DESIGN OF NOISE-ROBUST CLOCK AND DATA RECOVERY USING AN ADAPTIVE-BANDWIDTH MIXED PLL/DLL

A dissertation presented by

Han-Yuan Tan

to

The Division of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy in the subject of Engineering Sciences

Harvard University

Cambridge, Massachusetts

November 2006

Han-Yuan Tan

DESIGN OF NOISE-ROBUST CLOCK AND DATA RECOVERY USING AN ADAPTIVE-BANDWIDTH MIXED PLL/DLL

Dissertation Advisor: Associate Professor Gu-Yeon Wei

Abstract

As the continuing technology scaling keeps increasing the maximum on-chip clock frequency, the demand for the high-speed link that bridges the faster on-chip world and slower off-chip world is rapidly growing. The challenges are stronger than before with more functions being integrated into a single chip, which has become a complicated mixed-mode System on Chip (SoC) design. The performance of the link system greatly depends on how well the noise is managed in the system. Unfortunately the noise conditions in a highly-integrated SoC are usually difficult to predict before chip fabrication and vary quite a lot among different systems. The noise problem is particularly more troublesome on the receiver side than the transmitter side, because the receiver usually consists of more circuit blocks than the transmitter. If one can design a noise-robust receiver that can adapt to various noise conditions, such a macro can be used within a variety of systems and therefore reduce the cost of custom design. This dissertation presents a receiver design that can adjust itself under time-varying noise conditions in order to minimize jitter and achieve optimum performance.

This dissertation first describes an adaptive-bandwidth mixed PLL/DLL (MX-PDLL)-based multiphase clock generator to achieve the optimum jitter performance

under different noise conditions. The MX-PDLL uses a phase mixing interpolator to merge traditional PLL and DLL loops into a single loop. The resulting wide range of bandwidth adjustment enables this mechanism for adapting across different noise conditions to minimize jitter.

The dissertation then introduces a new clock and data recovery (CDR) architecture using this MX-PDLL clock generator in a receiver. This CDR uses a digitally-controlled phase rotator to shift the phase of the reference clock that feeds into the MX-PDLL to track the phase and frequency of the data. By tuning the bandwidth of the MX-PDLL, the CDR can find the optimum bandwidth setting under different amount of power supply noise, reference clock noise, and digital control-induced quantization noise to minimize clock jitter and thereby enhance performance.

A prototype chip was fabricated in a 0.18μm CMOS technology, and all the measurement results verify the above claims.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Continuous technology scaling from one generation to the next has enabled an impressive growth in single-chip integration for complex systems. While increasingly complex functions are being integrated on-chip and on-chip clock frequency has increased dramatically over the generations, the speed of the off-chip world lags behind the logic operation speed on-chip. This disparity has strengthened more than ever before the demand for the high-speed interface that bridges the two worlds, on-chip and off-chip. The challenges in the high-speed interface or link design primarily come from the bandwidth limitations of the channel and process technologies, and the 'noise' in today's highly integrated mixed-signal chips. Much effort has been made to overcome these obvious obstacles, and various solutions have been provided over the past decade, such as channel equalization, time-interleaved transceiver design, and low-jitter clock generators. Among all these challenges, the timing uncertainty in the link system is particularly difficult to address in today's System on-chip (SoC) environment. Most of the high-speed links employ synchronous transmission to fully exploit the channel's bandwidth. In a synchronous transmission system, the timing information is the most important factor that determines the link performance and the data rate. Clock timing uncertainty can be categorized as either dynamic or static. The dynamic type is also referred to as 'timing jitter' on the clock. The static type is referred to as 'static phase mismatch'. The dynamic jitter is caused by various noise sources in the high-speed link system and the static phase

mismatch results from process variation and systematic mismatch in the physical design (i.e., layout). The static phase mismatch can be calibrated in a static fashion after chip fabrication. However, on-chip and off-chip noise can contribute to dynamic timing uncertainty in a high-speed link system, therefore it can degrade the performance and data rate of the link. The noise can come from many different sources on-chip and off-chip, and is highly dependent on applications, process technologies, and designs. For example, in a DVI link for a video application, the reference clock is transmitted along with the data over the cable. The reference clock noise at the receiver is quite high (> 0.3U.I.) [38]. For the PCI Express interface standard, which enables high-speed chip-to-chip communications, transceiver circuitry is often located inside a GPU or a chipset and the on-chip power supply noise might be higher than other applications. Therefore, the demand for a reliable and noise-robust link macro or Intellectual Property (IP) that can accommodate a variety of different applications with various noise conditions (instead of needing to customize the link for each application), is rapidly growing.

This dissertation presents an adaptive-bandwidth mixed phase-locked loop (PLL)/delay-locked loop (DLL)–based multiphase clock generator to achieve the best timing accuracy under various noise conditions by tuning the clock generator's bandwidth. Then a noise-robust clock and data recovery (CDR) architecture incorporating this mixed PLL/DLL (MX-PDLL) is proposed, which ought to enable reliable high-seed links for different applications. This CDR architecture has the flexibility to adapt the bandwidth of its clock generator under different noise conditions in order to achieve a reliable performance. Lastly, a static phase offset detection method and several compensation schemes to mitigate the static phase offset among different

clock phases in the MX-PDLL are described. With all three of these measures, a flexible and reliable high-speed link system can be achieved.

At this point, two questions may arise: What exactly are PLL, DLL, and CDR? Where are they being used in a high-speed link system? Let us first understand the basic link structure. An illustration of a high-speed link system, which is comprised of a transmitter (TX), a channel, and a receiver (RX), is shown in Figure 1.1. The transmitter

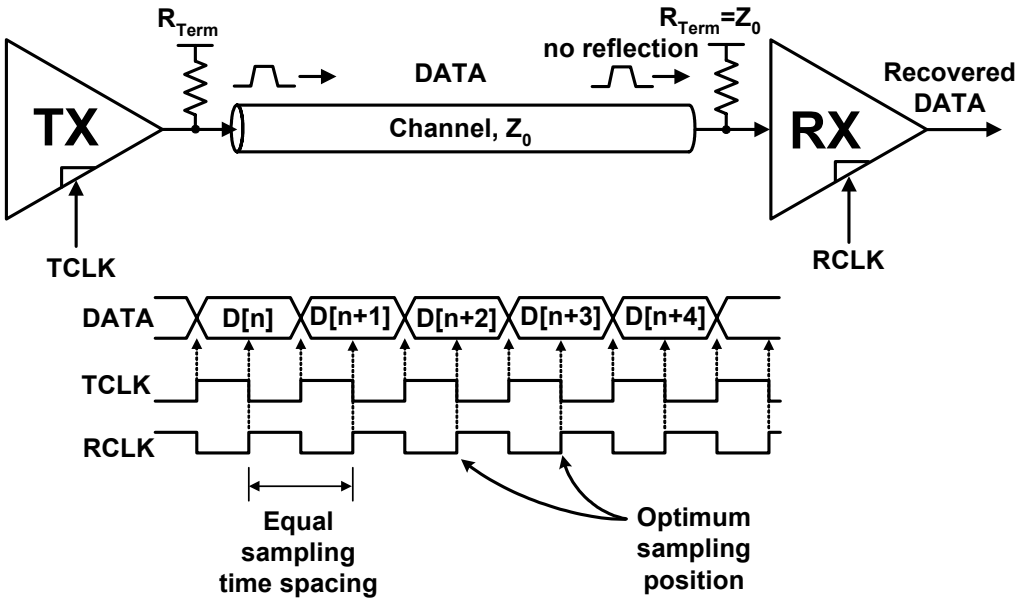


Figure 1.1: High-speed link system and 2-PAM singling convention (Recreated from [39]).

maps the digital information to a waveform and sends it over the channel, which can be any transmission medium, with respect to a local transmitter clock (TCLK). The channel needs to be properly terminated with its characteristic impedance, Z_0, to avoid any reflected waveform along the channel. The signaling convention used in Figure 1.1 is a

typical binary signal waveform (2-PAM, i.e., two-level, pulse-amplitude modulated). The receiver on the other end of the channel recovers the waveform to the original digital information usually by sampling and quantizing it using a local receiver clock (RCLK). Since the TCLK might not be available to the receiver explicitly, a clock and data recovery circuitry, called CDR, is usually embedded in the receiver. The CDR block usually uses either a PLL or a DLL to maintain a controllable timing relationship between the RCLK and the incoming data. This kind of link system usually requires accurate timing recovery in the receiver in order to sample the signal waveform at the optimal position. Therefore the timing accuracy of the PLL or DLL is an extremely important factor and determines how fast and efficiently the link system can operate.

The link's performance is determined by the aggregated performance of all of the blocks. Moreover the noise can exist everywhere throughout the link to degrade the clocks' timing accuracy. However, we focus only on the design of the receiver, more specifically the CDR block in this dissertation. The receiver's detailed architecture is described in Chapter 2.

In this chapter, first the motivation for designing a noise-robust CDR is discussed by introducing several widely adopted high-speed link systems for different applications. After that, a simplified block diagram of the adaptive-bandwidth mixed PLL/DLL (MX-PDLL)-based multiphase clock generator topology is presented. Then the block diagram of the proposed dual-loop CDR architecture incorporating the MX-PDLL is presented. In the last section of this chapter, an overview of the rest of this dissertation is provided.

1.1 Demand for Noise-Robust Clock and Data Recovery

The demand for high-speed links with high data rate, low power consumption, small footprint, and high flexibility is tremendous. Mainly because the on-chip clock frequency has been increasing as the technology scales, however, the number of I/O (Input/Output) pins is scaling at a lower rate. Therefore the off-chip data bandwidth per pin is scaling faster than the on-chip clock frequency. The fast growing demand for multi-Gigabit/s data communication applications, e.g., Gigabit Ethernet, hard disk drive interface (Serial ATA), and general purpose I/O (PCI Express), just to name a few, has motivated the high-speed link design to meet the stringent challenges posed from the limitations of process technology, lossy communication channels, and noise in the system.

The clock timing uncertainty in the CDR is one of the most significant factors that determine overall link performance. Both static phase mismatch and dynamic jitter degrade CDR performance, especially dynamic jitter, which can be caused by various noise sources in the CDR. Three of the most common noise sources are (1) power supply noise, (2) quantization noise in a digital CDR, and (3) reference clock noise. In order to meet the demand for a faster data rate, careful planning must be undertaken to keep the system's noise low, and sometimes even by making necessary tradeoffs to reduce the noise at a cost of higher power consumption, more expensive technology and packaging.

However, even with careful planning, the resulting noise conditions in the system may still change quite differently from the expected noise conditions due to process-

voltage-temperature (PVT) variations, unexpected noise sources, and using a wrong model for the noise source. Most importantly the noise conditions also change across different high-speed links for different applications. Therefore if we can build a flexible system that can be adjusted upon different noise conditions in order to achieve the best clock jitter performance, it can be used as a reliable macro for different link systems to reduce custom design costs. This CDR macro can be used for a multi-standard Serializer/Deserializer (SerDes) core, which can be easily configured to meet different jitter specifications for different applications as opposed to customizing the CDR for different applications. This can be very attractive in terms of lowering the cost of IP development.

This dissertation focuses on the design of a noise-robust clock and data recovery architecture to be used as a macro for different applications with different noise conditions, such that the CDR can always adapt itself to achieve the best clock jitter performance under a given noise condition. It also provides solutions to mitigate the static phase mismatch in the CDR to further improve overall system performance.

Even though many important building blocks constitute a high-speed link, in this dissertation, we focus on designing a noise-robust clock and data recovery architecture in the receiver. Similar techniques can also be applied to other building blocks in the system, but the analyses are beyond the scope of this dissertation.

1.2 Adaptive-Bandwidth Mixed PLL/DLL-Based Multiphase Clock Generator

The proposed mixed PLL/DLL (MX-PDLL)-based multiphase clock generator topology is shown in Figure 1.2. The most important feature of the MX-PDLL is the mixing element through which the bandwidth of this clock generator can be tuned across

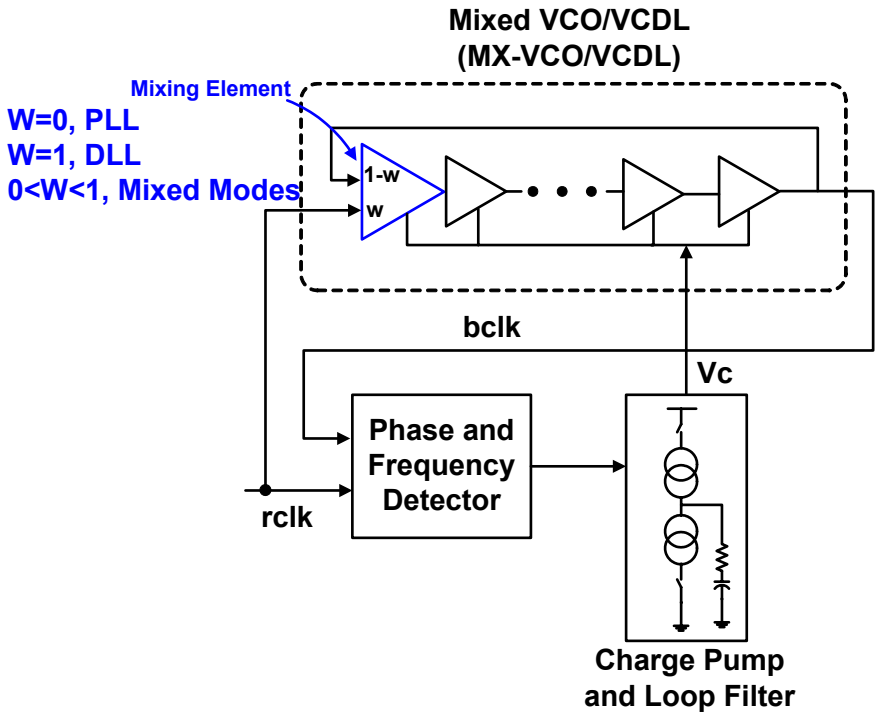


Figure 1.2: Mixed PLL/DLL loop topology.

a wide frequency range. Also the configuration can be easily tuned through the mixing element in order to optimize the rejection of different noise sources. This bandwidth

tuning ability can help minimize the clock jitter under different noise conditions in order to achieve the best timing accuracy.

This mixed PLL/DLL structure is achieved by merging the PLL and DLL loops into one single loop. And the structure can be configured as a PLL, a DLL, or a mixture of the two by controlling the mixing weight, *w*. In the example shown here, where *w*=0, the topology is a PLL. When *w*=1, the topology is a DLL. When *w* is between 0 and 1, the topology is a mixed PLL/DLL. The PLL- and DLL-based clock generators have very different noise responses with respect to different sources of noise. Throughout the rest of the dissertation, the tradeoffs between *w* and the rejection of different noise sources in the clock generator is discussed. This dissertation also discusses how you can achieve optimum jitter performance under various noise conditions by tuning *w*.

Then the next question is, can we incorporate this adaptive-bandwidth MX-PDLL in a CDR and take advantage of the noise-robust feature of this clock generator?

1.3 Dual-Loop CDR Architecture Using the MX-PDLL as the Core Clock Generator

A dual-loop CDR architecture that incorporates the MX-PDLL as the core clock generation loop is presented in Figure 1.3. This CDR architecture consists of two loops, a core clock generation loop and peripheral phase/frequency tracking loop. The clock generation loop uses the MX-PDLL to generate multiphase sampling clocks that sample the incoming serial data stream. Then some phase error information between the data and

sampling clock is sent over to the digital CDR control to update the phase rotator. An external reference clock, *xclk,* feeds into the phase rotator to generate *rclk*, which feeds into the MX-PDLL. In this type of system, the frequency of the inputs to the phase rotator, *i, q, i_b, and q_b*, can be the same as the data rate divided by m. Otherwise, there

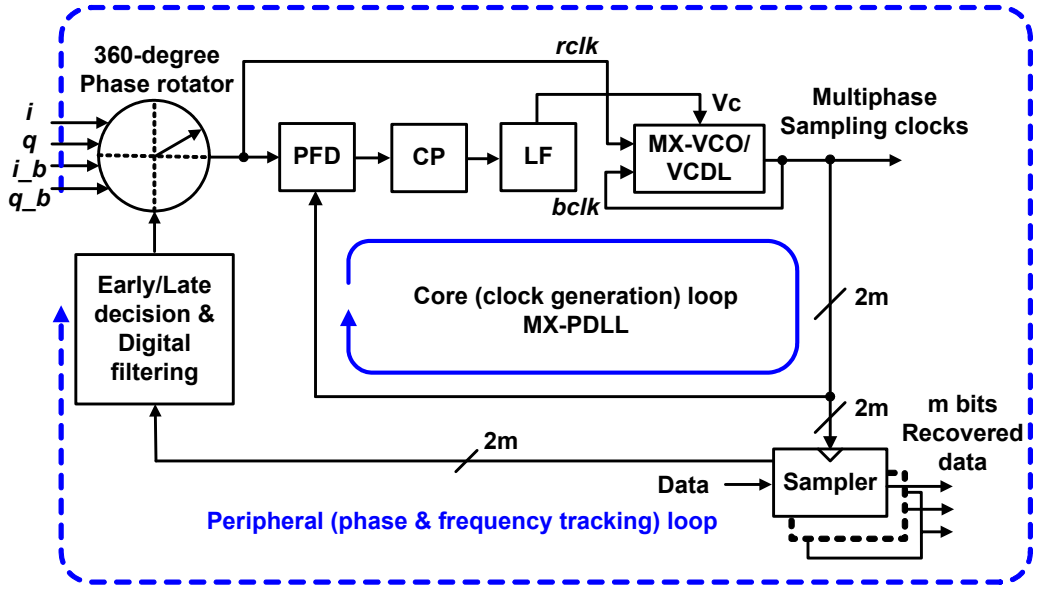


Figure 1.3: Proposed dual-loop CDR architecture using the MX-PDLL as the core clock generator.

could be several hundred ppm of frequency deviation between the *i, q, i_b, and q_b* and the recovered clock, *bclk*, due to different clock sources between the receiver and the transmitter. The CDR needs to adjust the phase and frequency of the sampling clocks in order to be aligned with the data. The phase rotator will shift the *rclk* phase according to the phase error information to update the sampling clock phases. When the CDR is in-lock, the sampling clocks will sample the data at the optimum timing location and the

rclk and one of the sampling clocks, *bclk*, will be aligned in phase and frequency. This completes the clock and data recovery function of the system.

By tuning the mixing weight of the MX-PDLL, which is equivalent to tuning the bandwidth of the MX-PDLL, we can find an optimum bandwidth setting to minimize sampling clock jitter under different noise conditions. Some other systematic noise sources also exist in the CDR due to the digitally controlled peripheral loop. The tradeoffs between bandwidth setting in the MX-PDLL and rejection of different sources of noise in this CDR are discussed. Because of all these tradeoffs, we can find an optimum mixing weight that renders the smallest output clock jitter in the MX-PDLL. In this dissertation, most of the blocks shown in Figure 1.3 are addressed.

1.4 Overview of the Dissertation

This dissertation proposes a noise-robust CDR architecture using an adaptive-bandwidth mixed PLL/DLL-based multiphase clock generator and presents the measurement results of the system. The bandwidth tuning capability of the clock generator allows for good timing accuracy across various noise conditions. Therefore this CDR architecture is a good candidate to be integrated with digital circuitry in today's complicated mixed-signal environment, where unpredictable noise conditions impose enormous challenges to the timing budget. In addition to the dynamic timing jitter, the static timing uncertainty (static phase mismatch) also consumes the link's timing budget. Therefore this dissertation also proposes a static phase spacing mismatch detection

method and several compensation schemes to mitigate the mismatch (skew). The rest of the dissertation is organized as follows.

Chapter 2 presents the background of the high-speed link system with special attention paid to the receiver, PLL and DLL clock generators, and CDR architectures. Three very important sources of noise in the receiver are discussed in order to provide a basic understanding of where the noise is coming from and the impact from difference noise sources. Then the PLL topology with both continuous-time and discrete-time modeling representations, and its corresponding input noise transfer function and output noise transfer function are described. The DLL topology with a discrete-time model representation follows. Some prior art of adaptive-bandwidth PLL design and injection-locked PLL design are introduced in order to compare those systems and mine. Following this, some popular CDR architectures that are widely used in today's high-speed link systems are introduced. Each one of them has advantages and disadvantages, therefore, design tradeoffs are necessary for different applications. Lastly, some CDR performance metrics are discussed at the end of Chapter 2.

Chapter 3 presents the proposed adaptive-bandwidth mixed PLL/DLL (MX-PDLL)-based multiphase clock generator. This design enables loop bandwidth adjustment over a wide range of magnitudes to achieve minimum output jitter across various noise conditions. The high-level architecture is first described along with the simulation results of various noise transfer functions based on a discrete-time z-domain model. The simulation results show at least one order of magnitude of bandwidth tuning from the PLL-mode operation to the DLL-mode operation. The circuit implementation of

the clock generator is presented next followed by measurement results from the test chip prototype.

Chapter 4 presents a proposed dual-loop CDR architecture using the MX-PDLL as the core loop clock generator, the loop bandwidth of which can be adaptively tuned, in order to minimize the sampling clock jitter under various noise conditions. Therefore bandwidth tuning can help improve the timing margin in the receiver. First the high-level architecture is presented, with a description of how phase and frequency tracking can be achieved through a 360° phase rotator in front of the MX-PDLL by controlling the amount of the reference clock's phase shifting. Four 90° phase-shifted clocks, which are fed into the phase rotator to accomplish a 360° phase rotation, are generated by an on-chip divide-by-2 (÷ 2) circuitry. Then the prototype system built and tested is presented. This system actually does not form a closed CDR loop; instead it uses an off-chip digital controller, which facilitates the testing of various control schemes. Lastly, the measurement results of the output clock jitter in the prototype chip are presented. The results demonstrate how the sampling clock jitter of the system can be minimized across various noise conditions with different digital control signals.

Chapter 5 is dedicated to mitigating other static type sources of clock uncertainty in the system. For example, to calibrate the static phase offset due to delay mismatches caused by systematic imbalances and random process variations, a charge-pump compensation scheme and a passive phase-averaging network-based phase spacing mismatch reduction scheme are presented. Also to detect static phase spacing mismatch between multiphase clocks, an XOR-based phase detector is included in the test chip. Lastly, a duty-cycle correction circuitry to compensate for the duty-cycle mismatch on

the external reference clock is proposed, along with the divide-by-2 block to minimize quadrant spacing mismatch in the phase rotator, since the quadrant mismatch aggravates the phase rotator non-linearity and the CDR performance. The measurement results of these compensation schemes are also presented in this chapter.

Chapter 6 concludes this dissertation and summarizes the contributions made to field of high-speed link design with capabilities to minimize timing uncertainty. The design of the proposed noise-robust dual-loop CDR architecture using an adaptive-bandwidth MX-PDLL-based multiphase clock generator is particularly suitable for the high-speed links in a highly integrated mixed-signal chip.

Chapter 2

Background

This dissertation focuses on the design and implementation of noise-robust, reliable clock and data recovery (CDR) blocks for high-speed links. In this effort, it is important to first understand what high-speed links are and how they operate. Based on the basic link structure seen in Chapter 1, this chapter provides a detailed overview of a high-speed link system and focuses on providing an understanding of the limitations for achieving high-speed operation, the employment of an interleaving architecture to overcome the potential bandwidth limitation of process technology, and the effects of clock-timing uncertainty on link performance.

In order to understand how a high-speed link operates, it is important to also review the operation of phase-locked loops (PLL) and delay-locked loops (DLL), which are used to generate the clock signal, i.e., time base with respect to which data is transmitted and received. This chapter provides an overview of the basic topologies traditionally used to implement a PLL and DLL. By looking at how these loops respond to different noise sources, one can identify the basic design tradeoffs that each presents. Furthermore, these tradeoffs motivate the desire to implement an "adjustable bandwidth" clock generator. A brief discussion of prior art in adjustable-bandwidth clock generators reveals limitations that must be overcome.

Building upon a working knowledge of PLLs and DLLs, this chapter proceeds to provide an overview of CDR operation and presents several topologies commonly used in

high-speed link designs. Traditional CDR designs generally fall into one of two categories – PLL-based and interpolator-based designs. Similar to the comparison made between PLL- and DLL-based clock generators, each design presents complimentary design tradeoffs. As a result, a comparison of the two types behooves one to consider utilizing an adjustable-bandwidth clock generator as the core component of an adjustable-bandwidth CDR that leads to noise-robust CDR operation.

Phase uncertainty of the sampling clock in a link receiver is one of the dominant sources of uncertainty that compromises link performance. Phase uncertainty can be categorized into two types – static and dynamic. These types of phase uncertainty are also often classified as deterministic jitter and random jitter, which result from various noise sources. To set the context for the main contributions of the presented work, Section 2.2 describes the characteristics of the most relevant sources of noise in a CDR, namely, power supply noise, quantization noise in a digitally-controlled CDR, and noise on the reference clock with respect to which a CDR operates.

2.1 High-Speed Link System

This section takes another look at the high-speed link architecture and discusses some of the design limitations in more detail. Figure 2.1 illustrates the high-speed link system that was already seen in Chapter 1. A transmitter converts digital information into an analog waveform and sends it across the channel, which is usually a board trace, a coaxial cable, or other transmission media. When the channel is properly terminated with its characteristic impedance, Z_0, the transmitted waveform is fully absorbed by the

receiver. Ideally, there is no reflected energy within the channel and a succession of symbols can be transmitted in a pipelined fashion without having to wait for the reception

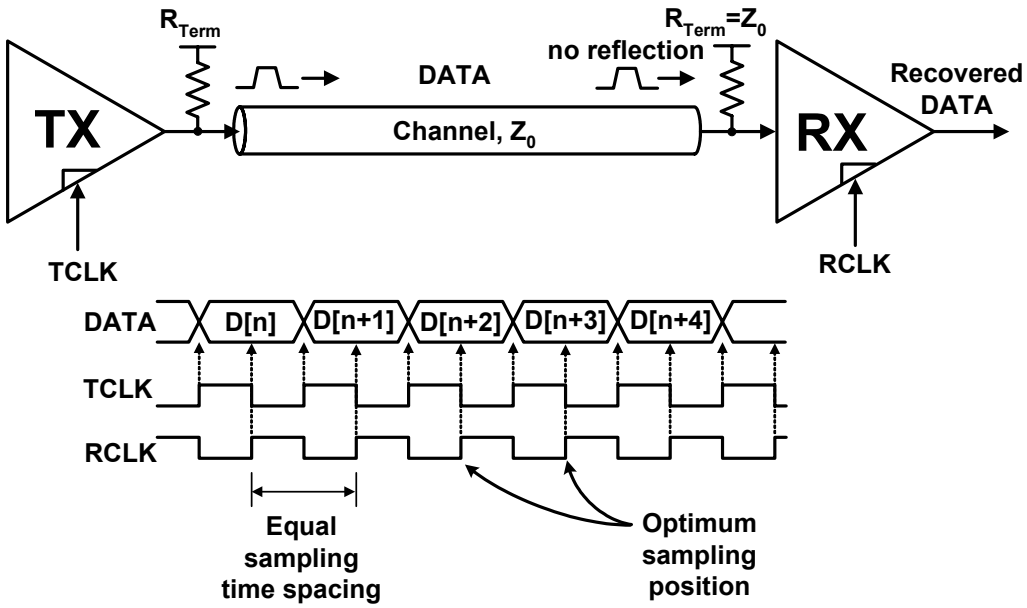


Figure 2.1: High-speed link system and corresponding timing between data and clocks. (Recreated from [39])

of each symbol at the receiver. A receiver on the opposite end of the channel recovers the analog waveform and converts it back to the original digital information, usually by sampling and quantizing it.

In this example, synchronous-mode transmission is assumed. This means data is transmitted and received with respect to a fixed time base, clocks TCLK and RCLK, as shown by the idealized waveforms in Figure 2.1. A precise clock generator is used to create TCLK, either a PLL or DLL, to transmit data at well-defined time intervals. In order to perform data recovery, the receiver needs accurate timing recovery in order to

align RCLK to sample the received DATA at the optimum position, typically at the center of each symbol interval. A precise timing generator is often used to synchronize the receiver sampling clock with respect to the incoming data stream [14][15].

Besides accurate timing, other factors limit the fastest operation speed that the link can achieve, one of which is the limited bandwidth of the process technology used to implement the link circuitry. Basic CMOS circuit speed improves as technology scales, and a metric called fan-out-of-four (FO-4) delay[1] is commonly used to represent CMOS circuit performance. The FO-4 delay can be roughly approximated to be $500ps/\mu m * L_{drawn}$ (Gate length) [16]. The minimum pulse width that can reliably propagate through an inverter chain, or clock distribution network, is around 3 to 4 FO-4 delays [39]. This leads to a minimum clock cycle time of 6 to 8 FO-4 delays. As an example, the maximum clock frequency that can be distributed in a 0.18μm process is about 1.4GHz. With one bit per clock cycle, this limits off-chip data rates to 1.4Gbps.

To meet the demand for higher off-chip data rates, designers have employed parallelism (interleaving) in time to send and recover data with the help of multiphase clocks. High data rates are achieved by relying on short timing intervals between multiple clock phases while the clock signal itself has a frequency much lower than the off-chip data rate [17][18]. Another approach to exploit parallelism in high-speed links is to use multilevel signaling. Multiple signal levels can encode more than one bit of information per symbol period (i.e. pulse amplitude modulation) and data rates higher than the clock frequency can be achieved [19]. These parallelism techniques can be

[1] A FO-4 delay is the delay through one stage in a chain of inverters, in which each inverter drives a capacitive load (fan-out) four times larger than its input capacitance [16].

employed to achieve high off-chip data rates without being limited by the maximum on-chip clock frequency of a given technology.

While multilevel signaling has been demonstrated to provide superior performance for severely band-limited channels, it does not preclude the use of interleaving [accelerant ref and others]. In order to investigate noise-reduction techniques associated with clock generators, this dissertation focuses on the design of a two-level receiver with time interleaving.

An interleaved receiver utilizes time-division multiplexing (TDM), as shown in Figure 2.2, to achieve higher off-chip bit rates while on-chip circuitry operate at a lower

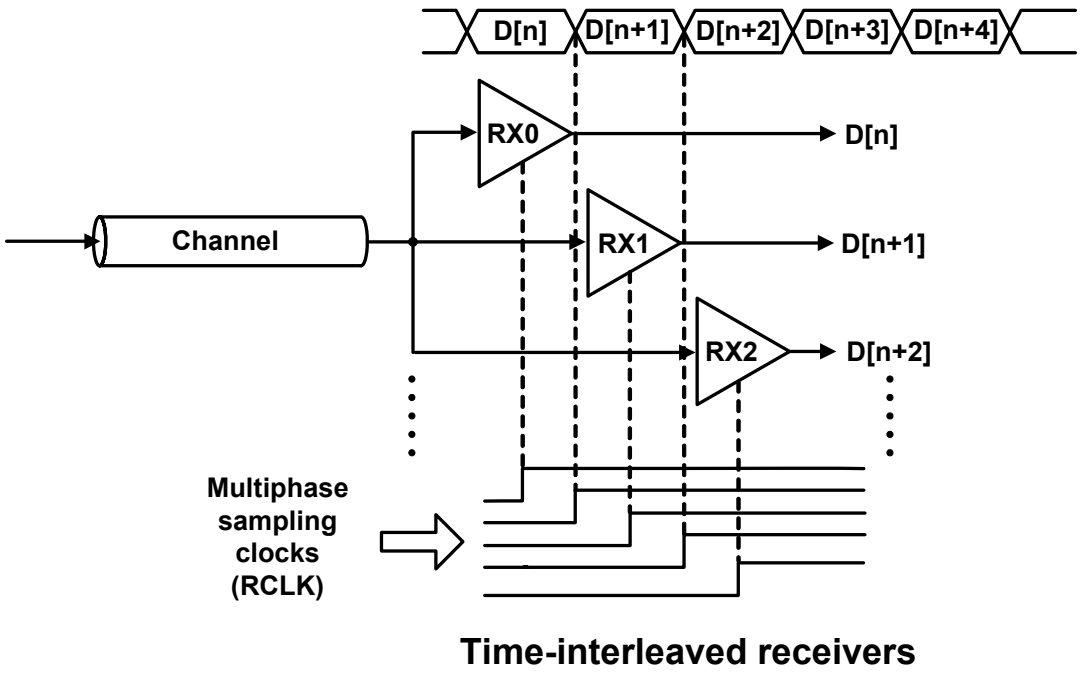


Figure 2.2: Time-division multiplexing with interleaved receivers.

clock frequency. The active period of each receiver is controlled by equally-spaced multiphase clocks that sample the data bits in the middle of their data symbol intervals. The receiver is also responsible for recovering embedded timing information from the data stream, assuming there is no clock signal being transmitted along with the data. Another set of equally-spaced multiphase clocks shifted by half of one symbol period with respect to the data-sampling clocks is aligned with data transitions by using a PLL or a DLL. With M clock phases that span one clock cycle, and clock frequency, F_{clk}, the bit rate is $M^* F_{clk}/2$ for 2-PAM signaling. M is usually limited by mismatch in the receivers, static phase spacing offset between multiphase clocks, and dynamic timing uncertainty on the clocks (jitter).

2.2 Review of Phase-Locked Loops

PLLs are commonly used to provide accurate timing signals for both transmit and receive sides of a high-speed link. This section reviews the operation of a PLL and derives the input phase transfer function and output noise transfer function using both continuous-time s-domain and discrete-time z-domain PLL models. These models can then be used to understand how a PLL reacts to different noise sources and compare its noise-rejection capabilities to other clock generators.

2.2.1 PLL Topology

In its simplest form, a PLL is a 2^{nd} order feedback system that generates a clock signal whose output phase is aligned with respect to the phase of an input reference clock.

Since phase is the integration of frequency, once the phases are aligned, both phase and frequency are "locked". This alignment is achieved by comparing the phase of the output clock with the phase of the reference. Any resulting difference in phase, the phase error, feeds into a block that filters this error and generates a control signal, typically a voltage. A voltage-controlled oscillator, where the oscillation frequency changes as a function of the control voltage, then creates the output clock. Through negative feedback, the control signal forces the output clock phase to be aligned with the input clock phase, resulting in zero phase error.

A block diagram of a commonly-used charge pump based PLL is presented in Figure 2.3, comprised of a voltage-controlled oscillator (VCO), a phase-frequency

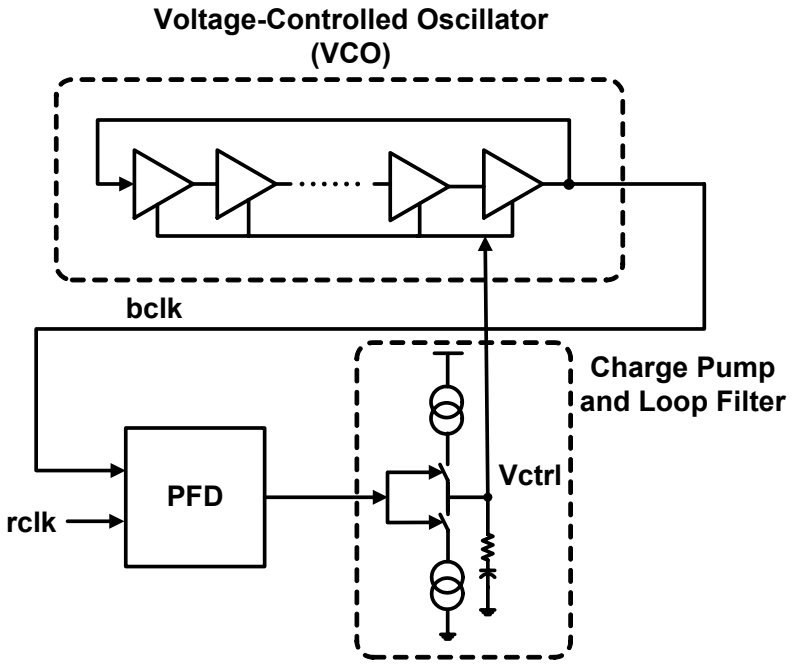


Figure 2.3: Charge-pump PLL block diagram.

detector (PFD), a charge pump (CP), and a loop filter (LF). The PFD compares the phase difference between the VCO output clock (bclk) and a reference clock (rclk), and then sends the phase difference information to the CP. When close to lock, only phase differences are detected. When out of lock, the PFD also detects frequency differences in order to prevent harmonic locking. The CP is comprised of two switched current sources, providing charging current and discharging current to the loop filter. The current sources are activated through two switches which are controlled by the output of the PFD. A net charge is dumped into or withdrawn from the LF, which contains a charge-integrating capacitor, depending on the phase difference information. The resulting control voltage (Vctrl) varies the VCO output frequency. Charge integration in loop filter and integration of frequency to generate a phase out of the VCO leads to a second-order loop. In order to stabilize the system, a zero must be introduced into the loop, by adding a resistor in series with the filter capacitor. When phase is locked, the phase error is zero and Vctrl remains stable, along with the VCO output frequency.

