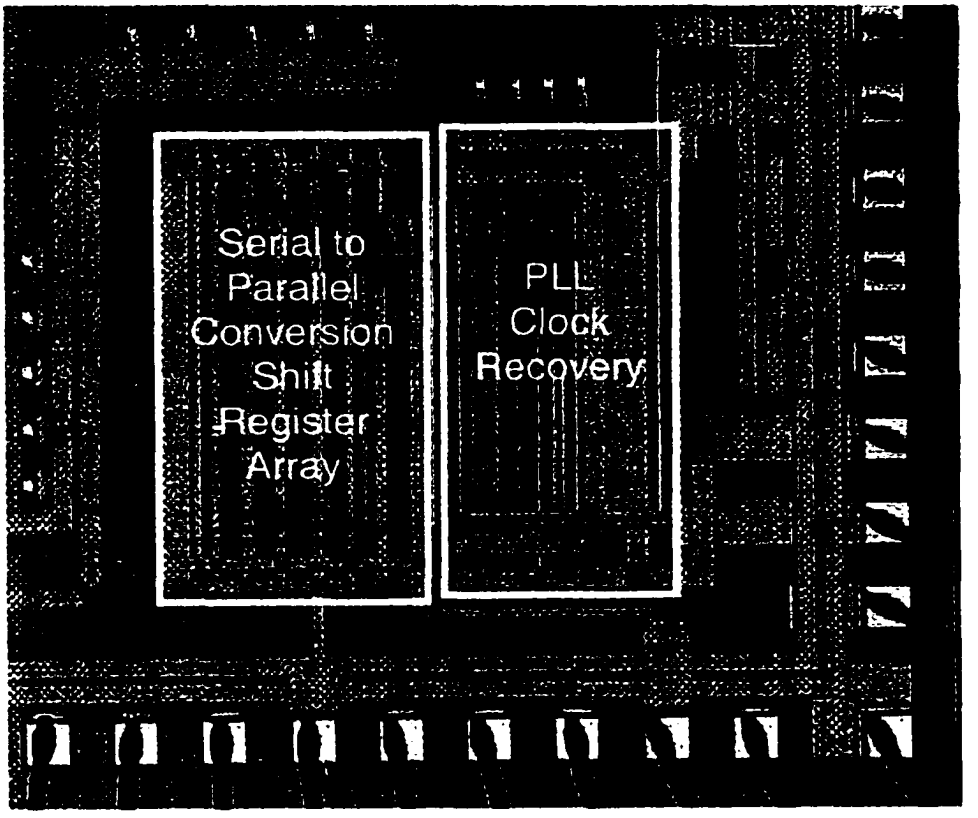


Figure 6.22 Die photo of receiver

is 109.3ps, which is within the specification. For the overall jitter, the r.m.s value is 22ps. Figure 6.25 shows the stability of jitter performance. The x-axis is elapsed time. It shows the peak- to-peak jitter and r.m.s jitter varying with time. The statistic shows that the r.m.s jitter is always around 20ps and the peak-to-peak jitter is always around 100ps. Both meet the specification.

Characteristics of this chip are summarized in Table 6.3.

Table 6.3 PLL clock recovery die summary

Fabrication process	HP0.35μm CMOS
Power supply	3.3V
Active area	$0.3 \times 0.9mm^2$
Operation frequency	1.25GHz
Jitter	$\sigma \approx 20ps$, peak-to-peak $\approx 100ps$
Power dissipation	160mW

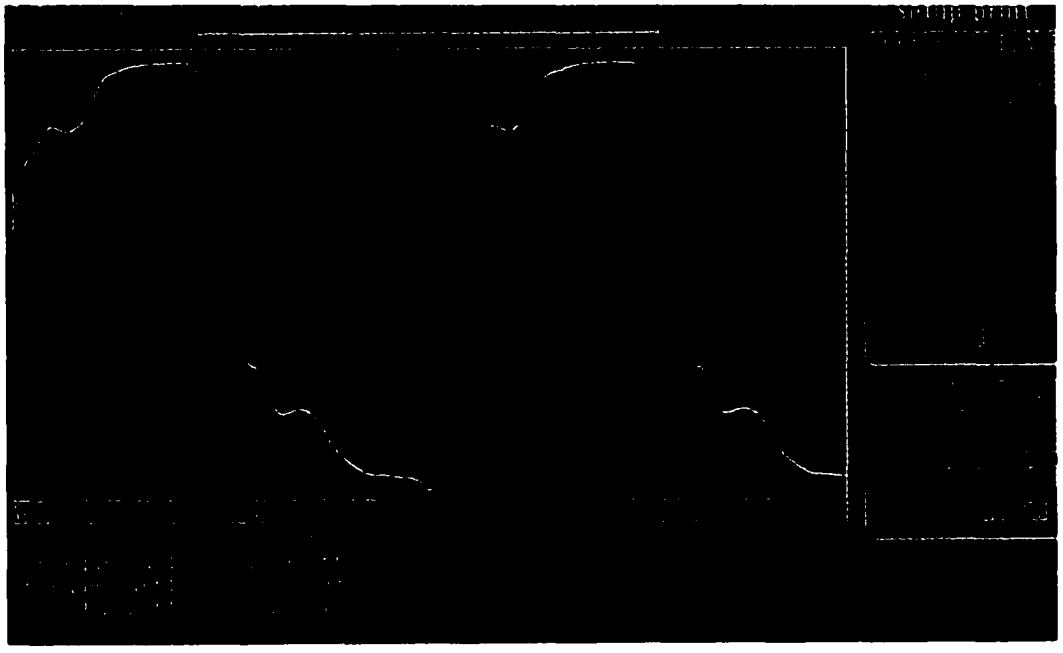


Figure 6.23 Recovered clock at $62.5MHz$

6.5 Summary

The applications of PLLs in Gigabit fibre channel transceivers were explored. The main goal is to use CMOS technology to achieve low power and low cost. Fabricated chips show that the designed circuits have met required performance specifications. For PLLs as clock emphasized for recovery on the receiver side, an analog PLL based on current steering technique is also proposed to reduce the power supply coupling noise and to improve the robustness of the circuit.

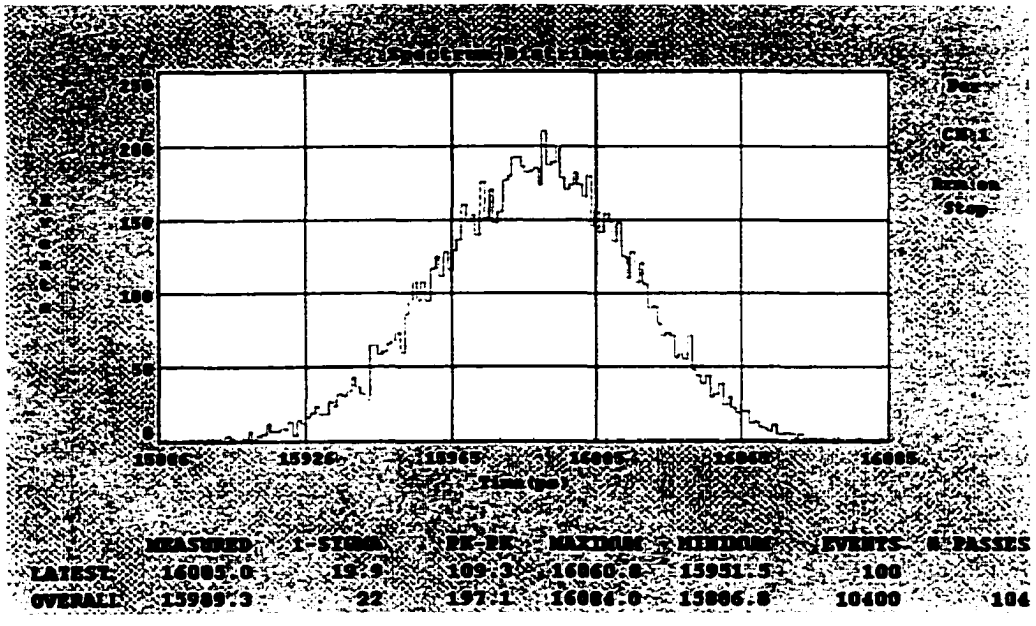


Figure 6.24 Output clock jitter histogram

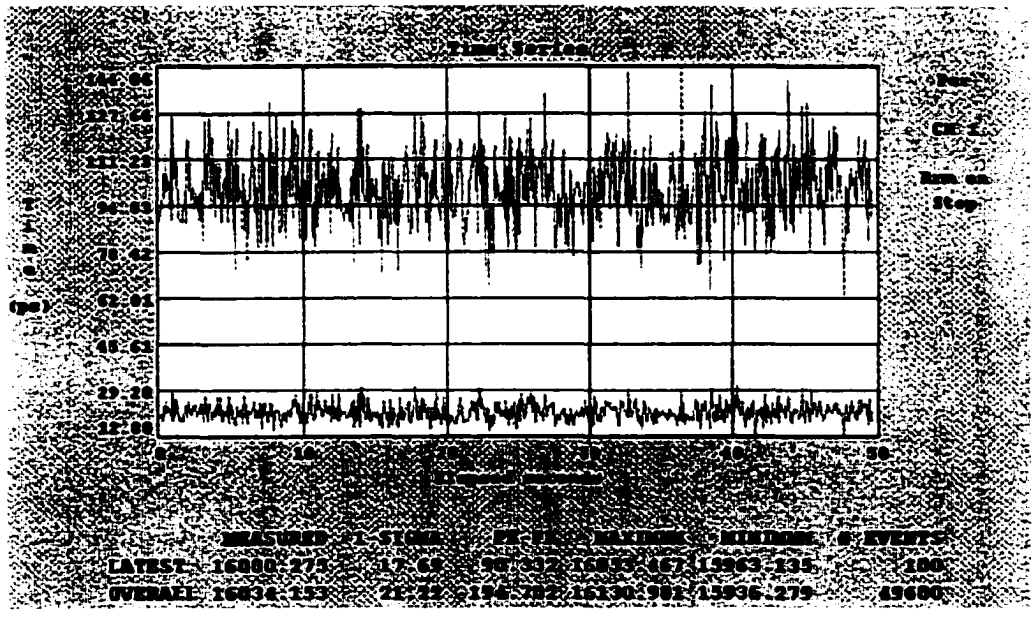


Figure 6.25 rms jitter and peak-to-peak jitter vs. time

CHAPTER 7 LOW JITTER PRECISE DELAY MULTI PHASE CLOCK GENERATOR FOR TIME-INTERLEAVED SYSTEMS

In previous chapters, we have discussed PLL/DLL as clock generators. The focus was on how to improve the jitter performance. Besides providing a clock with reduced jitter, a PLL incorporating a ring oscillator and a DLL with a delay line also generate clock phases that are uniformly distributed within one clock period. This is shown in Figure 7.1. These multi-phase clocks are very useful in many applications. By appropriately combining these output phases, they can generate timing signals that meet rather complicated system timing requirements. In multi-channel architectures, these clocks are necessary to implement time-interleaved functions.

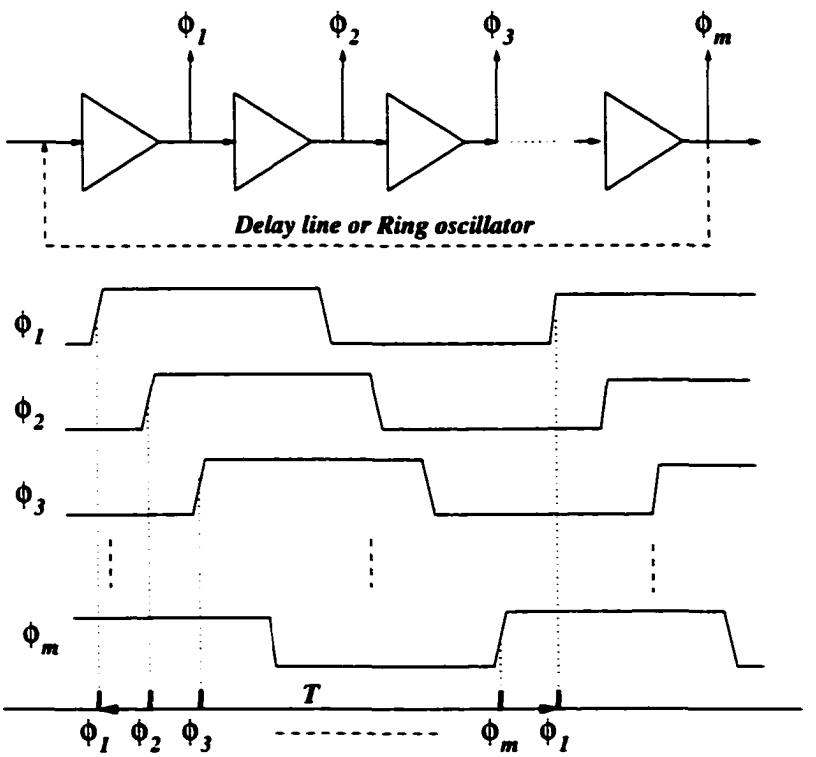


Figure 7.1 Multi-phase clocks in ring oscillator or delay line

In this chapter, PLL/DLL as multi-phase clock generator for time-interleaved application will be discussed. Unlike previous chapters, there are two issues that need to be addressed:(1) *skew*: delay mismatch between clock phases; (2) *jitter*: single clock phase stability.

In traditional PLL or DLL multi phase clock generators, there is no active control of the delay of individual phase. They rely only on the matching characteristics to maintain the uniformity of phase distributions. In this chapter, an idea for calibrating channel delay mismatch in such a clock generator is originated. This idea is mainly composed of two parts: (1) **architecture**: additional delay cells are added in the output path of each phase, in which the delay can be individually adjusted. This allows the fine adjustment of each phase; (2) **calibration**: with the new architecture, the fine adjustment of each phase is done as follows: besides for the PLL or DLL main loop, additional feedback loops are added into the architecture to dynamically control the delay of each stage to maintain the delay accuracy between all phases.

The advantages of the proposed method are: (1) It realizes on-chip real time calibration. It can track the circuit behavior as well as environmental change dynamically and respond accordingly; (2) It is very cost efficient. The idea of the new architecture does not increase the hardware significantly. And the calibration does not involve a complicated procedure and circuit. It is a simple but elegant solution [64].

7.1 Motivation

As the demand of circuit speed as well as resolution keeps increasing, a time interleaved (or multi-channel) architecture becomes an attractive approach to meet performance requirements. The very obvious advantage of such an architecture is the speed relaxation by the mechanism of parallelism. Suppose the final speed is F_s, if m channels are used, then each channel only needs to work at the speed of $\frac{F_s}{m}$. Thus the circuit design on each individual channel is greatly relaxed.