In order to better understand the loop dynamics and noise response of a PLL, we can use a linear model in the s-domain to model this feedback system.

2.2.2 S-Domain Modeling

A linear model of the PLL shown in Figure 2.3 is presented in Figure 2.4, using a continuous-time s-domain approximation[2]. Φref represents the phase of the input

[2] The continuous-time approximation is valid if the closed-loop bandwidth is much smaller compared to the input reference clock frequency [31].

reference clock (*rclk*), and *Φout* represents the phase of the output clock (*bclk*). Two noise sources are also added into the model – reference clock noise (*Nref*), which represents noise on the input reference clock (*Φref*), and output noise (*Nout*), which represents noise out of the VCO. Both of these noise sources lead to jitter on the output

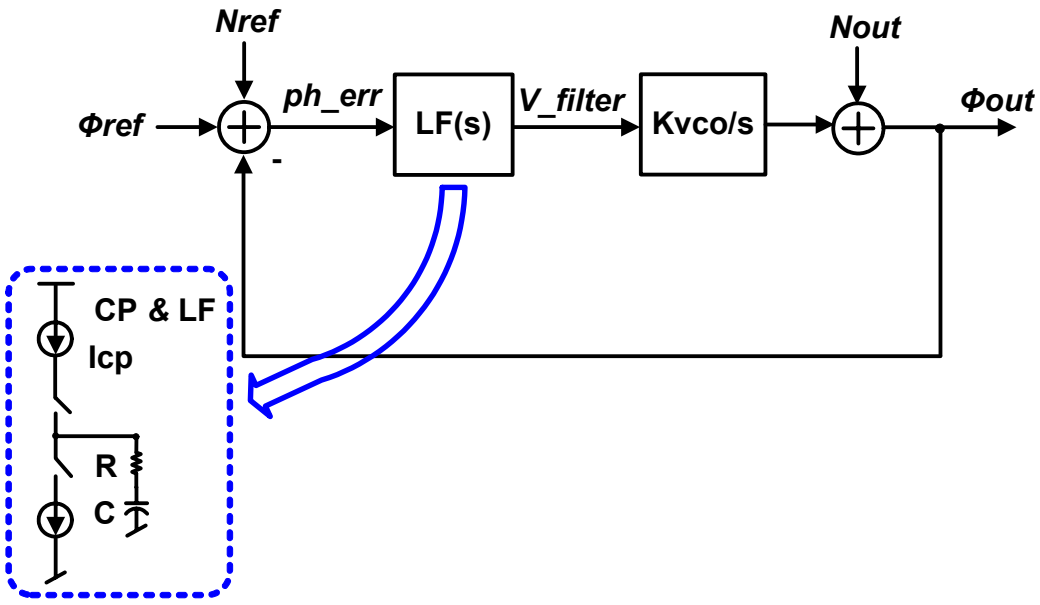


Figure 2.4: S-domain model for PLL.

clock phase. In order to derive the transfer functions corresponding to each noise source, let us first derive the open-loop transfer function, LG(s).

$$LG(s) = \frac{I_{CP}}{2\pi}\left(R + \frac{1}{sC}\right)\frac{K_{VCO}}{s} \tag{2.1}$$

I_{CP} is the charge pump current, and K_{VCO} is the gain of the VCO, which represents the change in oscillation frequency with respect to the control voltage (expressed in

Hertz/V). Given the open-loop transfer function, we can derive the closed-loop transfer function of the output phase with respect to the input phase.

$$\frac{\Phi_{out}}{\Phi_{ref}}(s) = \frac{2s\varsigma\omega_n + \omega_n^2}{s^2 + 2s\varsigma\omega_n + \omega_n^2} \tag{2.2}$$

$$\omega_n = \sqrt{\frac{I_{CP}K_{VCO}}{2\pi C}}, \varsigma = \frac{R}{2}\sqrt{\frac{I_{CP}K_{VCO}C}{2\pi}} \tag{2.3}$$

$$\omega_{-3dB} = \omega_n\sqrt{1 + 2\varsigma^2 + \sqrt{(1+2\varsigma^2)^2 + 1}} \tag{2.4}$$

Equation 2.2 is often called the input phase transfer function (PTF). ω_n is the natural frequency and ω_{-3dB} is the -3dB corner frequency of the PTF.

Now let us derive two key noise transfer functions (NTFs) in a PLL. For the reference noise transfer function, we notice that it is exactly the same as the input phase transfer function as shown in Equation 2.2. The PTF is a second-order low-pass transfer function suggesting that if *Φref* varies rapidly due to high-frequency noise, then *Φout* cannot track this variation. In other words, low-frequency jitter at the input propagates to the output un-attenuated, but high-frequency jitter gets filtered out.

Now, let us consider *Nout*, which represents jitter at the output of the VCO. *Nout* usually comes from device thermal noise and power-supply noise. To derive the PLL output noise transfer function, from *Nout* to *Φout,* notice that the output noise transfer function is just (1- *Φout*/*Φref*), shown in Equation 2.5.

$$\frac{\Phi_{out}}{N_{out}}(s) = \frac{s^2}{s^2 + 2s\varsigma\omega_n + \omega_n^2} \tag{2.5}$$

The PLL output noise transfer function has a high-pass characteristic. This indicates that low-frequency jitter components generated by the VCO are suppressed by the loop response, but high-frequency jitter components pass through the loop. This can be understood as follows. The PLL loop needs some time for the phase error generated at the VCO output to propagate thought the LF and to adjust the VCO frequency to correct this error. If the frequency content of the jitter is too high, the error produced by the PFD is mostly attenuated by the poles in the loop.

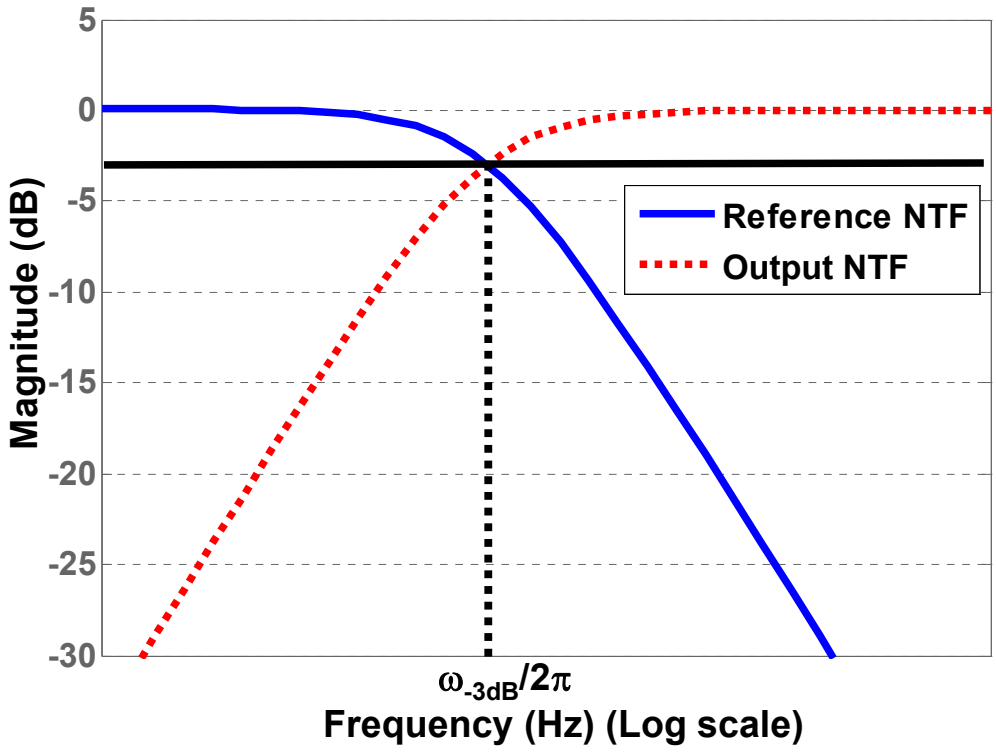


Figure 2.5: PLL reference noise transfer function and output noise transfer function in s-domain.

Figure 2.5 plots the two noise transfer functions, which summarize the responses of a PLL to input jitter and output jitter. Given that the corner frequencies of both noise

transfer functions are the same, there is a simple tradeoff in PLL design. Systems dominated by noise on the reference clock benefit from a low-bandwidth PLL while systems dominated by noise at the output require high bandwidths to minimize jitter. Depending on applications and environments, one or both noise sources may be significant, requiring an optimum choice for the loop bandwidth.

2.2.3 Z-Domain Modeling

This section investigates a discrete-time z-domain model of the PLL as shown in Figure 2.6. Since the PLL is a sampled system, a continuous-time approximation is only

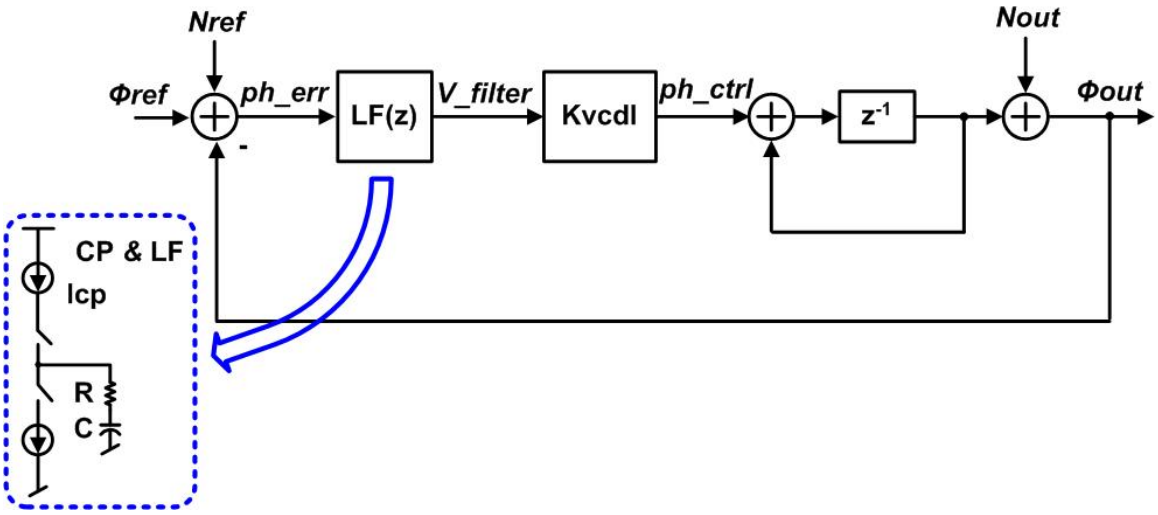


Figure 2.6: Z-domain model for PLL.

valid up to frequencies that are much lower than the reference clock frequency and the loop bandwidth is much lower than the sampling frequency. The z-domain model is a

more accurate model that captures the sampled nature of the system, and also facilitates the comparison between a PLL and DLL, presented later.

The CP and LF combined transfer function is replaced by its equivalent discrete-time model LF(z), and the K_{VCDL} is the delay line gain in radians per cycle per volt. T_{ref} is the period of the input reference clock. First, the LF(z) in s-domain is equivalent to (R+1/sC), therefore the pole at the origin can be replaced by a pole at z=1 in z-domain. The zero can be replaced by the expression shown in Equation 2.6[3]. The z-domain accumulator is equivalent to the transfer function of VCO in s-domain. The open-loop transfer function, LG(z), is expressed in Equation 2.7. The PLL input phase transfer function ($\Phi out/\Phi ref$) and VCO output noise transfer function ($\Phi out/Nout$) can be derived as shown in Equation 2.8 and Equation 2.9.

$$LF(z) = \frac{I_{CP} T_{ref} R}{2\pi} \frac{z-\beta}{z-1}, \beta = \exp(-\frac{T_{ref}}{RC}) \tag{2.6}$$

$$LG(z) = \frac{LF(z) K_{VCDL}}{z-1} \tag{2.7}$$

$$\frac{\Phi_{out}}{\Phi_{ref}}(z) = \frac{LG(z)}{1+LG(z)} = \frac{LF(z) K_{VCDL}}{z-[1-LF(z) K_{VCDL}]} \tag{2.8}$$

$$\frac{\Phi_{out}}{N_{out}}(z) = \frac{1}{1+LG(z)} = \frac{z-1}{z-[1-LF(z) K_{VCDL}]} \tag{2.9}$$

[3] Each factor of the form (s-a) in s-domain transfer function is mapped into the factor $(1-e^{aTref} z^{-1})$ in z-domain transfer function [40].

Figure 2.7 plots the PLL reference noise transfer function and VCO output noise transfer function in z-domain. If we compare Figure 2.5 with Figure 2.7, we notice that the continuous-time approximation is very close to the discrete-time representation.

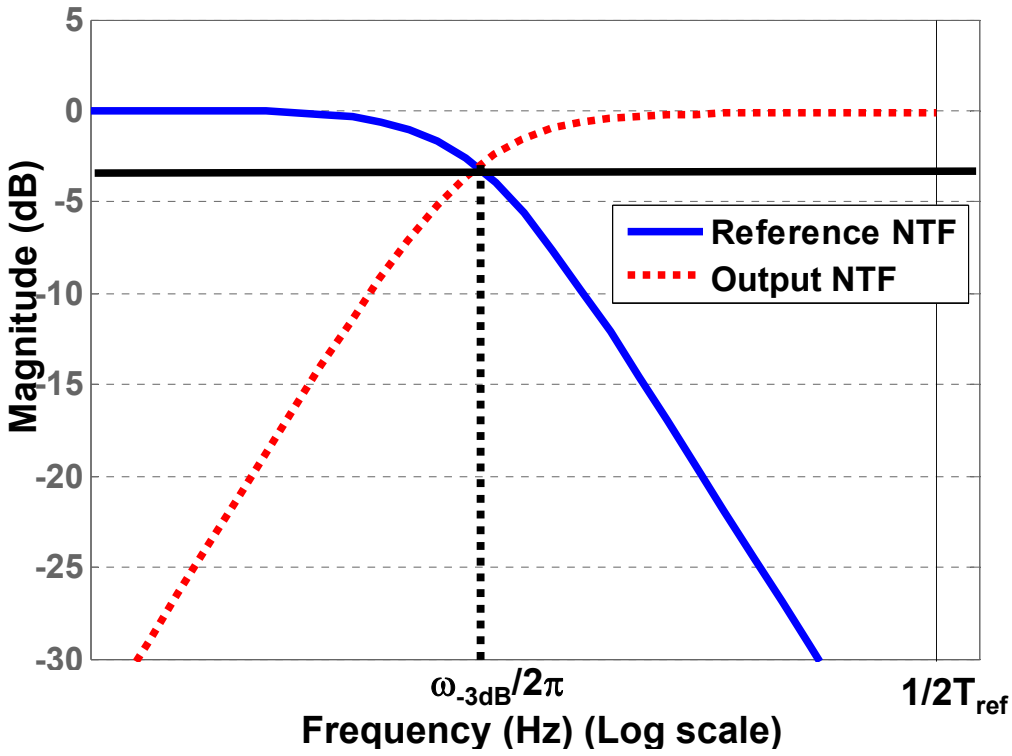


Figure 2.7: PLL reference noise transfer function and output noise transfer function in z-domain.

However, the z-domain model is only valid up to half of the reference clock frequency due to the sampled nature of the system. This confirms that when the bandwidth of the PLL is much lower than the reference clock frequency, the s-domain approximation is sufficiently accurate.

Another variant of clock generators that has become popular in the past decade is the DLL, which has very different phase and noise transfer function characteristics, when compared to those of a PLL. We take a look at the DLL system in the next subsection.

2.3 Review of Delay-Locked Loops

Like the PLL, a DLL can also provide an accurate timing relationship between an input clock and output clock. Therefore, it is also a viable option for clock generation in a high-speed link. The DLL is also a sampled system and so a discrete-time z-domain model can again be used to understand its characteristics. In particular, the input phase noise transfer function and output noise transfer function are derived, but let us first understand how a DLL works.

2.3.1 DLL Topology

A DLL is also a feedback system. Instead of using a VCO, a DLL utilizes a voltage-controlled delay line (VCDL) to generate an output clock that is a delayed version of the input clock, where the delay through the VCDL is a fixed fraction (often 50% or 100%) of the input clock period. While there are several ways to implement a DLL, Figure 2.8 presents a block diagram of a commonly-used charge pump based DLL topology. A phase detector (PD) measures the phase difference between coincident edges of the input and output clocks of the VCDL. The charge pump (CP) generates pulses of charge, proportional to the phase error, which is then integrated by the capacitor in the

loop filter (LF). The resulting voltage out of the LF sets the magnitude of the control voltage (Vctrl) that determines the delay through the VCDL. Since the reference clock (rclk) also feeds into the input of the VCDL, delay through the VCDL is fixed with respect to the period of rclk through negative feedback.

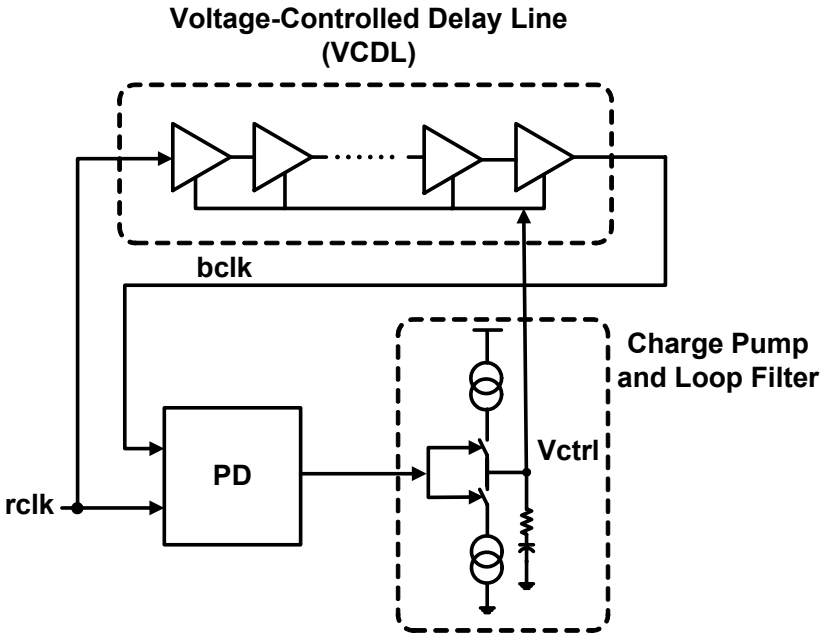


Figure 2.8: Charge-pump DLL block diagram.

A DLL is inherently simpler and easier to stabilize than a PLL. The VCO in a PLL must change frequency in order to adjust its output phase, introducing an integrator. In comparison, the phase of the clock edge out of the VCDL is directly controlled by Vctrl, which corresponds to a simple gain without integration. Hence, a DLL can be designed as a first-order loop and obviates adding proportional control to stabilize the loop. Moreover, frequency detection is not needed since the input and output clocks nominally have the same frequency. Given these differences between a PLL and DLL,

the DLL reacts differently to input and output noise when compared to a PLL. In order to understand the different characteristics, the next subsection briefly describes the loop dynamics of the DLL using discrete-time z-domain analysis.

2.3.2 Z-Domain Modeling

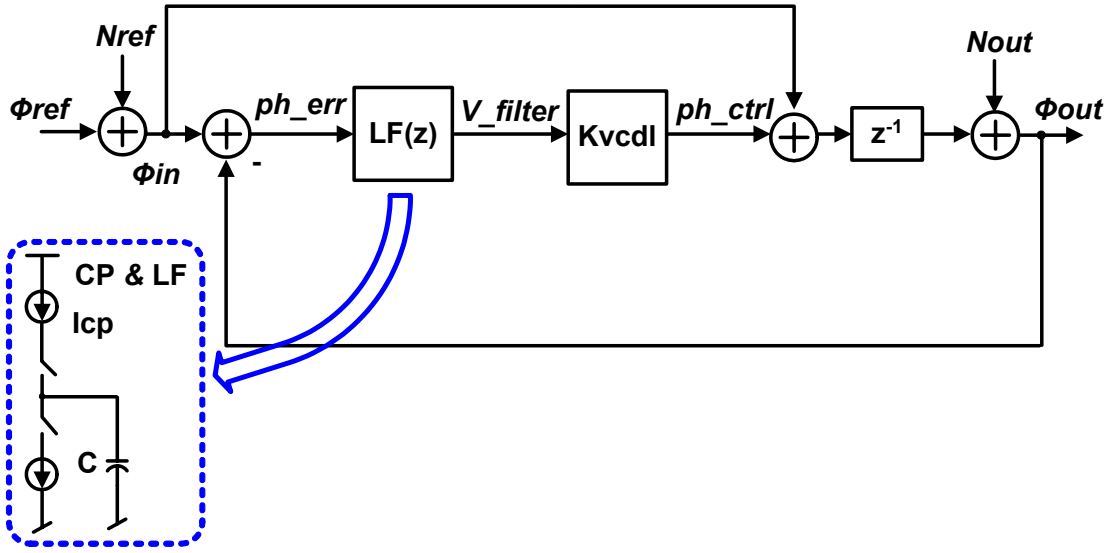


Figure 2.9: Z-domain model for DLL.

It is possible to analyze the frequency response of the DLL with a continuous-time approximation, where the sampling operation of the PD is ignored. However, this approximation fails to capture important phase transfer characteristics that govern DLL operation. Given the inherent discrete-time nature of the DLL, a z-domain analysis is more appropriate.

Figure 2.9 presents a z-domain model of the DLL discussed above. Remember that a DLL delays the input clock by a controlled amount of time. Therefore, the output clock phase is actually derived from the input clock phase one cycle before plus a corrected phase term from the output of PD and LF. The phase error term, *ph_err*, is filtered by LF and the resulting voltage, *V_filter*, is transformed into a phase, *ph_ctrl*. K_{VCDL} is the delay line gain in radians per cycle per volt. After combining the phase correction term with the input reference clock phase, a one-cycle delay is applied. Supply noise modulates the delay of the VCDL and its effects are modeled by simply adding a noise term, *Nout*, before the output node.

Again, we can derive the PTF and NTF for the DLL based on the z-domain model shown in Figure 2.9. Since a compensating zero is not required, the combined CP and LF transfer function is simplified with a single integrating capacitor. T_{ref} is the period of the input reference clock. Like what we have done in the PLL z-domain model, the pole at the origin formed by the integrating capacitor in LF can be replaced by the factor $1/(1-z^{-1})$ in z-domain.

$$LF(z) = \frac{K_{CP} z}{z-1}, K_{CP} = \frac{I_{CP} T_{ref}}{2\pi C} \tag{2.10}$$

Based on the above expression for LF, the input-to-output phase transfer can then be represented by the following expression.

$$\frac{\Phi_{out}}{\Phi_{in}}(z) = \frac{(1 + K_{VCDL} K_{CP})z - 1}{z[z - (1 - K_{VCDL} K_{CP})]} \tag{2.11}$$

Plotting the frequency response of this expression, in Figure 2.10, reveals that the input-to-output phase transfer function has an all-pass characteristic, which implies that high

frequency jitter on the reference clock passes straight through to the output without being filtering, as was the case in a PLL. Actually it is even worse. The high frequency jitter on the reference clock is slightly amplified for frequencies greater than

$$\frac{1}{2\pi T_{ref}} \ln\left(\frac{1}{1 + K_{VCDL} K_{CP}}\right),$$

which is the frequency of the zero in Equation 2.11. This jitter peaking problem is inevitable in this type of DLL design, because for positive values of $K_{VCDL*} K_{CP}$, the frequency of zero, $(1/(1+K_{VCDL*} K_{CP}))$, is always larger than the frequency of the pole, $(1-K_{VCDL*} K_{CP})$, in Equation 2.11. When the z-domain frequencies are mapped into s-domain frequencies, the zero will be mapped into a frequency that is lower than that of the pole, thus causing the peaking. The peaking can be mitigated by adding higher-order poles into the LF at a cost of potential instability of the loop [29].

The z-domain model can also be used to derive the NTF of the DLL to understand how it reacts to jitter injected at its output. The expression is given below.

$$\frac{\Phi_{out}}{N_{out}}(z) = \frac{z-1}{z-(1-K_{VCDL} K_{CP})} \tag{2.12}$$

Figure 2.10 presents a frequency response plot of Equations 2.11 and 2.12, and it shows that the DLL's output noise transfer function also has a high-pass characteristic, similar to that seen for a PLL. The high-pass corner frequency corresponds to the bandwidth of the loop. This high-pass behavior can be explained with the same reasoning used in the case for the PLL. For low-frequency jitter injected at the output that is within the bandwidth of the loop, the negative feedback loop is able to suppress its effects in order to keep the output clock phase fixed with respect to the reference clock. However,

as the frequency content of the injected jitter increases, the loop is no longer able to filter

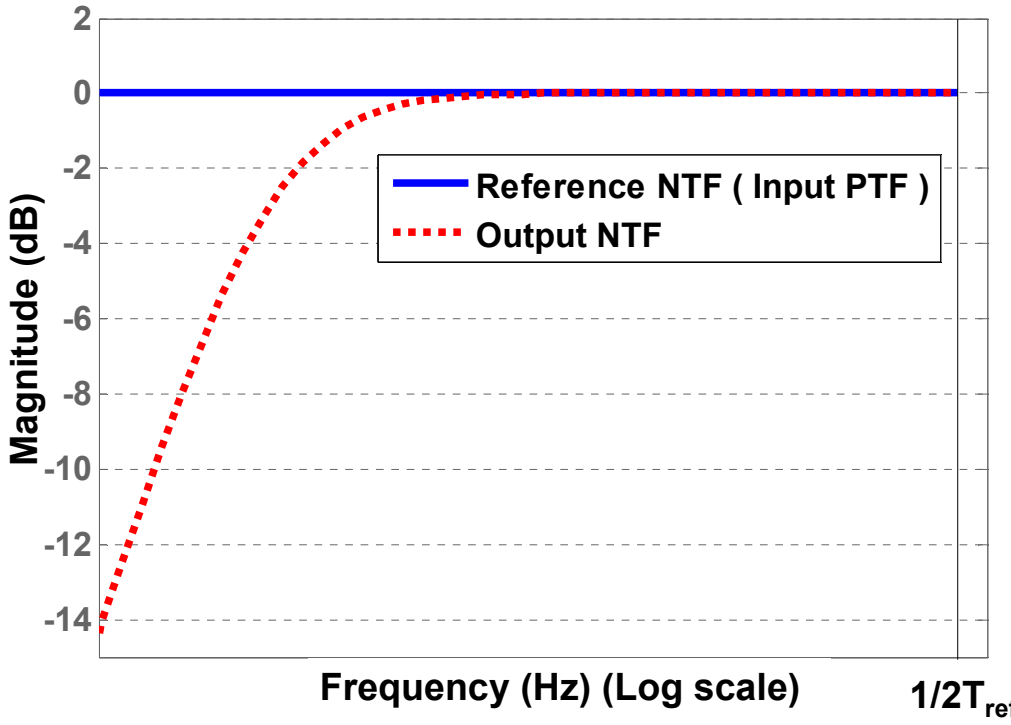


Figure 2.10: DLL input phase transfer function and output noise transfer function.

out the jitter and it passes straight through to the output. While the NTF responses of both the PLL and DLL are similar, there is an important distinction to point out. In a PLL, the output phase can only move by changing the VCO frequency, while in a DLL, the loop can directly move the output phase by adjusting the delay through the VCDL. This difference enables the DLL to respond much more quickly to noise injected at the output. Moreover, power supply noise is a dominant source of noise that must be dealt with in order to achieve low jitter. The VCDL used in a DLL has the added advantage of not accumulating the noise, as opposed to the VCO in a PLL. Therefore, power-supply noise

that may couple into the VCDL only affects the output phase over a single clock period. These distinctions behoove us to take a closer look at the comparison between PLL and DLL characteristics.

2.4 PLL and DLL Comparison

Since clock timing uncertainty is one of the most important factors that determine the performance of high-speed links, this subsection takes a closer look at the comparison of the noise responses of a PLL versus a DLL. The difference between using a VCO and a VCDL makes the noise responses of a PLL and a DLL very different. There are several dominating noise sources in a PLL and a DLL as reported in [26] and [27]. Two of the most significant noise sources in the system are: (1) Input reference clock noise and (2) Supply- or substrate-induced noise. Therefore, the noise responses of a PLL and a DLL are presented with respect to these two noise sources based on the z-domain models I derived in Section 2.2 and 2.3.

A clean input reference clock might not be available for some high-speed applications in order to reduce the cost. The output phase responses of the PLL and the DLL with respect to the reference clock noise are characterized by their input-to-output phase transfer functions (PTF) based on their respective z-domain models, plotted together in Figure 2.11. The PLL is a by design a higher order system, while the DLL can be approximated as a first-order system [22]. The PTF of a PLL typically exhibits low-pass filtering characteristics while the DLL exhibits all-pass filtering characteristics.

Therefore, the PLL loop can filter high-frequency noise on the reference clock, which

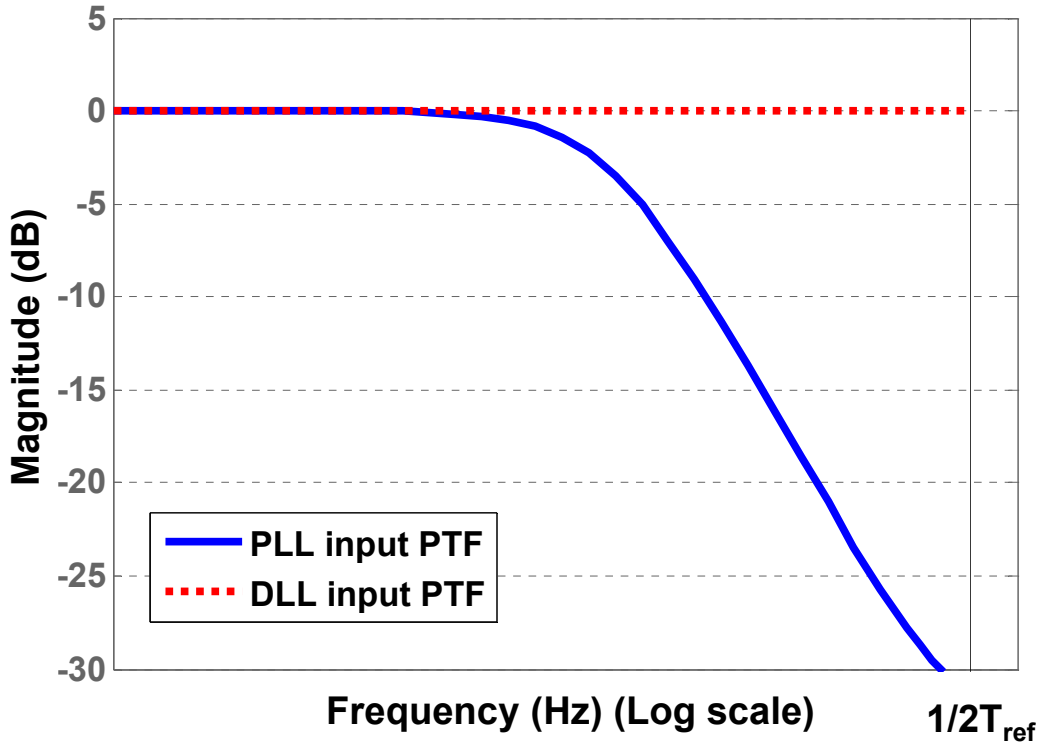


Figure 2.11: PLL and DLL reference-to-output transfer functions (PTF) in z-domain.

extends beyond its bandwidth. On the contrary, the DLL does not filter out any of the noise on the reference clock noise. Moreover the DLL's PTF also reveals high-frequency jitter peaking, making it unsuitable for applications where there is a significant amount of high-frequency noise associated with the input reference clock. In such cases, it is desirable to utilize a low-bandwidth PLL that can suppress noise on the reference clock.

Both output noise transfer functions (NTF) of the PLL and DLL have high-pass characteristics as was shown in Section 2.2 and Section 2.3.

To better understand how these clock generator loops respond to power supply noise, Figures 2.12 and 2.13 recreate the two discrete time z-domain models of the PLL

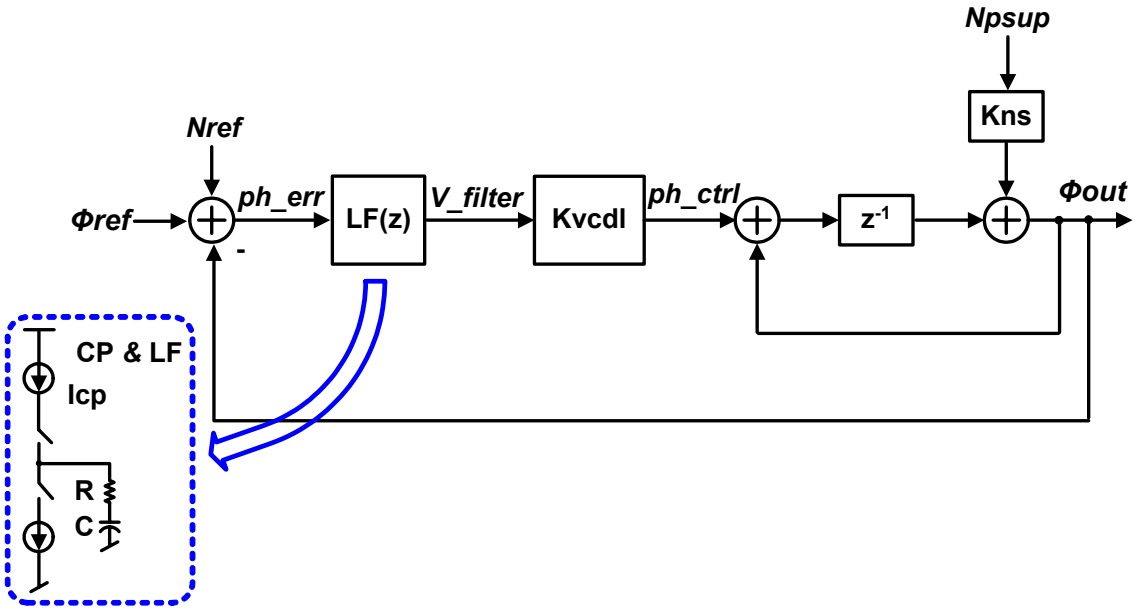


Figure 2.12: Z-domain model for PLL with power supply noise.

and DLL with some minor differences. Notice that power supply noise, Npsup, has been added to the models, which is scaled by a power supply noise sensitivity gain term, Kns. If identical delay elements are used in both the PLL and DLL, then Kns is the same in both models. The resulting expressions for the power supply noise to output phase transfer functions for the PLL and DLL are as follows.

$$PLL: \frac{\Phi_{out}}{N_{p\sup}}(z) = \frac{K_{ns} z}{z - [1 - LF(z)K_{VCDL}]} \tag{2.13}$$

$$LF(z) = \frac{I_{CP} T_{ref} R}{2\pi} \frac{z-\beta}{z-1}, \beta = \exp(-\frac{T_{ref}}{RC}) \tag{2.14}$$

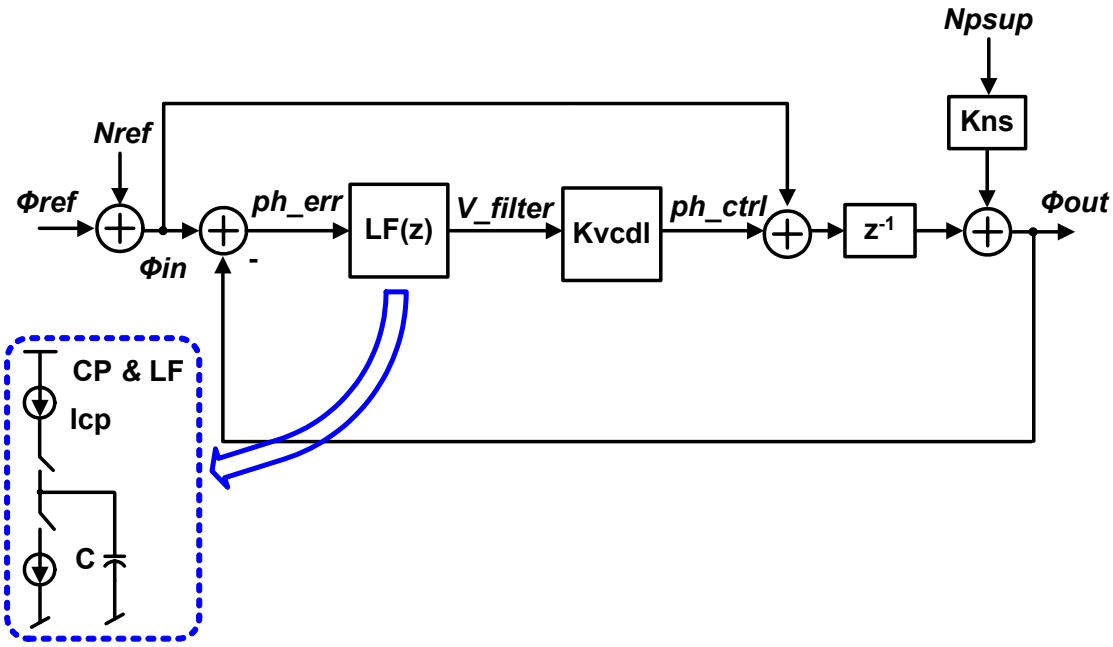


Figure 2.13: Z-domain model for DLL with power supply noise.

$$DLL: \frac{\Phi_{out}}{N_{p\,\sup}}(z) = \frac{K_{ns}(z-1)}{z-(1-K_{VCDL}K_{CP})} \tag{2.15}$$

$$K_{CP} = \frac{I_{CP} T_{ref}}{2\pi C} \tag{2.16}$$

As shown in Figure 2.14, plotting the frequency response of these expressions reveals a remarkable difference in how each loop responds to power supply noise. Notice that due to integration in the VCO, power supply noise is accumulated in a PLL. In comparison, the DLL does not exhibit any integration of power-supply noise but exhibits the same

response as the VCDL output noise transfer function does. Figure 2.15 plots the power supply noise transfer functions of the PLL and DLL with several different -3dB

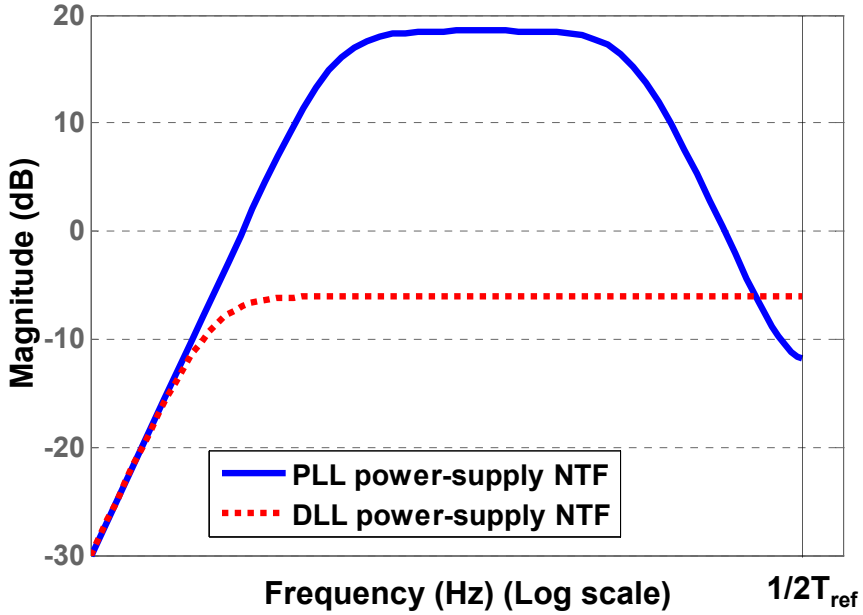


Figure 2.14: PLL and DLL power-supply noise transfer functions (with Kns=0.5) in z-domain.

bandwidths while keeping the same damping ratio of the PLL. It clearly shows that in order to reduce the accumulated power supply noise, a PLL with a higher bandwidth is desirable. In the DLL case, a higher bandwidth is also preferred. However, this advantage in the DLL case is not as significant as that in the PLL case. Hence, while a PLL is able to filter a noisy reference clock, it is sensitive to power supply. On the other hand, a DLL cannot reject noise on the reference clock, but it is more resilient to power supply noise.

These comparisons of loop dynamics reveal there are clear tradeoffs between PLL and DLL designs. In a noisy environment, such as a mixed-signal system-on-a-chip

(SoC) that suffers from switching activity in neighboring digital circuitry, a DLL-based clock generator offers advantages. However, if the reference (system) clock quality is extremely poor and requires additional jitter filtering, a PLL-base clock generator may provide cleaner clock signals. While the choice is clear for the above two extreme cases, the exact noise conditions may not be known or sufficiently characterized in advanced.

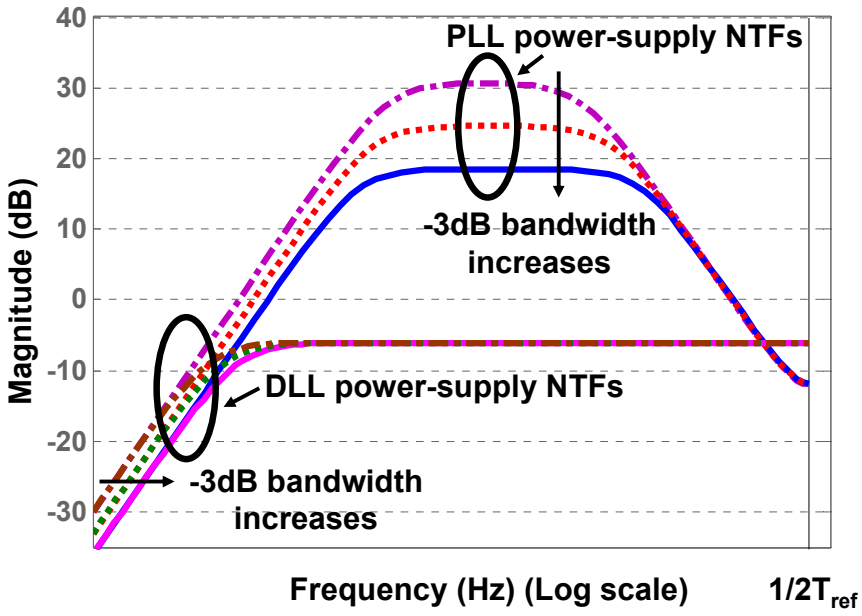


Figure 2.15: PLL and DLL power-supply noise transfer functions (with Kns=0.5) with different -3dB bandwidths in z-domain.

Hence, the complimentary requirements resulting from power-supply noise versus noise on the reference clock, motivates the need for a clock generator that can adapt to different noise conditions. Chapter 3 presents a clock generator design that specifically addresses this need by implementing a clock generator that merges the PLL and DLL into a single loop and can be tuned to operate as a PLL, a DLL, or a combination of the two. Another

possible solution is to design a PLL whose bandwidth can be adjusted depending on the noise environment. The next subsection reviews two pieces of prior art that attempt to achieve that objective – an adaptive-bandwidth clock generator and a phase-realigned clock generator.

2.5 Prior Art in Adaptive-Bandwidth PLL Design

Based on discussions in Section 2.4, the PLL has conflicting bandwidth requirements with respect to reference noise filtering and supply noise rejection. So, if we can somehow "tune" the bandwidth of the PLL such that it can adapt to different noise conditions, the jitter can be minimized. The bandwidth of the PLL is tightly coupled with the charge pump current, gain of the VCO, and the parameters in the loop filter. Normally, for a given process technology, the gain of the VCO is a relatively fixed parameter. Tuning the passive components in the LF is not very efficient either [34]. Therefore, the natural choice would be to tune the CP current as described by Mansuri and Yang in [26]. By implementing the stabilizing resistor in the LF using a charge pump, the resistor value can also be tuned efficiently. Mansuri and Yang show that the VCO-induced jitter can be minimized by increasing the loop bandwidth and improving the damping of the loop. However, this design requires more area and power than a conventional PLL design in order to get a wide tuning range.

Another interesting approach, which does not tune CP or LF parameters, is to "clean up" the accumulated VCO-induced jitter periodically [32][33]. This realigned-PLL (RPLL) topology performs VCO phase realignment by injection locking the VCO to a strongly buffered version of the PLL reference clock at fixed intervals of the reference clock. Figure 2.16 presents a block diagram of this topology. A buffered version of reference clock (Ref_clk) is injected into one of the delay cells in the VCO every reference cycle to pull the VCO internal clock edge toward the ideal position. If the

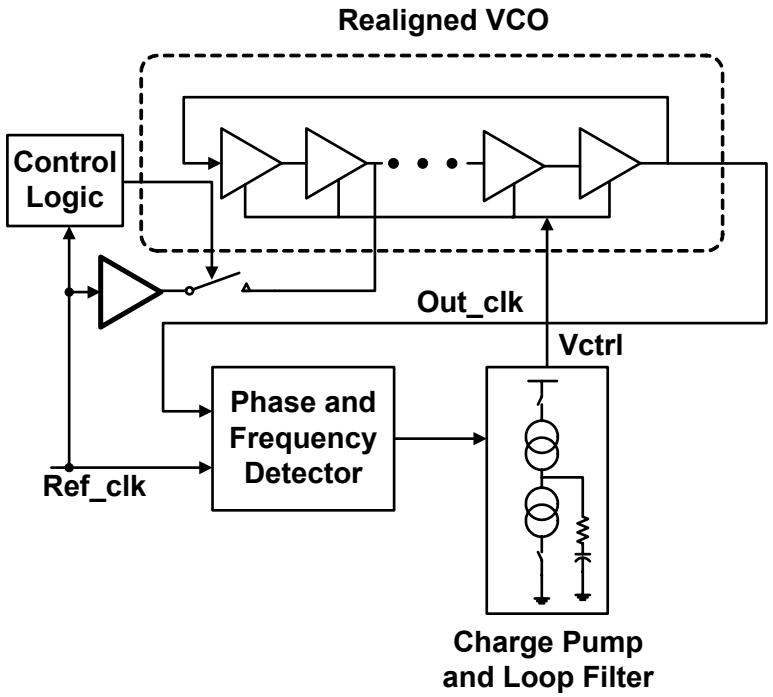


Figure 2.16: Realigned PLL (RPLL) topology.

amount of phase error correction is big enough to correct the phase difference between the buffered Ref_clk and the internal clock edge prior to the realignment, all the memory

of the past errors at the VCO output clock (Out_clk) can be completely suppressed. If the reference clock is clean, the accumulated noise in the VCO can be suppressed by resetting the accumulated phase error; thus, improving the supply noise rejection of the RPLL by realignment. The relative strength of the injection locking buffer to the delay cells in the VCO determines the "amount" of phase realignment performed.

There are several drawbacks associated with this design. The most significant drawback is that if there's a difference in delay (τ) through the reference clock injecting buffer versus the VCO buffers, identified by arrows in Figure 2.16, the injection can create a phase mismatch between the reference clock edge (Ref_clk) and the output edge out of the VCO (Out_clk). In other words, the injection can fight with the normal locking function of the feedback loop and can lead to pattern jitter at the VCO output set by the periodicity of injection. This corresponds to a strong injection-induced spur when observing the spectrum of the output clock. The other drawback is that the amount of realignment is very difficult to control due to the coupling between the injection buffers and VCO delay cells, which varies with process and temperature. A similar design, presented in Chapter 3, overcomes some of these drawbacks to enable a clock generator with wide bandwidth tuning capabilities.

2.6 Clock and Data Recovery Architecture

Review of Delay-Locked Loops

We have seen that a synchronous transceiver transmits data symbols at fixed timing intervals with respect to a clock signal. In the absence of a clock that is forwarded

along with the data, this type of link requires a clock and data recovery (CDR) block that can recover both data and timing information from the incoming symbol stream. The CDR needs to sample the symbol at an optimum position within the symbol and convert the sampled value to digital bits. Moreover, the CDR has circuitry to monitor transitions between symbols in order to generate the requisite sampling clock signal. There are a variety of CDR architectures that can be used with different tradeoffs to meet the requirements of different transceiver signaling schemes and applications. We will focus our understanding of CDR operation based on a generic interface that uses a CDR to recover two-level NRZ (non-return-to-zero) data, where the voltage for each symbol lasts the entire period without returning to a reference value. In order to recover this type of data, some minimum data transition density is required to prevent the CDR from drifting away from the optimal sampling position when long consecutive "0"s or "1"s are transmitted along the channel. We can assume an 8B/10B encoding scheme, which maps 8-bit words to 10-bit symbols, to ensure sufficient transition density. Such encoding schemes are often used in many link applications [23][24].

The next two subsections review three commonly-used CDR architectures to solidify our understanding of how CDR blocks operate and identify several tradeoffs. A subsequent subsection then reviews several metrics that are used to evaluate the performance a CDR.

2.6.1 PLL-based CDR

One of the most commonly-used CDR designs is a PLL-based CDR [25] to recover NRZ data, as shown in Figure 2.17. The example shown is also called a full-rate

CDR where the data symbol time corresponds to one full period of the recovered clock. This architecture is essentially operates like a PLL and consists of a VCO, a charge pump (CP), a loop filter (LF), a data sampler, a phase detector (PD), and a frequency detector (FD). Let us first focus on the upper CDR phase-tracking loop. The data feeds into a PD/Sampler block, which is clocked by the clock signal out of the VCO. When locked,

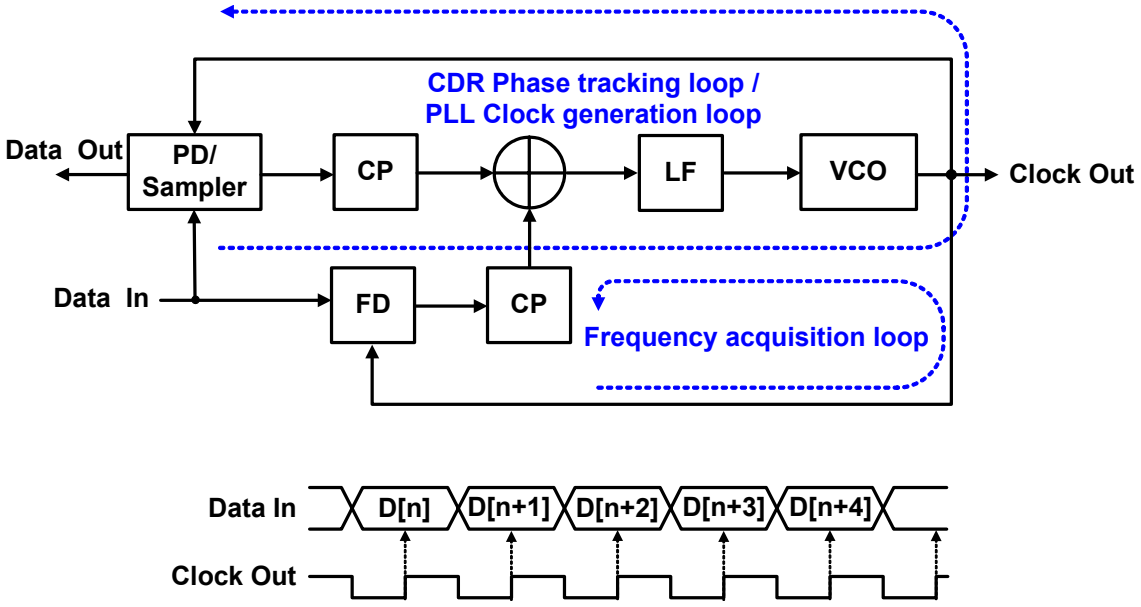


Figure 2.17: A PLL-based single-loop CDR architecture with no reference clock [25].

the falling edge of the clock samples the data in the middle of the symbol and the rising edge samples data transitions. This sampling of data edge transitions performs the phase detection function required by the phase tracking loop. The resulting phase error feeds the CP and subsequent loop filter to control the frequency and phase of the VCO output. Through negative feedback, the recovered clock, Clock Out, is aligned in phase and

frequency to the incoming data symbol. This operation is much like that of a traditional PLL design seen in Section 2.2.

Since data transitions are not guaranteed to exist on every rising transition of the recovered clock, the FD is used in a frequency-acquisition loop to aid the process of locking the recovered clock frequency with respect the frequency at which the data was transmitted. The additional frequency-acquisition loop is necessary, because the PD in the phase-tracking loop does not provide frequency information, which may lead the loop into false lock conditions. The FD reduces the frequency difference between data and VCO output. Once the frequency difference is small enough, the PD takes over to align the VCO's output clock with the data. When both phase and frequency of VCO output are in lock with the data, the FD output is zero, and will not affect the operation of the loop.

This architecture has many advantages. It is a relatively simple and compact design. It does not require a separate reference clock and the switching between frequency tracking and phase tracking is continuous and smooth. Moreover, this PLL-based design enables the CDR to track some frequency deviations on the input data.

The biggest disadvantage of this CDR is that the clock generation loop and the CDR phase tracking loop is essentially the same loop, and this results in a tradeoff between the jitter transfer and jitter tolerance requirements. The jitter transfer function of the CDR is the transfer function from the jitter on the input data to the recovered clock. In this particular architecture, the jitter transfer function of the CDR is the same as the input phase transfer function of the PLL when in lock. Jitter tolerance is a measure of the ability of the CDR to track jitter on the input data. It is often described as a plot of the

maximum amount of jitter allowed on the input data in terms of U.I. (unit interval) that leads to some preset BER (bit-error-rate) versus the frequency of the jitter. Implementing a higher closed-loop bandwidth for the CDR increases the amount of jitter it can "tolerate." In other words, the CDR can track large amounts of jitter while maintaining small phase differences between the input data and clock[4]. A large bandwidth also helps suppress on-chip noise. However, given that data transitions are not available every clock period, there is a limit to how high the bandwidth can be in order to guarantee stability. Moreover, a higher loop bandwidth results in more jitter on the data getting transferred to the VCO output. Thus, the jitter transfer suffers. For some applications, e.g., SONET (Synchronous Optical Network), the jitter-transfer and jitter-tolerance requirements cannot easily be met at the same time with this single-loop architecture (where the CDR and PLL share the same loop configuration).

To decouple the tradeoff between the jitter-transfer and jitter-tolerance requirements, a dual-loop architecture that separates the clock generation loop and the CDR phase tracking loop is commonly used and described next.

2.6.2 Dual-Loop CDR Architectures

A dual-loop CDR architecture was proposed by Sidiropoulos and Horowitz in [3]. Figure 2.18 presents a block diagram of this CDR architecture. The major difference

[4] The jitter tolerance is related to the open loop transfer function of the CDR phase tracking loop. When the CDR and PLL share the same loop configuration, the CDR can track larger low-frequency jitter if the PLL has a higher open-loop crossover frequency.

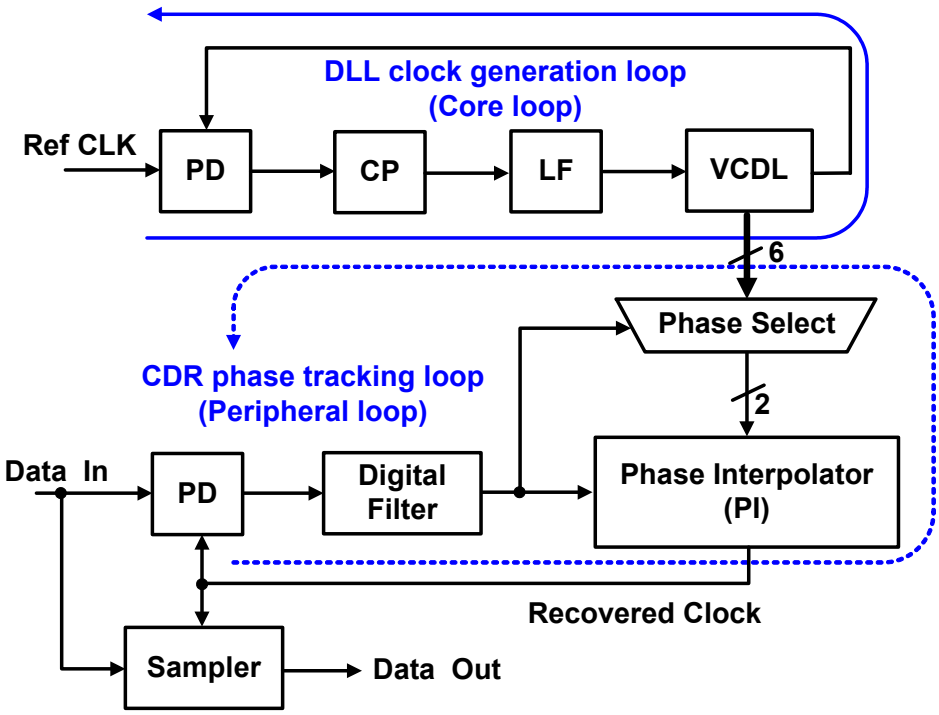


Figure 2.18: Dual-loop DLL-based CDR architecture [3].

between this architecture and the one shown in Figure 2.16 is the separation of the clock generation loop and the CDR phase tracking loop. A DLL block locks the output clock phase to a reference clock and generates six evenly-spaced clock phases[5] for the phase selection circuitry. This DLL is often referred to as the *core loop*. The CDR phase tracking loop (peripheral loop), consisting of a digital phase detector (PD), a digital filter, a phase select block, and a phase interpolator (PI), then locks recovered clock with respect to the data in order to recover and retime the data via the sampler. Based on a digital code out of the digital filter, the phase selection circuitry selects two adjacent phases that are then interpolated by the PI. The PI has the characteristics of creating a

[5] The number of clock phases generated by the core clock generation loop is a parameter of the designer's choice for different applications.

clock signal whose position lies somewhere between the two input clock signals. An analog PI can arbitrarily position the output edge with respect to an analog signal while a digital PI is limited to quantized positions, which is used in this example. This peripheral loop can track the phase of the incoming data by controlling the phase select and PI block via the digital filter output. Frequency tracking is also possible by sweeping the digital filter output at a rate that corresponds to the difference in the frequency of the incoming data and the local reference clock, when the two clock frequencies differ. Decoupling the phase-tracking loop from the clock generation loop offers several advantages, but this implementation also suffers from some drawbacks.

Cascading two loops in dual-loop DLL architectures offers some advantages. It does not suffer the same tradeoff between jitter-transfer and jitter-tolerance requirements as was the case for the single-loop CDR architecture, previously seen. Designers can choose different loop bandwidths for the core loop and the peripheral loop, independently. For example, the DLL clock generation loop can be replaced with a PLL if the input reference clock is noisy. Moreover, this CDR architecture can benefit from an adjustable-bandwidth clock generator that can be tuned depending on differing noise conditions.

While this architecture offers the flexibility of decoupling bandwidths in the two loops, one must also be cognizant of its drawbacks. For example, if this architecture were to used in an interleaved receiver, where multiphase clocks are needed to sample parallel receivers, the number of phase interpolators (PI) would have to increase by the degree of parallelism desired [1]. If the degree of parallelism is large, this can lead to large power and area penalty. Another drawback of this architecture is the digitally-controlled

peripheral loop creates quantized phase step at the output of the PI. This quantization noise is directly transferred to the sampling clock, therefore, either fine phase steps in the PI is required to minimize the quantization noise, or a jitter-filtering block is needed to low-pass filter the quantization noise [11][12].

Since the quantization noise due to the digitized phase rotator step usually sets the limit of the maximum peak-to-peak jitter on the sampling clock, it always needs to be minimized in order to increase the maximum data rate. Another dual-loop CDR architecture based on feedback clock selection was proposed by Larsson in [9]. Figure 2.19 presents the block diagram of this CDR architecture. Again, it consists of two loops, but now the core clock-generation loop is embedded within an outer phase-tracking peripheral loop. This approach contains nearly the same components as the previous design, but configured in a different manner. Instead of using the PI's output to directly sample the data, the phase selection circuitry selects and the PI interpolates between appropriate VCO output phases to be fed back to the PFD. Therefore, the digital code out of the filter can adjust the phase of the VCO output clock with respect to Data In while maintaining phase and frequency lock with respect to Ref CLK. It is important to note that in this topology, the core loop must be a PLL. Frequency tracking can be achieved by, again, sweeping through the digital codes generated by the digital filter, which then effectively changes the frequency between Ref CLK and the VCO.

This design offers some advantages over the previous one. It attenuates the quantization noise on the sampling clock by using the low-pass filtering characteristics of the PLL's PTF to filter the quantization noise resulting from the digital controller. Moreover, now that the VCO in the core loop is locked in frequency and phase with

respect to the incoming data, this topology can be used in an interleaved receiver design by directly tapping out phases out of the VCO.

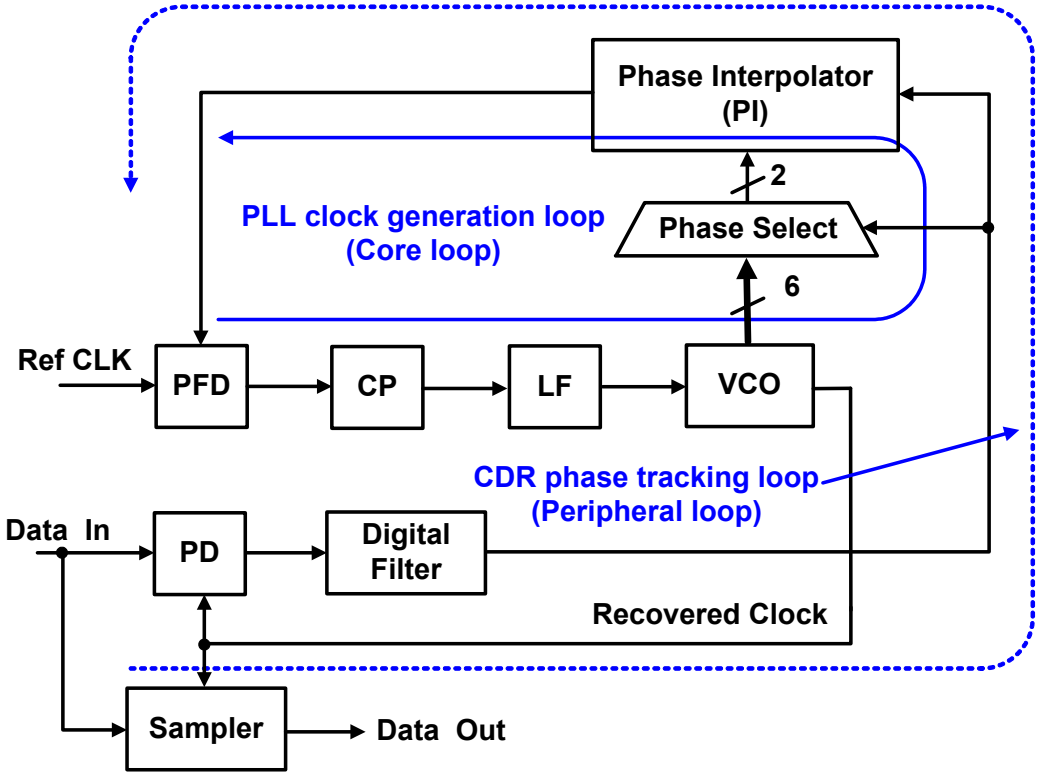


Figure 2.19: A PLL-based feedback phase selection dual-loop CDR architecture [9].

Unfortunately, with the above advantages come a few disadvantages. While this architecture also keeps the clock generation loop and CDR phase tracking loop somewhat separate, their bandwidths cannot be set independently. Given the nested loops, to avoid loop interactions and possible instability, the two bandwidths must be separated by at least an order of magnitude. Moreover, since the core loop is used to filter quantization noise, the PLL bandwidth must be sufficiently low. This means the core loop is more susceptible to on-chip noise, such as power-supply noise.

We have seen that the dual-loop CDR architectures are preferred, because they offer the ability to decouple the tradeoff between jitter-transfer and jitter-tolerance requirements. However, an explicit reference clock is needed in such architectures. Both PLL- and DLL-based dual-loop architectures have been widely used and reported in research literature. While they offer several advantages, the appropriate design must be chosen based on expected noise conditions. For example, for noisy on-chip environments, the cascaded loops are preferred in order to use DLLs. On the other hand, if quantization noise (or noise on the reference clock) dominates, the nested dual-loop design is preferred. Again, two solutions are available for two different noise conditions, which behoove us to ponder whether another CDR topology exists; one that can adapt to different noise conditions. Remember that if we can have an adaptive-bandwidth clock generator to accommodate various noise conditions, we can achieve the minimum sampling clock jitter for a given design. Moreover, if we can incorporate the adaptive-bandwidth clock generator into the CDR architectures we discussed above, one may also be able to adjust the closed-loop bandwidth of the CDR.

Before proceeding to the next chapter, we take a short detour to review the various dominating sources of noise that link designers must contend and metrics commonly used to characterize CDR performance.

2.7 Dominating Sources of Noise in a CDR

In order to understand how timing uncertainty on the sampling clock can degrade high-speed link receiver performance, Figure 2.20 presents an illustration of a data eye

diagram, where t_b represents the bit time or symbol interval. Due to timing uncertainty of data transitions, t_U, the sampling clock uncertainty cannot exceed t_{SH}. Otherwise, errors in recovering the data can occur. Therefore, sampling clock uncertainty needs to be minimized in a high-speed link receiver in order to reduce bit times and maximize data rates.

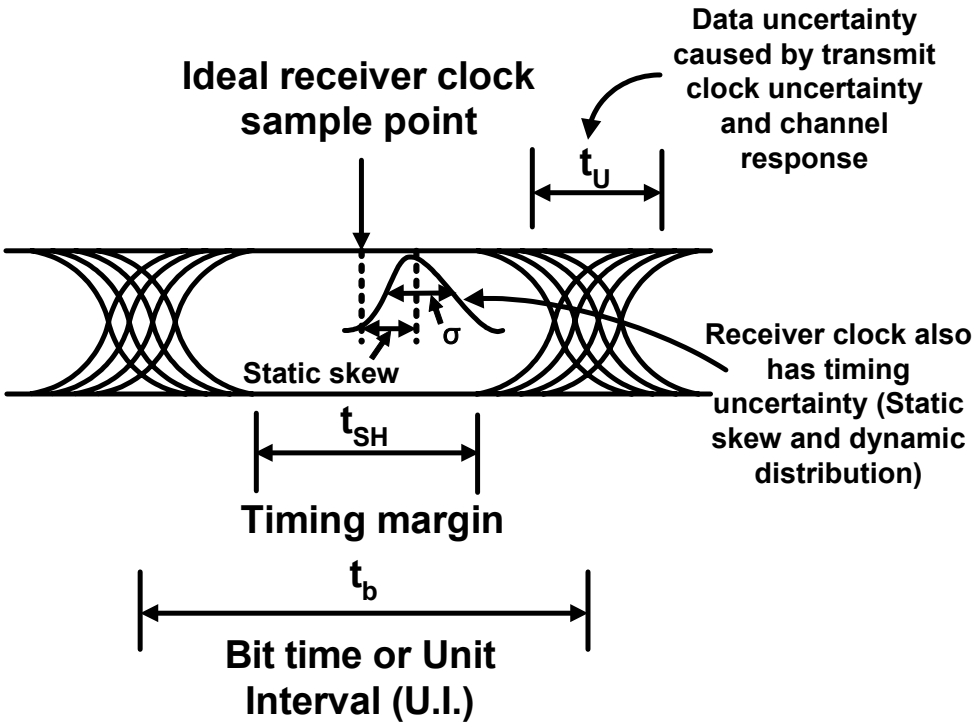


Figure 2.20: Receiver timing margin. (Recreated from [16])

Phase uncertainty of the sampling clock, both static (due to static phase mismatch) and dynamic, are the dominant sources of sampling clock uncertainty. While there may be ways to compensate for static phase errors using a variety of calibration techniques, dynamic or random jitter cannot be completely eliminated and must therefore be dealt with diligently. There are several sources of noise that can cause sampling clock

jitter in a receiver. Three of them arouse major concerns: First, on-chip power-supply and substrate noise; second, quantization noise in digitally-controlled CDRs; and finally, noise on the reference clock with respect to which links operate. The following subsection discusses the characteristics of these different noise sources in detail.

2.7.1 Power Supply Noise

Power-supply and substrate noise are important noise sources to watch out for in today's digital-circuitry dominated chips. Thorough characterization of on-chip power-supply noise usually requires exhaustive post-fabrication simulations and greatly depends on different aspects of the system, application, board designs, etc, which may be difficult to achieve. To mitigate the deleterious effects of noise created by digital circuitry, a separate analog power supply is commonly used for high-speed links, providing better isolation from the noisy digital power supply. However, due to the proximity of both analog and digital circuitry in a compact design, the power supply for the digital circuitry and the power supply for the analog circuitry may exhibit some coupling. Moreover, in standard, low-cost CMOS processes, all circuitry shares a common substrate, providing another path through which digital switching noise can corrupt the analog power supply [30]. On-chip power-supply noise measurement results for a high-speed serial link operating at multi-Gb/s, presented in [35], show that an ASIC (Application-Specific Integrated Circuits) core can generate noise that couples through the substrate and appear on the analog power supply lines. These results also show that the stationary (time-averaged) noise is essentially white except for several deterministic components (e.g., the reference clock that is fed to the PLLs in the links). However, the random noise is

actually cyclostationary in nature. That is to say, the noise exhibits periodicity (e.g., the chip has multiple clocks or exhibits somewhat repetitive modes of operation). Moreover, this periodic noise is not known *a priori* for most designs. In many link applications, where links must coexist with large digital cores (e.g., CPU-to-CPU interface, CPU-to-memory interface, etc.), the power-supply noise presents a big challenge for high-speed link designers. The resulting power-supply noise can lead to significant amounts of timing uncertainty on the sampling clocks and the magnitude of the noise is difficult to predict prior to chip fabrication.

2.7.2 Quantization Noise in a Digitally-Controlled CDR

Descriptions of several CDR topologies, provided in the previous section, have alluded to quantization noise in digitally-control CDR designs. Digital CDR controllers are commonly used, because they can facilitate designing loops with very low bandwidths. In many high-speed link designs (e.g., PCI Express [36] and Serial ATA [37]), a digital CDR control is often used to reduce the cost. Analog circuit equivalents would require extremely large loop filter components that consume large area and suffer process, voltage and temperature (PVT) variations. However, digital CDR control often limits, or quantizes, the resolution of sampling clock positions that can be created. Therefore, the sampling clock can never be perfectly aligned to the center of a data symbol. Negative feedback in the CDR can further exacerbate this quantization noise in the form of dither jitter. Therefore, this digital control-induced quantization noise cannot be ignored.

The detailed effects of this quantization noise depend on the specific CDR architecture chosen and relevant specifications. In later chapters, we will see how this quantization noise can be minimized for some CDR architectures.

2.7.3 Reference Clock Noise

Although not all CDRs need a reference clock, commonly-used dual-loop CDR designs require one. Also, it is a necessary component for clock generators such as a PLL and DLL. The reference clock may come from an off-chip crystal oscillator or an on-chip clock source. Noise on the reference clock can get transferred to the sampling clock or even amplified, causing a reduction in the receiver's timing budget [36]. Therefore, the effects of jitter on the reference clock also need to be minimized in a high-speed link system.