Any such time-interleaved architecture needs a precise delay multi-phase clock generator to generate pulses p_1, p_2, ... , p_m for each channel (shown in Figure 7.2). These pulses can be easily derived from the phases $\phi 1$, $\phi 2$, $\phi 3$, ..., ϕm in Figure 7.1. Ideally, the delay between any adjacent phases should be exactly the same, i.e. $\phi 1$, $\phi 2$, ..., ϕm are distributed uniformly within one channel clock period T, where $T = m \times \frac{1}{F_s}$, $\Delta td1 = \Delta td2 = ... = \Delta tdm = \frac{1}{F_s}$. However, random error of the sampling instant $\phi 1$, $\phi 2$, ... ϕm is inevitable. This includes two types of error: (1) the phase random variation of every individual clock, i.e. jitter, which comes from various noise sources on chip; (2) the delay mismatch among $\Delta td1, \Delta td2, ..., \Delta tdm$, i.e. skew. This is due to device mismatch as well as channel

mismatch. Compared with jitter, delay mismatch is rather a static error. In the output spectrum of a time-interleaved system, jitter raises the noise floor, reducing SNR (Signal-to-Noise-Ratio), while skew introduces tones, reducing SFDR (Spurious-Free-Dynamic-Range). Both errors are critical error sources that limit the performance of time-interleaved architectures.

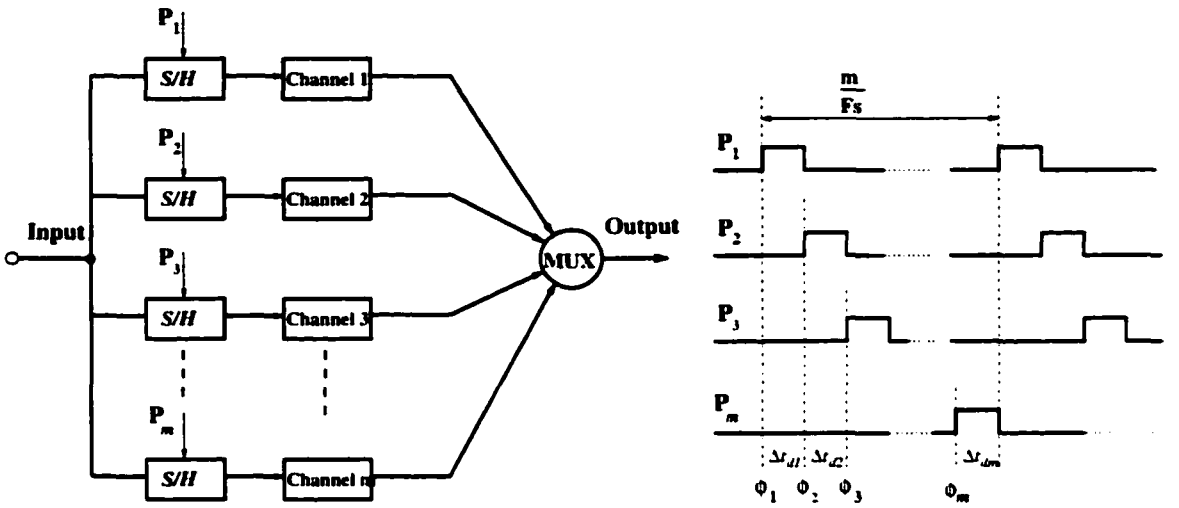


Figure 7.2 Time-interleaved system and required multi-phase clocks

Ring oscillator-based Phase-Locked-Loops or delay-line-based Delay-Locked-Loops are the most popular ways to generate such multi phase clocks. Numerous papers can be found dealing with jitter in PLLs or DLLs [46] [59] etc., and very low jitter (rms as low as ~10ps) integrated CMOS PLLs and DLLs have been reported recently [65] [66] [67]. By contrast, delay mismatch has not been addressed so far. In many cases, delay mismatch is the critical error that limits the performance of the whole time-interleaved system. Therefore, it is an important topic. The motivation of this project is to explore an effective way to calibrate the delay mismatch on-chip in such a multi-phase clock generator.

In the following sections, the delay calibration idea will first be introduced. Low jitter circuit design will then be discussed. As a proof of this idea, testing results on a prototype chip will be given, which not only show that this design achieves by far the best single channel jitter performance reported but also strongly support the proposed delay calibration scheme.

7.2 Proposed Delay Mismatch Calibration System Architecture

7.2.1 Prior Art — PLL and DLL based multi-phase clock generator architecture discussion

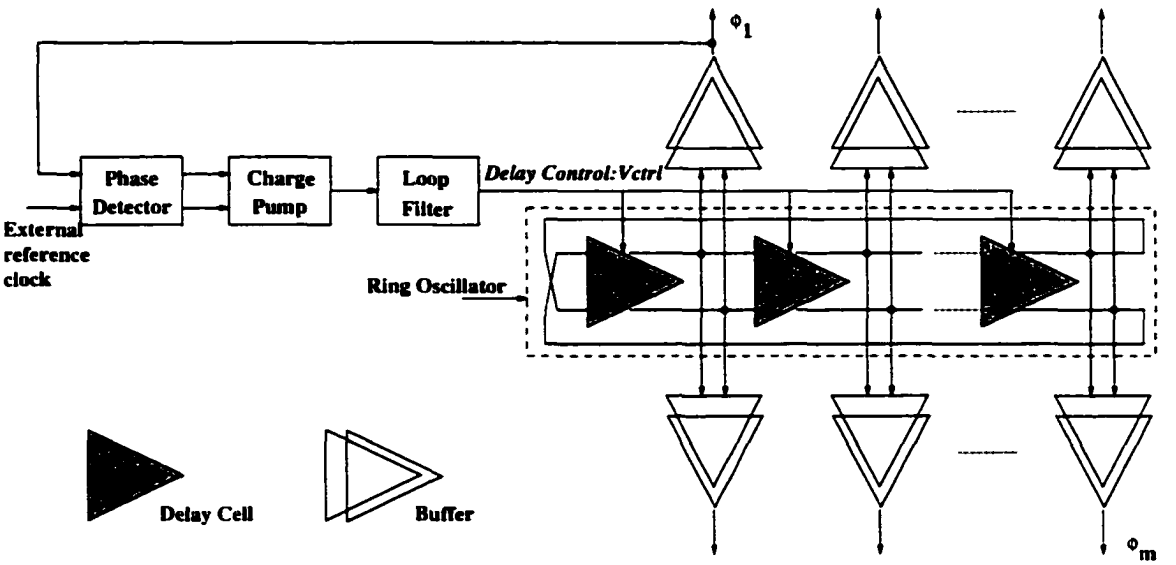
As an illustration, typical PLL and DLL based multi-phase clock generators are shown in Figure 7.3(a) and (b). In both systems, $\phi 1$, $\phi 2$, ..., ϕm are generated by appropriately buffering the output of the delay stage in the ring oscillator or in the delay-line. Usually $\phi 1$ is aligned to a rather stable external reference clock phase through the feedback loop. One observation of all reported PLLs and DLLs is that only one control voltage V_{ctrl} generated by the global loop is used to control the delay of every delay cell in ring oscillator or delay line. Theoretically, if every channel is perfectly matched with each other, the delay between any two adjacent stages is exactly the same. However, since channel mismatch always exists, using only one global control voltage V_{ctrl} cannot yield the same delay.

Based on the above discussion, in order to make the delay of every stage be the same despite mismatches, every delay should be adjusted independently without affecting the global loop. That is, for every stage, besides the global V_{ctrl}, there must be additional delay adjustment capability that enables the delay calibration. This is the principal idea of the proposed method.

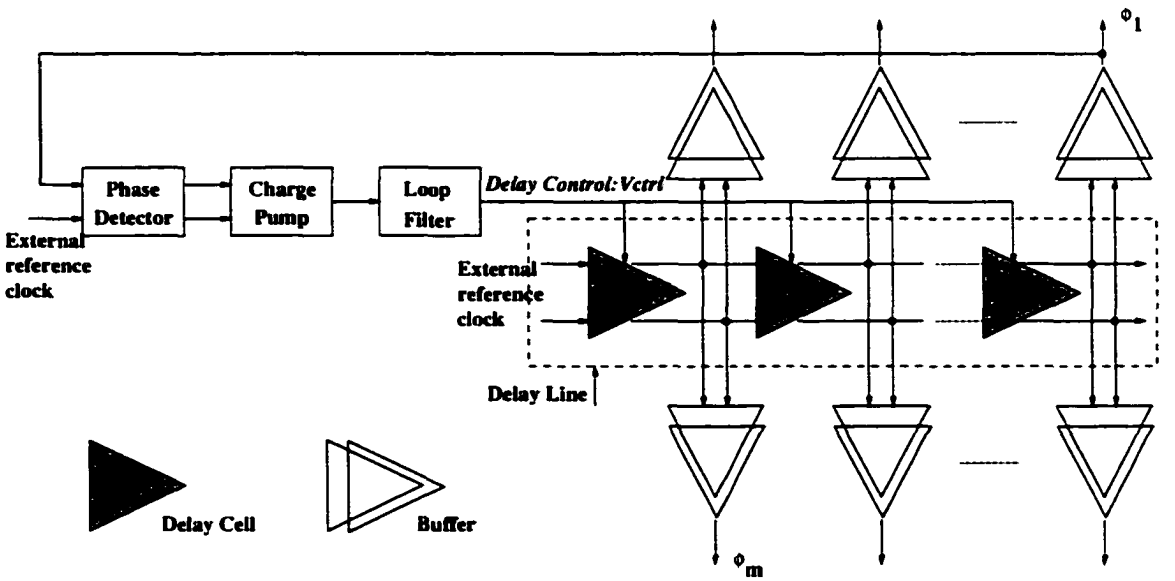
7.2.2 Delay calibration architecture — new ring oscillator and delay line

In order to not interfere with the global loop, the adjustment should be carried out outside either the ring oscillator or the delay line. A new ring oscillator as well as a delay line architecture with individual delay adjustment is shown in Figure 7.4. Figure 7.4(a) is the new ring oscillator which has the ability of delay adjustment of every single stage. Figure 7.4(b) is the corresponding delay line. Notice that the output buffer is included because every output clock phase needs to drive an on-chip circuit. For clarity, **one channel** is defined as one clock phase generator from the delay stage to the output buffer only, not including the circuit it needs to drive.

Unlike a traditional ring oscillator or delay line, here an additional delay cell is inserted between the major delay cell and the buffer in every channel. It is because of this additional delay cell that the independent calibration of every channel is possible. With this modification, for the main loop, V_{ctrl} is used as the global control voltage to align one output clock with the reference clock, while every channel delay can be adjusted individually by V_{ctrli}. Note that V_{ctrl1} is connected to V_{ctrl}. The reason for this will be discussed in the next section. With this new architecture, additional feedback loops can be introduced to calibrate the delay mismatch in real time.



a)



b)

Figure 7.3 Typical multi-phase clock generator (a) PLL (b) DLL

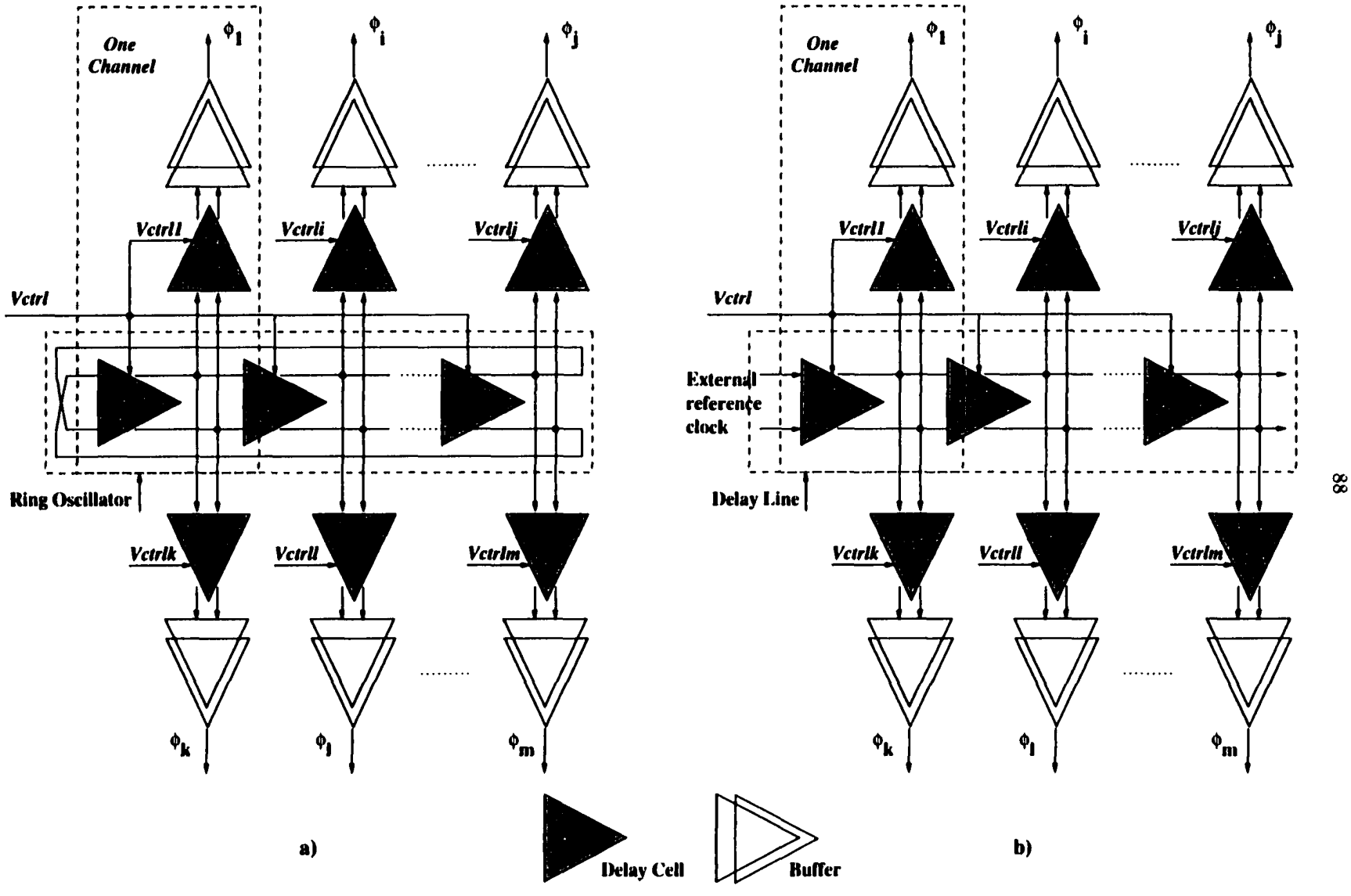


Figure 7.4 Architecture: (a) new ring oscillator (b) new delay line

7.3 Proposed Delay Mismatch Calibration

With the basic architecture proposed in the previous section, the next step is considering how to construct the calibration feedback loop to achieve our final goal.