We have just seen three of sources of noise that can degrade the performance of a CDR. Although there are other sources of noise in high-speed link systems, these three require the most attention. The rest of this dissertation focuses on how to deal with these different noise sources to build a noise-robust and reliable CDR.

2.7.4 Clock and Data Recovery Performance Metrics

Because the main purpose of a CDR is to recover the transmitted data correctly, the most important performance metric is the bit error rate (BER). BER is defined as the ratio of the number of bits incorrectly received to the total number of bits received during

a specified time interval. In most systems, a BER below 10^{-12} is required. The best way to minimize BER is to minimize noise that can cause errors.

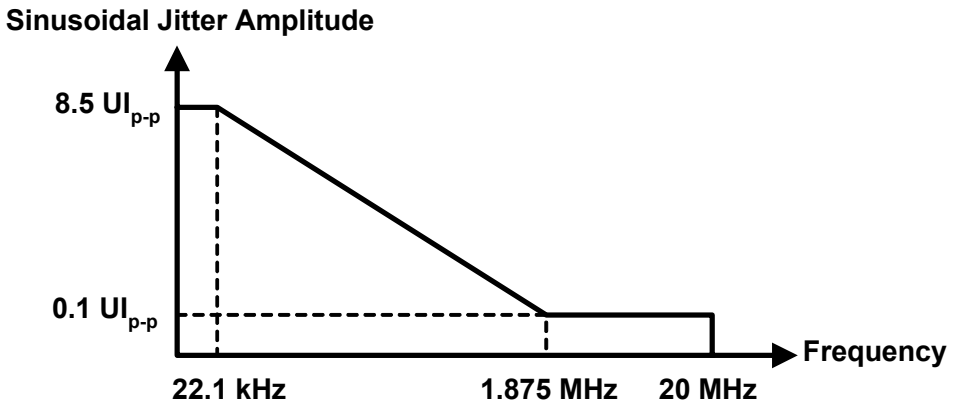


Figure 2.21: Jitter tolerance mask example.

Another very important metric is jitter tolerance, which determines the data "tracking" ability of the CDR. Jitter tolerance is defined as the maximum peak-to-peak amplitude in unit intervals (UI), or symbol times, of sinusoidal jitter applied on the input data that can lead to errors, whereby the CDR cannot meet a specific BER. This metric is best described using a plot of sinusoidal jitter versus frequency, as shown in Figure 2.21. A CDR ought to be able to tolerate or track different jitter amplitudes across a wide range of frequencies. At lower frequencies, the jitter amplitude can be several UIs, but reduces to small fractions for a UI at higher frequencies. Good tolerance can be achieved with CDRs that exhibit high tracking bandwidths and have low-jitter sampling clocks.

2.8 Summary

This chapter has presented an overview of the high-speed link systems and introduced the concept of utilizing time-division multiplexing (TDM) in the receiver to achieve data rates higher than the on-chip clock operation frequency. Based on this basic high-speed link topology, we reviewed the necessary components that are required to enable high performance – low-jitter clock generators and noise-robust CDRs. An in-depth view of PLL- and DLL-based clock generators has revealed that those two topologies offer differing characteristics with respect to different noise. These differences motivate us to ponder other clock generator topologies that may offer robust, low-jitter operation under a wide range of noise conditions. Subsequently, a review of three commonly-used CDR architectures reveal that the same tradeoffs exist with respect to different noise sources, as was seen for clock generators. Given the three dominant sources of noise identified in the latter part of this chapter, the rest of this dissertation investigates new clock generator and CDR designs. In particular, the goal is to find topologies that can readily adapt themselves to different noise conditions in order to minimize jitter and maximize performance. The next chapter begins this endeavor by describing the architecture, characteristics, and experimentally measured results of a mixed PLL/DLL clock generator.

Chapter 3

Adaptive-Bandwidth Mixed PLL/DLL-Based Multiphase Clock Generator

To provide reliable clocks for clock and data recovery (CDR) in order to support the higher bandwidth requirements of high-speed links, a low-jitter, low-cost clock generator is needed. There are two popular choices for multiphase sampling clock generation: the (1) phase-locked loop-based (PLL) clock generator, and (2) delay-locked loop-based (DLL) clock generator. If a PLL clock generator is used as the core clock generator as in [1] and [9], the output clock jitter is largely affected by the on-chip supply noise but relatively immune to the reference clock noise because the PLL has the advantage of filtering out reference clock noise due to its low bandwidth. However, the PLL suffers from large supply- or substrate-induced jitter accumulation due to the use of a voltage-controlled oscillator (VCO). What if we use a DLL clock generator instead [2]? The DLL clock generator has the benefit of suppressing on-chip jitter accumulation via the use of a voltage-controlled delay line (VCDL), but any noise on the reference clock is passed to the output clock without any attenuation.

However, the on-chip and off-chip noise conditions are usually hard to predict precisely before chip fabrication. It is challenging to choose one clock generator over the other in order to minimize output clock jitter. We have seen some adaptive-bandwidth PLL designs in Chapter 2 that attempted to overcome the problems of low bandwidth and

jitter accumulation in a PLL. But both designs present some problems, e.g., power and area penalty, limited tuning range, and static phase mismatch between the reference clock and internal clock's edge. We consider a better way to design an adaptive-bandwidth PLL with a wide bandwidth tuning range and minimum overhead in order to minimize the dynamic clock jitter under different noise conditions.

What if we combine these two types of clock generators to create a hybrid clock generator that can be configured as a PLL, a DLL, or a combination of the two? We can then tune the loop bandwidth over a wide range of frequencies between the PLL and DLL bandwidths to take advantage of this flexibility under different noise conditions.

In this chapter, an adaptive-bandwidth mixed PLL/DLL-based (MX-PDLL) multiphase clock generator is proposed to achieve flexibility to adapt the clock generator's loop bandwidth under different noise conditions. The proposed mixed PLL/DLL architecture can provide a solution to the previous quest for a PLL design that has a wide bandwidth tuning range with minimum overhead. A detailed description of this design is presented in the following sections.

The high-level architecture will be presented in detail in Section 3.1 followed by z-domain models of the clock generator and MATLAB® simulation results in Section 3.2. First, a simplified model of the MX-PDLL is presented. Following this, a complete model is presented with added complexity based on the actual circuit implementation. The detailed circuit implementation is presented in Section 3.3 and experimental measurement results from a test-chip prototype are presented in Section 3.4.

3.1 Architecture

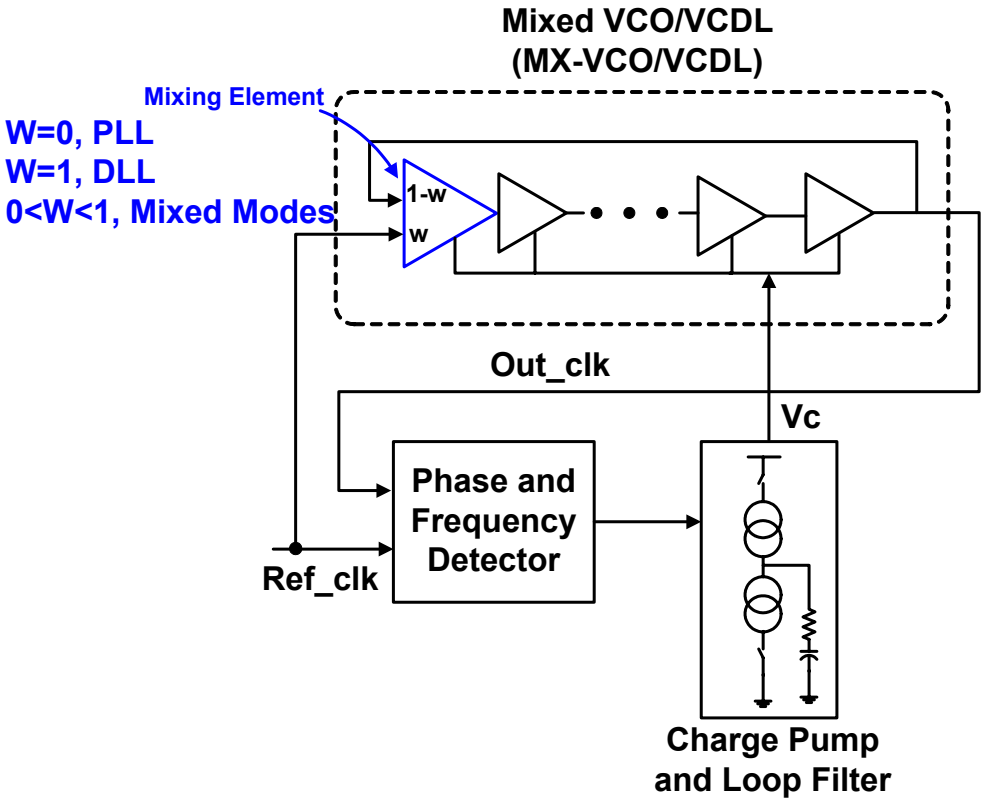


Figure 3.1: MX-PDLL loop topology.

First, let us look at the simplified top-level block diagram of the MX-PDLL in Figure 3.1. This circuit uses a phase mixing interpolator to configure the loop as a PLL, DLL, or a mixture of the two. If the on-chip supply is noisy, the high-bandwidth DLL-mode is preferred to avoid supply noise accumulation. On the other hand, if the reference clock is noisy, the low-bandwidth PLL-mode is preferred to take advantage of low-pass filtering characteristics of the loop. When both noise sources, whose relative amounts

might not be known *a priori*, are present, a mixed-mode may yield an optimum bandwidth setting to minimize output clock jitter [4].

The detailed block diagram of the MX-PDLL is shown in Figure 3.2. We can see that the MX-PDLL resembles a traditional PLL except for the mixed voltage-controlled

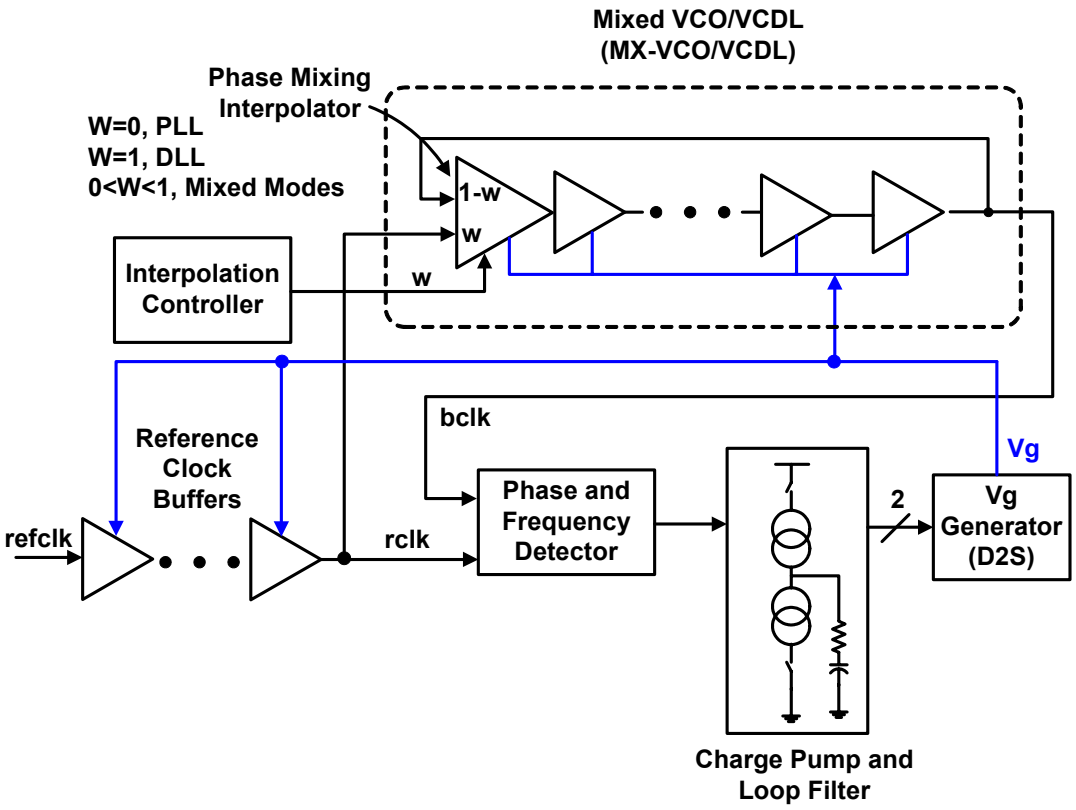


Figure 3.2: Block diagram of a mixed PLL/DLL-based multiphase clock generator.

oscillator/voltage-controlled delay-line (MX-VCO/VCDL), which can operate in several modes. It is implemented with a phase mixing interpolator as the first delay element that mixes signal energy between the output clock (*bclk*) and a buffered reference clock (*rclk*) based on a 3-bit digital code (*w*[2:0]). A mixing weight of 0 corresponds to the PLL-

mode without any reference clock injection (VCO-mode); a mixing weight of 1 corresponds to the DLL-mode with the full reference clock injection strength (VCDL-mode), and four intermediate mixing weights (0.2, 0.4, 0.6, and 0.8) correspond to mixed-mode operations with relative weighted strengths between *rclk* and *bclk* (MX-VCO/VCDL-mode). These different modes facilitate a wide range of the clock generator's bandwidth adjustment. We can see this wide range of bandwidth tuning capability when the z-domain model of the MX-PDLL is presented in Section 3.2. The measurement results of the input phase transfer functions across different interpolation weights are presented in Section 3.4.

The rest of the system consists of a phase and frequency detector (PFD), a differential charge pump (CP), a loop filter (LF), a Vg (MX-VCO/VCDL control voltage) generator, which serves as a differential-to-single-ended (D2S) voltage conversion block, the reference clock (*rclk*) conditioning buffers, and an interpolation controller. The MX-VCO/VCDL has eight stages and produces 16 evenly-spaced, single-ended clock phases. The PFD measures the phase difference between *rclk* and *bclk*, and then the loop integrates the difference through a differential CP and LF blocks. The integrated voltage output is then fed into the Vg generator to create the single-ended control voltage, Vg, which controls the output clock frequency of MX-VCO/VCDL. The interpolation controller generates control signals for the phase mixing interpolator based on different weight settings to configure the loop as a PLL, a DLL, or a combination of the two.

Like the z-domain models of the PLL and DLL that we saw in Chapter 2, a z-domain model can be used to represent the MX-PDLL shown in Figure 3.2, and derive the input phase transfer functions and noise transfer functions for all operation modes.

We first look at a simplified model that captures the essence of the MX-PDLL. Then additional modifications will be added into the model to represent the complete MX-PDLL that we built and tested.

3.2 Z-Domain Modeling

In order to analyze the frequency responses with respect to different noise sources across all operation modes in the MX-PDLL, we will derive the input phase transfer functions and supply noise transfer functions based on the z-domain models in this section. First a simplified model that uses a pure gain element to represent the Vg generator is presented in Section 3.2.1. Then we replace the Vg generator block with a more accurate model of the block. In Section 3.2.2, we further modify the system to include a feed-forward control path corresponding to the actual circuit implementation.

3.2.1 Z-Domain Model of the Simplified Mixed PLL/DLL

First let us look at the z-domain model shown in Figure 3.3. *Φref* represents the *refclk* phase, *Φin* represents the *rclk* phase, and *Φout* represents the *bclk* phase. The phase difference between *rclk* and *bclk* is measured by the PFD to create a phase error term, *ph_err*. This phase error is integrated by the CP and LF and converted to a control voltage *V_filter*. LF(z) is the z-domain representation of the CP and LF continuous-time transfer function. We saw the same representation in Chapter 2 when we were modeling the PLL in the z-domain. VG is a pure gain element that represents the gain from *V_filter*

to *V_g* that controls the delay of the delay-line. Kvcdl represents how much delay the delay line has per 1 volt change of *V_g* in radians per volt per cycle. After the voltage-controlled delay line (Kvcdl) block, the correction term, *ph_ctrl*, is generated to correct the phase error. Remember that in a PLL, the *bclk* circulates back to the input of the delay line. In a DLL, the input of the delay line is *rclk*. Therefore in the MX-PDLL, we first assign different weights to *bclk* and *rclk*, and add them up along with the *ph_ctrl* correction term at the delay line input. *I* is the injection strength of *rclk* whose phase is represented by *Φin*, and (1-*I*) is the injection strength of *bclk*. After one cycle delay, the next cycle *bclk* phase can be derived.

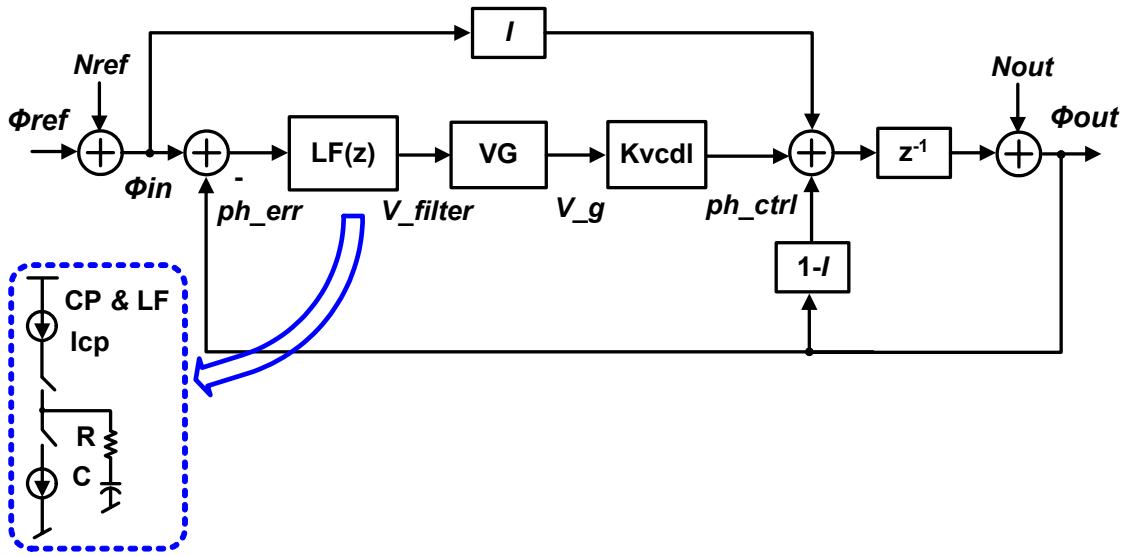


Figure 3.3: Z-domain representation of a simplified MX-PDLL.

The circuit has two main noise sources – *Nout* represents the phase modulation of the MX-VCO/VCDL output (e.g., supply noise), and *Nref* represents noise on *refclk* from off-chip sources. Notice that *Nout* will accumulate within the accumulator loop in PLL

mode. The injection point of *Nout* is different from the output noise injection point, as we saw in Figure 2.6, in order to capture the supply noise accumulation effect in the VCO.

Now let us derive the input phase transfer function (PTF) of the MX-PDLL from *Φin* to *Φout*, and the supply noise transfer function (NTF) from *Nout* to *Φout*. Notice that the PTF is also the transfer function from *Nref* to *Φout* and from *Φref* to *Φout*.

$$\frac{\Phi_{out}}{\Phi_{in}} = \frac{I + LF(z) * VG * K_{VCDL}}{z + LF(Z) * VG * K_{VCDL} - (1 - I)} \tag{3.1}$$

$$LF(z) = \frac{I_{CP} T_{ref} R}{2\pi} \frac{z - \beta}{z - 1}, \beta = \exp(-\frac{T_{ref}}{RC}) \tag{3.2}$$

$$\frac{\Phi_{out}}{N_{out}} = \frac{z}{z + LF(z) * VG * K_{VCDL} - (1 - I)} \tag{3.3}$$

The simulation results of the PTF and NTF across different mixing weights are shown in Figure 3.4 and Figure 3.5.

The input phase transfer function plot demonstrates that the loop bandwidth of the MX-PDLL can be tuned over a wide range of frequencies, from the lower PLL bandwidth all the way to half the reference clock frequency by changing the relative mixing weights between *rclk* and *bclk*. The bandwidths of the mixed-mode operations are linearly related to the normalized mixing weights. This suggests that if the bandwidth of the PLL is designed to be very low, the maximum tuning range is also increased. It also clearly shows that the PLL can filter high frequency reference noise. On the other hand, the DLL passes all the reference noise without attenuation. Therefore from the reference noise perspective, a PLL is better than a DLL. The jitter peaking in the PLL mode is less than 0.5dB with a large damping ratio. Some jitter peaking is also observed at high

frequencies in the DLL mode. This peaking can be reduced by linearly scaling the loop filter resistor with *bclk* injection strength, (1-*I*), such that the zero is removed in DLL mode.

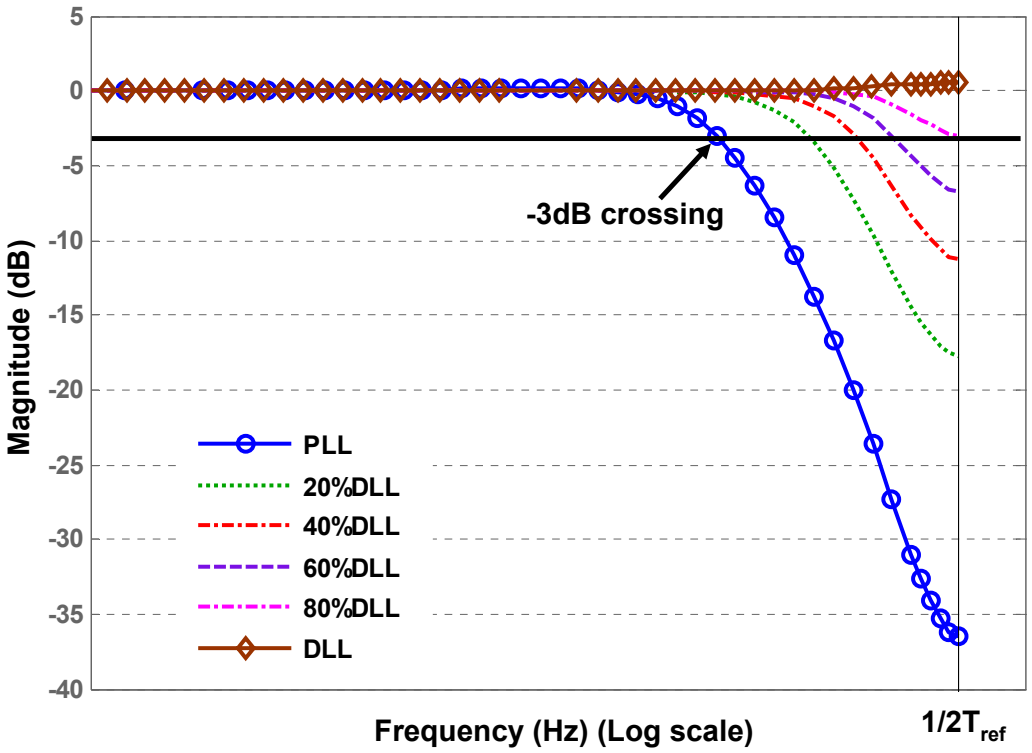


Figure 3.4: Simplified MX-PDLL model input phase transfer function (*Φout*/*Φin*).

Before we compare the supply noise responses of a PLL and a DLL, we must understand how the supply or substrate noise gets into a PLL or DLL. If the supply or substrate reference level changes, a change on the delay of the delay elements in the VCO in a PLL, or the VCDL in a DLL, occurs. A performance measurement called *supply sensitivity* is usually expressed in a normalized *percentage of delay change per percentage of supply change* (% delay / % volt). The delay modulation of the delay

elements in the VCO/VCDL[6] requires the use of delay element designs with good

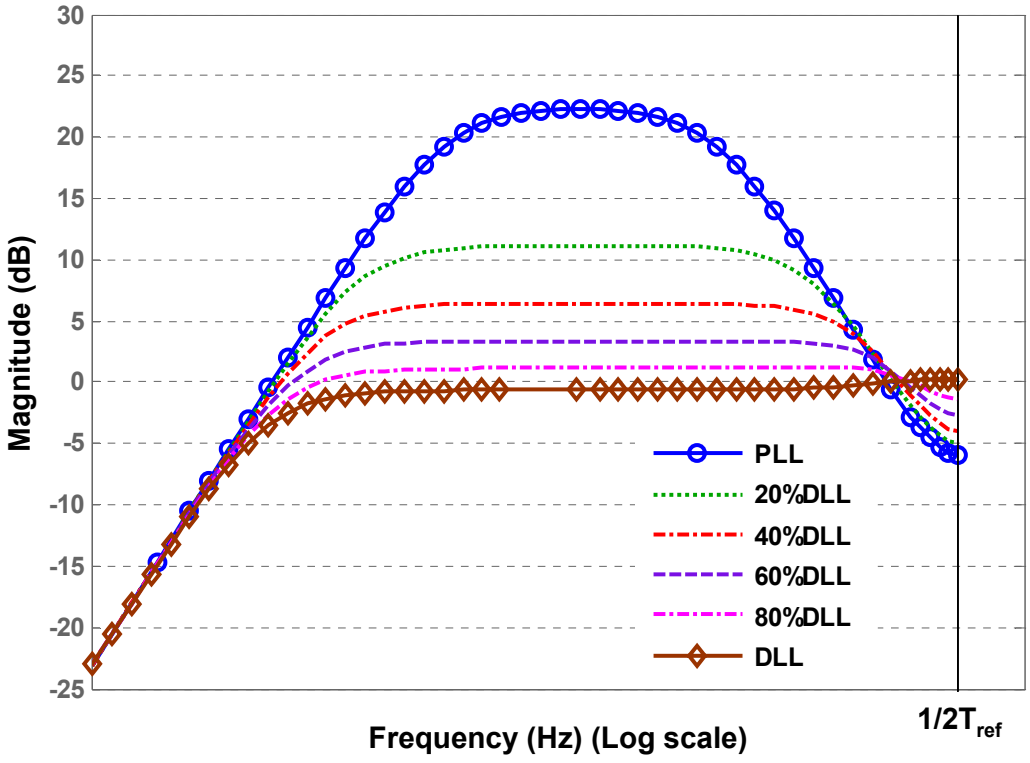


Figure 3.5: Simplified MX-PDLL model supply noise transfer function (*Φout*/*Nout*).

supply-noise rejection. A PLL usually has higher supply noise sensitivity than a DLL using the same delay element because a change in supply voltage results in a change in the VCO's output frequency, which integrates the phase error within the feedback loop until the loop's correcting action takes effect. Since a PLL typically has a low bandwidth in order to guarantee stability, the phase error accumulation can result in a large amount

[6] The supply-noise sensitivity of the inverter-based ring oscillator VCO is 0.25% -f_{VCO} / 1% - Vdd with no regulated supply in the proposed clock generator design.

of output jitter under a noisy supply. On the other hand, a change of a VCDL's supply voltage results in only a delay change occurring once through the delay line. There is no phase error accumulation within the loop due to a fresh reference clock edge that feeds into the VCDL every reference cycle. Therefore a PLL has more supply-induced jitter than a DLL [21].

The supply noise transfer function plot shows a very different result than the input phase transfer function plot. Figure 3.5 shows that the PLL accumulates a significant amount of jitter at high frequencies because in the PLL mode, *bclk* is always circulating back to the input of the delay line to form an oscillator, which then accumulates supply noise-induced jitter within the loop. The DLL does not amplify the supply noise-induced jitter at high frequencies, since *bclk* does not feed back to the input of the delay line but is simply a delayed version of *rclk*. All of the mixed-mode operations accumulate some jitter, the amount of which depends on the relative weight of *bclk*.

The supply and substrate noise are important noise sources in today's digital circuitry-dominated chips, where the digital power and analog power supplies might couple noise. The measurement results and analysis of the supply noise-induced jitter with several PLL power configurations shown in [30] have shown that higher PLL bandwidth is desirable to reduce the jitter caused by the digital switching noise. This is because the supply noise transfer function in the PLL mode has a jitter-accumulation characteristic at high frequencies [26][27]. Since the PLL can suppress the supply noise that has a frequency within its loop bandwidth, a higher PLL bandwidth is preferred to suppress more supply-induced jitter. These are conflicting bandwidth requirements in terms of supply noise suppression and reference clock noise filtering. Based on these

observations, choosing the best operation mode depends on the relative amount of supply noise and reference clock noise.

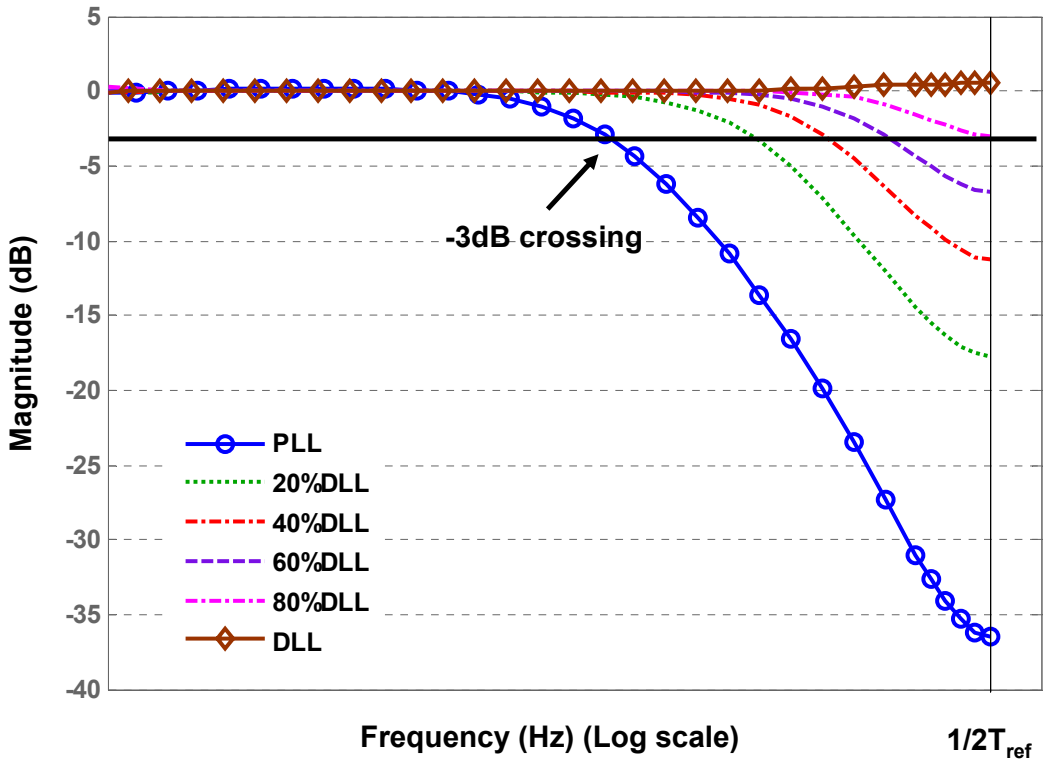


Figure 3.6: MX-PDLL input phase transfer function with "lag-lead" VG(z) frequency response.

What happens to the PTF and NTF if VG(z) has a "lag-lead" frequency response, i.e. a pole, ω_p, followed by a zero, ω_z, at frequencies much lower than the closed-loop bandwidth? We will see why the VG(z) has this type of frequency response in this particular MX-PDLL design as we talk about the Vg generator design in more detail later. For now, let us consider what happens when we replace VG in Equations 3.1 and 3.3 with the expression as shown in Equation 3.4 and see how the input phase transfer function and supply noise transfer function differ from Figure 3.4 and Figure 3.5.

$$VG(z) = VG * T_{ref} * \frac{z - \exp(-\omega_z T_{ref})}{z - \exp(-\omega_p T_{ref})} \tag{3.4}$$

The new input phase transfer function and supply noise transfer function are plotted in Figure 3.6 and Figure 3.7, respectively. There is not much noticeable change on input phase transfer functions, but there is slight jitter peaking on supply noise transfer functions in DLL mode and mixed modes at lower frequencies due to the additional pole-zero pair in VG(z).

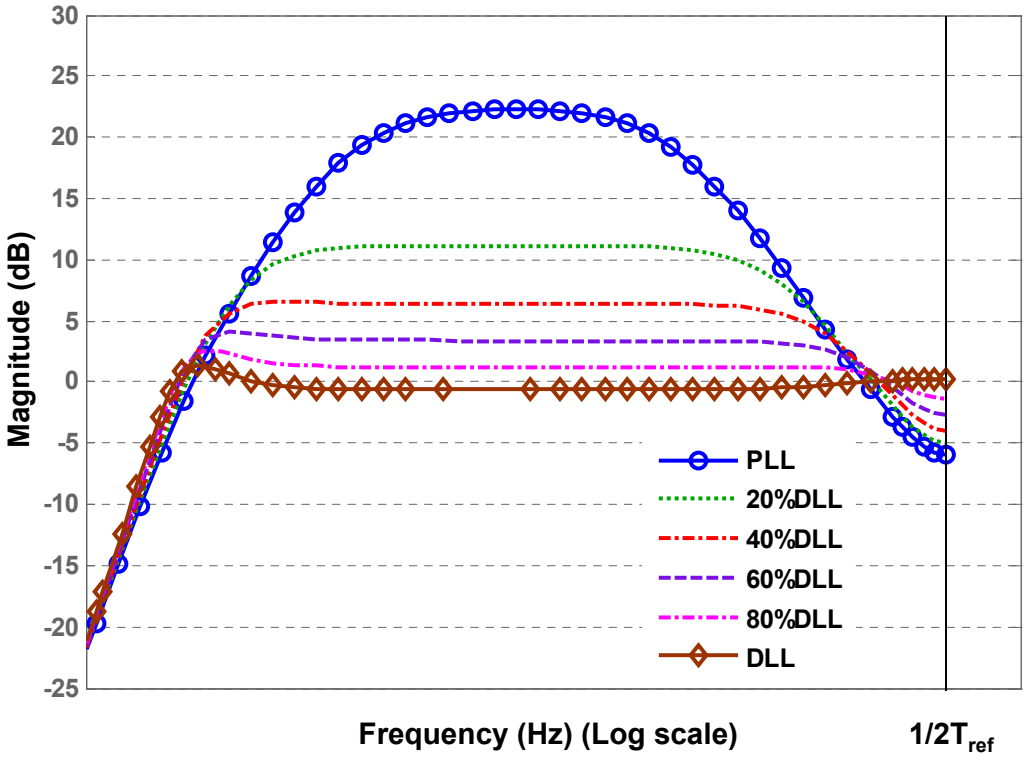


Figure 3.7: MX-PDLL supply noise transfer function with "lag-lead" VG(z) frequency response.

3.2.2 Z-Domain Model of the Complete Mixed PLL/DLL

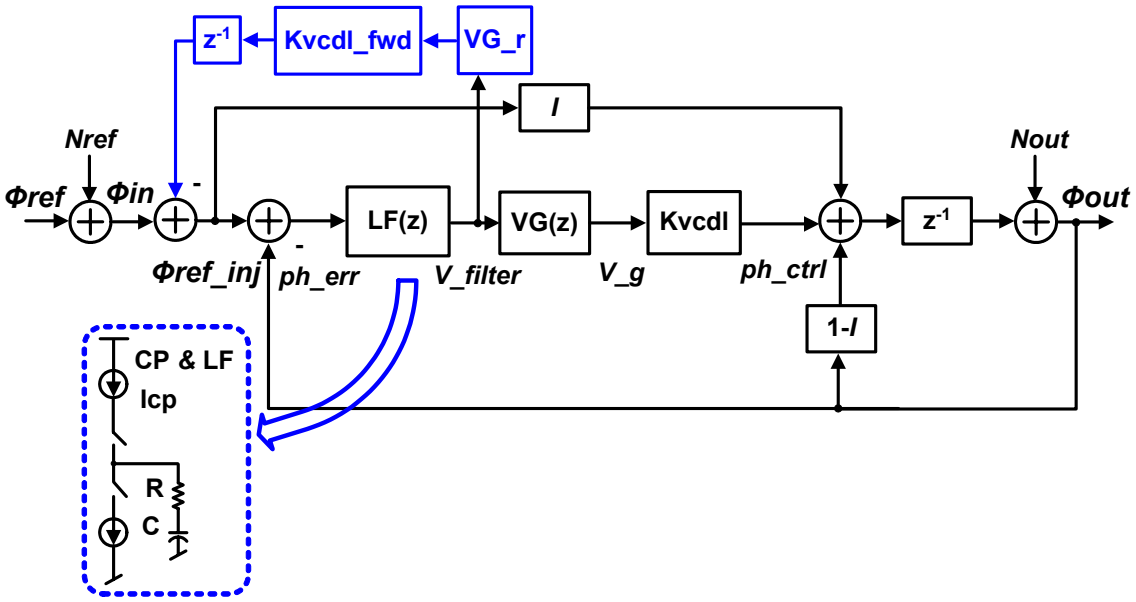


Figure 3.8: Z-domain representation of the complete MX-PDLL with feed-forward delay modulation path.

After analyzing the simplified model of the MX-PDLL, we now analyze a complete model of the MX-PDLL built by considering additional circuit implementation details. First the z-domain model of the complete MX-PDLL is presented in Figure 3.8. Notice that there is a feed-forward path from LF output (*V_filter*) to the input of the MX-VCO/VCDL modulating *Φref_inj*. This is due to the fact that the *V_filter* also modulates the delay of the *rclk* buffers, which have a gain of Kvcdl_fwd in radians per volt per cycle. VG_r represents the transfer function gain from *V_filter* to the control voltage that controls the *rclk* buffer delay. The buffers are inserted to make sure that the *rclk* and *bclk* signal conditions closely match each other. This additional path creates a pole-zero pair

that results in high-frequency peaking in the phase transfer functions of all operation modes, which we will see later in this section. Another difference between the complete model and the simplified model is that we now also consider the frequency response of the Vg generator. VG(z) represents the Vg generator's transfer function with a "lag-lead" frequency response as was shown in Equation 3.4. The derivation of the "lag-lead" frequency response will be discussed in more detail when we look at the detailed circuit implementation of Vg generator in Section 3.3.5. It is actually a higher-order system with one low-frequency pole, one low-frequency zero, and several higher-order poles.

The input phase-transfer function from *Φin* to *Φout* and the supply noise transfer function from *Nout* to *Φout* are given by Equations 3.5-3.8. LF(z) and VG(z) have the same expressions as those shown in Equation 3.2 and Equation 3.4.

$$\frac{\Phi_{out}}{\Phi_{in}} = \frac{z[A(z)+I]}{z[A(z)+z-(1-I)]+(z-1)B(z)} \tag{3.5}$$

$$A(z) = K_{VCDL} * LF(z) * VG(z) \tag{3.6}$$

$$B(z) = K_{VCDL\_fwd} * LF(z) * VG\_r \tag{3.7}$$

$$\frac{\Phi_{out}}{N_{out}} = \frac{z[z+B(z)]}{z[A(z)+z-(1-I)]+(z-1)B(z)} \tag{3.8}$$

Figures 3.9 and 3.10 plot the input phase transfer function (*Φout*/*Φin*) and the supply noise transfer function (*Φout*/*Nout*) respectively based on the complete z-domain model. The reference clock frequency, $1/T_{ref}$, is 750 MHz in Figures 3.9 and 3.10. When Kvcdl_fwd is not zero, the feed-forward path from *V_filter* to *Φref_inj* results in a peaking around 200 MHz in both transfer functions. Moreover, DLL-mode operation sees

the largest impact from this extra "pole-zero pair", because all the output energy is coming from *rclk*. This peaking can be mitigated by reducing Kvcdl_fwd or adding a low-pass filter in the feed-forward path. Also the higher-order poles in the Vg generator are also included in the simulation, which shows a slight shifting of the bandwidths of the input phase transfer functions compared to the simplified model. The supply noise transfer function also exhibits low-frequency peaking due to the additional pole-zero pair in VG(z). Moreover, the peaking is even worse than before due to the inclusion of other high-order poles in VG(z).

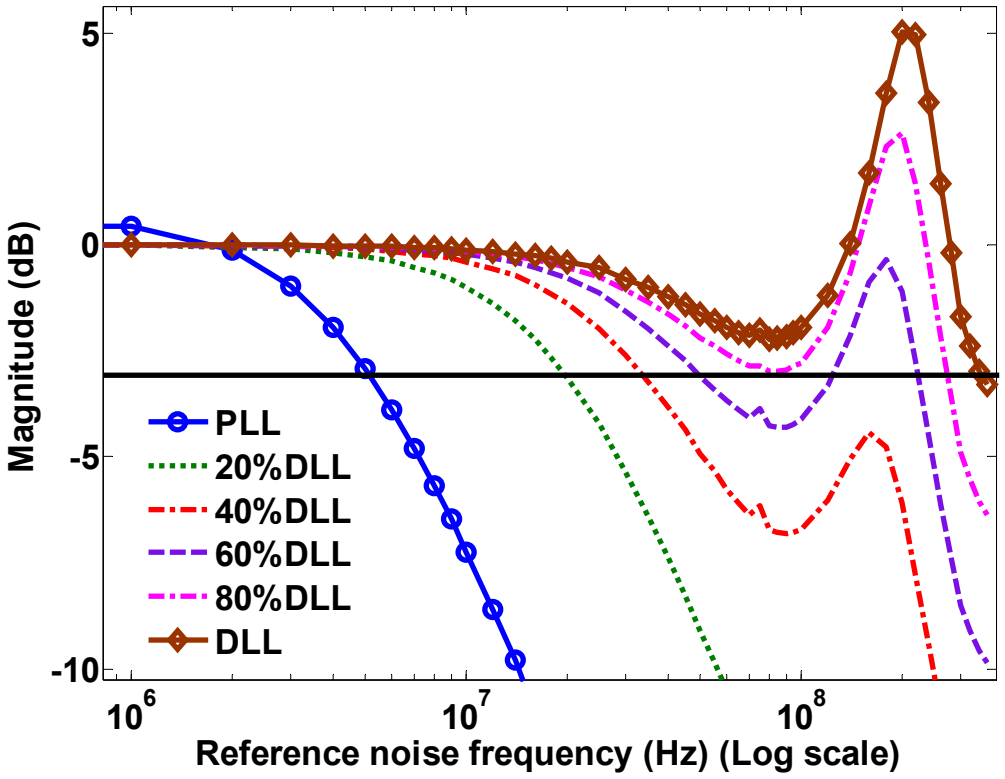


Figure 3.9: MX-PDLL input phase transfer function (*Φout*/*Φin* vs. frequency) w/ Kvcdl_fwd feed-forward path.

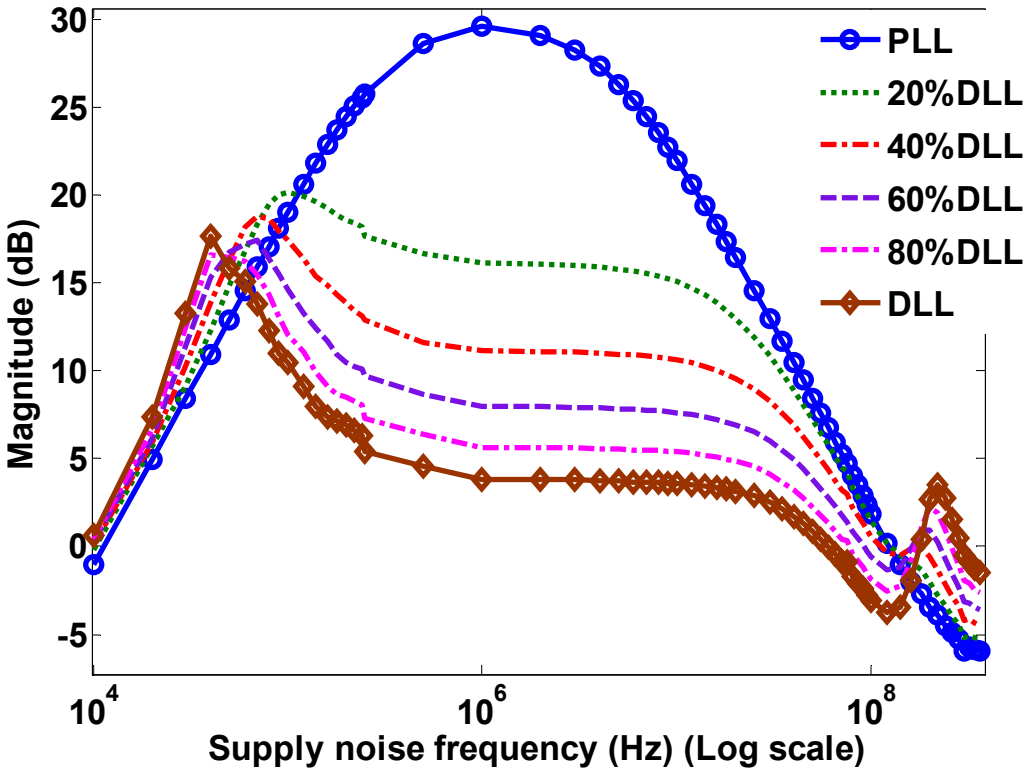


Figure 3.10: MX-PDLL supply noise transfer function (*Φout*/*Nout* vs. frequency) w/ Kvcdl_fwd feed-forward path.

All the input phase transfer functions and supply noise transfer functions we have derived so far demonstrate that there is a tradeoff between input noise filtering and supply noise suppression in terms of choosing the PLL or DLL. We can take advantage of the wide range of bandwidth-adjusting capability of the MX-PDLL to minimize clock jitter under various noise conditions. Therefore the adaptive-bandwidth MX-PDLL-based multiphase clock generator can achieve optimum jitter performance across different noise conditions for different high-speed link systems.

3.3 Circuit Implementation

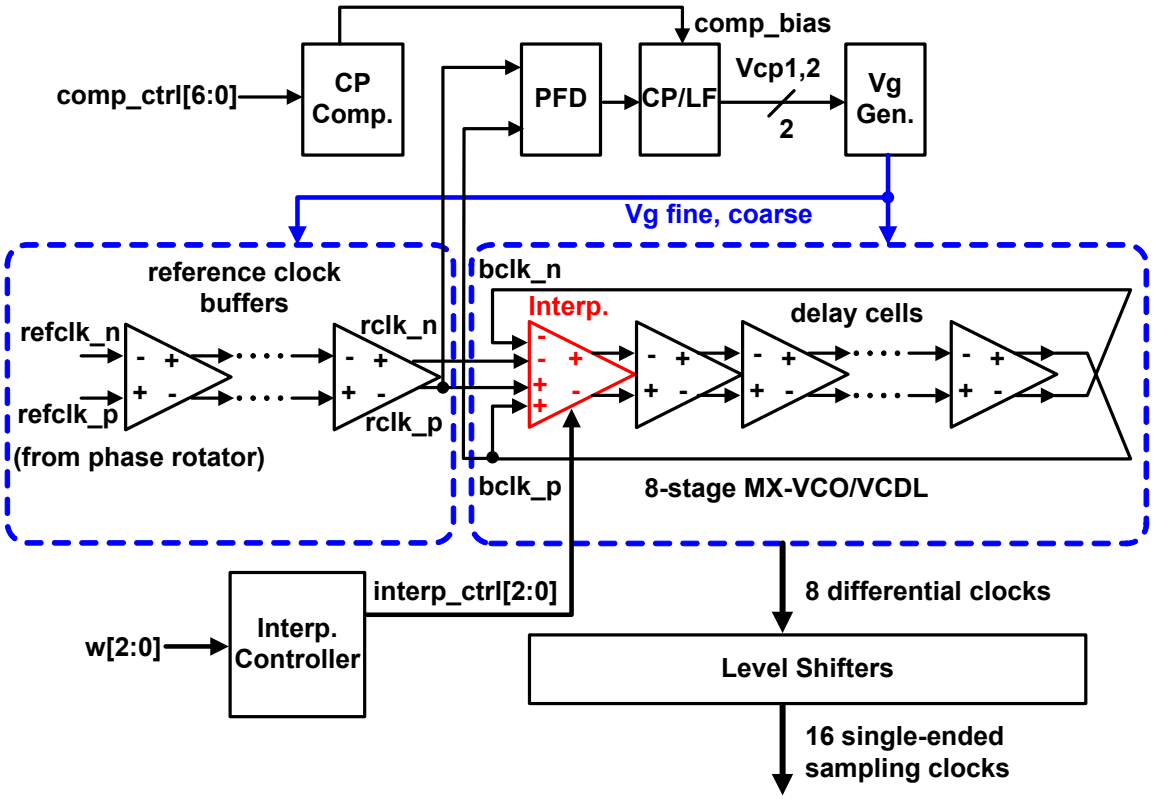


Figure 3.11: Detailed block diagram of a mixed PLL/DLL-based multiphase clock generator.

The design of the individual building blocks in the MX-PDLL as shown in Figure 3.11 is discussed in detail in this section. First, the differential reference clock (*refclk*) goes through a buffer chain to condition the signal, and then feeds into the MX-PDLL. The PFD measures the phase difference between *rclk* and *bclk*, and generates up and down pulses to control the differential CP. The LF integrates the differential current difference to create a differential control voltage (*Vcp1*/*Vcp2*). *Vcp1*/*Vcp2* then feeds into

a differential-to-single-ended block Vg generator to generate the final control voltages (*Vg_fine* and *Vg_coarse*) for the MX-VCO/VCDL. The mixing weight of the phase mixing interpolator in the MX-VCO/VCDL is controlled by the interpolation controller. In this section, detailed descriptions of the building blocks shown in Figure 3.11 are presented, except the CP compensator (CP Comp.), which is described in Chapter 5.

The phase interpolator-based MX-VCO/VCDL is presented in Section 3.3.1 followed by the interpolation controller design in Section 3.3.2. Next the phase frequency detector is presented in Section 3.3.3, and the differential charge pump with a loop filter will be presented in Section 3.3.4. Lastly, the Vg (MX-VCO/VCDL control voltage) generator is described in Section 3.3.5.

3.3.1 Interpolator-Based Mixed Voltage-Controlled Oscillator/Voltage-Controlled Delay Line

The MX-VCO/VCDL consists of eight delay cells with pseudo-differential outputs, and uses a phase mixing interpolator to mix the signal energy from *bclk* to *rclk*. Figure 3.12 presents the schematic of the interpolator. The subsequent seven delay cells use the same interpolator circuitry with both *in1* and *in2* tied together to guarantee that equally-spaced clock phase can be tapped out of the 8-stage MX-VCO/VCDL. The relative strengths by which *in1* and *in2* affect the interpolator output are controlled by three weighted differential pairs with weights of 1, 2, and 2. A total of six mixing weights can be configured to make the eight delay cells operate as a VCO, a VCDL, or a combination of the two.

The PMOS device gate control voltages, *Vg_coarse* and *Vg_fine*, are set by the Vg generator block to control the propagation delay of the cells. By separating the control

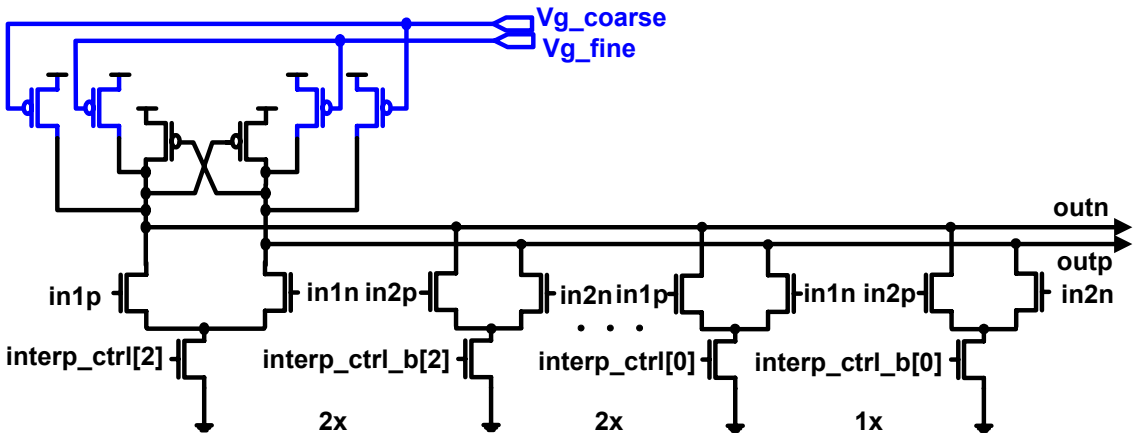


Figure 3.12: Schematic of the phase interpolator in the MX-VCO/VCDL.

into a low-frequency, high-gain coarse path and a high-frequency, low-gain fine path, the gain of the VCO (fine control path), when it is near the locked condition, can be greatly reduced [5]. Therefore a smaller sized capacitor in the LF can be used to conserve on-chip area. The simulated VCO transfer function is presented in Figure 3.13. The scaling factor of the coarse device to the fine device is set at 5 in the design. The VCO gain is 800 MHz per volt for *Vg_coarse* and 160 MHz per volt for *Vg_fine*.

The NMOS input differential pair is chosen to increase driving capability and to minimize load capacitance at the input nodes. The cross-coupled PMOS pair is added to ensure that the outputs swing differentially. The voltage swing of the output is not fully scaled due to the bleeding PMOS devices, and will be restored to the full swing and converted to single-ended clocks after level-shifting.

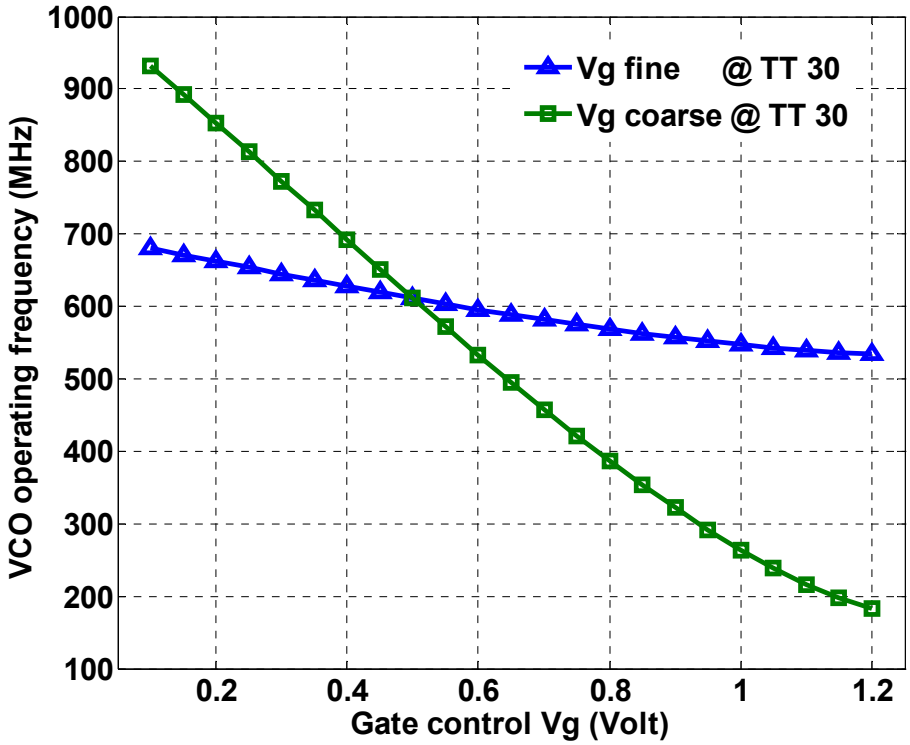


Figure 3.13: Simulated VCO gain transfer functions of *Vg_coarse* and *Vg_fine*.

3.3.2 Interpolation Controller

The timing window of the *interp_ctrl* signals must be valid before the rising edge of *rclk* arrives and closes before the arrival of its falling edge while the loop is almost in lock as shown in Figure 3.14. The carefully controlled aperture can guarantee the completion of interpolation without glitches on the output [13]. To ensure that the *rclk*/*bclk*'s rising edge is centered within the *interp_ctrl* aperture, and to maximize the timing window, the delay of all the buffers, NAND gates, and NOR gates are controlled by *Vg_replica* (a replica of *Vg_coarse*), which tracks the *rclk/bclk* frequency, and clock

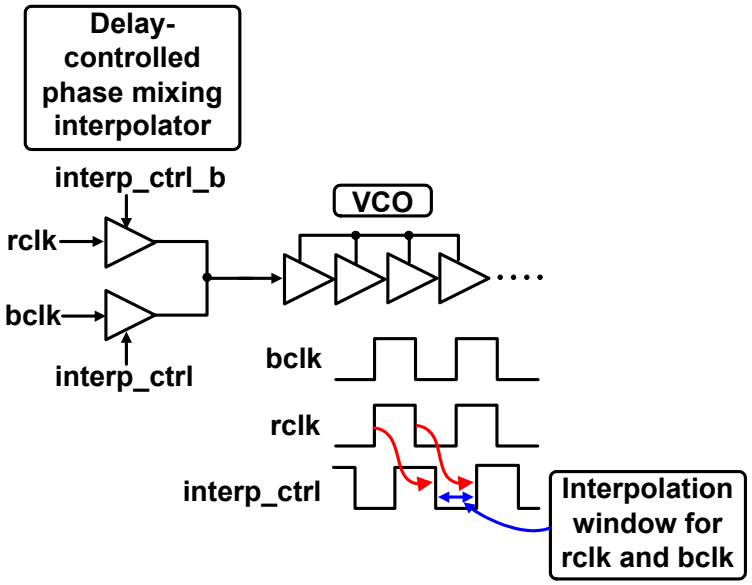


Figure 3.14: Interpolation control window timing diagram.

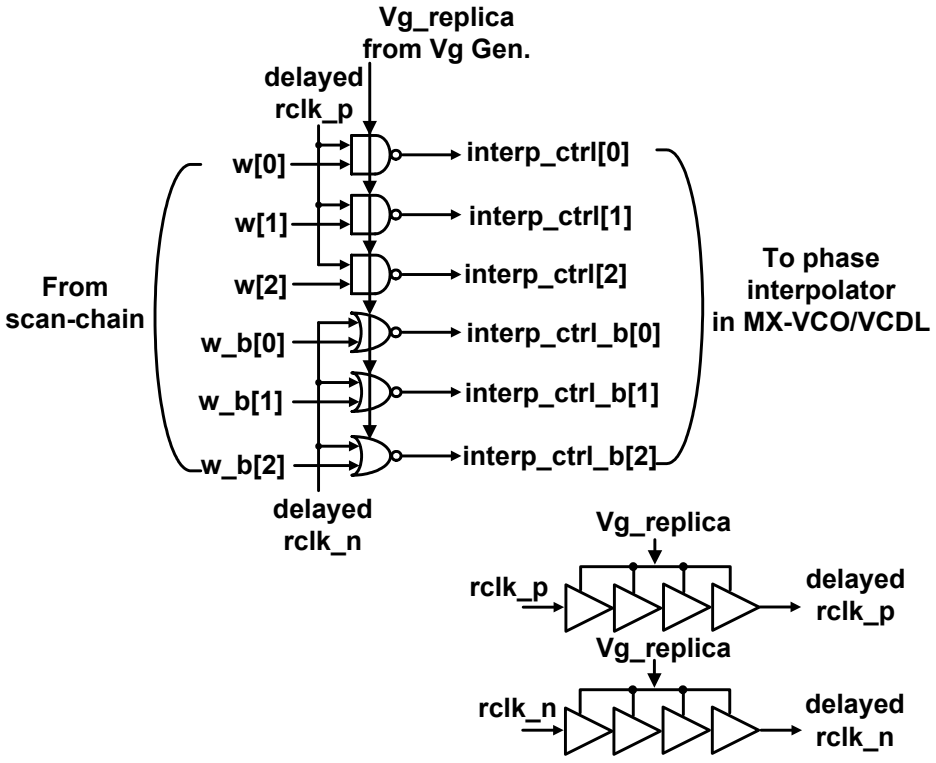


Figure 3.15: Schematic of the interpolation controller.

gated by *rclk*. Therefore the *interp_ctrl* aperture tracks the *rclk*'s rising edge across all process corners as shown in Figure 3.15. By separating *Vg_replica* and *Vg_coarse*, the noise generated by the digital circuitry does not directly affect the analog blocks in the core clock generator.

3.3.3 Phase and Frequency Detector

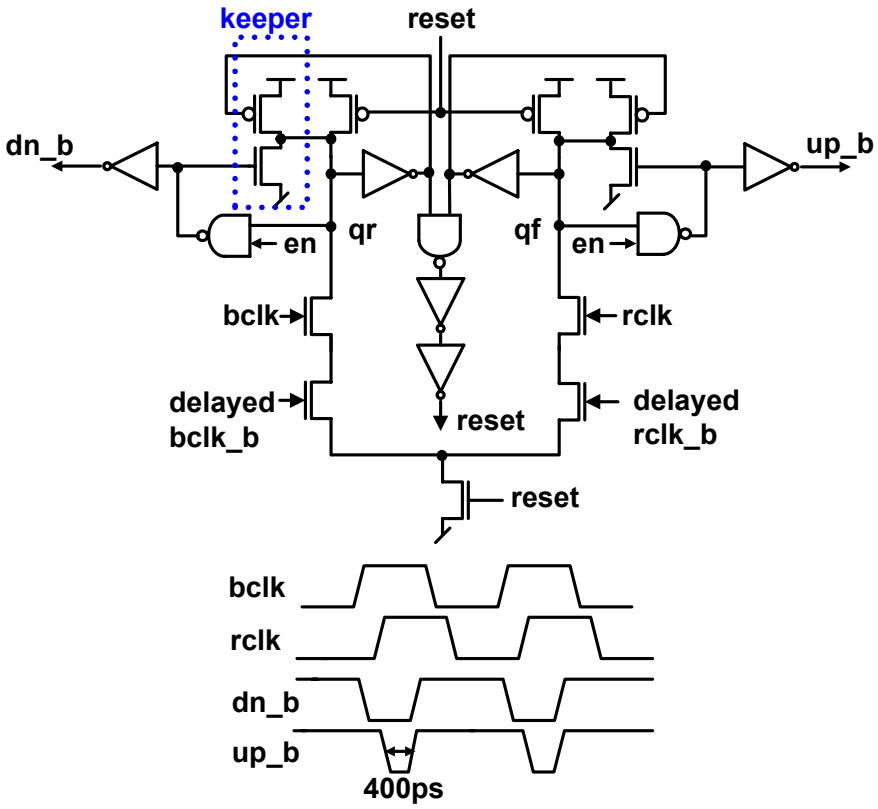


Figure 3.16: Schematic of phase and frequency detector.

The linear phase frequency detector (PFD) used in the MX-PDLL is presented in Figure 3.16. Two short low-true pulses are triggered by the rising edges of the input

clocks, *rclk* and *bclk*. When either *bclk* or *rclk* rises, *qr* or *qf* drops until the rising edge of the other clock arrives to reset both *qr* and *qf* to *Vdd*. The weak keeper will keep *qr* and *qf* at *Vdd* when *reset* is set at high. To minimize the voltage ripple at the output of the CP during the differential-mode operation, the pulse width should be minimized. The minimum pulse width of about 400ps is designed to be wide enough to fully turn on the current source devices in the CP.

Mismatch between the *up* and *dn* paths will create pulse-width mismatches and lead to static phase offset when the loop is locked. A careful layout can minimize this mismatch, but process variations can still create residual mismatch. A charge pump compensation scheme that compensates for the residual mismatch will be presented in Chapter 5.

3.3.4 Differential Charge Pump and Loop Filter

Figure 3.17 presents the schematic of the differential charge pump. It consists of two PMOS current sources that are controlled by *upb*/*dnb* pulses out of the PFD and two NMOS current sinks biased by a common-mode feedback (CMFB) circuit. The bottom NMOS switch activates the differential operation of the CP when either input pulse is low. Otherwise, the pull-down current paths are disabled to minimize common-mode ripple on the outputs. Two auxiliary NMOS current sink devices, controlled by a CP compensator block, are included to compensate for mismatch, which can cause static phase offset, in the CP and PFD.

The loop filter is implemented with transmission gate resistors and accumulation-mode PMOS gate capacitors. The secondary capacitors are achieved using the parasitic capacitors at the internal nodes (*Vcp1i* and *Vcp2i*).

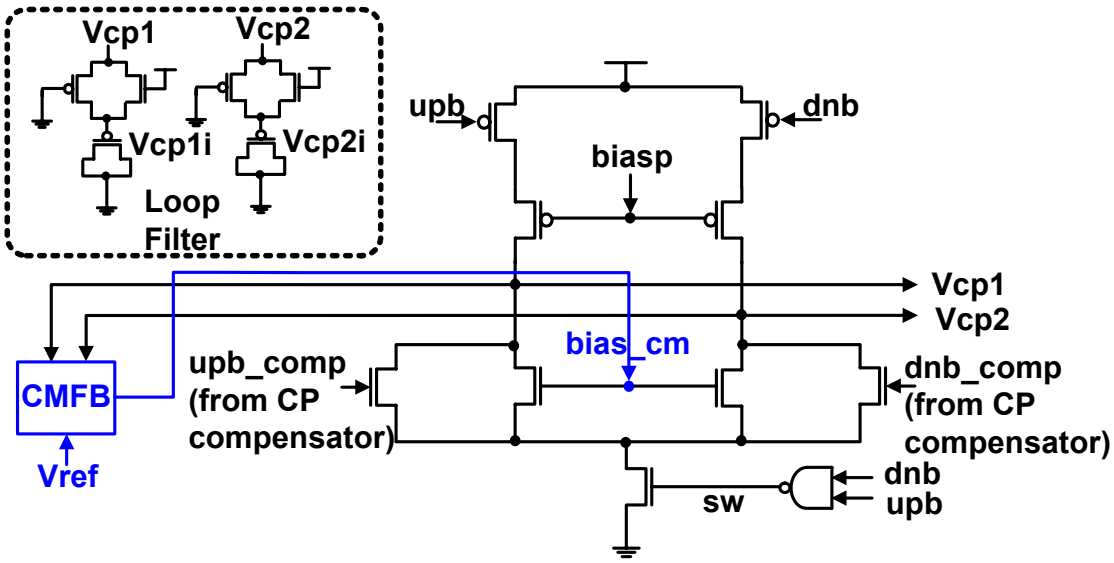


Figure 3.17: Schematic of the pseudo-differential CP and LF.

3.3.5 Vg Generator

The Vg generator takes in the differential charge pump outputs, *Vcp1* and *Vcp2*, and generates fine and coarse control voltages for the MX-VCO/VCDL as shown in Figure 3.18. The *Vg_fine* and *Vg_coarse* outputs provide high- and low-frequency control paths, respectively. A pair of parallel amplifiers with shorted outputs generates *Vg_fine*, where the unity-gain amplifier nominally sets *Vg_fine* with respect to *Vg_coarse* as expressed in Equation 3.9. Also the differential voltage from the charge pump skews

Vg_fine around *Vg_coarse* at high frequencies as shown in Equation 3.9. A_0_unity is the open-loop gain of the unity-gain amplifier and A_0_skew is the open-loop gain of the skew amplifier.

$$Vg\_fine = A_0\_skew\Delta Vcp + \frac{A_0\_unity}{1 + A_0\_unity} Vg\_coarse \tag{3.9}$$

$$\Delta Vcp \equiv Vcp2 - Vcp1 \tag{3.10}$$

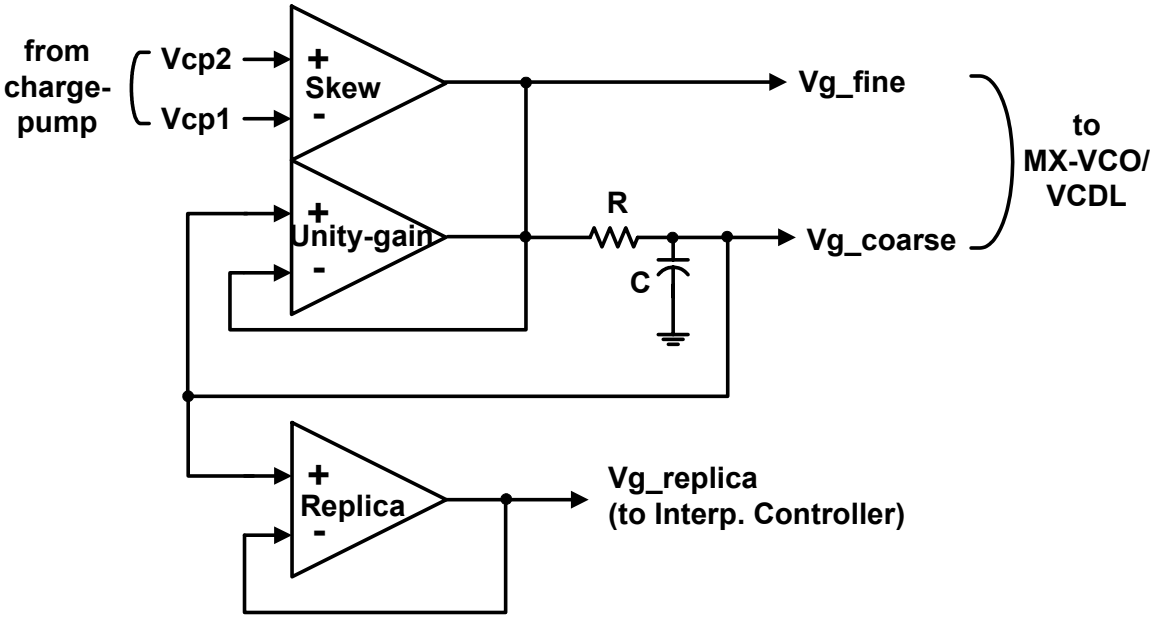


Figure 3.18: Vg generator block diagram.

$$Vg\_coarse = Vg\_fine * \frac{1}{RCs + 1} \tag{3.11}$$

$$\frac{Vg\_fine}{\Delta Vcp} = \frac{A_0\_skew(A_0\_unity + 1)(RCs + 1)}{RC(A_0\_unity + 1)s + 1} \tag{3.12}$$

$$\frac{Vg\_coarse}{\Delta Vcp} = \frac{A_0\_skew(A_0\_unity+1)}{RC(A_0\_unity+1)s+1} \tag{3.13}$$

Given that *Vg_coarse* is an RC low-pass filtered version of *Vg_fine* as shown in Equation 3.11, the two control voltages converge to approximately the same level when the loop is locked. This RC filter introduces a pole-zero pair ("lag-lead") into the overall loop response and must, therefore, be set low enough to ensure loop stability. Equation 3.12 presents the transfer function from (*Vcp2-Vcp1*) to *Vg_fine*. The transfer function has a pole at roughly $1/(A_0\_unity*RC)$ and a zero at $1/RC$. The transfer function from (*Vcp2-Vcp1*) to *Vg_coarse* also has a pole at $1/(A_0\_unity*RC)$ as expressed in Equation

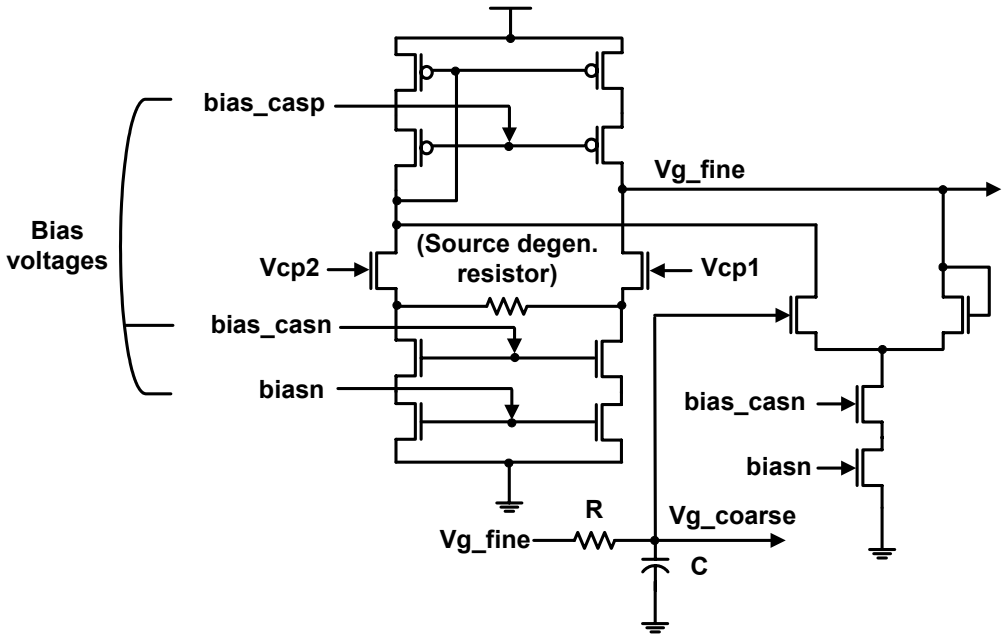


Figure 3.19: Schematic of the skew amplifier and unity-gain amplifier in the Vg generator.