Suppose there are 8 channels: $\phi 1$, $\phi 2$, ..., $\phi 8$. We define $\phi 1$ as the one which is phase aligned to the external reference clock. From another point of view, $\phi 1$ is the only phase that is calibrated in a traditional a PLL and DLL. In such PLL or DLL, the delay of any other phase $\phi 2$, $\phi 3$, ... $\phi 8$ is left uncontrolled but is passively depending upon the matching between all channels.

Since a feedback loop is used as the method of implementing real time calibration, the stability issue needs to be considered carefully. As mentioned above, $\phi 1$ is already calibrated by the main loop, thus it needs not be calibrated by an additional loop. That is why in the previous section V_{ctrl1} is the same as V_{ctrl}. Once $\phi 1$ is in lock with reference clock, it will work as the second level reference clock. From this level, the generation of next level of reference clock is shown in Figure 7.5: Δtd_{1-5} and Δtd_{5-1} are compared to get V_{ctrl5} to adjust $\phi 5$ (this is the third level where $\phi 1$ and $\phi 5$ are calibrated). Then, in the fourth level, Δtd_{1-3} and Δtd_{3-5} are compared to get V_{ctrl3} to adjust $\phi 3$, and Δtd_{5-7} and Δtd_{7-1} are compared to get V_{ctrl7} to adjust $\phi 7$. Finally, with the fourth level calibration, i.e. $\phi 1$, $\phi 3$, $\phi 5$, $\phi 7$ already calibrated, Δtd_{1-2} and Δtd_{2-3} are compared to get V_{ctrl2} to adjust $\phi 2$, Δtd_{3-4} and Δtd_{4-5} are compared to get V_{ctrl4} to adjust $\phi 4$, Δtd_{5-6} and Δtd_{6-7} are compared to get V_{ctrl6} to adjust $\phi 6$, Δtd_{7-8} and Δtd_{8-1} are compared to get V_{ctrl8} to adjust $\phi 8$. In this manner, feedback loop stability can be guaranteed because in each level, the calibration does not affect the lock of the previous level of feedback loops.

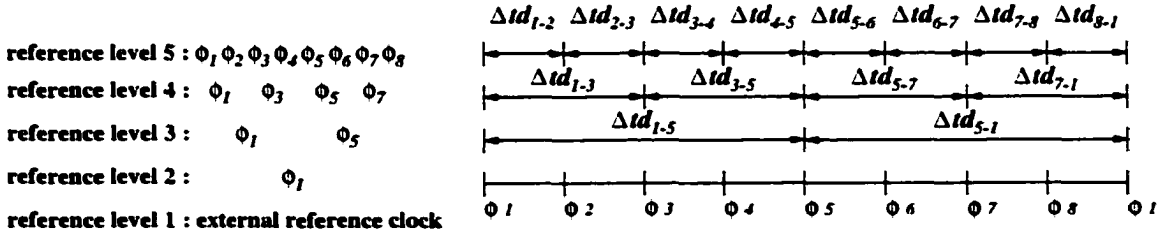


Figure 7.5 Calibration loop construction

It is worth mentioning that in the time domain, all levels of calibration are carried out **simultaneously**. Figure 7.5 illustrates that lock can be guaranteed in this manner. The final system architecture is shown in Figures 7.7 and 7.8. Figure 7.7 is the calibrated PLL, while Figure 7.8 is the calibrated DLL. Note that the DLL delay line is further modified. Here, the input of the next delay cell is not the

output of the previous delay cell but is the output from the buffer of the previous stage. The reason is that if there is no such modification, the distribution of $\phi 1$ to $\phi 8$ will have a delay offset coming from the extra buffer delay shown in Figure 7.6. This is a problem inherent in a delay line but not in a ring oscillator. In order to get uniformly distributed clock phases, such a modification is necessary.

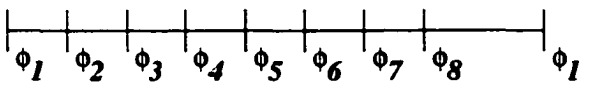


Figure 7.6 Illustration of the delay offset problem of DLL in Figure 7.4(b)

7.4 Low Jitter Circuit Implementation

7.4.1 Self-biased technique based on a differential delay cell with symmetric load

As discussed on Section 7.1, both precise delay and low jitter are very important for such multi-phase clock generators. The final performance cannot be evaluated if there is one without the other. Though no former research has been done in achieving precise delay, low jitter, which is the focus of this project is also very important. Low jitter circuit design in this project is based on the self-biased technique introduced by Maneatis in 1996 [59].

From its name, self-biased technique uses the control voltage of the ring oscillator or delay line itself to generate all of the internal bias voltages. It avoids the necessity for external biasing, which can require special band gap circuit or other. The key idea behind self-biasing is that it allows circuits to choose the best operating bias levels dynamically according to the state they are in. Moreover, based on a differential delay cell with symmetric load, PLLs and DLLs designed with this method have the advantages of a fixed damping factor, fixed loop bandwidth to operating frequency ratio (i.e. loop bandwidth is adaptive to the working frequency), broad frequency range, input phase offset cancellation, and low sensitivity to power supply variation.

7.4.1.1 Delay cell with symmetric load

The delay cell, shown in Figure 7.9, is a fully differential pair with a load called a symmetric load. There are two control taps in this cell: VBP and VBN, which will be generated by the replica bias circuit. With replica biasing, VBP actually defines the lower voltage swing limit of the delay cell. The effective resistance of the load elements changes with VBP. It has been shown that these load elements have broad control over delay and high dynamic supply noise rejection. VBN is generated

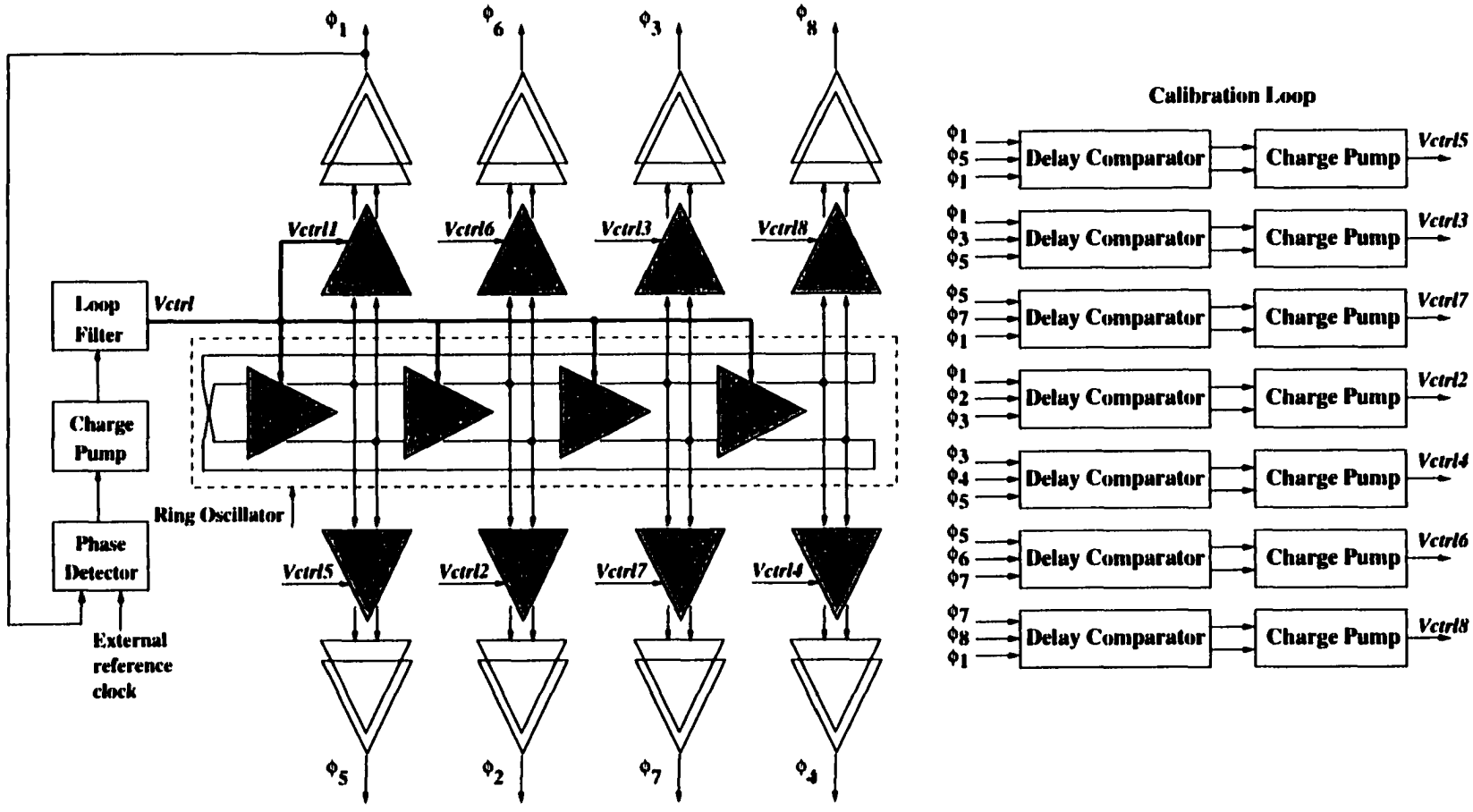


Figure 7.7 Delay calibrated PLL multi-phase clock generator system architecture

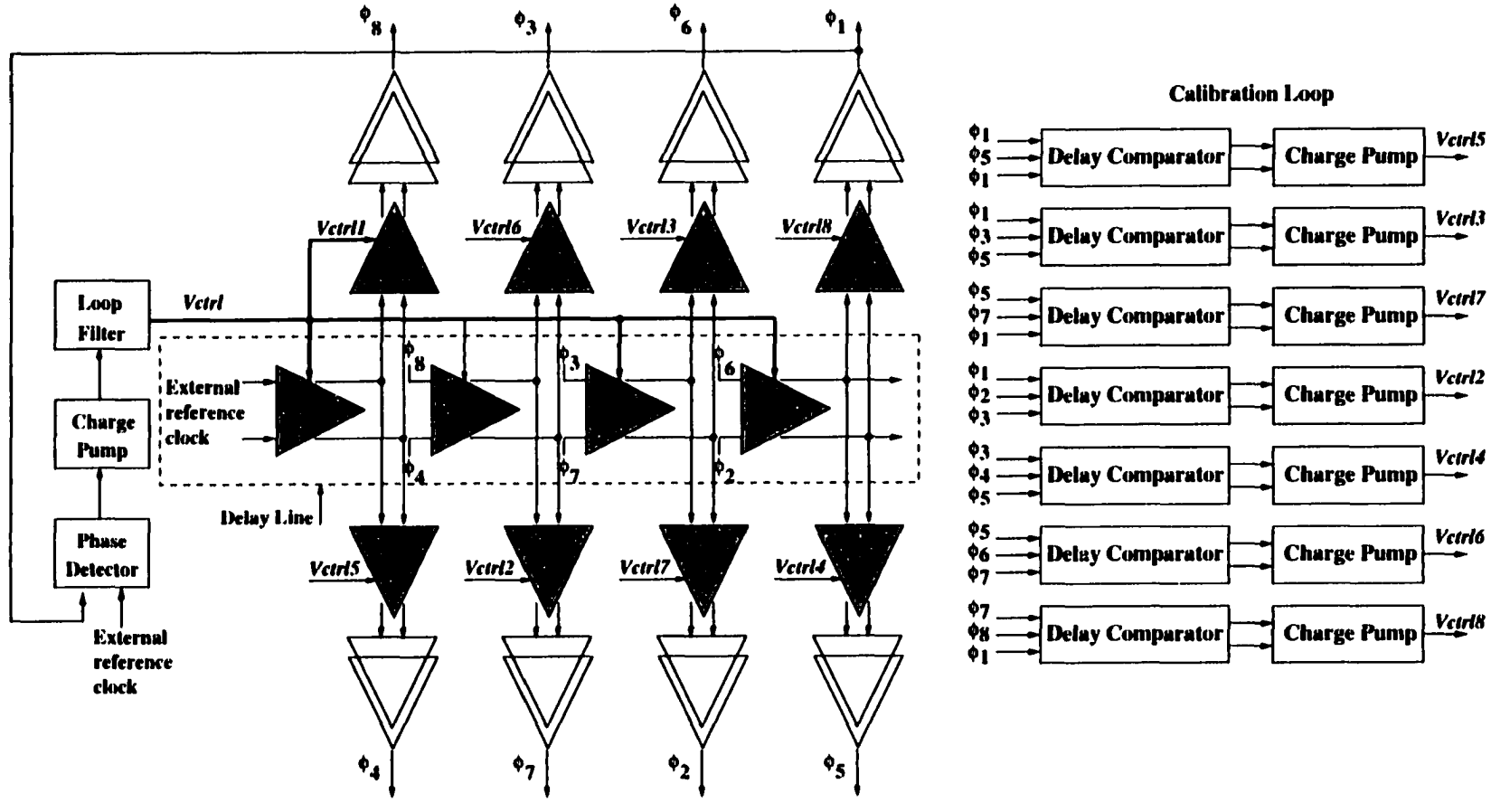


Figure 7.8 Delay calibrated DLL multi-phase clock generator system architecture

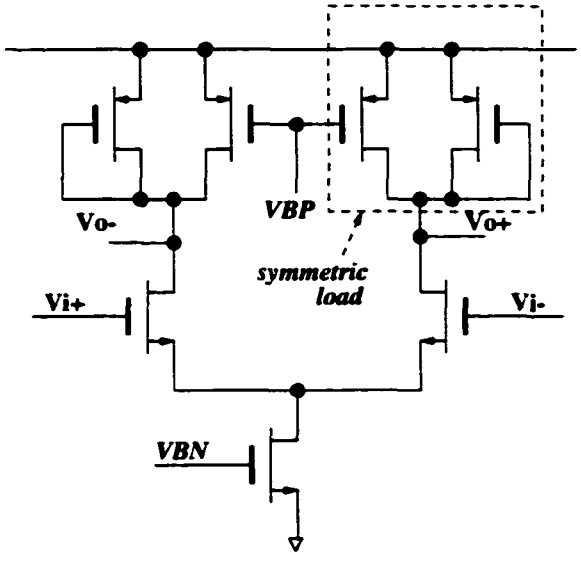


Figure 7.9 Delay cell with symmetric load

dynamically to compensate for the tail transistor drain variation, achieving the effective performance of a cascode current source.

7.4.1.2 Self-bias circuit—replica feedback bias

The bias generator, shown in Figure 7.10, uses a half delay cell replica and an amplifier to force the lower swing limit of the delay cell to be exactly the same as the delay cell control voltage V_{ctrl}. When there is variation on the power supply, it continuously adjusts the delay cell bias current so that the swing can remain constant. Therefore, the delay cell has good power supply noise rejection.

With this delay cell and replica bias, adaptive bandwidth PLL and DLL can be constructed with low jitter (especially power supply induced jitter) performance.

7.4.2 Traditional PLL based multi-phase clock generator without calibration

As the first design, a four channel PLL based clock generator was designed to work from $10MHz \sim 300MHz$. This self-biased PLL is composed of a dead zone free phase-frequency detector, two charge pumps, a four-stage ring oscillator with bias generator and four output buffers. The system block diagram is shown in Figure 7.11. This PLL has two characteristics: (1) Under different working frequencies, the damping factor ξ remains the same; (2) the loop bandwidth ω_n to the working frequency ω ratio is a constant, i.e. $\frac{\omega_n}{\omega} = C$, which means the loop bandwidth is adaptive to operating frequency.

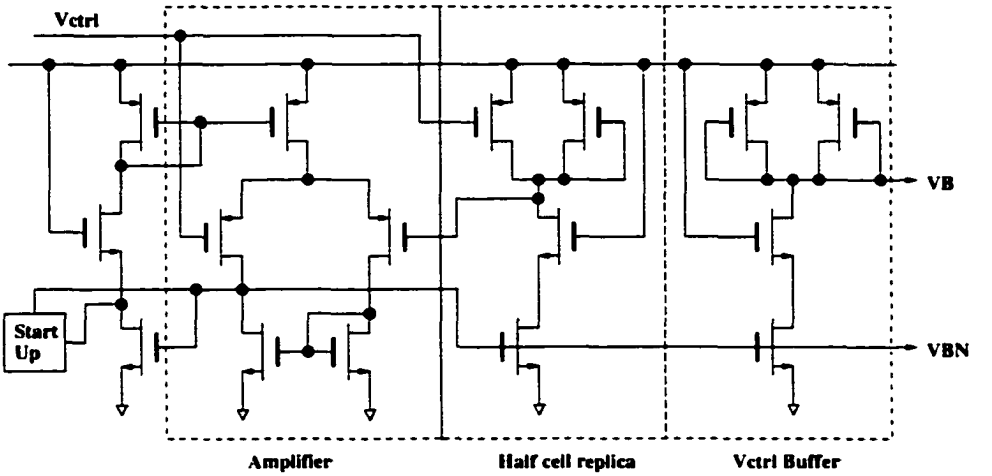


Figure 7.10 Bias generator

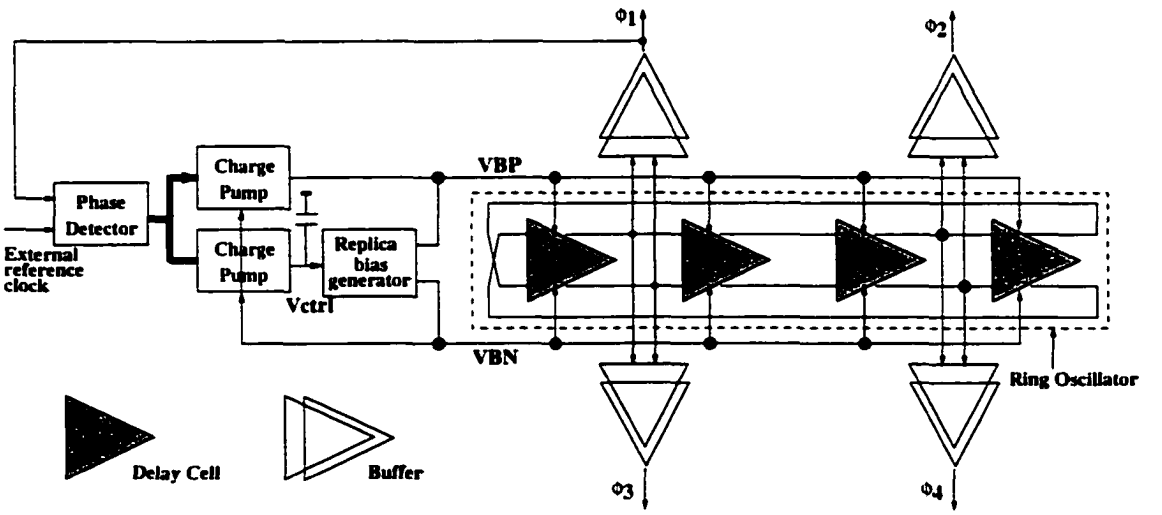


Figure 7.11 Traditional PLL multi-phase clock generator based on self-bias technique

There are many advantages due to the above characteristics: (1) During capture, the PLL will slew toward lock at the fastest rate possible using the maximum charge pump current without jeopardizing the stability. Thus, this PLL will exhibit much faster locking time. (2) The frequency range at which this PLL can work is very broad. (3) Input tracking jitter due to power supply coupled noise is low. Detailed analysis can be found in [59].

In the following section, building blocks such as the phase-frequency detector, charge pump and output buffer will be introduced.

7.4.2.1 Phase-Frequency-Detector

The dead zone of a phase detector is the range of input phase differences for which the PLL takes no corrective action. In order to avoid dead zone effectively, both *Up* and *Down* outputs should be asserted for an equal and short period of time when the PLL is in lock. Otherwise, it will take a finite phase difference before a large enough pulse could turn on the charge pump, which leads to a dead zone. A phase and frequency detector which is dead-zone free is shown in Figure 7.12. Unlike traditional PFD, this PFD asserts *Up* and *Down* signals when in lock without sacrificing the speed.

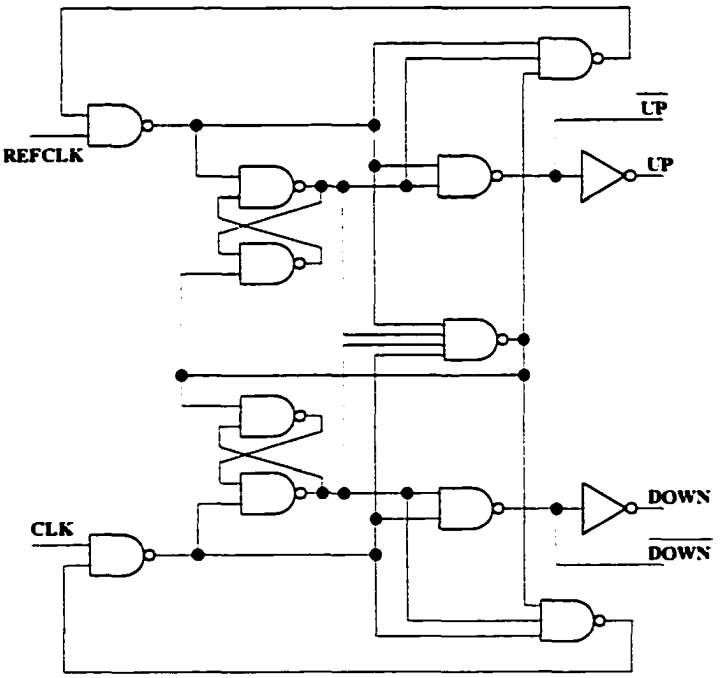


Figure 7.12 Dead zone free phase-frequency detector

7.4.2.2 Zero offset charge pump

The function of a charge pump is to convert phase difference information into control voltage by charging or discharging the loop filter. It is very important that the charging and discharging currents be well matched. When in-lock, there should be no leakage to the loop filter, i.e. zero offset. This is done by the circuit shown in Figure 7.13.

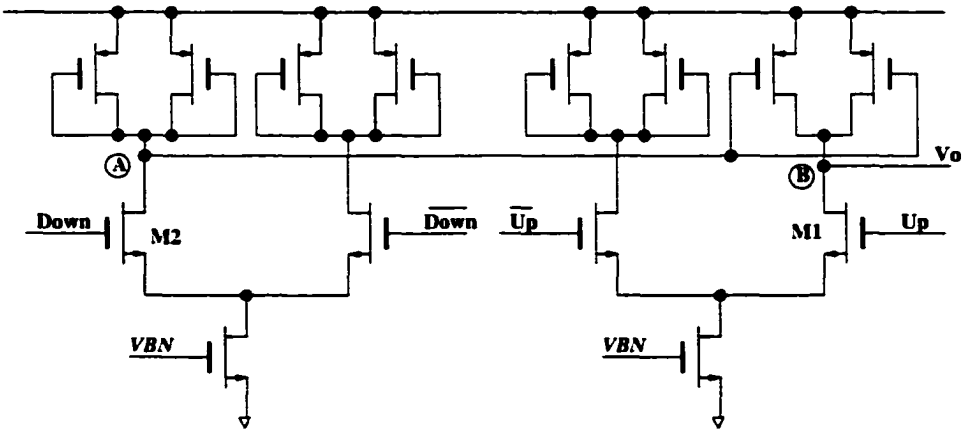


Figure 7.13 Zero offset charge pump based on symmetric delay cell and self bias technique

By constructing the charge pump from the symmetric load delay cell, it can be guaranteed that the charging and discharging currents are well matched. When *Down* is asserted, the left source coupled pair will behave like the half buffer replica in the bias circuit and produce V_{ctrl} at the current mirror node A. The PMOS devices in the right source coupled pair will then have V_{ctrl} at their gates (node A) and drain (node B) at the same time so they source the same bias current that is sunk by M1 when *Up* is on. In doing so, it overcomes the finite output impedance effect of the charge pump and has zero phase offset.

7.4.2.3 Output buffer

Since the clock generator needs to drive the internal circuit, it should have enough driving capability and be single-ended. The major function of the output buffer is to convert the differential output from the ring oscillator to single ended and enhance the driving capability of the converted single ended clock signal. The circuit is shown in Figure 7.14.

This buffer also has a duty cycle correction function. This first stage of the output buffer is also constructed from a symmetric load delay cell, and it uses the same bias voltage VBN as the delay cell, so that it receives the correct common mode input voltage level. It provides signal amplification and a DC bias point for the successive PMOS pairs. The PMOS pair converts the signal into single-ended output and this output is further amplified by additional inverters. Because the two stages of amplification are differentially balanced, the opposing differential input transitions have equal delay to the output, thus 50% duty cycle is maintained.

7.4.3 Traditional DLL based multi-phase clock generator without calibration

As a comparison, the second design is a DLL that shares the basic building blocks of the above PLL. What is different is the top level architecture. The designed DLL is shown in Figure 7.15.

In this DLL application, a phase detector can provide sufficient information because the DLL does not have a frequency component other than the reference frequency. Shown in Figure 7.16 is the simplified circuit of a phase detector which may be compared with the phase-frequency detector. shown in Figure 7.12.

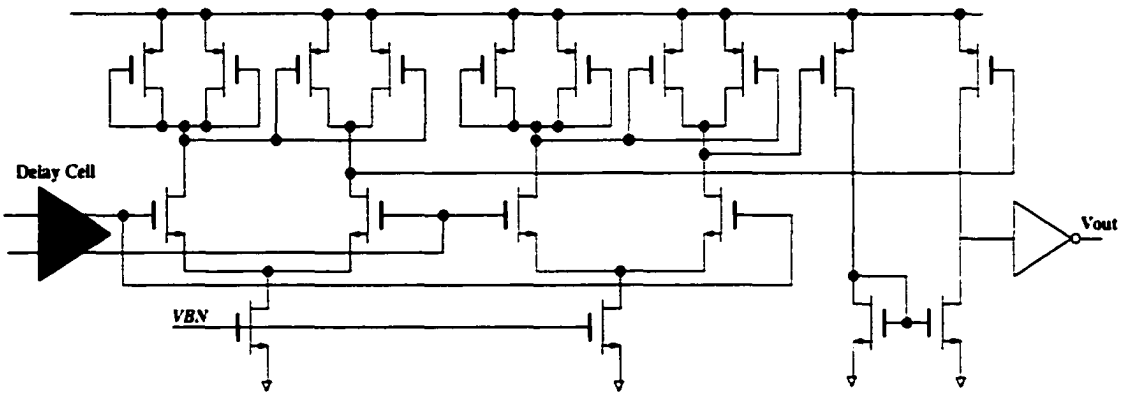


Figure 7.14 Duty cycle correction output buffer

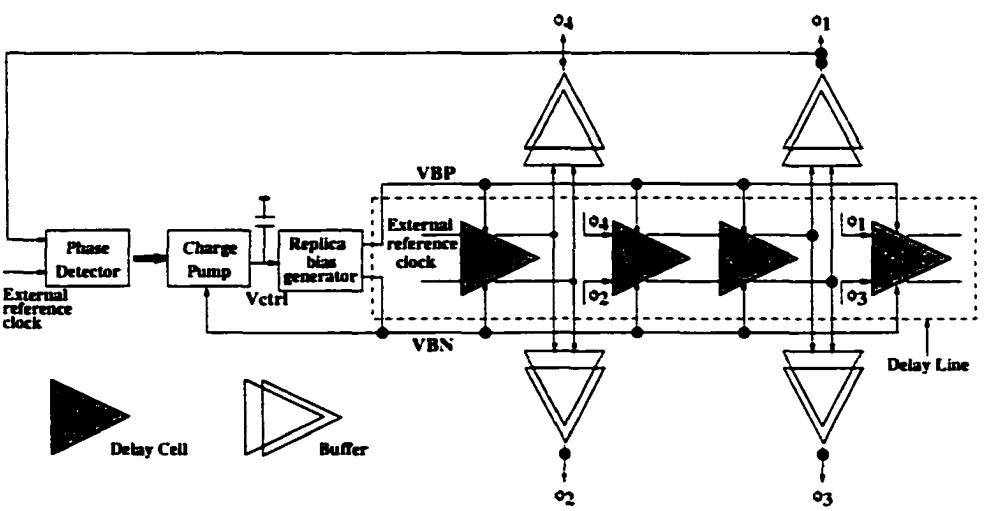


Figure 7.15 Traditional DLL multi-phase clock generator based on self-bias technique

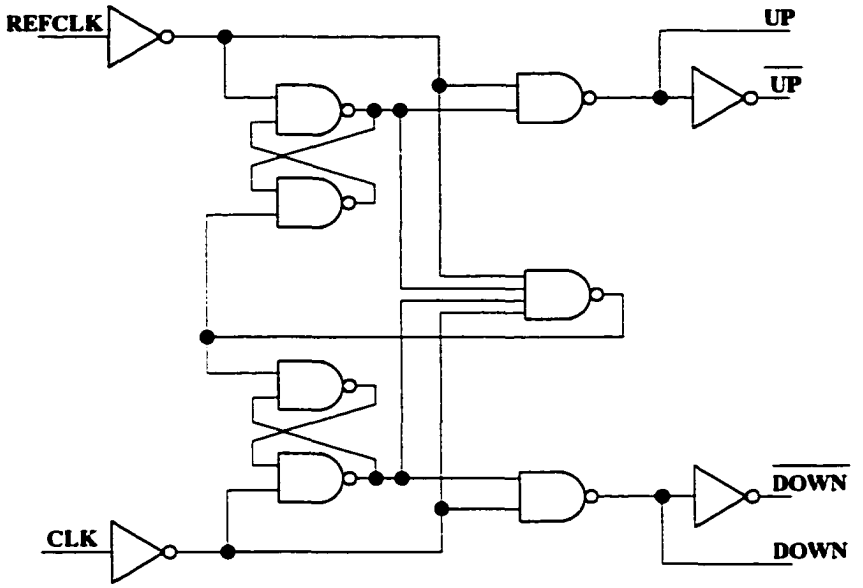


Figure 7.16 Dead zone free phase detector

7.4.4 Traditional PLL and DLL design summary

So far, all the building blocks as well as the system architectures of PLL and DLL have been discussed. The whole system is based on one special cell, symmetric load delay cell, and one technique, self biased technique. This resulting system has fixed damping factor ξ under all working frequencies and fixed loop bandwidth ω_n to working frequency ω ratio. As a result, on one hand, the system has very fast tracking behavior and broad speed range. On the other hand, it has low input tracking jitter, especially due to power supply variation.