3.13. Since the pole occurs at a very low frequency, the gain of *Vg_coarse* control path is very small at the higher frequencies of the interest. Therefore when the loop is closed and near lock, we only need to consider the transfer function of *Vg_fine*. By separating the fine and coarse control voltages, the fine control gain of the VCO is reduced to only about 160 MHz per volt.

A replica of *Vg_coarse* (*Vg_replica*) is also generated to provide the delay control for the interpolation controller such that the control signal aperture window tracks the clock operating frequency. This is done to guarantee that the rising edges of *rclk* and *bclk* are always in the middle of the control window.

Figure 3.19 presents the amplifier schematic that is used to implement the Vg generator. A source-degenerated differential pair is used for the skew amplifier to reduce and linearize the gain. A differential pair connected in a unity-gain configuration shares the pMOS loads with the skew amplifier. *Bias_casp*, *bias_casn*, and *biasn* are bias voltages generated by a simple bias block, which is not shown.

3.4 Measurement Results

The measurement results that verify the wide bandwidth tuning capability of the MX-PDLL are presented in this section. The results are presented as follows: first the test setup is explained in Section 3.4.1 and in Section 3.4.2 the on-chip power supply noise generator that is used to inject different amount of supply noise into the MX-PDLL is described. Then the measured input phase transfer function and supply noise-induced jitter vs. noise frequency are presented in Sections 3.4.3 and 3.4.4, respectively. The jitter

histogram plots followed by the measured *bclk* root-mean-square (rms) jitter vs. normalized mixing weight (bandwidth setting) with various amount of reference clock noise and supply noise complete the measurement results of the MX-PDLL clock generator in Section 3.4.5.

3.4.1 Test setup

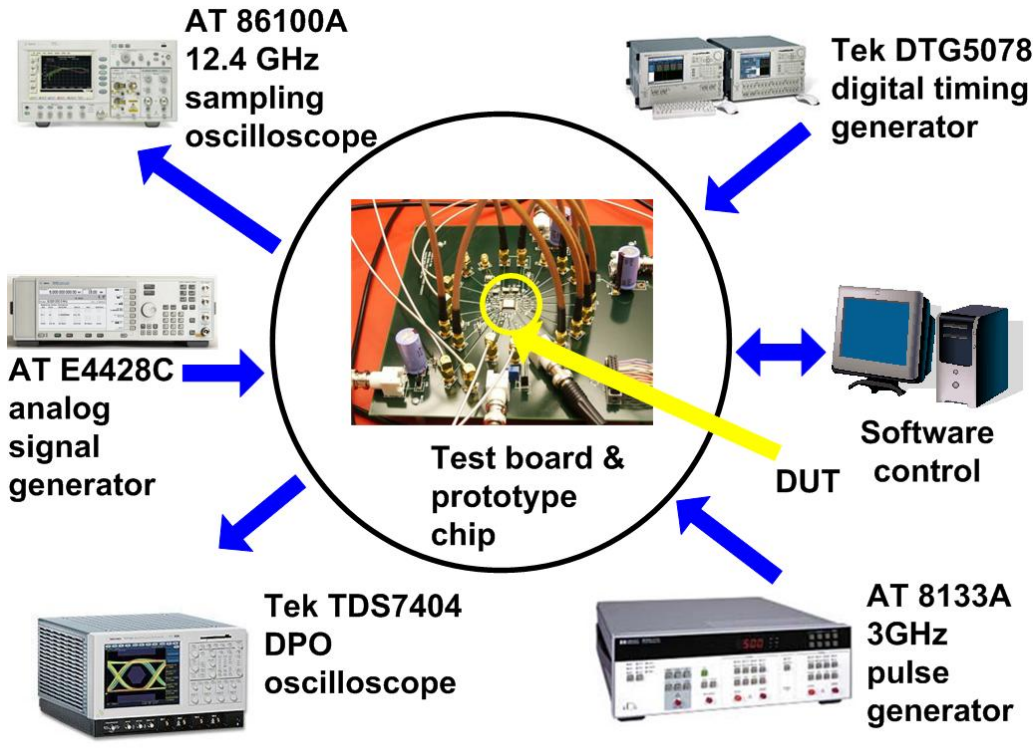


Figure 3.20: Measurement setup.

The test setup for verifying the design objectives of the adaptive-bandwidth MX-PDLL multiphase clock generator is shown in Figure 3.20. The Tektronix® (Tek) real-

time Digital Phosphor Oscilloscope (DPO) equipped with TDSJIT3 v2.0 advanced jitter and timing analysis software is the main equipment for jitter measurement. All of the measured jitter is defined as the phase jitter [36]. The real-time oscilloscope captures at least 15,000 clock edges and extracts an ideal clock signal based on the captured samples. The phase jitter is then defined as the timing difference between the captured clock edge and the extracted ideal clock edge. The self-triggered mode is used to capture the clock edges.

The Agilent Technologies® (AT) 12.4GHz sampling oscilloscope is also used to measure the jitter on the output clock. The Agilent Technologies® 3GHz pulse generator and analog signal generator are used as the reference clock and noise clock sources. The Tektronix® digital timing generator is used to generate Pseudo Random Bit Sequence (PRBS) and digital control signals. The chip is packaged in a 44-pin open cavity LQFP (Low-profile Quad Flat Pack) package. The I/Os of the test chip use ground-shielded and impedance-matched traces on the Printed Circuit Board (PCB) and SMA (Sub-Miniature version A) connectors to connect to the test equipment.

3.4.2 Power Supply Noise Generator

In order to inject noise into the on-chip power supply for the core MX-PDLL, an on-chip power supply noise generator is used as shown in Figure 3.21. The noise input controls a NMOS switch, which shorts *Vdd* and *GND* to create a dip on the core power supply [3]. Four binary weighted NMOS switches are used in parallel to change the amount of injected noise.

There are three power domains on this test chip — *Vdd_core*, *Vdd_ref*, and *Vdd_digital*. *Vdd_core* is the power supply voltage for the core MX-PDLL. *Vdd_ref* supplies reference clock buffers and the interpolation controller. *Vdd_digital* supplies the rest of the digital blocks including configuration registers. Each power domain has dedicated pins on the test chip.

Before presenting the measured phase transfer functions and jitter results across different interpolation weights, a performance summary of this prototype chip is first presented. Table 3.1 summarizes the performance of the prototype chip built and tested. Figure 3.22 shows the die photo delineating the MX-PDLL and power supply noise generator. The nominal supply voltage is 1.8V and the nominal operating frequency is 750 MHz.

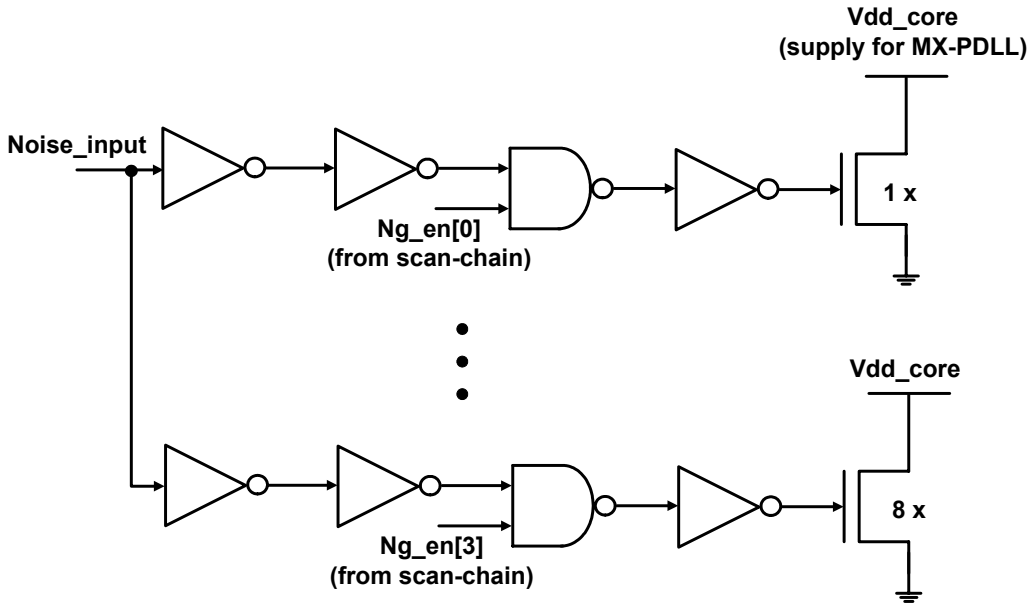


Figure 3.21: Schematic of the power supply noise generator.

Table 3.1: Test chip performance summary - MX-PDLL.

Process:	TSMC 0.18µm CMOS logic
Supply voltage:	1.8V typical
PLL frequency range @ 1.8V:	537MHz~818MHz
Quiescent *bclk* jitter:	
PLL @ 750MHz	rms: 1.37ps; pk-to-pk: 12.87ps
DLL @ 750MHz	rms: 1.27ps; pk-to-pk: 11.84ps
Power @ 750MHz, 1.8V:	77mW

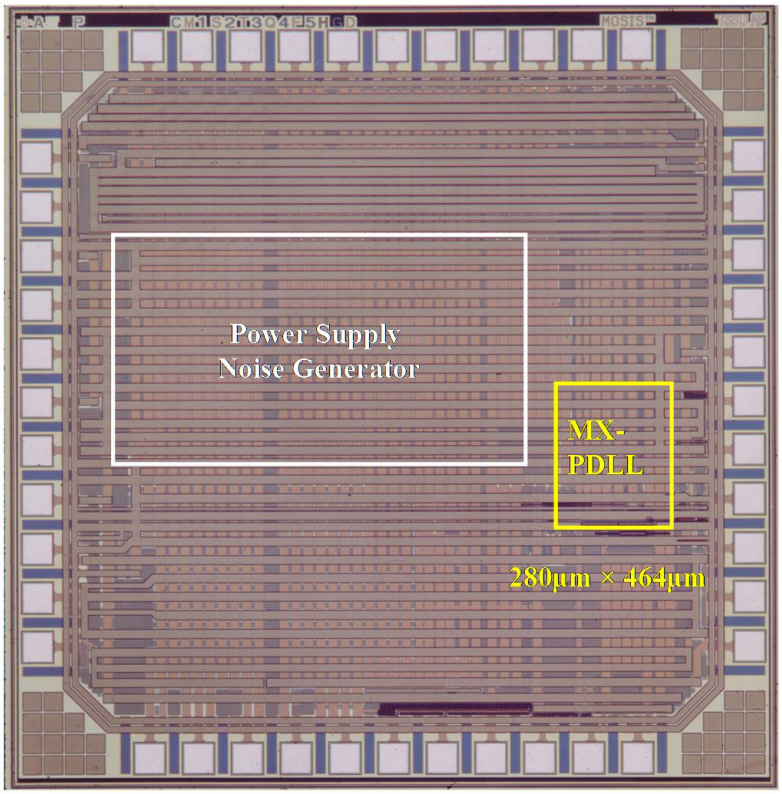


Figure 3.22: Die photo of the prototype chip.

3.4.3 Phase Transfer Function

The measured adaptive-bandwidth MX-PDLL input phase transfer function from *refclk* to *bclk* is presented in Figure 3.23. The *bclk* frequency is 750 MHz under a 1.8 V power supply throughout this measurement. A single frequency sinusoidal signal is coupled to *refclk* to enable the sweeping of the *refclk* phase noise frequency. First the phase jitter that is defined before is calculated from 15,000 captured clock edges, and then this measured timing jitter is converted to the frequency domain by taking the Fourier transformation of the samples in order to calculate the ratio of the *bclk* jitter spectrum and *refclk* jitter spectrum for a given *refclk* phase noise frequency.

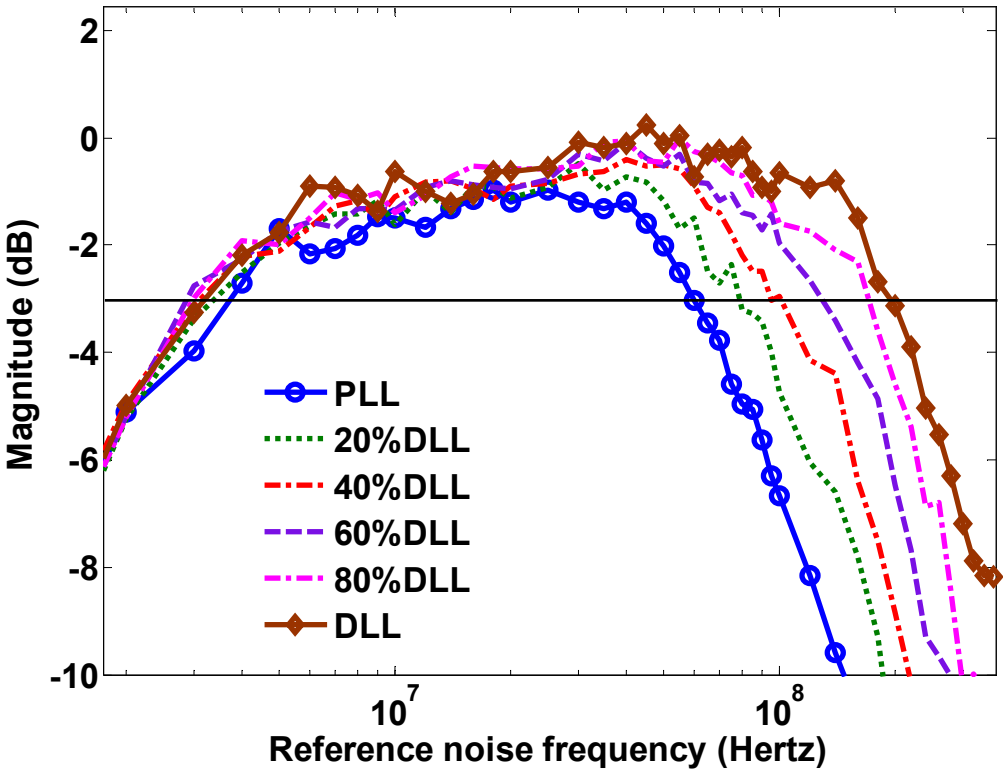


Figure 3.23: Measured MX-PDLL phase transfer function (*bclk*/*refclk*) for various operation modes.

The measured PLL bandwidth is higher than intended, which may be attributed to the non-zero injection of the *rclk* signal through the parasitic capacitors at the input nodes in the phase mixing interpolator even in the PLL-mode operation. This results in a mixed-mode operation even when the interpolation weight is set at zero in the PLL-mode operation. This coupling between the interpolator's two inputs also affects the input phase transfer function of the DLL, which should exhibit an all-pass transfer function as the z-model model predicts. Instead the DLL shows limited measured bandwidth due to

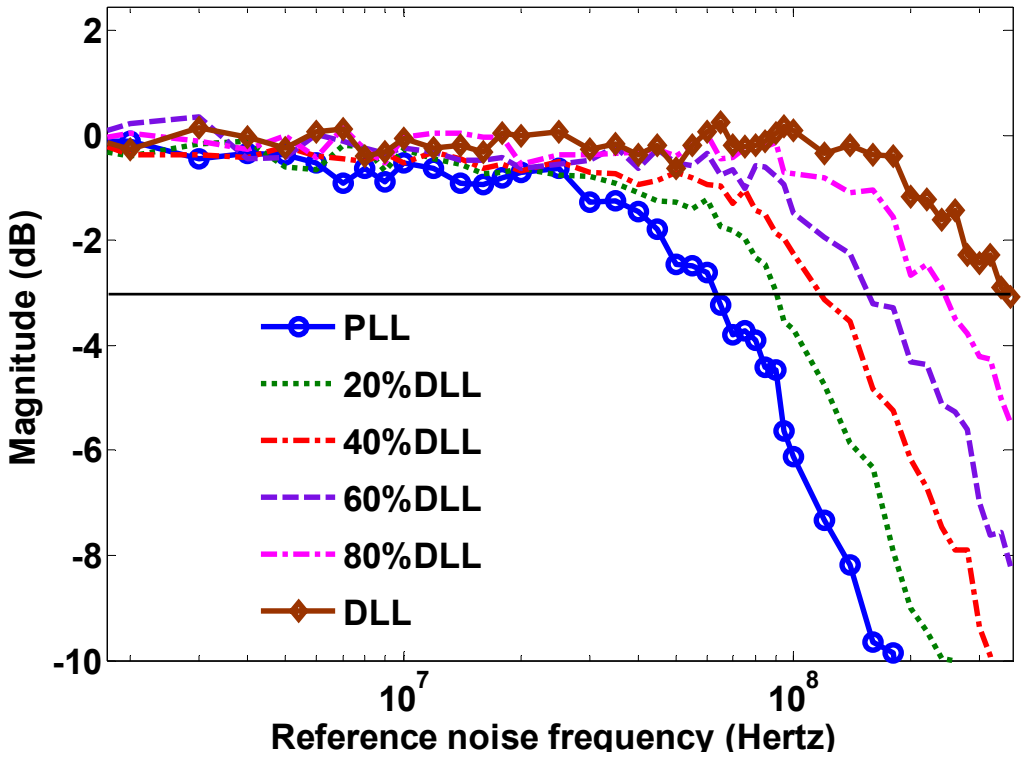


Figure 3.24: Measured MX-PDLL phase transfer function (*bclk*/*rclk*) for various operation modes.

the unintended mixed-mode operation. To lower the bandwidth in the PLL-mode, good isolation between *bclk* and *rclk* is required in order to minimize the coupling from the interpolator's inputs to its output.

Figure 3.24 plots the phase transfer functions of *bclk* with respect to *rclk,* and exhibits the -3dB-bandwidth of 60 MHz in PLL-mode operation, and over 300 MHz in DLL-mode operation. Notice that Figure 3.23 plots the measured input phase transfer function from *rclk*, which is the input to the MX-VCO/VCDL as shown in Figure 3.11, to *bclk*, whereas Figure 3.24 plots the measured input transfer function from *refclk*, which is the input to the reference clock buffers, to *bclk*. Since *rclk* is the output of the reference clock buffers, it does not exhibit the feed-forward delay modulation from Vg generator as *refclk* does.

The MX-PDLL's wide range bandwidth adjustment capability has been verified from these measurement results. The measured *bclk* jitter in response to injected wide-spectrum supply noise is presented in the next section.

3.4.4 Output Noise vs. Supply Noise Frequency

In order to measure the supply noise-induced jitter across all the operation modes, a frequency-controlled sinusoidal differential input is applied to the on-chip noise generator to intentionally create voltage noise on the power supply. Given the difficulty of measuring the magnitude of the on-chip supply noise-induced phase noise at specific frequencies, a conventional noise transfer function plot cannot be presented. Instead, Figure 3.25 plots the *bclk* rms jitter vs. power supply noise induced across a wide range

of frequencies. As expected, the PLL-mode operation exhibits the biggest low-frequency supply noise-induced jitter compared to the DLL-mode.

The multiple sets of peaks in this plot can be attributed to a variety of factors. The first set of peaks corresponds to the different loop bandwidths. The second set comes from resonance in the on-chip power supply network. Lastly, the discrete-time model

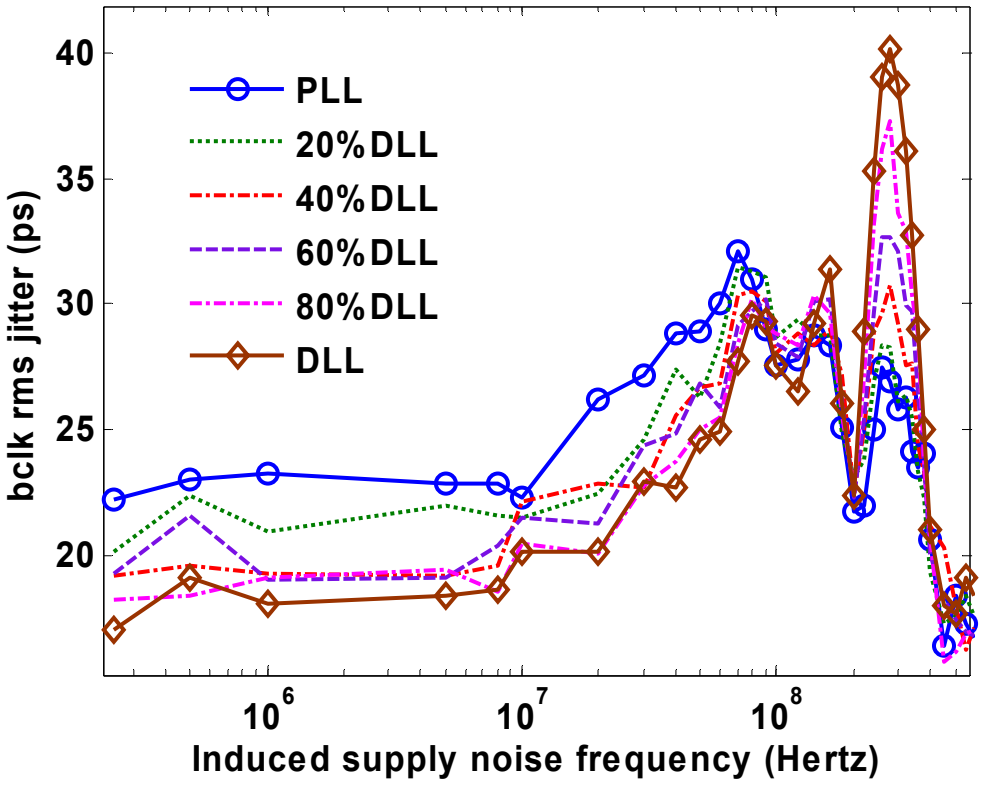


Figure 3.25: Measured *bclk* rms jitter vs. induced supply noise frequency.

shows that the third set in the 200–300 MHz range arises from feed-forward delay modulation of the *rclk* buffers that drive the MX-VCO/VCDL. The z-domain model also predicts this peak in Figure 3.10, which can be eliminated by further filtering the control

signals that drive these buffers or by completely eliminating the feed-forward path. Nevertheless Figure 3.25 still shows that due to the jitter accumulation in the PLL loop, the supply noise-induced jitter in the PLL mode is the biggest among all the operation modes across a wide range of frequencies.

With the measurement results of the phase transfer function and supply noise-induced jitter vs. noise frequency, we are now ready to look at the jitter performance of the MX-PDLL across all the operation modes under various noise conditions. The measured *bclk* rms jitter with different amount of injected supply noise is presented first. Then the measured *bclk* rms jitter with different amount of injected reference clock noise is presented. Last, the optimum bandwidth settings in the mixed-modes when both the supply noise and reference clock noise are present are found.

3.4.5 Jitter Results

Before we look at the jitter performance with additional injected noise, the jitter histograms in the PLL-mode and DLL-mode with a quiescent power supply of 1.8 V at 750 MHz are presented in Figure 3.26. The jitter histogram is measured with a sampling scope using the reference clock as a trigger in order to measure the relative phase jitter between *bclk* and *rclk*. The histograms show that the PLL-mode operation leads to slightly higher jitter due to noise accumulation. In the PLL-mode, the bclk rms jitter of 1.37 ps and peak-to-peak jitter of 12.87 ps are recorded. In the DLL-mode, the bclk rms jitter of 1.27 ps and peak-to-peak jitter of 11.84 ps are recorded. For all of the following jitter measurement results, the *bclk* frequency is 750 MHz under a 1.8 V power supply.

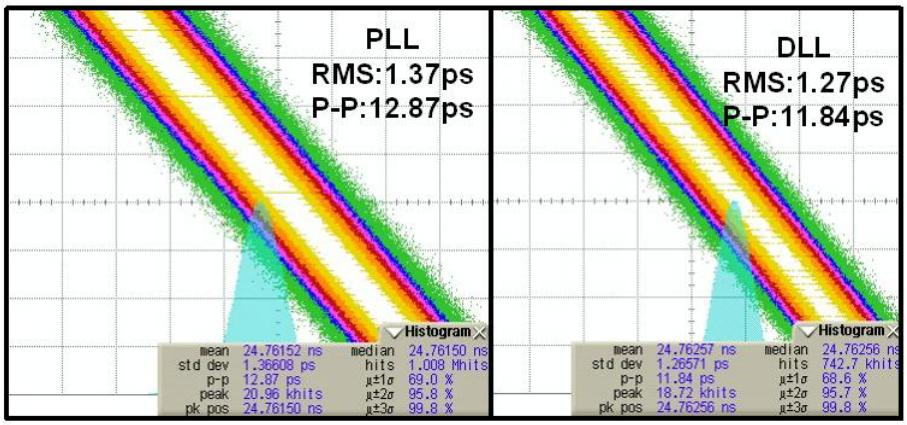


Figure 3.26: Measured jitter histograms at PLL and DLL modes at 750MHz with a quiescent supply.

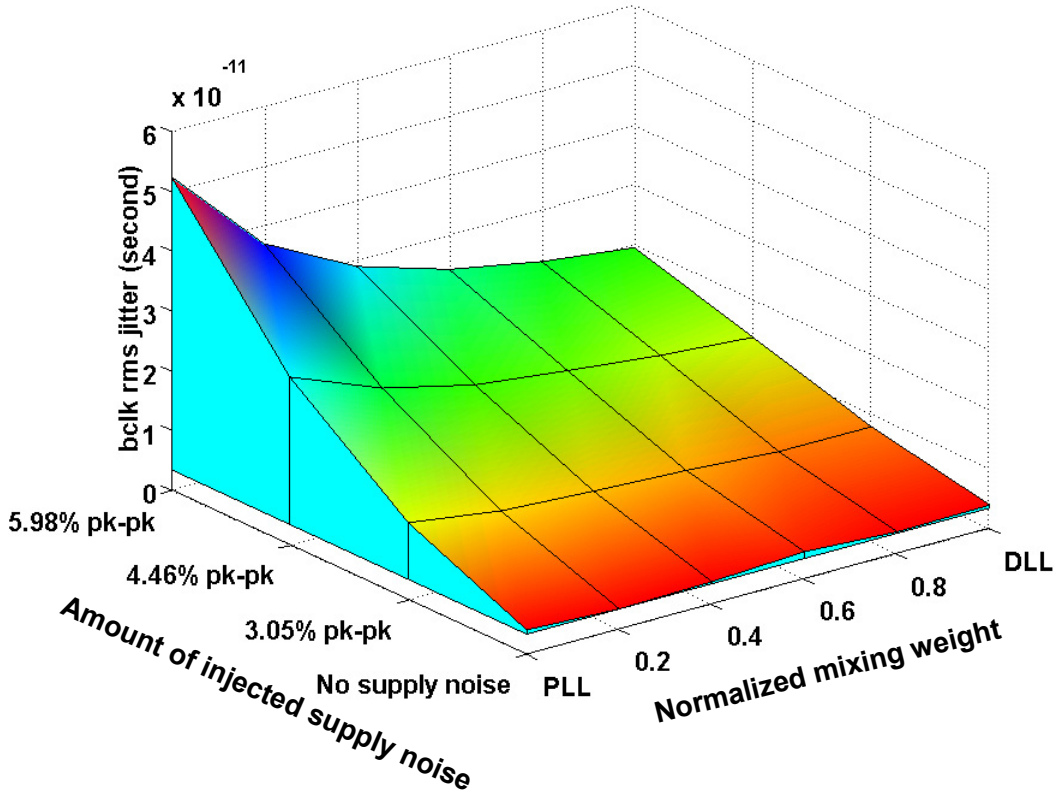


Figure 3.27: Measured rms jitter on *bclk* across different mixing weights vs. supply noise amount.

Figure 3.27 shows the measured *bclk* rms jitter across normalized mixing weights vs. multiple levels of artificially induced on-chip supply noise. The induced wide-spectrum supply noise is generated by the 2^{31}-1 Pseudo Random Bit Sequence (PRBS) at

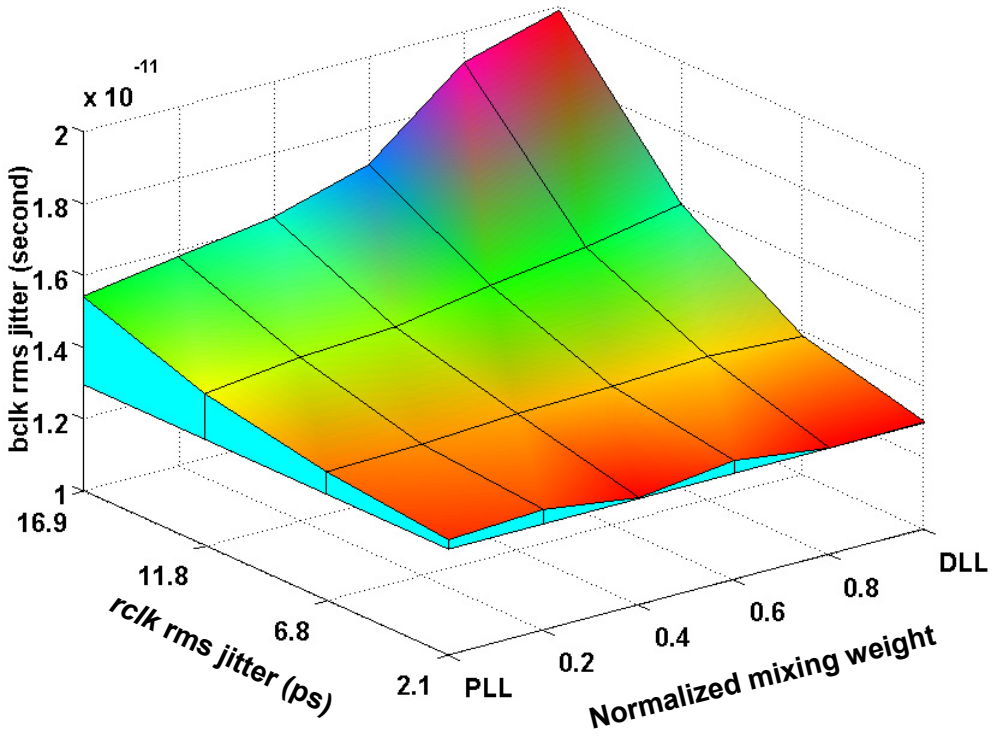


Figure 3.28: Measured rms jitter on *bclk* across different mixing weights vs. *refclk* noise amount.

23.4375 Mb/s from the Tek DTG5078 digital timing generator. An I/O pin is dedicated to monitoring the on-chip supply noise level. The phase jitter is measured in the self-triggering mode. The *bclk* jitter can pick up all the noise along the output buffer chain up to the output pin. This explains why the jitter is unusually large, when large amount of supply noise is injected into the power supply.

When there is no additional supply noise, the output clock jitter suggests that PLL and DLL have comparable jitter performance. As the induced supply noise increases, the PLL-mode operation suffers from jitter accumulation and the optimum bandwidth setting (to minimize jitter) shifts toward the DLL-mode operation as expected.

Figure 3.28 presents the measured *bclk* rms jitter across normalized mixing weights vs. multiple levels of artificially induced *refclk* noise. The wide-spectrum *refclk* noise is induced onto chip by coupling a 2^{31}-1 PRBS at 750 Mb/s from the AT 8133A pulse generator to the clock source. As expected, the PLL-mode operation yields lower jitter by filtering more of the induced *refclk* noise at high frequencies. Now that we have seen the jitter performance with only injected supply noise or reference clock noise, what if both types of noise are present and comparable?

Lastly, we consider what happens when we couple different amounts of wide-spectrum noise onto the reference clock while there is 3% peak-to-peak supply voltage variation. Figure 3.29 shows the shifting of the optimum mixing weights toward the PLL-mode as the amount of reference clock noise increases. The jitter across all operation modes also increases when the reference clock noise increases as expected.

From the above measurement results, we have verified the tradeoffs between input reference noise filtering and on-chip power supply noise suppression in terms of the clock generator's loop bandwidth. By tuning the interpolation weight, the bandwidth of the MX-PDLL can be tuned across a wide range of frequencies. This bandwidth tuning capability proves to be advantageous in order to minimize the output clock dynamic jitter under different on-chip and off-chip noise conditions. The next question is: Can we use the MX-PDLL in the CDR to take advantage of this bandwidth tuning capability? The

answer is yes, and Chapter 4 presents a CDR architecture that incorporates the MX-PDLL to achieve optimum receiver clock jitter performance.

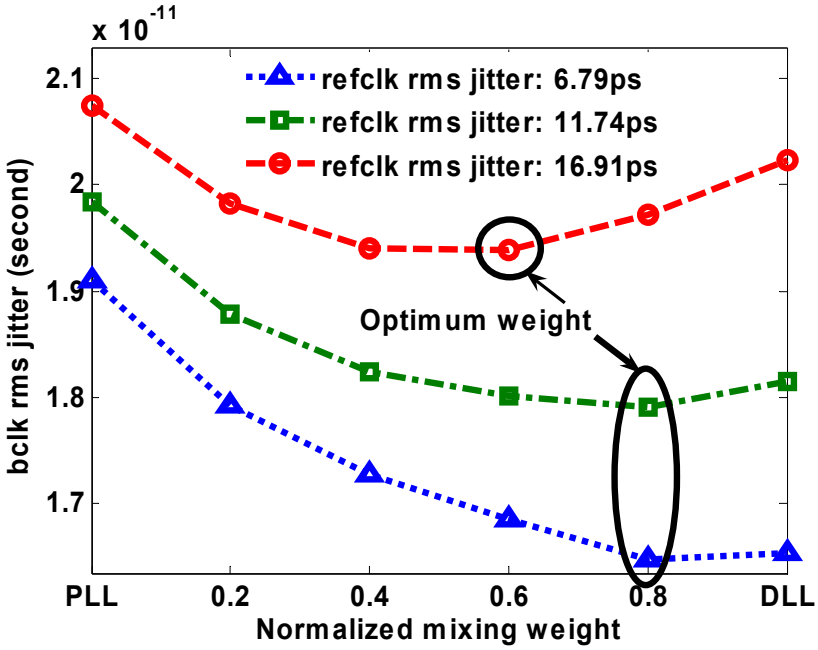


Figure 3.29: Optimum mixing weight vs. *refclk* noise for a given 3.05% peak-to-peak supply noise.

3.5 Summary

An adaptive-bandwidth MX-PDLL-based multiphase clock generator that can operate as a PLL, DLL, or a mixture of the two, and which enables wide range bandwidth adjustment to minimize the output clock dynamic jitter under various noise environments, is presented. The bandwidth tuning ability is predicted by MATLAB® simulation results

and verified with measurement results from the test-chip prototype. The experimentally measured results verify that while the PLL mode has a greater ability to filter high-frequency reference noise, shifting toward the DLL mode can reduce more of the jitter induced by the on-chip supply noise. When both noise sources are present, and the relative amount might not be known *a priori*, a mixed mode might be the best choice for minimizing jitter.

Having achieved this wide range bandwidth tuning capability to minimize the output clock dynamic jitter with the adaptive-bandwidth MX-PDLL, such a clock generator may be used in a dual-loop clock and data recovery architecture that incorporates the MX-PDLL as the core multiphase clock generator. We see how to again take advantage of the bandwidth tuning ability of the MX-PDLL in a CDR in Chapter 4.