Our new delay calibration scheme will depend on these two low jitter PLL and DLL architectures to achieve precise delay time between different phases.

7.4.5 New delay calibrated PLL multi-phase clock generator

As shown in Figure 7.7, delay calibration is achieved by inserting an additional delay cell, of which the delay can be adjusted individually while independently. A calibration loop is then formed by sensing the delay differences between certain stages (as shown in Figure 7.5) to generate control voltages V_{ctrli} accordingly, so as to adjust the delay of additional delay cell. In the following sections, the circuit implementation of the calibration loop will be given.

7.4.5.1 Additional delay cell

By studying the basic delay cell and the duty cycle correction output buffer (Figure 7.14), one observation is that by small modification, the first output stage can be used as the additional delay cell, which has independent delay tuning tap. Therefore, the output buffer and the additional delay cell are combined together, which is shown in Figure 7.17. In this case, the first stage of the output buffer is also used as the extra delay cell. No additional circuitry is needed. Notice that the modified first stage is exactly two basic symmetric load delay cells.

With this new output buffer, the system can be constructed as follows: for $\phi 1$, since it is already calibrated by the main loop, the output buffer still uses the one shown in Figure 7.14. For $\phi 2$, $\phi 3$, $\phi 4$, the output buffer uses the new one in Figure 7.17.

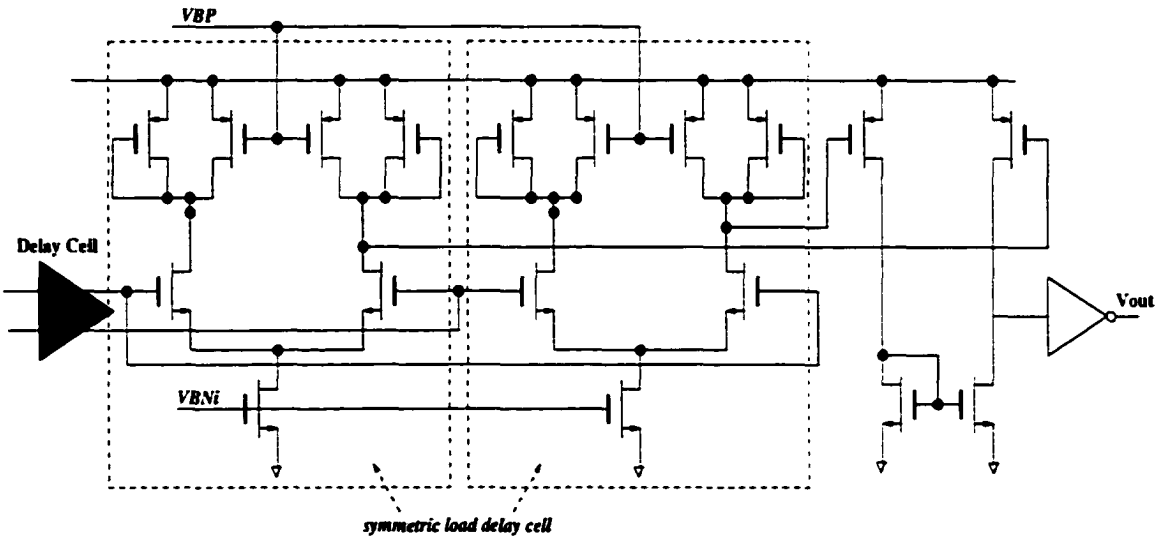


Figure 7.17 New output buffer for delay calibration

7.4.5.2 Delay comparison

Delay calibration is based on correctly sensing two delays, Δtd_{i-j} and Δtd_{j-k}, then comparing them and generating V_{ctrlj} accordingly, to adjust the phase of the clock in the middle ϕj until $\Delta td_{i-j} = \Delta td_{j-k}$. The purpose of the delay sensing circuit is to get Δtd and this comparison is then sent to the charge pump and loop filter. This rising edge triggered circuit is shown in Figure 7.18. The detail of the calibration loop is shown in Figure 7.19. The bias generator is the same as in Figure 7.10.

The final delay calibration PLL system is shown in Figure 7.20

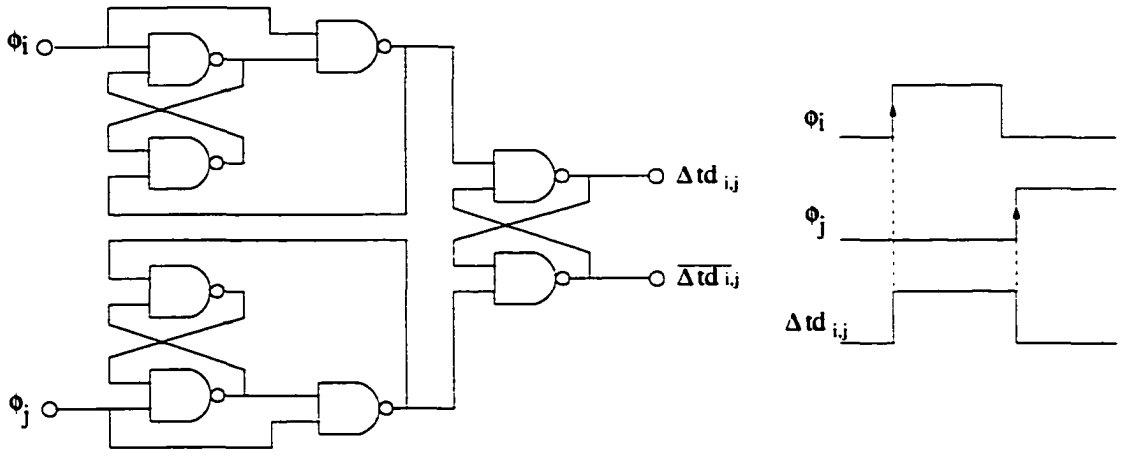


Figure 7.18 Delay sensing circuit

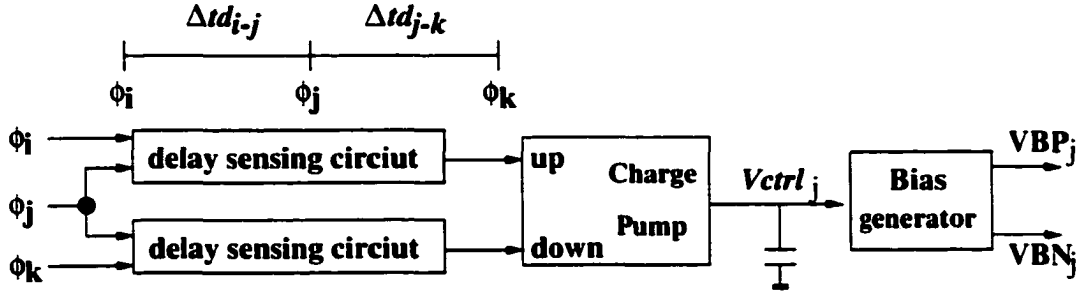


Figure 7.19 Calibration loop

7.4.6 New delay calibrated DLL multi-phase clock generator

Using building blocks discussed so far in the previous sessions, a calibrated DLL based multi-phase clock generator is shown in Figure 7.21. Notice that all the blocks are the same as those inside of the calibrated PLL.

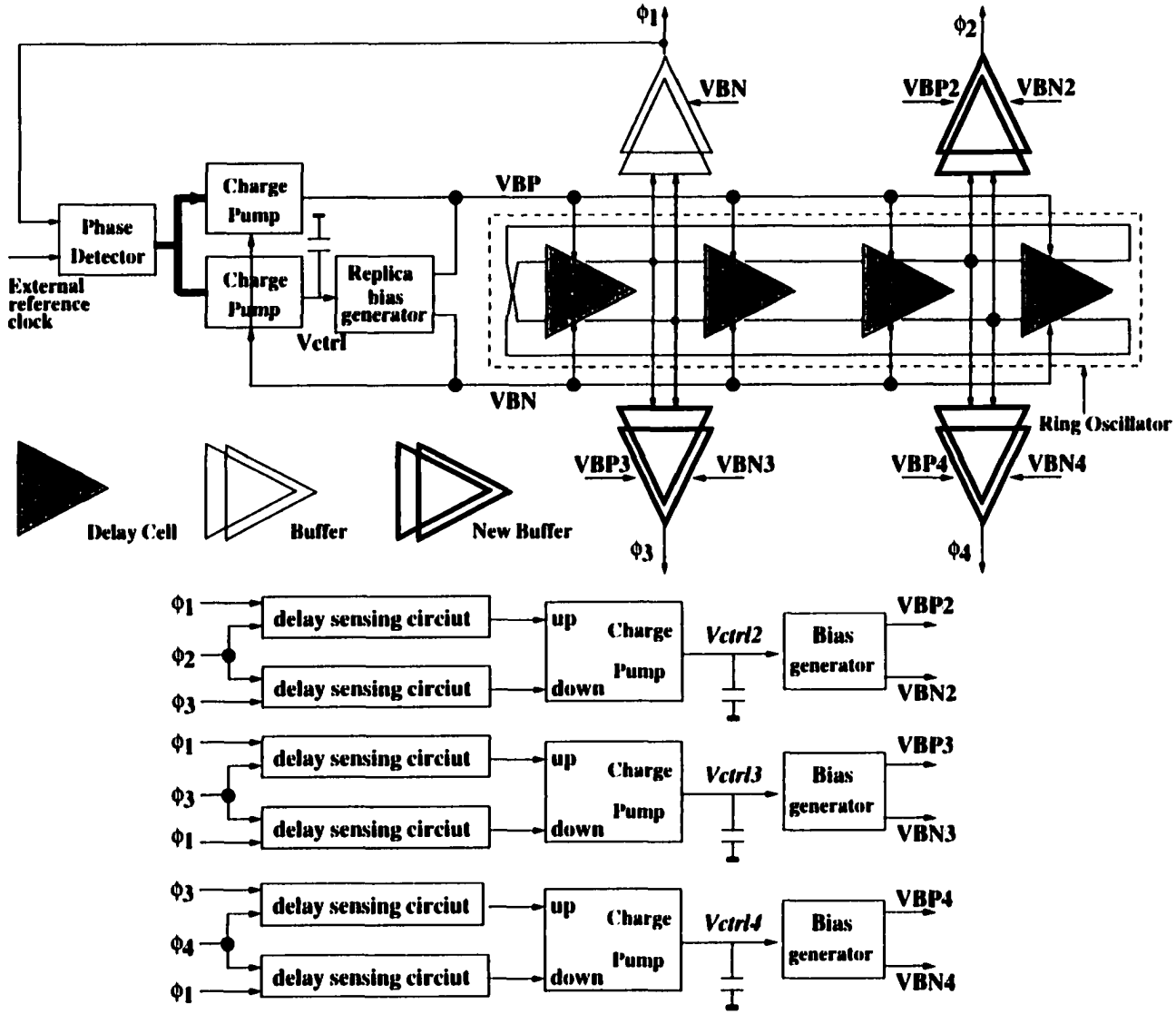


Figure 7.20 Delay calibrated PLL multi-phase clock generator

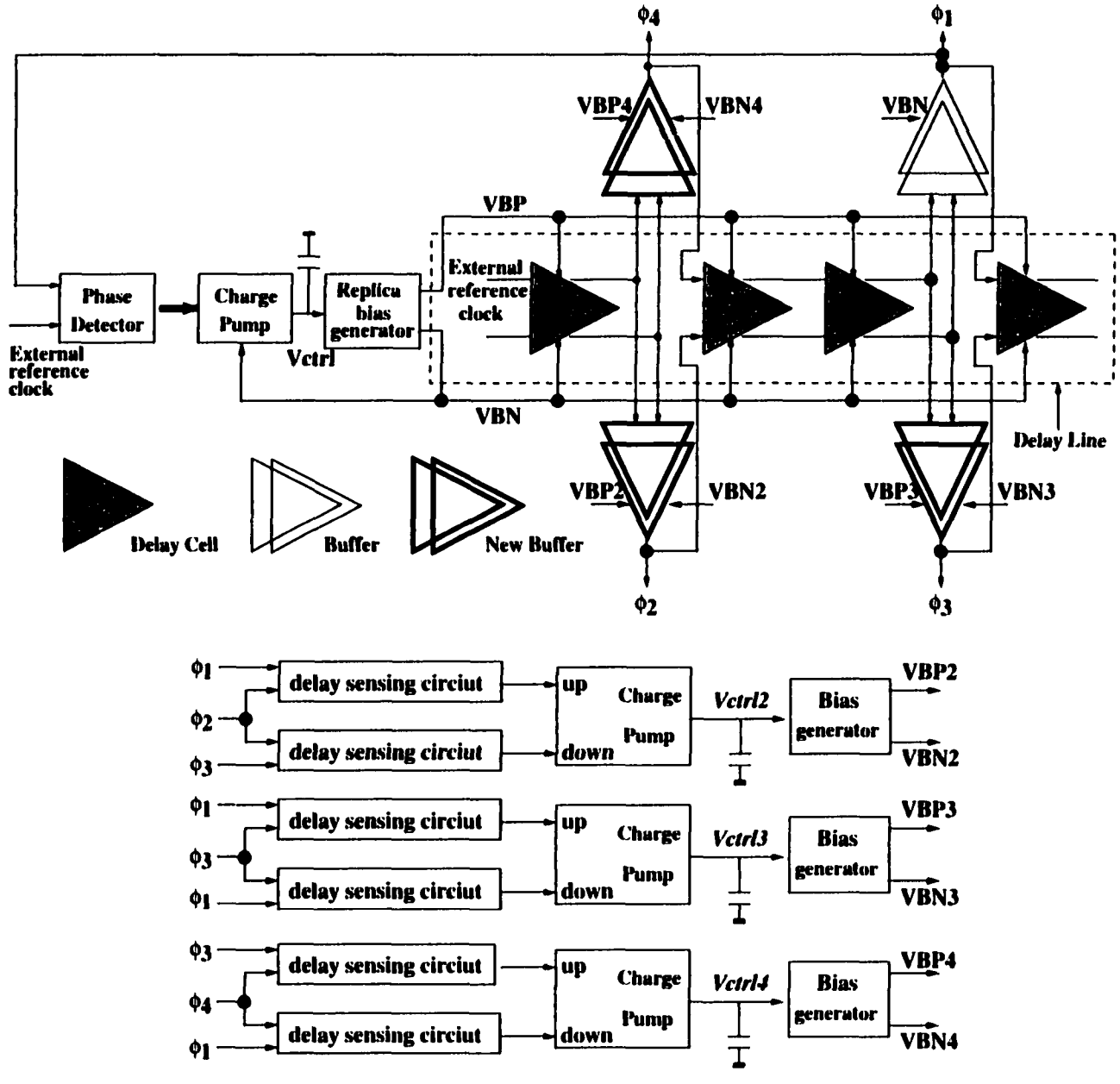


Figure 7.21 Delay calibrated DLL multi-phase clock generator

7.5 Experimental Results

Based on the idea introduced in the previous sections, four experimental multi-phase clock generators were implemented with $TSMC0.25\mu$ CMOS process: (1) Traditional PLL based clock generator without calibration; (2) New PLL based clock generator with delay calibration; (3) Traditional DLL based clock generator without calibration; and (4) New DLL based clock generator with delay calibration. With these four combinations, the jitter performance of PLL and DLL can be compared, and the effectiveness of the delay calibration idea can be validated. In order to make the four designs comparable, all designs have four channels targeting a channel speed of $10MHz \sim 300MHz$, and they share as many building blocks as possible. The prototype chip die photo shown in Figure 7.22 shows the boundary of the four designs.

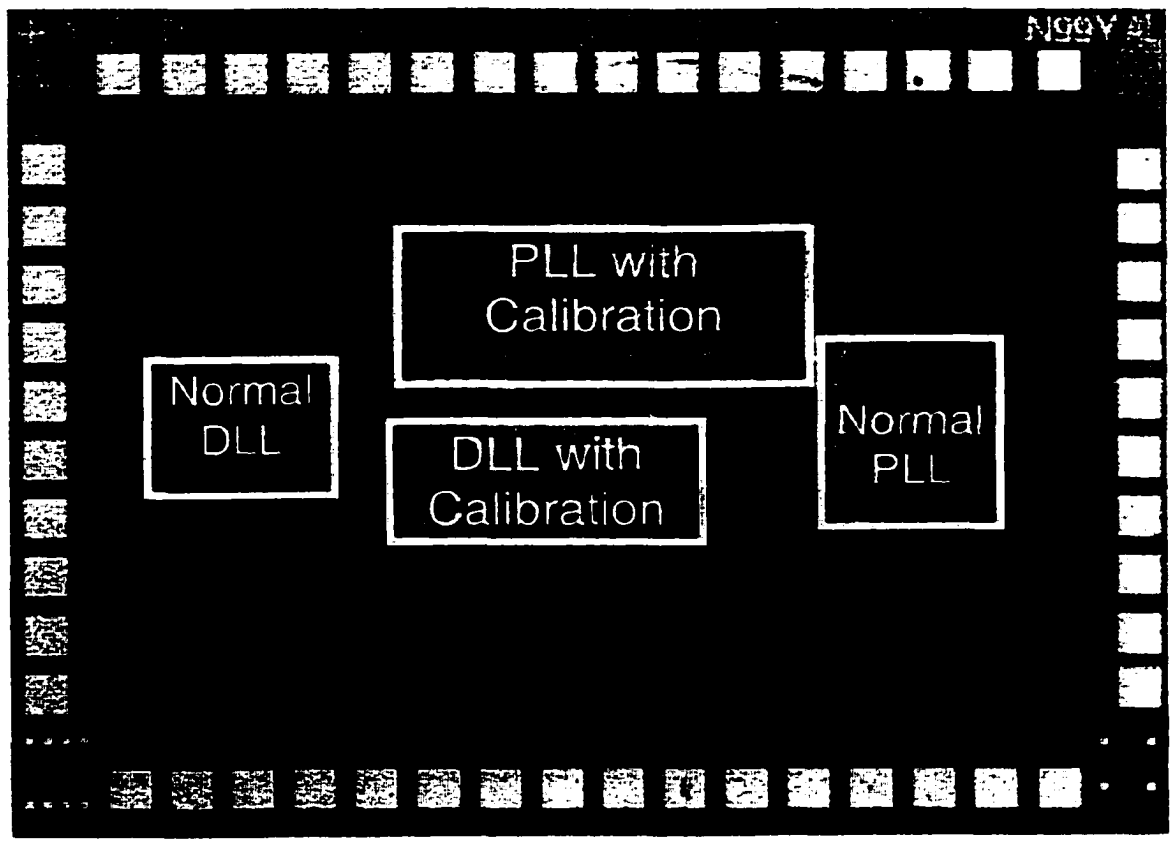


Figure 7.22 Prototype die photo — PLL and DLL multi-phase clock generator

7.5.1 Testing PC board design

Since this is a low jitter design, noise is the major concern during testing. A 4-layer board was designed which provides good ground and power planes to the circuit. Much effort was put on reducing the coupling between signal and signal, signal and power, input and output, etc. On board impedance matching was also done to reduce the reflection. The board is shown in Figure 7.23.

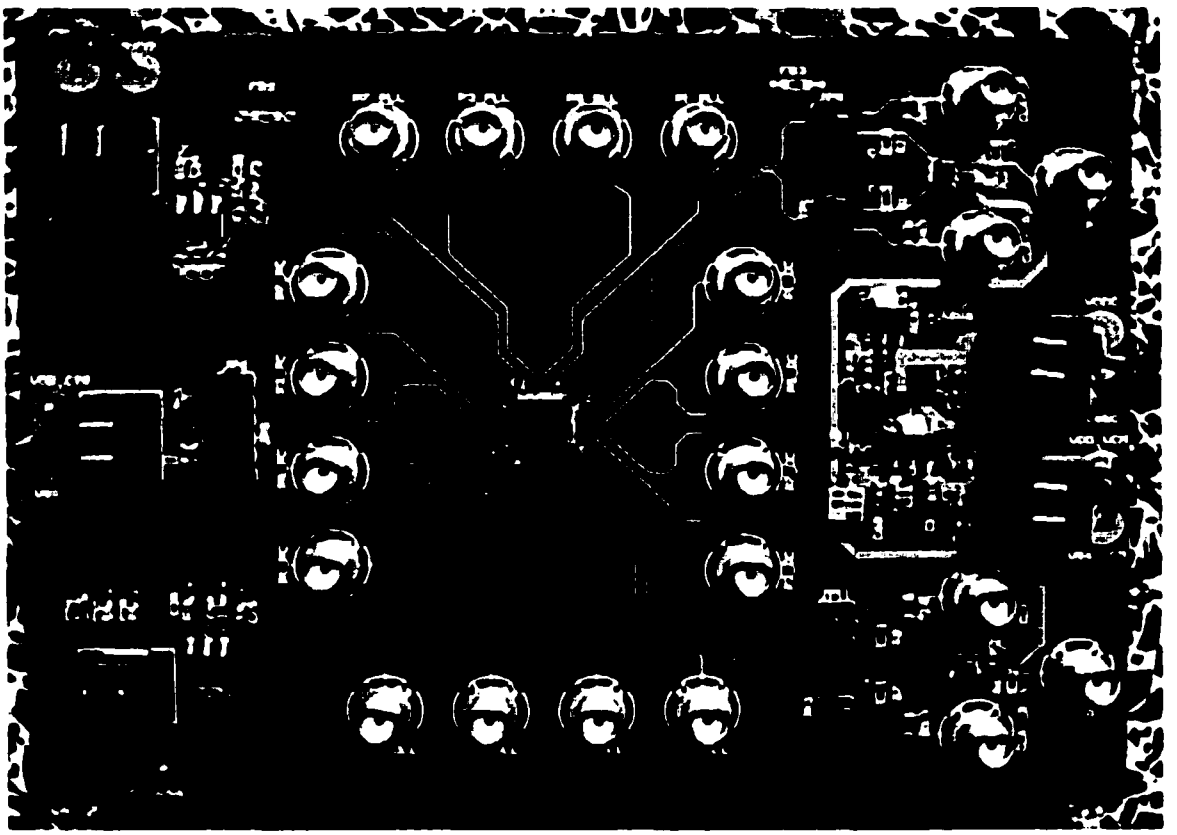


Figure 7.23 Testing 4-layer PC board

7.5.2 Operation range testing

As the first experimant, the operation ranges of both PLL and DLL are tested. Figure 7.24 shows the operation range of the PLL. In these four results, the upper waveforms are the input sine wave reference and the lower waveforms are the PLL outputs. This testing clearly shows that the PLL works well from $25MHz$ to $250MHz$, which is a very broad range. Similar waveforms were obtained for the DLL. The operation range for the DLL is $10MHz$ to $350MHz$. This test was done with a Tektronics high speed sampling oscilloscope with an input bandwidth of 1GHz.

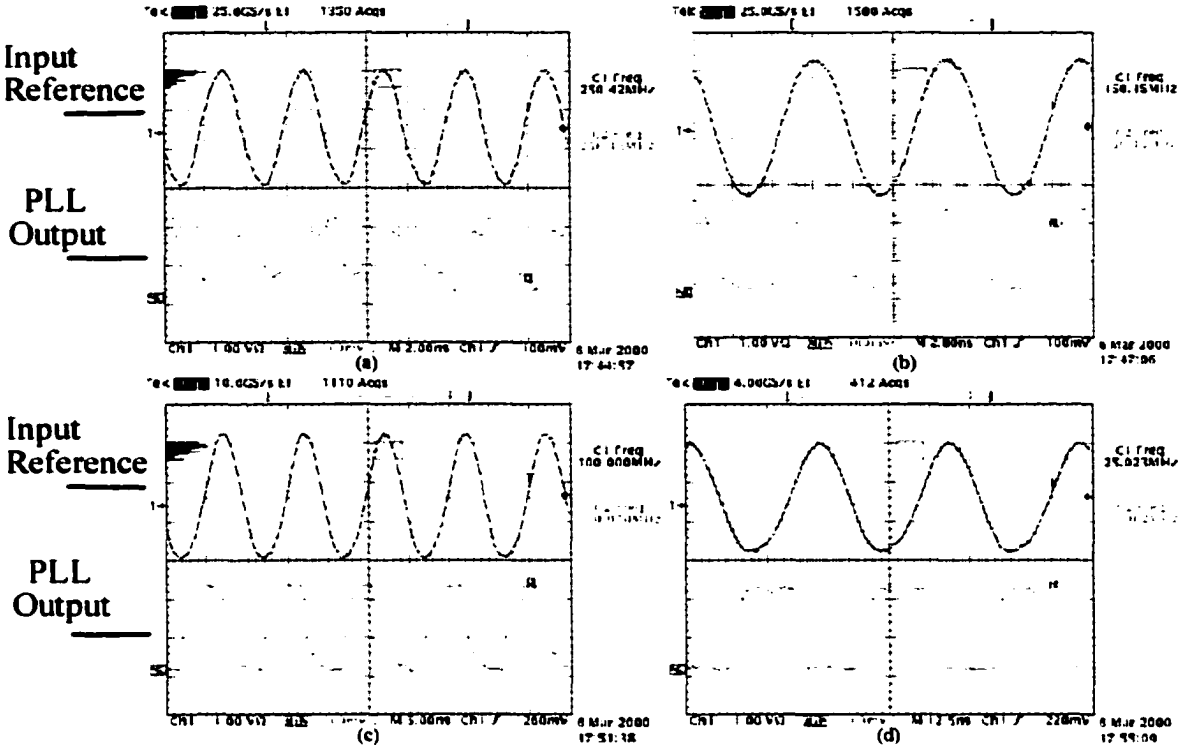


Figure 7.24 PLL operation range testing results

7.5.3 Jitter testing

The single channel jitter testing was done with an HP83480A communication analyzer. To illustrate the testing result of the PLL, when working at $250MHz$, is shown in Figure 7.25.

This result shows that the measured r.m.s jitter of the PLL output at $250MHz$ is only 2.27ps and the peak-to-peak jitter is only 18.9ps. If the noise floor($\sim$ $1ps$) of the equipment and the triggering source jitter ($\sim$ $1ps$) are excluded in the measured data, as stated in Chapter 3 Equation 3.32, the real r.m.s. jitter of the output will only be 1.77ps.

Figure 7.26 is the corresponding measured result of the DLL. Again, the operation speed is $250MHz$. The measured r.m.s jitter is only 1.9541ps and the peak-to-peak jitter is only 16.7ps.

It is worth mentioning that from the above results, we can see that the jitter performance of the prototype chip, which has already approached the noise floor of the high quality testing equipment is very good. It is possible for jitter performance to be better, but there is no direct way to measure it accurately, so far. The testing results are summarized in the Table 7.1. Equation 3.32 in Chapter 3 is used to calculate the real signal jitter.