Chapter 4

Clock and Data Recovery Using the Adaptive-Bandwidth Mixed PLL/DLL-Based Multiphase Clock Generator

Chapter 3 presented an adaptive-bandwidth MX-PDLL-based multiphase clock generator that can operate as a PLL, DLL, or a combination of the two, which enables wide range bandwidth adjustment to minimize jitter under various noise environments. That design also raised the question as to whether we can use the MX-PDLL in a CDR to build a noise-robust high-speed link system because most of the CDR in the high-speed link receivers use either a PLL- or DLL-based clock generator [1][2]. Various on-chip and off-chip noise conditions make it challenging to design a clock generator that can achieve optimal jitter performance under different noise environments. This chapter describes a dual-loop CDR architecture using this adaptive-bandwidth MX-PDLL-based multiphase clock generator.

A feature of the proposed architecture that is distinguishable from previously published dual-loop architectures [1][2][3] is that the bandwidth of the core clock generating loop can be tuned across a wide range of frequencies in response to various noise conditions. Therefore we can minimize output clock jitter as is shown in the measurement results in Chapter 3. Moreover this dual-loop CDR relies on a phase rotator

to track the phase and frequency of the data. A digitally-controlled 360° phase rotator that rotates the reference clock that feeds into the MX-PDLL is used to achieve phase and frequency tracking for plesiochronous[7] interfaces. The phase rotator is commonly implemented with four 90-degree phase-shifted clocks (I-clock and Q-clock) and their complements (I_b-clock and Q_b-clock), which drive a phase interpolator [7]. In this CDR, a divide-by-2 (÷ 2) circuit is used to generate four 90-degree phase-shifted clocks from only one external reference clock. This approach simplifies the generation of quadrature clocks at a cost of requiring a reference clock with a higher frequency than the output clock.

The proposed CDR uses a time-interleaved receiver architecture to achieve high data rates that do not suffer process speed limitations [6]. The CDR uses multiphase sampling clocks to recover the data. This type of receiver design requires very precise clocking to improve the performance. Otherwise clock timing uncertainty can result in receiver timing margin degradation and sub-optimal sampling positions, which can both degrade performance. Clock timing uncertainty can be categorized into two types – dynamic and static. The 'dynamic' type, known as random clock jitter, can be minimized by dynamically adapting the bandwidth of the core clock generation loop. The 'static' type, which introduces uneven phase spacing among multiphase clocks, can be calibrated using a phase mismatch detector and several compensation schemes on-chip. In this chapter, we focus on minimizing dynamic clock jitter by tuning the clock generator's

---

[7] Plesiochronous system refers to a system in which each component operates at nominally the same rate with a variation in rate being constrained within specified limits. On the contrary to a plesiochronous system, in a mesochronous system all the components run at exactly the same frequency.

bandwidth. The discussion of the static phase mismatch compensation can be found in Chapter 5.

First the architecture of a dual-loop CDR that employs the MX-PDLL as its core loop is presented in Section 4.1, and then the circuit designs of the main building blocks are discussed in Section 4.2. Measurement results of the prototype system that evaluate important characteristics of this architecture are presented in Section 4.3.

4.1 Architecture

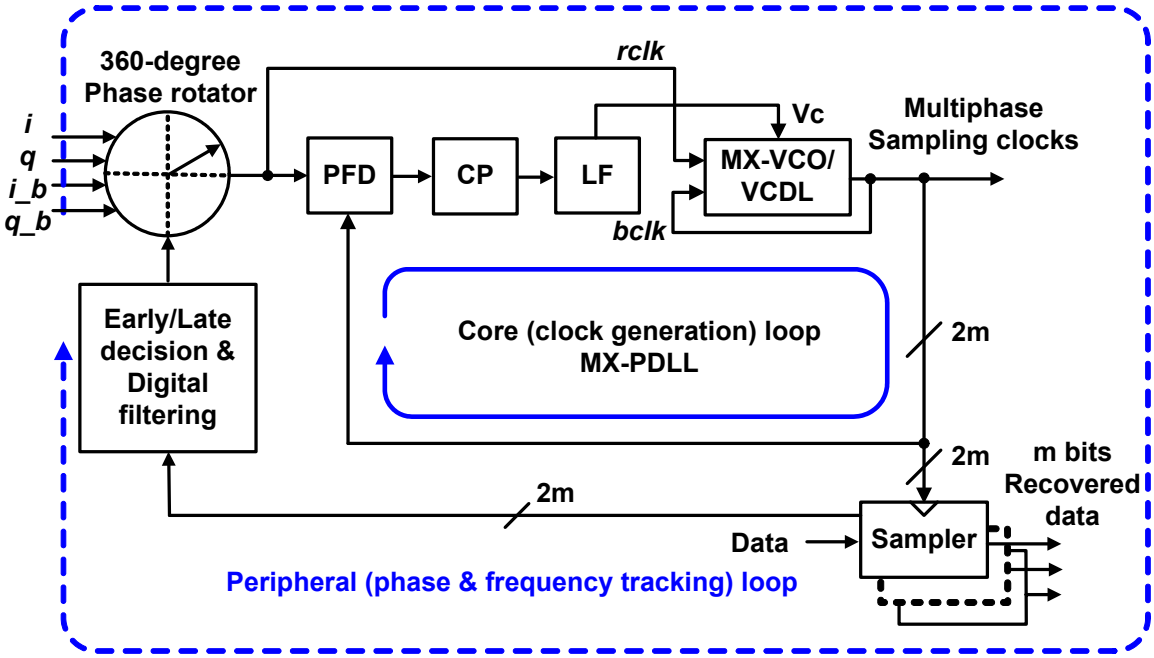


Figure 4.1: Dual-loop CDR architecture incorporating the MX-PDLL in the core clock generation loop.

A top-level block diagram of the proposed CDR, shown in Figure 4.1, consists of two loops, a core clock generation loop and a peripheral loop. The core loop consists of the MX-PDLL that generates multiphase sampling clocks for data and edge samplers. Early/late information out of the samplers is digitally filtered and used to control the phase rotator to shift the reference clock (*rclk*) phase that feeds into the MX-PDLL, thereby tracking the phase and frequency of the data. This completes the peripheral phase and frequency tracking loop. The phase rotator takes four 90-degree phase-shifted clocks and interpolates two adjacent clock phases in order to achieve the 360-degree phase-shifting range. The time-interleaved receiver architecture is employed to relax the on-chip clock frequency requirement and to improve the data rate for a given process.

An important attribute of this CDR architecture is that the quantization noise from the digitized phase rotator step can be filtered by the MX-PDLL loop. And since the core loop bandwidth is tunable, this provides another opportunity to make tradeoffs between quantization noise filtering and supply noise suppression. The measurement results are provided to verify this later in this chapter.

If bandwidth tuning is desired, the MX-PDLL cannot be used in the feedback clock selective architecture that is shown in [9], which can only be configured as a PLL. Instead, using a phase rotator to adjust *refclk* enables the core loop to operate in PLL, DLL, and mixed modes. Both architectures have the advantage of directly generating multiple, evenly spaced clock phases out of the VCO/VCDL. Therefore, these architectures are better suited for time-interleaved receiver designs.

In a time-interleaved receiver design, multiple clock phases are used to clock the data and edge samplers in order to recover both the data timing and the data. Uneven

static phase spacing in the MX-PDLL will result in a static skew of the sampling clock phase with respect to the ideal sampling point, therefore potentially causes an error. The static phase spacing mismatch primarily comes from process variations and systematic imbalances in the physical design (layout mismatch). In Chapter 5, several compensation schemes to overcome this issue are discussed in more detail. Also another source of mismatch in this system is stemming from the phase mismatch among the four 90-degree phase-shifted clocks (i.e., *i*, *q*, *i_b*, and *q_b*). This phase rotator quadrant mismatch is due to a non-50% duty-cycle of an external reference clock that is used to generate the four clock phases. This quadrant mismatch can further degrade the phase rotator non-linearity as well as the receiver performance. A duty-cycle correction circuit is also presented to overcome this problem in Chapter 5.

Detailing the rest of this chapter, first, the detailed circuit implementations of the main building blocks in the system (except the blocks, which have been described in Chapter 3) are presented in Section 4.2. Then the measurement results of running the system in an open-loop configuration are shown in Section 4.3.

4.2 Circuit Implementation

Since the phase rotator enables phase and frequency tracking of the CDR as well as the system's frequency synthesis ability, we focus on the detailed circuit implementation of this important block in the following subsection.

4.2.1 Phase Rotator

The phase rotator that enables phase and frequency tracking in the dual-loop CDR has three design considerations: (1) 360° phase rotation range, (2) the need to handle the meta-stability issue associated with asynchronous off-chip control, and (3) proper timing design for phase rotation codes to avoid incomplete phase mixing that can cause glitches on the output.

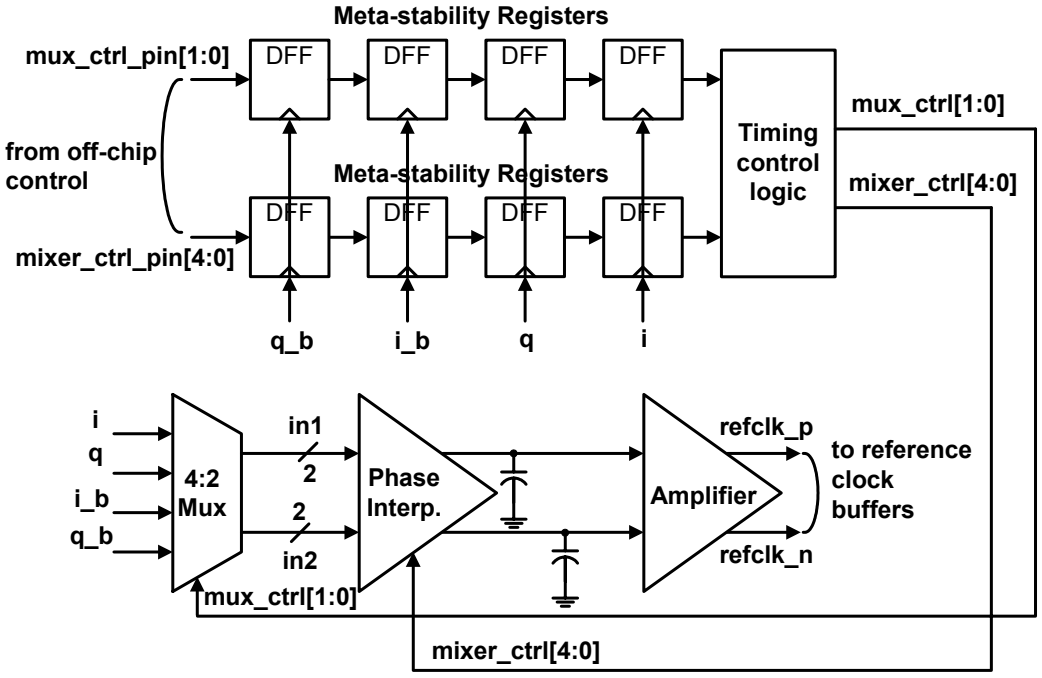


Figure 4.2: Phase rotator block diagram.

Figure 4.2 presents the phase rotator block diagram in detail. Four 90° phase-shifted clocks (*i*, *q*, *i_b*, and *q_b*) drive a 4:2 multiplexer (mux), which selects a pair of adjacent clock phases (quadrant selection) to be interpolated by the phase interpolator to achieve fine phase adjustment and 360° operation. The schematic of the phase

interpolator is shown in Figure 4.3, and consists of five pairs of binary-weighted tri-state buffers with their outputs shorted together. Scaling the relative driving strengths of the two sets of buffers, driven by *in1* and *in2*, enables 32 programmable output phases. The control switches are placed closer to the output in order to prevent the input from feeding through the parasitic capacitors to affect the output. The pseudo-differential outputs are filtered by the capacitors to slow down the voltage transition in order to improve linearity. The subsequent amplifier then restores the interpolator outputs to full swing signals. By selecting one of four quadrants, via the mux, a total of 128 clock phases that span 360° is achieved.

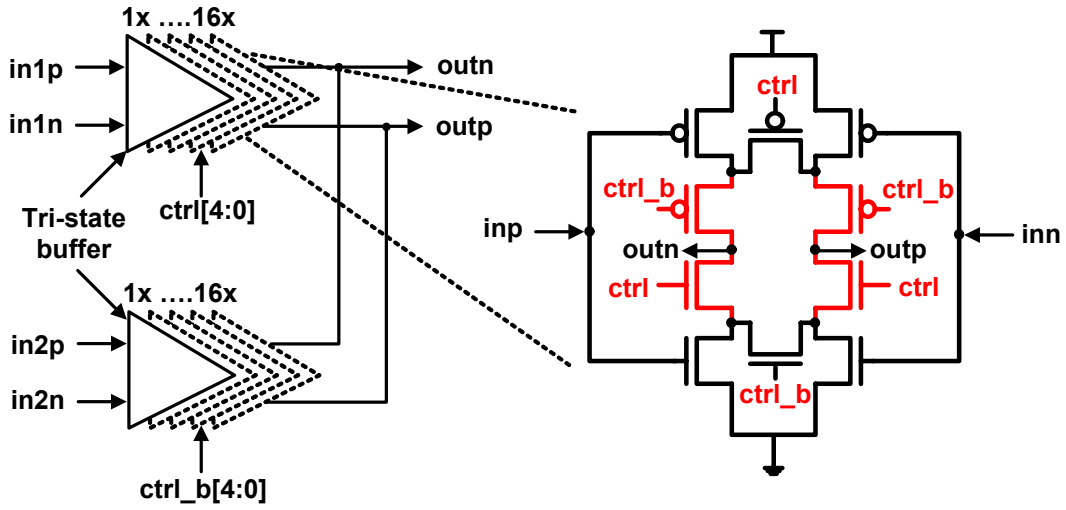


Figure 4.3: Schematic of the tri-state buffer-based phase interpolator.

Meta-stability registers are added to synchronize the off-chip control signals with the on-chip clocks. The timing of the code change needs to be carefully designed to avoid glitches or missing edges at the output. The change of the quadrant can only take place after the previous cycle phase-mixing is complete. A control block is added to keep track

of the current and previous mixing quadrant and to generate synchronized control signals for the mux (*mux_ctrl[1:0]*) and the interpolator (*mixer_ctrl[4:0]*). Due to the internal delay associated with the control logic, the fastest feasible code change frequency is half the reference clock frequency. This control scheme also limits the consecutive code change to occur within two adjacent quadrants.

In pleisochronous operation, the CDR needs to track a slight frequency deviation between *refclk* and the incoming data, which has the same frequency as *bclk* [11][12][2]. The 360° phase rotator enables this frequency tracking. The frequency tracking range, phase transfer function, and average step size of the phase rotator all significantly impact CDR performance. The measured performance of the digitally-controlled phase rotator with a maximum of 128 phase steps is presented next.

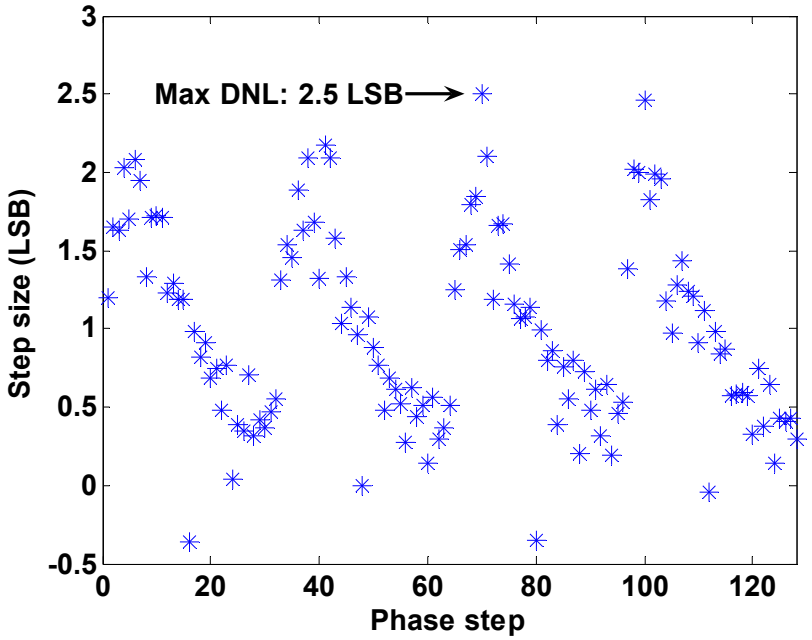


Figure 4.4: Measured phase step size of the phase rotator across one reference clock period (DNL).

Figures 4.4 and 4.5 present the measured phase step size and transfer function of the phase rotator with 128 phase steps across one reference-clock cycle, respectively. Figure 4.4 plots the measured phase step size across all phase rotator codes (differential non-linearity, DNL). The maximum step size is 2.5 LSB (least significant bit), which is equivalent to 7°, and the minimum is -0.36 LSB. Figure 4.5 plots the measured phase

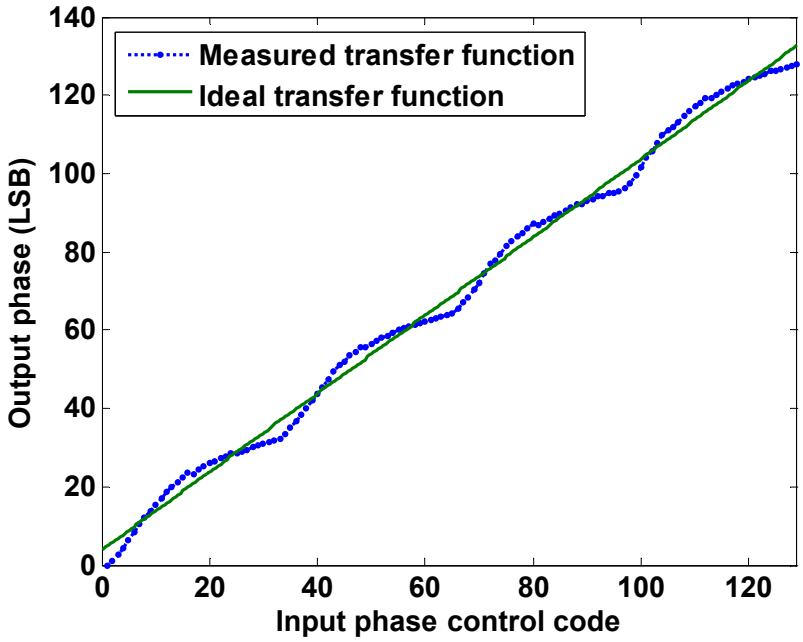


Figure 4.5: Measured phase rotator transfer function - output phase vs. input phase control code across one reference clock period.

rotator phase transfer function versus the ideal phase-transfer function. The integral non-linearity (INL) is less than ±5 LSB as shown in Figure 4.6. Notice that there is also non-monotonicity in the phase rotator transfer function due to the switching of the MSB (Most Significant Bit) devices, but this can be corrected at the software level. The non-

monotonicity comes from mismatch in the binary-coded phase interpolator, and it can be alleviated using a thermometer-coded phase interpolator.

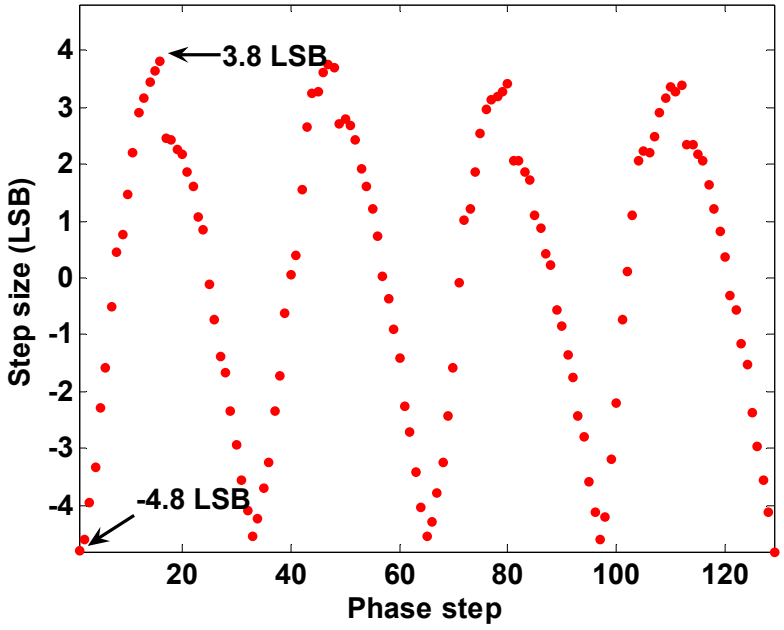


Figure 4.6: Phase difference between measured phase rotator transfer function and ideal transfer function across one reference clock period (INL).

A phase rotator generates a clock signal that spans 360° in 128 increments based on a digital controller is presented in this section. The measured DNL is less than 2.5 LSB with non-monotonicity due to the switching of the MSB devices. The measured INL is less than ±5 LSB. While we would like to make the DNL and INL perfect, the results are not that surprising given the use of full-swing signals. Given the non-linearity in the phase rotator, we can look at the output clock jitter in the MX-PDLL while the CDR architecture is configured open-loop in order to understand how the phase rotator's performance impacts the rest of the system. Moreover we can also look at the output

clock jitter while applying different digital controls to the phase rotator and investigate their corresponding impacts on the system. Therefore in the next section, the measurement results of the output clock jitter across all operation modes while applying different digital controls to the phase rotator under various noise conditions are presented.

4.3 Measurement Results

Figure 4.7 illustrates the prototype system built and tested. The MX-PDLL generates 16 clock phases for the samplers. In order to more easily investigate the impact of different control schemes on the CDR, the digital CDR control is implemented off-chip. Therefore the CDR peripheral loop cannot be closed on-chip. So instead we investigate an interesting application using this architecture in an open-loop configuration as a frequency synthesizer. Frequency synthesis is important for plesiochronous links, because the CDR needs to track a slight frequency deviation between the reference clock and the incoming data. In a closed-loop CDR, the phase and frequency tracking would be achieved via the 128-step digitally-controlled 360° phase rotator, which generates *refclk* that feeds into the MX-PDLL. A divide-by-2 (÷ 2) block takes a duty-cycle corrected *xclk* and drives four 90° phase-shifted clocks (*i*, *q*, *i_b*, and *q_b*) to the phase rotator. The frequency of *xclk* is two times faster than the frequencies of *refclk* and *rclk*. Therefore when the MX-PDLL is locked, the frequency of *bclk* is half the frequency of *xclk*. Phase tracking can be achieved by appropriately rotating *refclk* via the digital control. Frequency tracking can be achieved by constantly stepping (advancing or retarding) the

phase rotator at a nominally fixed rate to cover small frequency differences between *xclk* and the recovered clock in a CDR.

With the open-loop configuration, frequency synthesis can be easily achieved by controlling the phase rotator with the off-chip digital CDR control. The control schemes tested on the chip include (1) bang-bang limit-cycle phase dithering digital control, (2) fixed phase advancing/retarding rate frequency tracking control, and (3) second-order Delta-Sigma (ΔΣ) modulated (DSM) frequency synthesis control. The 7-bit control signal generates up to 128 phase steps that enable fine phase adjustment in the MX-PDLL.

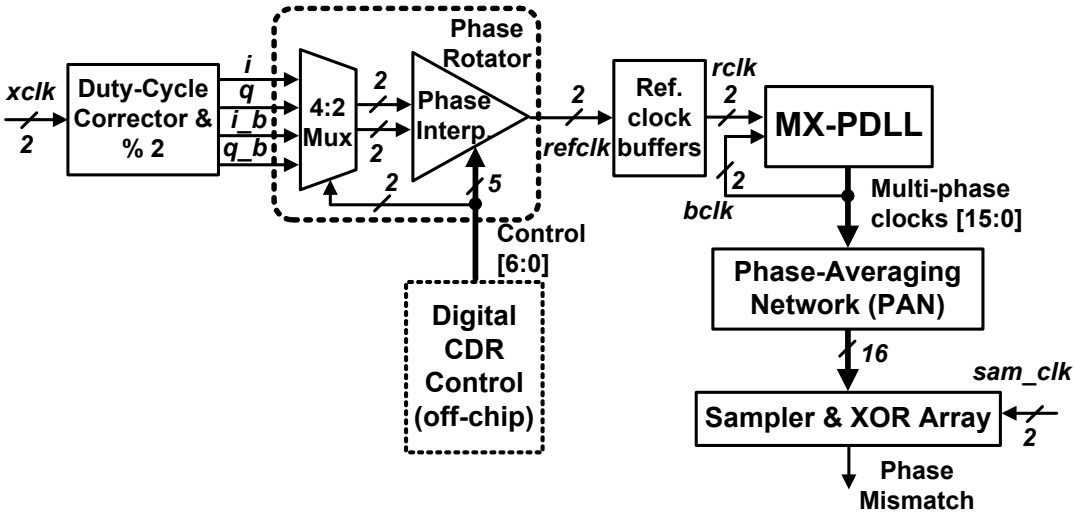


Figure 4.7: Block diagram of the prototype system built and tested.

The quantized phase step from the phase rotator due to the digital control manifests itself as dithering noise on *refclk*. This dither can be filtered out if the core loop is operating in the PLL mode. However, when the on-chip supply noise is also present, a shifting toward the DLL-mode operation might yield lower output clock jitter. Since the second-order DSM control shapes the quantization noise to higher frequencies, one

would expect PLL-mode operation to yield lower jitter when no additional injected on-chip supply noise is present. Again, the optimum core loop bandwidth setting might change when supply noise is present.

The quantized phase step from the phase rotator requires low-pass filtering through the core clock generating loop to reduce output jitter, which also means that a smaller phase step size would yield smaller output jitter resulting from the quantized steps. When the proposed CDR architecture is used as a frequency synthesizer in an open-loop configuration, it is much more sensitive to phase rotator linearity than the close-loop CDR configuration. The off-chip controller also facilitates the remapping of the phase rotator codes to reduce non-linearity at the cost of larger phase steps. The remapped codes can yield the smallest output jitter even with coarser phase resolution in the open-loop configuration.

The measurement results of the prototype system that was built are shown in the subsequent sections. The output clock rms jitter is defined as the phase jitter, which is the same kind of jitter as was discussed in Chapter 3. The real-time oscilloscope captures at least 15,000 clock edges in the self-triggering mode, and calculates the standard deviation of output clock jitter. This measuring method tends to yield larger jitter than that found in the reference-triggering mode. But again the results are presented in order to demonstrate how the optimum setting can shift with respect to different noise conditions and in order to investigate the advantage of bandwidth tuning of the clock generator.

The rest of this section is organized as follows. First the test setup for all the measurements is presented in Section 4.3.1. The output clock jitter performance with limit-cycle bang-bang phase dithering digital control is presented in Section 4.3.2. The

frequency tracking performance of the phase rotator with two different frequency tracking controls under various noise conditions is presented in Section 4.3.3. The impact of the quantization step size on system performance is presented in Section 4.3.4 followed by the impact of phase rotator linearity on frequency tracking and jitter performance in Section 4.3.5.

Table 4.1: Test chip performance summary with phase rotator.

Process:	TSMC 0.18μm CMOS logic
Supply voltage:	1.8V typical
PLL frequency range @ 1.8V:	537MHz~818MHz
Quiescent *bclk* jitter:	
PLL @ 750MHz	rms: 1.37ps; pk-to-pk: 12.87ps
DLL @ 750MHz	rms: 1.27ps; pk-to-pk: 11.84ps
Phase rotator:	
INL:	± 5 LSB
DNL:	< 2.5 LSB
Quadrant mismatch:	< 0.65% of the desired 25%
Maximum synthesizable frequency offset:	4200ppm
Power @ 750MHz, 1.8V:	77mW

Before presenting the detailed measurement results of the prototype chip, a performance summary of the system is presented in Table 4.1. The nominal output clock frequency is 750 MHz operating off of a 1.8 V power supply. The maximum phase rotator phase update rate is 375 MHz at the nominal operating frequency, and the

maximum synthesizable frequency offset is ±4200 ppm. Table 4.1 also presents the measured linearity of the phase rotator. Figure 4.8 shows the die photo of the prototype chip delineating the phase rotator, divide-by-2 block and duty-cycle corrector hidden under the global power supply lines.

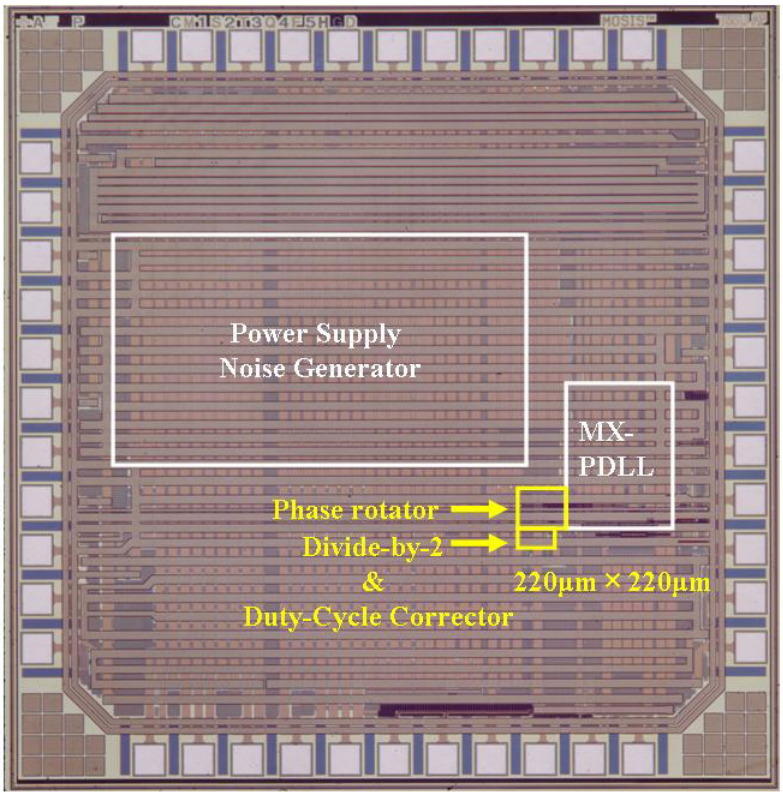


Figure 4.8: Die photo of the prototype chip.

4.3.1 Test Setup

The test setup and Tektronix® Digital Timing Generator (DTG5078) that is used to provide off-chip digital control for the phase rotator were shown previously in Figure 3.20. It is important to ensure that all seven input control paths have the same delay from

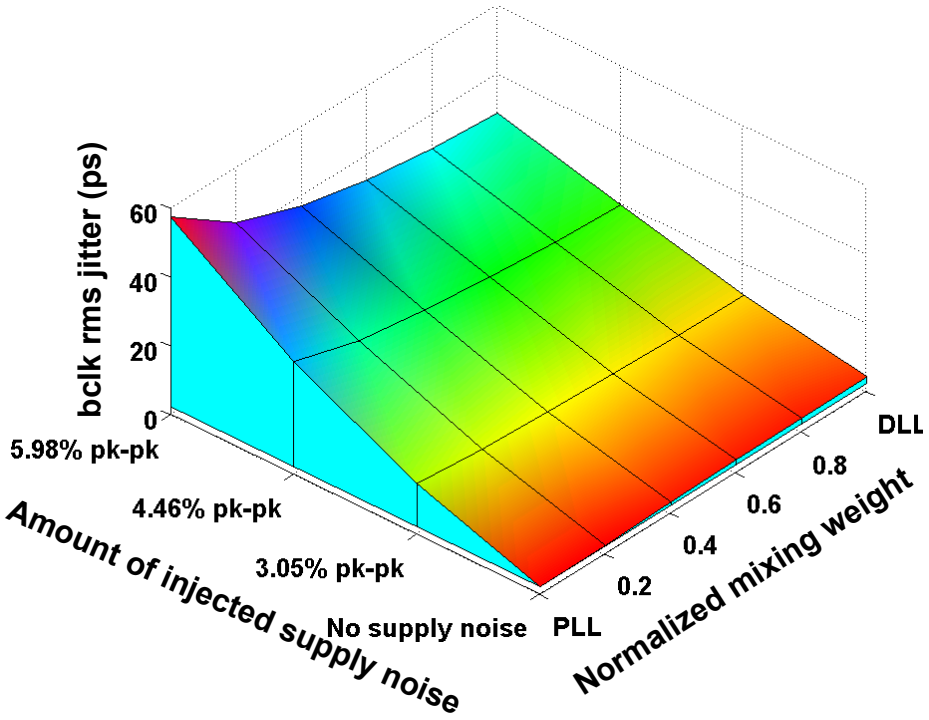
the external generator to the chip I/O pins. The DTG5078 comes with a self-calibration feature to compensate for the delay mismatch among different cables. The traces on the board were carefully laid out to minimize additional delay mismatch from the connectors to the I/O pins.

4.3.2 Limit-Cycle Dithering Digital Control

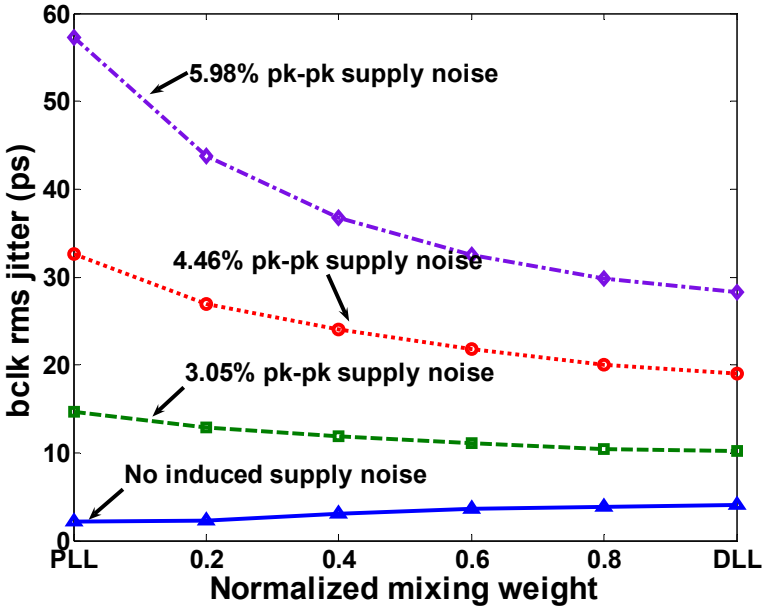
The phase tracking control of the dual-loop CDR creates quantized dithering noise on *refclk* because of the loop's digital control. To analyze the impact of the *refclk* dither noise on *bclk* jitter across different mixing weights, a bang-bang limit-cycle phase dithering digital control signal is applied to the phase rotator. A limit-cycle control is a periodic bang-bang signal with a period that has a normal distribution. The nominal dithering frequency of the test control signal is 1/143 of the *bclk* frequency (750MHz), which is about 5.24 MHz.

Figure 4.9 shows that the PLL-mode operation filters more of the controller-induced wide-band *refclk* noise due to its low-bandwidth loop and it yields lower *bclk* root-mean-square (rms) jitter without any additional injected on-chip supply noise. On the other hand, the DLL-mode operation suffers from direct phase modulation from the phase rotator and thus yields slightly higher *bclk* rms jitter. As the supply noise increases, the optimum bandwidth setting (to minimize jitter) shifts toward the DLL-mode due to jitter accumulation in low-bandwidth setting modes.

In the next section, the output jitter due to the frequency synthesis control to compensate for small frequency deviation between *refclk* and *bclk* is presented. The phase rotator linearity and frequency synthesizing performance is also plotted.



(a)



(b)

Figure 4.9: Measured *bclk* rms jitter across different mixing weights vs. power supply noise with limit-cycle digital control. (a) Surface plot. (b) Line plot.