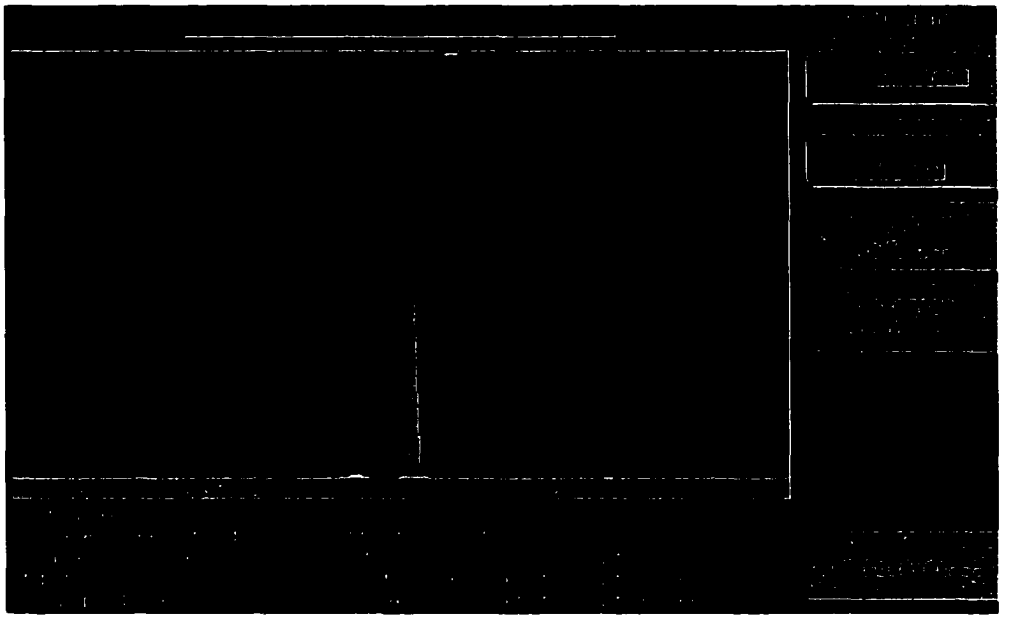


Figure 7.25 PLL single channel jitter at $250MHz$

Table 7.1 Jitter testing results

	PLL			DLL		
Speed	rms jitter		p-p jitter	rms jitter		p-p jitter
	measured	real		measured	real	
$300MHz$	2.12ps	1.58ps	18.9ps	1.97ps	1.37ps	16.9ps
$250MHz$	2.27ps	1.78ps	18.9ps	1.95ps	1.34ps	16.9ps
$200MHz$	2.82ps	2.44ps	24.4ps	2.67ps	2.26ps	22.2ps
$150MHz$	3.87ps	3.60ps	31.1ps	3.54ps	3.25ps	28.9ps
$100MHz$	4.39ps	4.16ps	40ps	4.28ps	4.04ps	35.6ps

7.5.4 Multi channel delay skew testing

To investigate the effectiveness of the delay calibration scheme, the skew between channels was tested before calibration and after calibration. The delay measurement was taken by a Wavecrest^{TM} time-interval analyzer DTS-2075. This equipment can directly give the statistic results on the delay between two channels, which is very appropriate for this specific application.

The first test is at an operating speed of $100MHz$. The period is 10ns. Ideally, the delay between adjacent phases should be 2.5ns or 2,500ps. Shown in Table 7.2 are the testing results before and after calibration.

The next testing is done at $125MHz$. At this example, the period is 8ns. If all four channels are

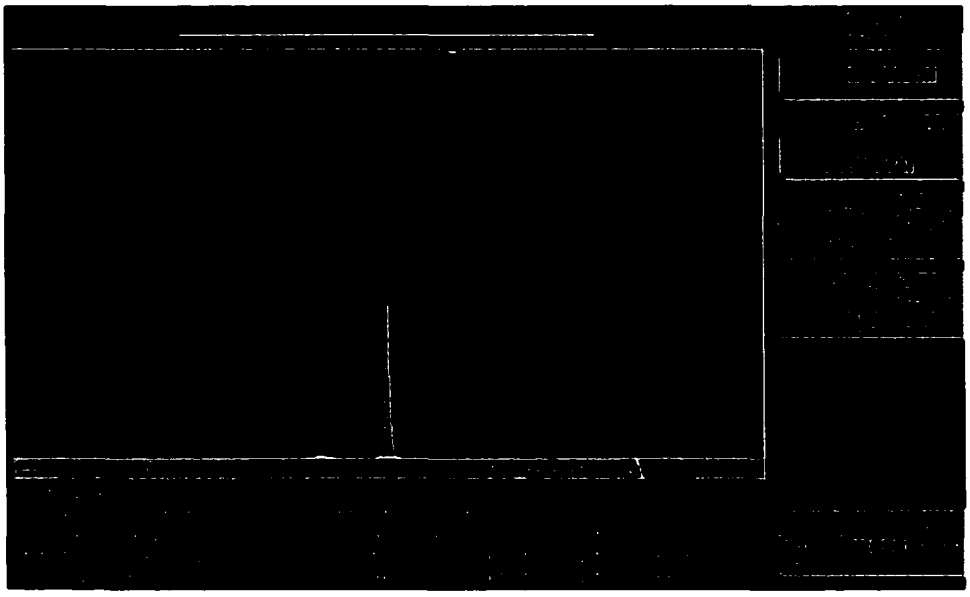


Figure 7.26 DLL single channel jitter at $250MHz$

distributed uniformly, the delay between adjacent phases should be 2,000ps. The measured results from the DTS-2075 before calibration and after calibration are summarized in Table 7.3.

The third test is done at $250MHz$, which corresponds to a period of 4ns or 4,000ps. Ideally, the delay between adjacent phases should be 1ns, or 1,000ps. Shown in Table 7.4 are the testing results before and after calibration.

As an example, the four-channel waveforms before and after calibration captured by a high speed oscilloscope are shown in Figure 7.27. Here the rising edges of the four phases are calibrated.

Table 7.2 Multi-phase delay testing at $100MHz$

	Ideal delay	Before calibration		After calibration	
$\phi 1 - \phi 2$	2,500ps	2,537.3ps	$\Delta = 37.3ps$	2,496.9ps	$\Delta = -3.1ps$
$\phi 1 - \phi 3$	5,000ps	5,051.5ps	$\Delta = 51.5ps$	5,010.2ps	$\Delta = 10.2ps$
$\phi 1 - \phi 4$	7,500ps	7,525.6ps	$\Delta = 25.6ps$	7,505.4ps	$\Delta = 5.4ps$

Table 7.3 Multi-phase delay testing at $125MHz$

	Ideal delay	Before calibration		After calibration	
$\phi 1 - \phi 2$	2,000ps	1,987.3ps	$\Delta = -12.7ps$	1,999.7ps	$\Delta = -0.3ps$
$\phi 1 - \phi 3$	4,000ps	4,056.8ps	$\Delta = 56.9ps$	4,010.5ps	$\Delta = 10.5ps$
$\phi 1 - \phi 4$	6,000ps	6,061.0ps	$\Delta = 61.0ps$	6,008.5ps	$\Delta = 8.5ps$

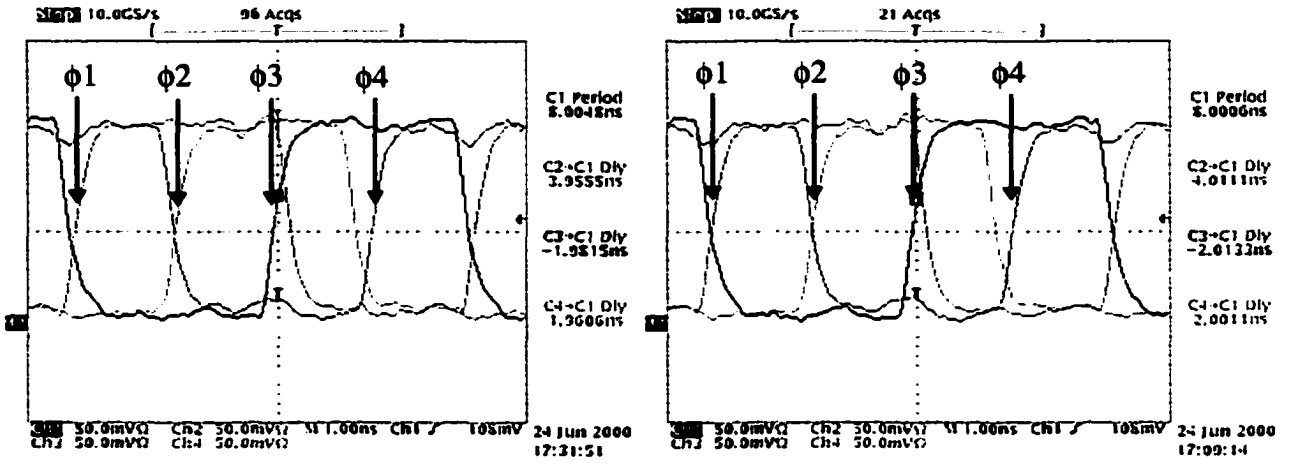


Figure 7.27 4 channel waveforms before and after calibration

Table 7.4 Multi-phase delay testing at $250MHz$

	Ideal delay	Before calibration		After calibration	
$\phi 1 - \phi 2$	1,000ps	978ps	$\Delta = -22ps$	991ps	$\Delta = -9ps$
$\phi 1 - \phi 3$	2,000ps	2,048ps	$\Delta = 48ps$	2,006	$\Delta = 6ps$
$\phi 1 - \phi 4$	3,000ps	3,039ps	$\Delta = 39ps$	3,003ps	$\Delta = 3ps$

From the above comparisons, it is clear that the proposed calibration scheme works well. The skew before calibration is around 50ps. After calibration, the skew is less than 10ps. It has significant improvement over the scheme without calibration.

7.6 Summary

A delay calibration algorithm was developed to deal with the delay mismatch problem in a PLL or DLL multi phase clock generator. The main idea of this method is to control the delay of each channel independently without affecting the main loop. This is done by adding additional feedback loops to do on chip real-time calibration. At the same time, low jitter circuit design techniques are used. Experimental results show that the final goal of achieving precise delay, as well as low jitter, is reached.

The delay mismatch calibration has not been previously reported in the literature. To the best of our knowledge, this is the first work that deals with this problem. Much has been reported, however, only on single channel jitter. Table 7.5 compares the achievement of this work with those of recently reported work. The chip performance is summarized in Table 7.6.

Table 7.5 Jitter performance comparison with prior art

Design	RMS jitter	P-P jitter	Operation speed	CMOS process
This design	**~ 2ps**	**~20ps**	$10MHz \sim 300MHz$	$0.25\mu m$
ISSCC'00(1) [68]	4.77ps	40ps	$622MHz$	$0.25\mu m$
ISSCC'00(2) [69]	14.52ps@$400MHz$ 24.61ps@$200MHz$	136ps@$400MHz$ 149ps@$200MHz$	$57.2MHz \sim 934MHz$	$0.6\mu m$
ISSCC'99(1) [65]	3.64ps	29ps	$250MHz$	$0.6\mu m$
ISSCC'99(2) [66]	5ps	35ps	2.5GHz	$0.28\mu m$

Table 7.6 Prototype chip summary

Fabrication process	TSMC0.25μm CMOS
Power supply	2.5V
Active area	$0.8 \times 1.5mm^2$
Operation frequency	10MHz~ 300MHz
Power dissipation	80mW of 4 designs together @ $250MHz$

CHAPTER 8 CONCLUSION

8.1 Conclusions

Timing jitter is a major concern in almost every type of communication system. Yet the desire for high levels of integration works against minimization of this error, especially for systems employing a phase locked loop or delay locked loop for timing generation or timing recovery. This thesis has explored system level as well as transistor level low jitter design techniques for integrated PLLs and DLLs. These are summarized as follows:

In chapter 2, the fundamentals of PLLs and DLLs are introduced. The emphasis has been placed on the monolithic implementation trends of both systems. It has been shown that charge-pump based PLLs and DLLs are the most popular contemporary design method, and the choices of the building blocks depend upon the application.

As the topic of this thesis, the noise issue of both PLLs and DLLs is discussed in detail in chapter 3. A rigorous jitter analysis based on a discrete z-domain model is performed in which the jitter is treated as a random event. Combined with statistical methods, the *rms* value of the accumulated jitter can be expressed with a closed form solution that successfully ties the jitter performance with loop parameters. For PLLs, the jitter accumulation factor is inversely proportional to the loop bandwidth. On the other hand, there is no such jitter accumulation effect in DLLs. The jitter measurement methods are also discussed at the end of this chapter which provides good understanding of the testing issue.

Based on the analysis in chapter 3, a cascaded PLL/DLL structure is proposed in chapter 4 that combines the advantages of both systems. The overall jitter accumulation factor is improved by the jitter filtering provided by the DLL. The resulting system is able to perform frequency synthesis with jitter as low as that of a DLL.

In chapter 5, a new nonlinear behavioral simulator for PLL/DLL jitter simulation is developed, which is based on a new oscillator and delay line model. Compared with prior art, this simulator simplifies the algorithm and thus enhances the computation efficiency. Moreover, it allows the inclusion of noise effects during simulation, which turns out to be a useful tool in predicting the jitter performance

both during tracking and lock conditions. This is also the first reported top-level simulation tool for DLL noise simulation.

In addition to the above system level work, three chips of PLLs and DLLs for different applications are implemented and tested:

In chapter 6, the application of PLLs in a high speed fibre channel transceiver is demonstrated. On the transmitter side, the PLL works as a frequency multiplication clock generator. A five-channel interleaved architecture is used to relax the speed requirement and thus to lower the power dissipation. On the receiver side, the PLL acts as clock recovery circuit working at 1.25GHz. The main effort is the design of high speed circuits with good noise immunity. Both chips are implemented with a CMOS bulk process. Testing results show that both chips have met the jitter specification with low power dissipation. Also, an alternative clock recovery approach based on a PLL is proposed in this chapter. This method uses a fully differential current steering technique that can minimize the power supply coupling effect.

In chapter 7, an on-chip dynamic delay calibration technique is proposed to achieve a low skew and low jitter multi-phase clock generator. The prototype chip is fabricated with $TSMC 0.25\mu m$ CMOS technology. Experimental results strongly support the effectiveness of the calibration scheme. At the same time, very low jitter is achieved. Compared with prior art, the jitter performance of this design achieves by far superior performance.

8.2 Recommended Future Work

Low jitter design of phase-lock loops and delay-locked loops will continue to be an important topic especially as the idea of system-on-chip becomes mature. The techniques proposed in this thesis are conceptual and therefore have endless potential of being further improved or expanded to meet future demands.

(1) For the cascaded PLL/DLL system, the major idea is to combine both systems appropriately to filter the noise. Even though in this thesis the structure is a cascaded system, the configuration is not limited to this. For different applications, the combination might be different.

(2) For any new system level structure, the jitter analysis method proposed in this thesis can be used to provide theoretical insight with proper simplifications.

(3) For the nonlinear behavioral modeling and simulation developed in this work, there is much that still can be added to make it more accurate and efficient. The RC delay model provides the basics of this whole method. It can be improved to include more real circuit behavior with the trade off of

CPU time. The noise mechanisms of especially power supply and substrate coupling noise are worthy of further investigation with this simulator. A suggested near future work is to use this method to evaluate the structure proposed in chapter 4.

(4) The current steering PLL clock recovery presented in chapter 6 will be useful when applied in large systems that have large portion of digital circuitry. The basic idea is to maintain the current on the power supply and the ground to be as constant as possible. With this idea in mind, the circuit blocks can have numerous variations.

(5) The delay calibration idea proposed in chapter 7 can be applied to any clock generator that requires rather accurate timing. Its application in time-interleaved architecture is one such example.

Any silicon implementation of the improvement on the above ideas in (1) (4) (5) in the future is a worthwhile goal.

BIBLIOGRAPHY

[1] F. M. Gardner, *Phaselock techniques*, second edition, John Wiley & Sons, New York, NY, 1979.

[2] D. Wolaver, *Phase locked loop circuit design*, Prentice Hall, 1991.

[3] R. Best, *Phase-locked loops, design, simulation, and applications*, fourth edition, McGraw-Hill, 1999.

[4] W. Egan, *Phase-lock basics*, John Wiley & Sons, New York, NY, 1998.

[5] R. Baker, H. Li, D. Boyce, *CMOS circuit design, layout, and simulation*, chapter 19, pp.417-425, IEEE Press, New York, NY, 1998.

[6] F. M. Gardner, "Charge-pump phase-locked loops", *IEEE Transactions on Communications*, vol. 28, pp. 1849-1858, Nov., 1980.

[7] W. Robins, *Phase noise in signal sources: theory and application*, Peregrinus, 1982.

[8] D. Leeson, "A simple model for feedback oscillator noise spectrum", *Proc. of IEEE*, pp. 329-330, Feb., 1966.

[9] A. Hajimiri, T. Lee, "A general theory of phase noise in electrical oscillators", *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 179-194, Feb., 1998.

[10] T. Lee, A. Hajimiri, "Oscillator phase noise: a tutorial", *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 326-336, Mar., 2000.

[11] A. Hajimiri, T. Lee, "Design issues in CMOS differential LC oscillators", *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 716-724, May, 1999.

[12] N. Nguyan, R. Meyer, "A 1.8GHz monolithic LC voltage-controlled oscillator", *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 444-450, Mar., 1992.

[13] B. Razavi, "A 1.8GHz CMOS voltage-controlled oscillator", *ISSCC'97*, pp. 388-389, Feb., 1997.

[14] L. Dauphinee, etc., "A Balanced 1.5GHz voltage-controlled oscillator with an integrated LC resonator", *ISSCC'97*, pp. 390-391, Feb., 1997.

[15] A. Dec, K. Suyama, "A 1.9GHz micromachined-based low phase noise CMOS VCO", *ISSCC'99*, pp. 80-81, Feb., 1999.

[16] J. Kim, B. Kim, "A low phase noise CMOS LC oscillator with a ring structure", *ISSCC'00*, pp. 430-431, Feb., 2000.

[17] D. Jeong, S. Chai, W. Song, G. Cho, "CMOS current controlled oscillator using multiple feedback loop architecture", *ISSCC'96*, pp. 386-387, Feb., 1996.

[18] S. Lee, B. Kim, K. Lee, "A novel high speed ring oscillator for multiphase clock generation using negative skewed delay scheme", *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 289-291, Feb., 1997.

[19] V. Kaenel, etc., "A 320MHz, 1.5mW at 1.35V CMOS PLL for microprocessor clock generation", *ISSCC'96*, pp. 132-133, Feb., 1996.

[20] B. Lai, R. Walker, "A monolithic 622Mb/sec clock extraction and data retiming circuit", *ISSCC'91*, pp. 144-145, Feb., 1991.

[21] S. Enam, A. Abidi, "NMOS ICs for clock and data regeneration in gigabit-per-second optical fibre receivers", *IEEE Journal of Solid-State Circuits*, vol.27, pp. 1763-1774, Dec., 1992.

[22] T. Weigandt, B. Kim, P. Gray, "Analysis of timing jitter in CMOS ring oscillators", *ISCAS'94*, pp. 27-30, June., 1994.

[23] B. Razavi, "A study of phase noise in CMOS oscillators", *IEEE Journal of Solid-State Circuits*, vol.31, pp. 331-343, Mar., 1996.

[24] J. Mcneill, "Jitter in ring oscillators", *IEEE Journal of Solid-State Circuits*, vol.32, pp. 870-879, Jun., 1997.

[25] A. Hajimiri, S. Limotyrakis, T. Lee, "Jitter and phase noise in ring oscillators", *IEEE Journal of Solid-State Circuits*, vol.34, pp. 790-804, Jun., 1999.

[26] B. Kim, *High speed clock recovery in VLSI using hybrid analog/digital techniques*, PhD dissertation, UC Berkeley, 1990.

[27] C. Park, B. Kim, "A low noise 900MHz VCO in 0.6μm CMOS", *IEEE Journal of Solid-State Circuits*, vol.34, pp. 586-590, May, 1999.

[28] L. Devito, J. Newton, R. Croughwell, J. Bulzacchelli, F. Benkley, "A 52MHz and 155MHz clock recovery PLL", *ISSCC'91*, pp. 142-143, Feb., 1999.

[29] J. Sneep, C. Verhoeven, "A new low noise 100MHz balanced relaxation oscillator", *IEEE Journal of Solid-State Circuits*, vol.25, pp. 692-698, Jun., 1990.

[30] M. Souyer, H. Ainspan, "A monolithic 2.3Gb/s 100mW clock and data recovery circuit", *ISSCC'93*, pp. 158-159, Feb., 1999.

[31] A. Abidi, R. Meyer, "Noise in relaxation oscillators", *IEEE Journal of Solod-State Circuits*, vol. sc-18, pp. 794-802, Dec., 1983.

[32] D. Sharpe, "A 3-state phase detector can improve your next PLL design", *EDN Magazine*, Sep., 1976.

[33] P. Gray, R. Meyer, *Analysis and design of analog integrated circuits*, third edition, John Wiley & Sons, New York, NY, 1993.

[34] P. Allen, D. Holberg, *CMOS analog circuit design*, Oxford University Press, 1987.

[35] D. Johns, K. Martin, *Analog integrated circuit design*, John Wiley & Sons, New York, NY, 1997.

[36] F. Herzel, B. Razavi, "Oscillator jitter due to supply and substrate noise", *IEEE CICC*, May, 1998.

[37] P. Heydari, M. Pedram, "Analysis of jitter due to power-supply noise in phase-locked loops", *IEEE CICC*, May, 2000.

[38] A. Papoulis, *Probability, random variables, and stochastic processes*, McGraw-Hill, 1965.

[39] W. Davenport, W. Root, *Random signals and noise*, McGraw-Hill, 1958.

[40] V. Kroupa, "Noise property of PLL systems", *IEEE Transactions on Communications*, vol. COM-30, pp. 2244-2252, Oct. 1982.

[41] VITESSE Semiconductor Co., *VSC7135—1.25Gbits/sec Gigabit Ethernet transceiver data sheet*, Minneapolis, Minnesota, 1997.

[42] AMCC Co., *S2052—Fibre channel and Gigabit Ethernet transceiver data sheet*, San Diego, California, 1997.

[43] AMD Co., *Am79761—Physical layer 10-Bit transceiver for Gigabit Ethernet data sheet*, Austin, Texas, 1998.

[44] J. Hein, J. Scott, "z-Domain model for discrete-time PLL", *IEEE Trans. Circuits and Systems*, vol. 35, pp. 1393-1400, Nov., 1988.

[45] J. Kovacs, "Analyze PLLs with discrete time modeling", *Microwaves & RF*, pp. 224-229, May 1991.

[46] B. Kim, T. Weigandt, P. Gray, "PLL/DLL system noise analysis for low jitter clock synthesizer design", *ISCAS'94*, pp. 31 34, June, 1994.

[47] C. Samori, A. Lacaita, A. Zanchi, F. Pizzolato, "Experimental verification of the link between timing jitter and phase noise", *Electronic Letter*, vol. 34, pp. 2024-2025, Oct., 1998.

[48] T. Weigandt, *Low-phase-noise, low-timing-jitter design techniques for delay cell based VCOs and frequency synthesizer*, PhD dissertation, UC Berkeley, 1998.

[49] Hewlett Parkard Inc., *HP83480A communication analyzer user's manual*, Palo Alto, California, 1998.

[50] Wavecrest Inc., *DTS-2075 time interval analyzer user's manual*, Minneapolis, Minnesota, 1998.

[51] L. Wu, H. Jin, W. Black, "Nonlinear behavioral modeling and simulation of phase-locked and delay-locked system", *Proc. of IEEE CICC'2000*, pp. 447 - 450 , May 2000.

[52] E. Liu, A.L. Sangiovanni-Vincentelli, "Behavioral representations for VCO and detectors in phase-lock systems", *Proc. of IEEE CICC'1992*, May, 1992.

[53] A. Demir, E. Liu, A.L. Sangiovanni-Vincentelli, "Behavioral simulation techniques for phase/delay-locked systems", *Proc. of IEEE CICC'1994*, May, 1994.

[54] J. Morris, *Computational methods in elementary numerical analysis*, John Wiley & Sons, New York, NY, 1983.

[55] G. Chien, P. Gray, "A 900MHz local oscillator using a DLL-based frequency multiplier technique for PCS application", *IEEE ISSCC'2000*, pp. 202-203, Feb., 2000.

[56] A. Waizman, "A Delay line loop for frequency synthesis of de-skewed clock", *IEEE ISSCC'1994*, pp. 298-299, Feb., 1994.

[57] T. Lee, J. Bulzacchelli, "A 155MHz clock recovery delay and phase-locked loop", *IEEE Journal of Solid-State Circuits*, pp. 1736-1746, Dec. 1992.

[58] L. Wu, etc., "A monolithic 1.25Gbits/sec CMOS clock/data recovery circuit for fibre channel transceiver", *Proc. of IEEE ISCAS'1999*, June, 1999.

[59] J. Maneatis, "Low-jitter process independent DLL and PLL based on self-biased technique", *IEEE Journal of Solid-State Circuits*, vol.32, pp. 1723-1732, Nov., 1996.

[60] L. Wu, W. Black, "A low jitter 1.25GHz CMOS analog PLL for clock recovery", *Proc. of IEEE ISCAS*, June 1998.

[61] D. Allstot, etc., "Current-mode logic techniques for CMOS mixed-mode ASICs", *Proc. of IEEE CICC'1991*, pp. 25.2.1-25.2.4, May, 1991.

[62] M. Mizuno, etc., "A GHz adaptive pipeline technique using MOS current-mode logic", *IEEE Journal of Solid-State Circuits*, vol.31, pp. 784-791, June, 1996.

[63] Alan Fieder, Ross Mactaggart, James Welch and Shoba Krishnam, "A 1.0625 Gbps transceiver with 2X-oversampling and transmit signal pre-emphasis", *IEEE ISSCC'1997*, P. 238-239, Feb., 1997

[64] L. Wu and W. Black, "A low jitter skew calibrated multi-phase clock generator for time-interleaved applications", *IEEE ISSCC'2001*, WP 25.3, Feb., 2001.

[65] J. Lee, etc., "A 250MHz low jitter adaptive bandwidth PLL", *IEEE ISSCC'1999*, Feb., 1999.

[66] R. Gu, etc., "A 0.5~3Gb/sec low power low jitter serial data CMOS transceiver", *IEEE ISSCC'1999*, Feb., 1999.

[67] D. Boerstler, "A low jitter PLL clock generator for microprocessors with lock range of $340MHz \sim 612MHz$", *IEEE Journal of Solid-State Circuits*, April, 1999.

[68] T. Saeki, etc., "A 1.3cycle lock time , non PLL/DLL jitter suppression clock multiplier based on direct clock cycle interpolation for 'Clock on Demand' ", *IEEE ISSCC'2000*, Feb., 2000.

[69] I. Hwang, etc., "A digitally controlled phase-locked loop with fast locking scheme for clock synthesis application", *IEEE ISSCC'2000*, Feb., 2000.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to the many people who have made my PhD study a rewarding experience.

First, I would like to thank my research advisor Professor William Black Jr. I have been honored to have worked under his supervision for the past four and half years of my stay at Iowa State University. Prof. Black provided many profound thoughts and valuable guidance for my research. I am also deeply indebted to him for providing me a good research environment and being a wonderful friend.

I would also like to express my gratitude to Dr. Marwan Hassoun, Dr. Robert Weber, Dr. Chris Chu and Dr. Wolfgang Kliemann for kindly agreeing to be my committee members during their very busy schedule. And especially I would like to thank Dr. Weber for providing much help on the PC board design that is critical for the success of the last chip.

During my Ph.D. years our VLSI group has grown from a small research group with few students to a sophisticated design center with projects reaching all the directions of cutting edge technology. We also have a multi functional computer lab and a state of the art bench lab. All of these provide a comprehensive and effective working environment to the graduate students. All of these privileges become possible due to the unceasing effort from our faculty members, especially Dr. Black and Dr. Hassoun.

There are many friendly staff members in our department who have helped me along the way. I would like to thank Maria Blanco for all the help she kindly provided, from ordering components, arranging travels, to many enjoyable group activities. A thank you also goes to our technician Jason Boyd, who accomplished wonderful PC board soldering work for my testing.

Through these years, all my research work was funded by RocketChips Inc. Therefore I would like to thank my sponsor. And I would also like to thank Carver lab that is funded by Roy J. Carver Charitable Trust, Texas Instruments Inc., Rockwell Inc., VTC Inc.

I wish to thank the students with whom I have worked over the past several years, who have been there to share in many good times and have always had time to offer their support and their knowledge:

Baiying Yu, Huimin Xia, Jing Ye, Bodhi Das, Xiaoyu Xi, Jian Zhou, Xiaohong Du, Kae Wong, Steve Osugi, Jie Yan and Huiting Chen.

Finally, I would express my deepest appreciation to my family: my father and mother, Dake and Shulin, and my sister Fan, who always encourage me to do my best and who have always been there to share my difficulties and times of joy.

I am especially grateful to the support and love from my husband, Huawen. Through all these years, he helped me on a lot of technical issues. I feel very lucky to have him as a partner not only of life but also of career. Without his love, I could not have achieved this.