4.3.3 Frequency Synthesis Results with Delta-Sigma and Fixed-Rate Frequency Synthesis Controls

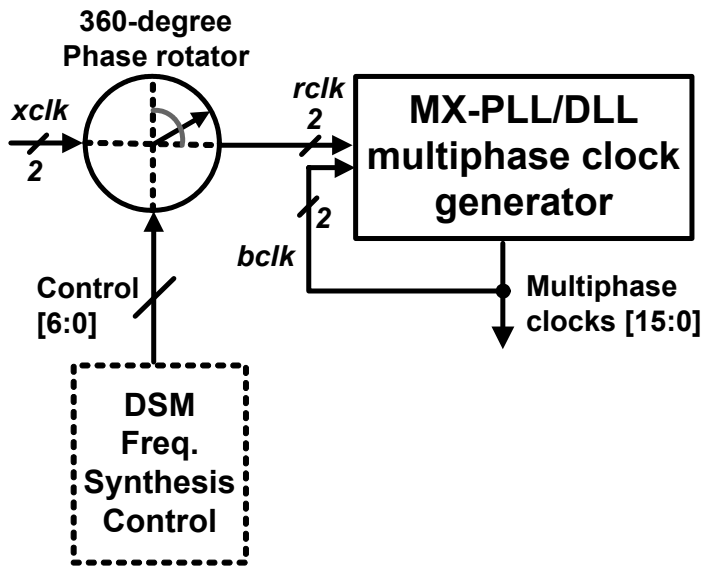


Figure 4.10: DSM frequency synthesizer block diagram.

In plesiochronous operation, the CDR needs to track a slight frequency deviation between *refclk* and the incoming data, which has the same frequency as *bclk* [11][12][2]. The 360° phase rotator can track frequency differences by rotating *rclk* phase, shown in Figure 4.10. If the phase change happens slowly enough, the MX-PDLL loop will make sure that *bclk* tracks the *rclk* phase. Therefore *bclk* frequency can be changed according to how often we shift *rclk* phase. Figure 4.10 shows the top-level block diagram of the frequency synthesizer using the proposed CDR architecture in an open-loop configuration. The frequency tracking range, the linearity of the phase transfer function, and average step size of the phase rotator have a significant impact on frequency

synthesis performance. In this section, two frequency synthesis control schemes are applied to the phase rotator in an open-loop configuration, fixed-rate and second-order Delta-Sigma Modulation (DSM), to investigate the frequency synthesizing performance of using a phase rotator followed by the MX-PDLL.

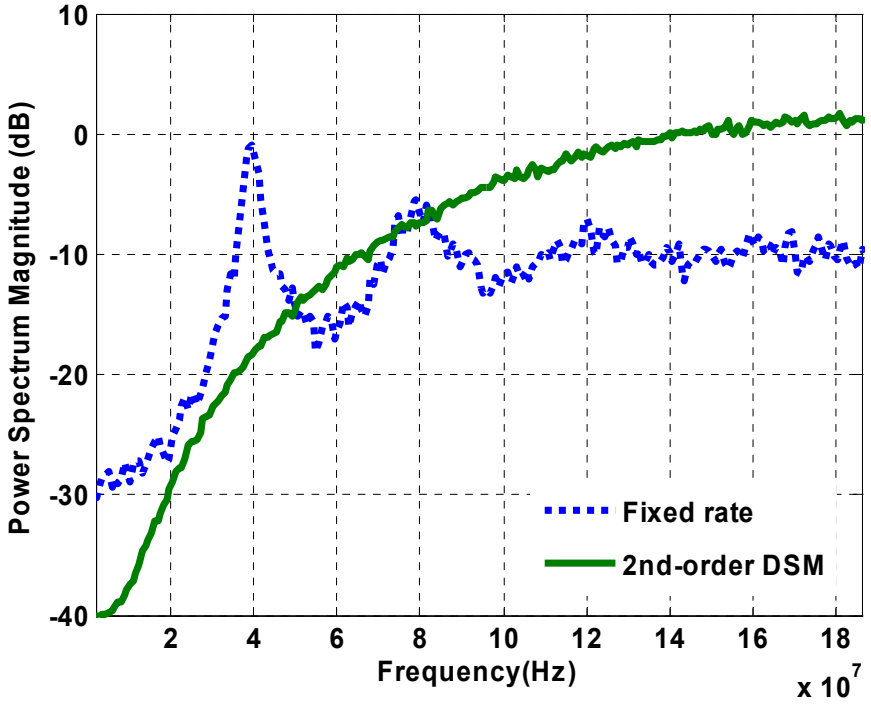


Figure 4.11: Power spectrum of frequency control-induced quantization noise for fixed-rate control and second-order DSM control.

Frequency tracking is achieved by constantly stepping (advancing or retarding) the phase rotator at a nominally fixed-rate or by using a second-order Delta-Sigma ($\Delta\Sigma$) modulated control to cover small frequency differences between *refclk* and *bclk*. In the case of a fixed-rate control, the quantization noise can be approximated as *white* noise, which requires very low bandwidth settings in the MX-PDLL to filter it out. On the other hand, DSM control has the advantage of shaping the quantization noise to higher

frequencies [13]. The simulated power spectrum of the fixed-rate control and the second-order DSM control that are used to synthesize a 400 ppm frequency offset between *refclk* and *bclk* are plotted in Figure 4.11. From this plot, we can see that the shaped DSM

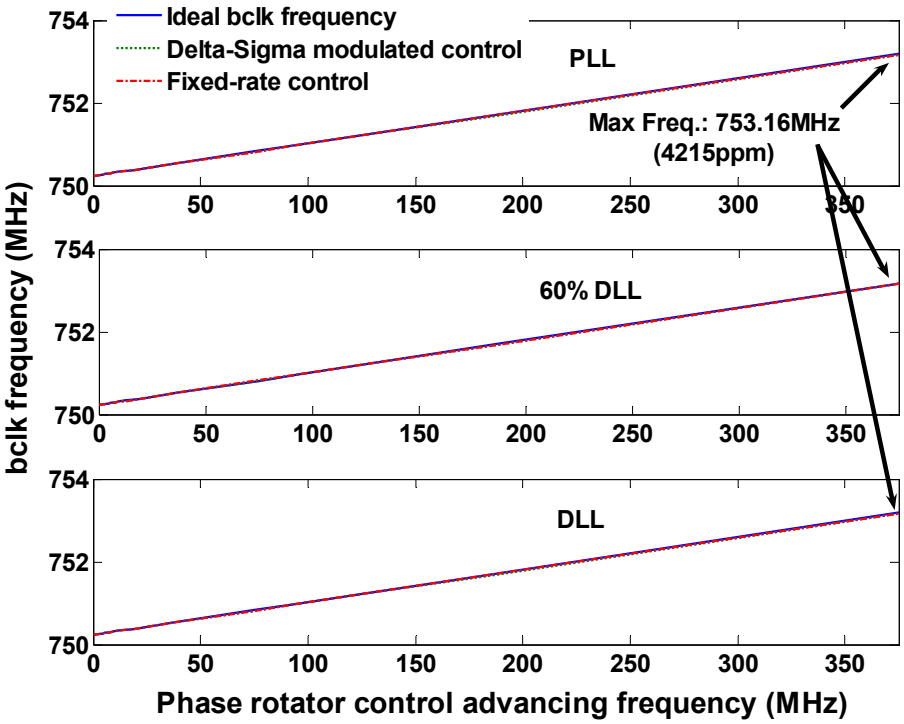


Figure 4.12: Measured phase rotator frequency tracking range for ΔΣ and fixed-rate controls across PLL, 60%DLL, and DLL modes with 128 phases across one reference clock cycle.

control-induced quantization noise can more easily be filtered than the fixed-rate control-induced quantization noise by the core loop in low-bandwidth setting modes. Hence, PLL-mode operation yields lower jitter in the absence of on-chip supply noise. However, as the on-chip supply noise increases, the optimum bandwidth setting again shifts toward DLL-mode operation.

The frequency tracking ranges of the two control schemes are plotted in Figure 4.12. This figure presents the *bclk* frequency vs. the phase rotator control advancing frequency for PLL-, 60% DLL-, and DLL-mode operations with fixed-rate and DSM controls. In all three modes, the maximum achievable *bclk* frequency is 753.16MHz, which equals a 4200 ppm frequency difference from the nominal operating frequency of 750 MHz, with the fastest *refclk* phase updating frequency at 375 MHz. For both control schemes in all three modes, the percentages of differences between the measured output frequency and calculated ideal frequency for given control updating frequencies are smaller than 20 ppm. Both fixed-rate and DSM controls can synthesize an output clock frequency according to the control updating rate with comparably great accuracy across three tested MX-PDLL operation modes.

Figure 4.13 presents the measured output jitter results vs normalized mixing weight for various amounts of on-chip supply noise while tracking a 400 ppm frequency offset. The DSM control has smaller jitter than the fixed-rate control under all supply noise conditions for nearly all of the mixing weight settings. Since the DSM control can shape the low-frequency fixed-rate control to higher frequencies as shown in Figure 4.11, this digital control-induced noise can be filtered out more by the MX-PDLL loop. In the case of the DLL-mode operation, the results from DSM and fixed-rate controls show no difference due to the fact that the DLL mode has a relatively high bandwidth such that the core loop cannot filter out much of the *shaped* quantization noise. Again, PLL-mode operation filters more of the digital control-induced noise and yields lower jitter in the absence of on-chip supply noise. As the supply noise increases, the optimum bandwidth setting shifts toward the DLL-mode operation.

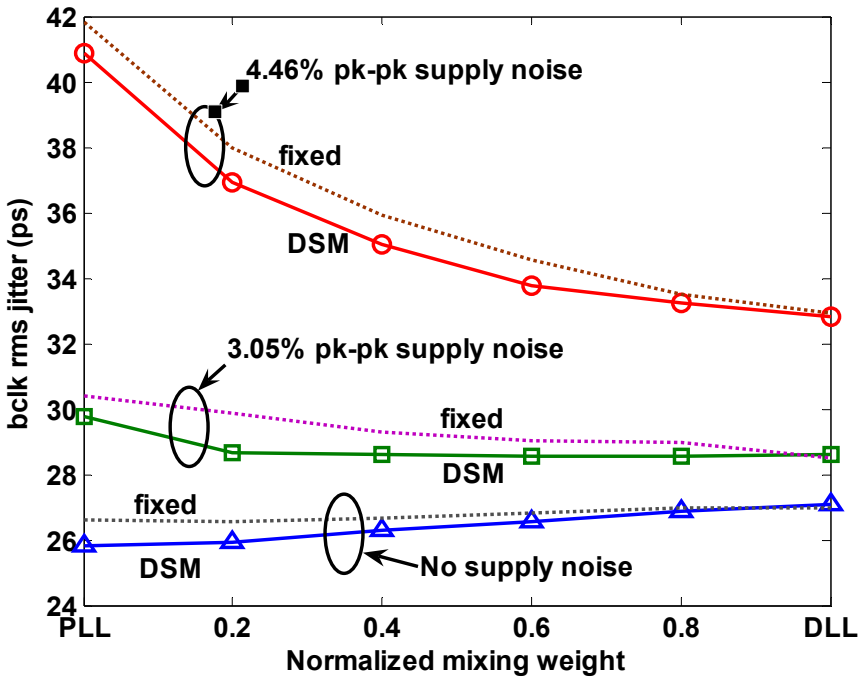


Figure 4.13: Measured *bclk* rms jitter with DSM and fixed controls to track a 400 ppm frequency offset across normalized mixing weight vs. supply noise.

We have verified the ability of the phase rotator to track frequency offset over 4,000 ppm with 2.5 LSB maximum phase step size and ± 5 LSB INL. Although the tracking performance is not affected by different operation modes, the jitter results have shown that second-order Delta-Sigma modulated frequency control leads to lower jitter than the straightforward fixed-rate control in low-bandwidth setting modes due to quantization noise filtering.

In the next section, the impact of the phase rotator's quantization step size on the output clock jitter will be investigated. A smaller number of steps (i.e., larger step size) in the phase rotator is sometimes preferred in order to reduce power consumption and area of the design. Therefore it is worth investigating the performance impact due to the larger phase rotator step size.

4.3.4 Quantization Step Size

Since the quantization step size determines the number of bits required for the phase rotator, finer resolution comes at a cost of higher power consumption and more hardware. The performance with various step sizes is investigated to see if there is some tradeoff between jitter, step size, control schemes, and supply noise across all operation modes.

To investigate the impact of quantization step size on the frequency tracking performance, various numbers of phase steps were tested on the chip. Figure 4.14 plots

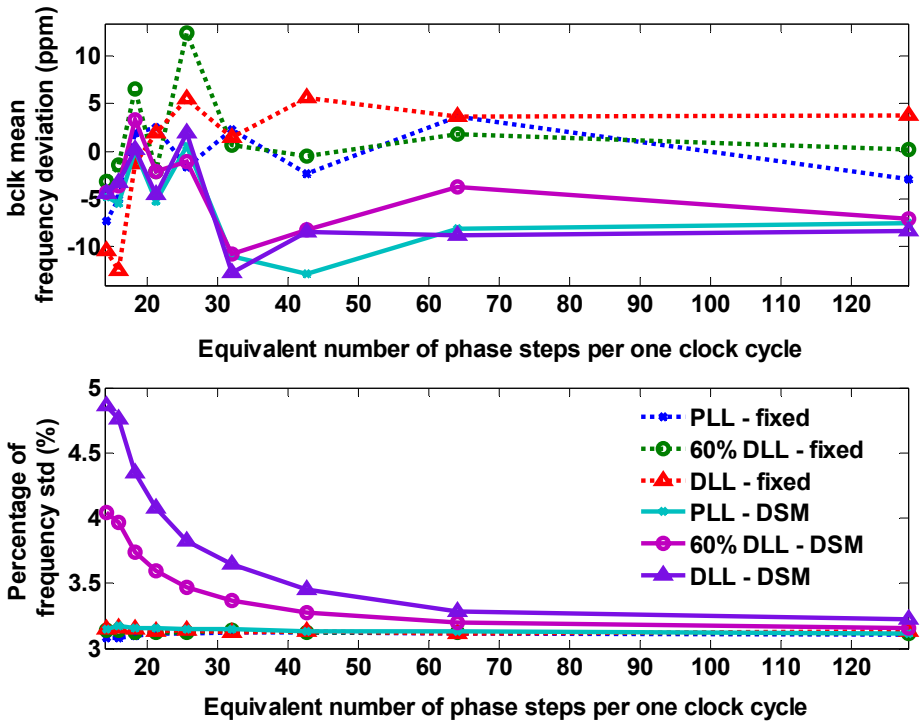


Figure 4.14: Measured phase rotator frequency tracking performance for DSM and fixed-rate controls to track a 400 ppm frequency offset across PLL, 60%DLL, and DLL modes vs. equivalent number of phase steps per one clock cycle.

the measured *bclk* mean frequency deviation from the calculated ideal frequency and frequency standard deviation (std)/mean value while tracking a 400 ppm frequency offset vs. number of phase steps per one clock cycle. Bigger quantization step sizes have very little impact on the mean value of *bclk* frequency. However, they do aggravate the frequency standard deviation especially in the case of DSM control with high-bandwidth settings in the core loop. The number of phase steps in the phase rotator may be reduced to save power and to reduce the required area to achieve the same frequency tracking performance. However, a larger quantization step size can create more output jitter in high-bandwidth setting modes due to larger quantization noise.

Figure 4.15 plots the *bclk* rms jitter for DSM and fixed-rate controls with various phase step sizes, which are 2 LSB, 4 LSB, and 8 LSB, while tracking a 400 ppm frequency offset under different supply noise conditions. Increasing the quantization step size increases the jitter across all operation modes for both control schemes. Moreover, if DSM control is used in low-bandwidth setting modes, it can achieve smaller jitter and reduce the penalty of having larger quantization step sizes. Again, the tradeoff between digital control-induced reference noise filtering and supply noise suppression in terms of the core loop bandwidth setting results in the same outcome as in the prior analyses.

A larger quantization step size in the phase rotator does not significantly impact frequency-tracking performance. The mean value of the *bclk* frequency is within ±15ppm of the desired frequency for all step sizes measured. However, the drawback of a large quantization step size is seen on the output clock jitter. By using DSM control in low-bandwidth setting modes, we can reduce the above quantization-induced jitter due to the quantization noise-shaping characteristic of the DSM control.

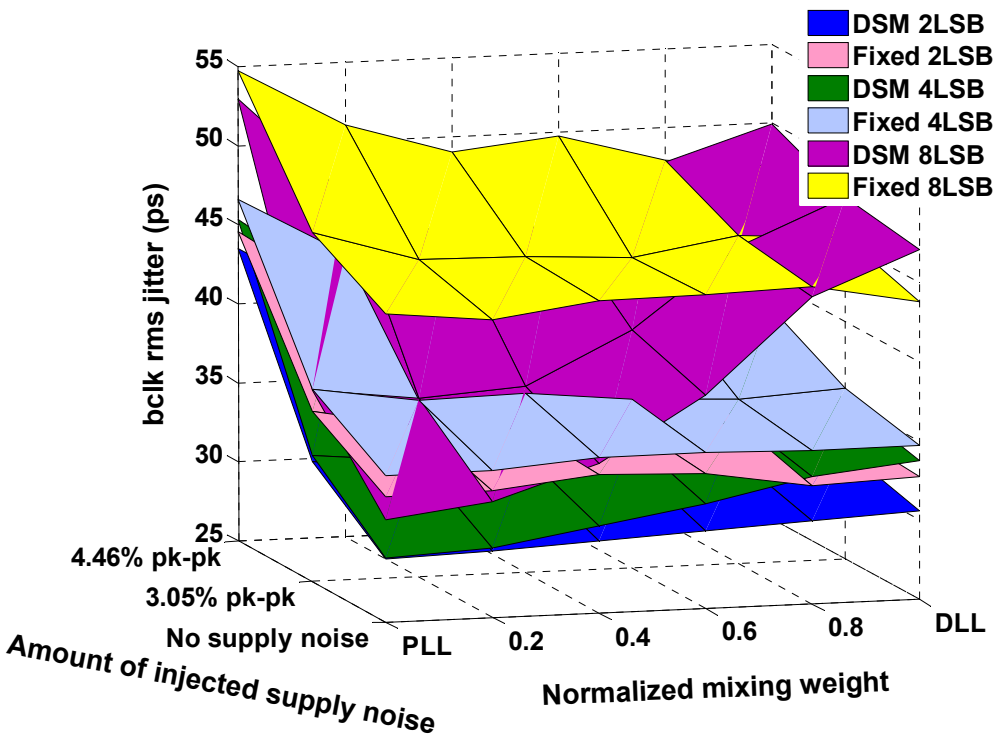


Figure 4.15: Measured *bclk* rms jitter for DSM and fixed rate controls while tracking a 400 ppm frequency offset with different quantization step sizes across normalized mixing weights vs. supply noise.

We have seen the impact of phase rotator's resolution on output clock jitter, but not the impact of the non-ideal phase transfer function of the phase rotator. In the next section, we investigate how non-linearity in the phase rotator can affect system performance.

4.3.5 Code Remapping

The phase rotator's non-linearity as seen in Section 4.2.1 can be compensated for at the software level by remapping the phase control codes. Figure 4.16 plots the remapped phase transfer function of the phase rotator after picking 32 of the original 128

codes and rearranging them to achieve the best linearity. After remapping, the integral non-linearity (INL) is almost zero.

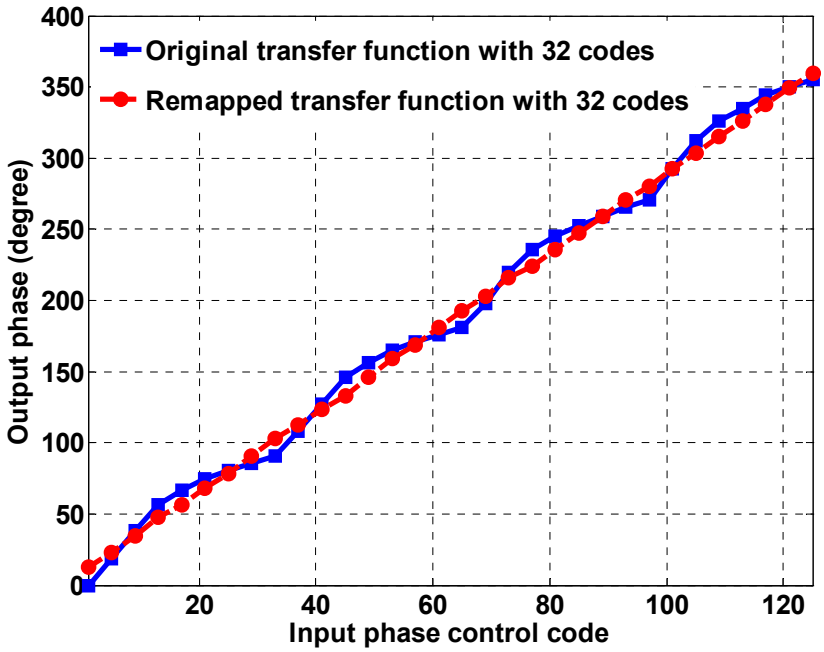


Figure 4.16: Phase rotator transfer function after code remapping with a total of 32 codes.

Figure 4.17 shows a comparison of the measured *bclk* rms jitter between the new linear 32 phase control codes and nonlinear control codes with different resolutions to track a 400 ppm frequency offset using second-order DSM frequency synthesis control. The general trend of each surface is similar to what we have seen before with injected supply noise and frequency synthesis control. As we increase the amount of on-chip supply noise, the low-bandwidth setting modes suffer from jitter accumulation. With 32 remapped codes, the output jitter is the smallest among other code options, which all possess non-linearity despite different quantization step sizes. Moreover, in the low-bandwidth setting modes the output clock jitter with new linear codes shows a larger jitter

difference from other jitter results with nonlinear codes than that in the high-bandwidth setting modes. In the low-bandwidth setting modes, the noise due to non-linearity is more

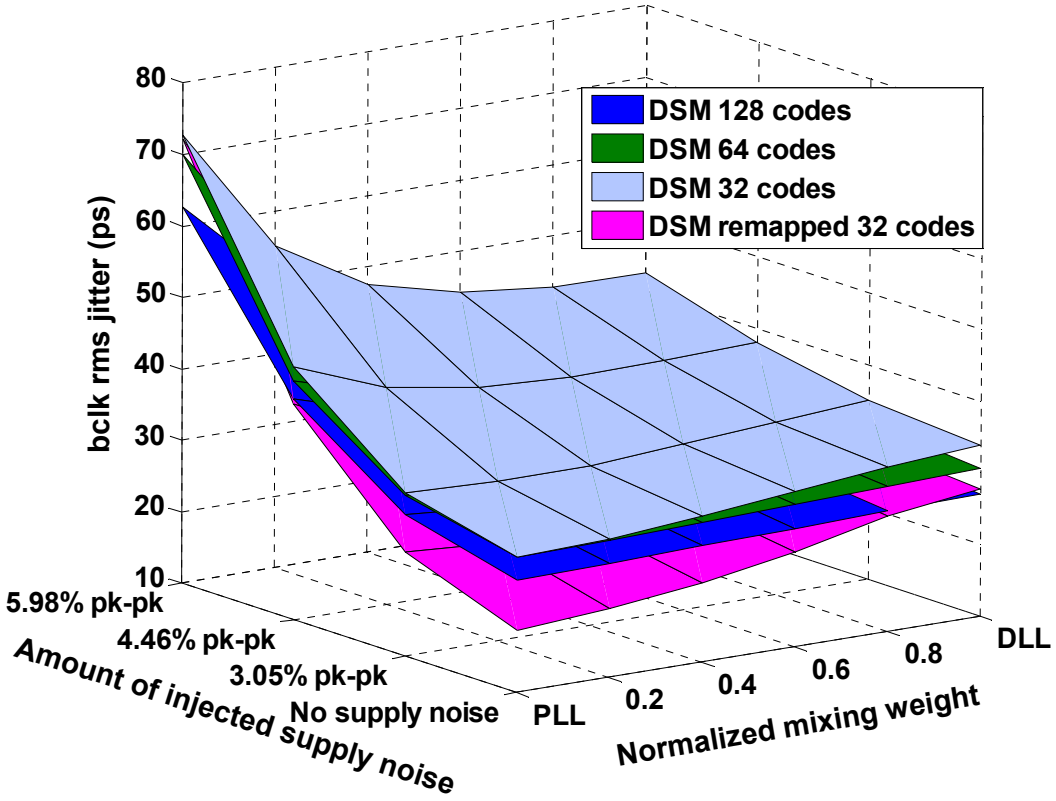


Figure 4.17: Measured *bclk* rms jitter for DSM control to track a 400 ppm frequency offset with remapped codes and other code options across normalized mixing weight vs. supply noise.

pronounced than the resolution-induced noise. But as we shift toward the high-bandwidth setting modes, the resolution-induced noise becomes more pronounced than the noise due to non-linearity. In DLL mode, the original 128 nonlinear codes have smaller output jitter than the 32 linear codes. With the same linearity, higher resolution always helps reduce the output clock jitter, especially in the high-bandwidth setting modes because the high-

bandwidth cannot attenuate the larger step size, therefore the resolution-induced noise dominates.

The proposed CDR architecture configured as a frequency synthesizer is much more sensitive to phase rotator non-linearity. The measured results show that improved linearity has a significant impact on reducing output jitter even with larger quantization step sizes. If the CDR data tracking loop is closed, the feedback loop should be able to *linearize* some of the INL and reduce jitter. The reduction of jitter will depend on the closed-loop bandwidth of the CDR system. If the closed-loop bandwidth is high, the feedback loop will be able to correct the non-linearity by selecting the appropriate codes quickly, and therefore minimize the accumulated error.

4.4 Summary

In this chapter, a dual-loop CDR architecture for a time-interleaved receiver has been presented. The dual-loop CDR utilizes the MX-PDLL as the core clock generator, which enables several modes of operation to accommodate a variety of noise conditions in order to minimize output clock jitter. The CDR then uses a digitally-controlled phase rotator, which rotates the reference clock that feeds into the MX-PDLL to achieve phase and frequency tracking for plesiochronous interfaces. The prototype chip utilizes an external digital controller to drive the phase rotator with a variety of control options in order to more easily investigate the merits of utilizing a MX-PDLL in the proposed CDR architecture. This architecture was investigated in an open-loop configuration as a frequency synthesizer. The test chip was fabricated in the TSMC 0.18 μm CMOS logic

process. Table 4.1 summarizes the chip performance, and Figure 4.5 presents the micrograph of the test chip.

There is a tradeoff between filtering digital control-induced quantization noise and suppressing on-chip supply noise in the CDR. By taking advantage of tuning the loop bandwidth of the MX-PDLL, we can minimize the output clock jitter under various noise conditions. The measurement results verify the above claim when the bang-bang limit-cycle phase dithering digital control and frequency tracking controls (fixed-rate and second-order Delta-Sigma modulated) are applied to the phase rotator.

The measured frequency tracking results show that Delta-Sigma modulated (DSM) frequency control renders lower output jitter than a straightforward fixed-rate control in low-bandwidth setting modes due to the quantization noise-shaping characteristic of DSM. Moreover, a larger quantization step size increases the output jitter for both controls, but it has a smaller impact on DSM control in low-bandwidth setting modes due to the quantization noise filtering of the MX-PDLL.

As a frequency synthesizer, the results also show that non-linearity of the phase rotator can aggravate the output jitter performance significantly. Although in the closed-loop CDR operation, the non-linearity of the phase rotator might be mitigated by the feedback loop gain, it is still important to have good linearity when tracking a large frequency offset.

So far, we have been looking only at the dynamic clock uncertainty, known as random jitter when we have compared the performance between different operation modes, noise conditions, and control schemes. We have not talked about static type clock uncertainty, or static phase spacing mismatch among the multiphase clocks. For example,

we have seen previously that the CDR uses a divide-by-2 block with a duty-cycle corrector to generate four 90° phase-shifted clocks for the phase rotator. The next chapter describes various techniques that can be used to minimize static clock uncertainty and explains why they are important.

Chapter 5

Static Phase Mismatch Detection and Compensation

We usually call the 'static' type clock timing uncertainty, static phase mismatch. What would happen if static phase mismatch existed in a CDR? If a multiphase clock generator is used in a CDR for a time-interleaved receiver, static phase spacing mismatch among the clock phases would further degrade the timing margin of the receiver. One can think about the static phase mismatch as the static "skew" of the phase error, and dynamic jitter as random "distribution" of the phase error. The good news is that since this skew is for the most part static, it can be calibrated using a phase mismatch detector and several on-chip compensation schemes. Before these different static phase mismatch compensation schemes are introduced, some of the main static phase mismatch sources in a CDR system are discussed.

In order to understand from where the static phase mismatch originates, let us look at the block diagram of the MX-PDLL-based multiphase clock generator again as shown in Figure 5.1. One of the static phase mismatch sources is the static phase offset between the reference clock and the output clock due to signal path and device mismatches in the MX-PDLL. For example, we may have delay mismatch through the PFD and current mismatch in the CP. The mismatch in PFD and differential CP can lead to static phase offset between the input reference clock (*rclk*) and the output clock (*bclk*)

of the MX-PDLL when the loop is locked, which can cause uneven phase spacing in the MX-PDLL if it is in DLL-mode operation. In order to calibrate this phase mismatch, a charge pump-based compensator is proposed in this chapter.

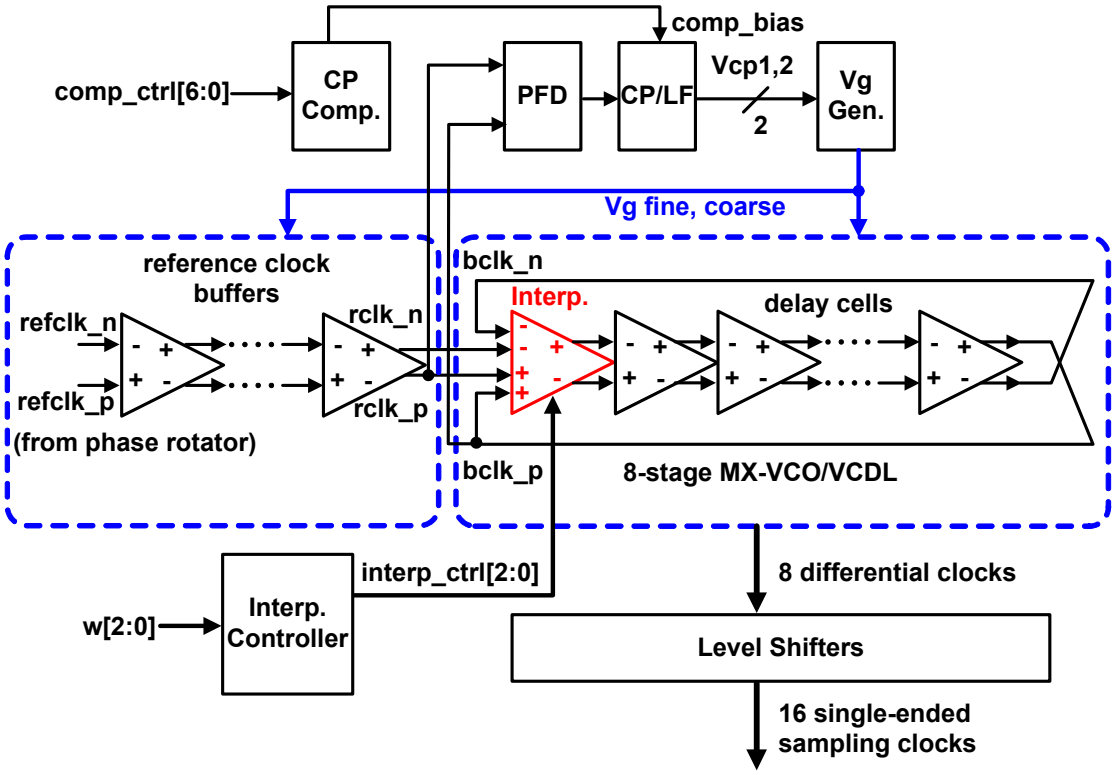


Figure 5.1: Block diagram of the MX-PDLL-based multiphase clock generator.

Although in PLL-mode operation the above static phase offset does not cause delay mismatch between the delay cells when configured as a VCO. Process variations and systematic imbalances in the physical design can also introduce delay mismatch between the delay cells when configured either as a VCO or VCDL. To mitigate the uneven phase spacing in the MX-PDLL caused by process variations and systematic imbalances in the physical design, we can use a passive resistor-ring-based phase-

averaging network (PAN) [8] in between the MX-PDLL and samplers. The PAN can "smear" the voltage transitions to help reduce the phase errors caused by mismatch along different clock paths.

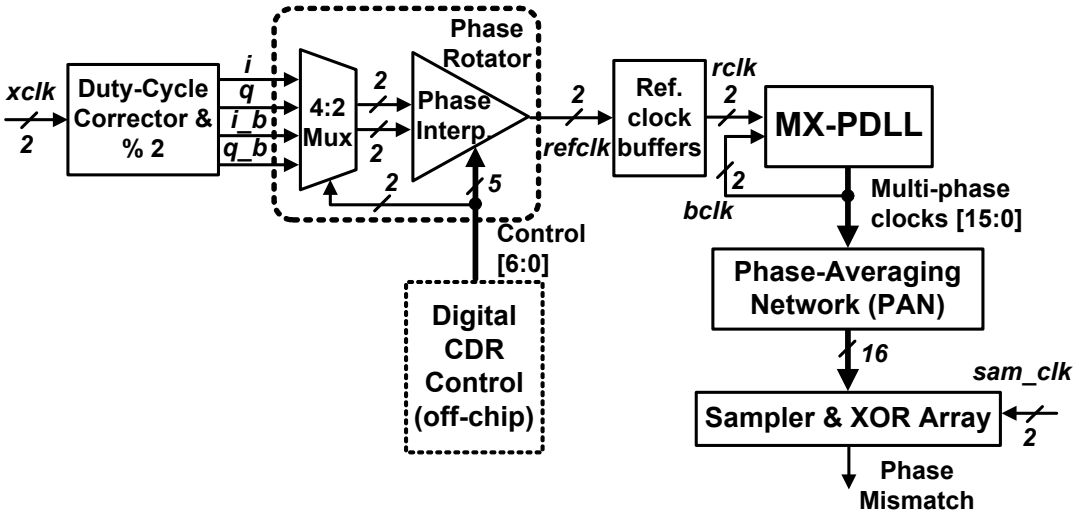


Figure 5.2: Block diagram of the prototype system built and tested.

Before we can calibrate the static phase mismatch, we first need a method to detect how big the mismatch is. Therefore, an on-chip static multiphase mismatch detector to measure the static phase offsets is presented. The goal is to track the edge transition of an external clock (*sam_clk*), which has a frequency that is very close (but not equal) to the sampling clock frequency, by subsampling it with multiphase clocks. By measuring the duration of time in which the edge transition occurs in between two sampling clock phases, the phase difference between the two clock phases can be derived with high precision.

Now let us look at the overall CDR architecture shown in Figure 5.2, and identify the other potential mismatch sources in the system. First let us assume *xclk* does not have

a 50% duty-cycle, which would be the quadrant spacing between i, q, i_b, and q_b clocks. Then the quadrant spacing will not be exactly 90 degrees. Therefore the phase step sizes in the phase rotator will be different among the four quadrants. We have seen in Chapter 4 that phase rotator step size also affects output jitter. Since the duty-cycle mismatch on *xclk* can lead to phase spacing mismatches among the four 90-degree phase-shifted clocks, a duty-cycle correction (DCC) circuit, which works in conjunction with the divide-by-2 circuit, is presented to minimize this quadrant phase mismatch.

5.1 Static Phase Mismatch Detector

Since we need a method to detect static phase spacing mismatch among the clocks before we can calibrate the mismatch, a static phase mismatch detector is introduced in this section first. The detector actually uses the data/edge samplers to sub-sample an external clock, and then XOR the outputs of two adjacent samplers. Therefore it consists of two parts, a data sampler array and a phase detection XOR array.

5.1.1 Data Sampler

A double-sampling latch-based sampler is shown in Figure 5.3. Double-sampling relaxes the gain requirement for each stage and enables the use of smaller input differential pairs to increase the speed of the sampler. Lastly, the sampled input is latched by a SR latch following the samplers. A single sampling stage is a modified StrongArm latch that was proposed in [10]. Five *clk*-controlled PMOS devices are used for pre-

charging the outputs and internal nodes. During the evaluation phase (*clk* is high), the input differential pair senses the difference between *inp* and *inn* and the cross-coupled inverters amplify this difference. The second sampler evaluates the input when the first sampler is in the pre-charging phase. The 16 sampler outputs (*sa1, sa2, ..., sa16*) are sent to the phase-detection XOR array, which is presented in the next section, for calibration and measurement purposes.

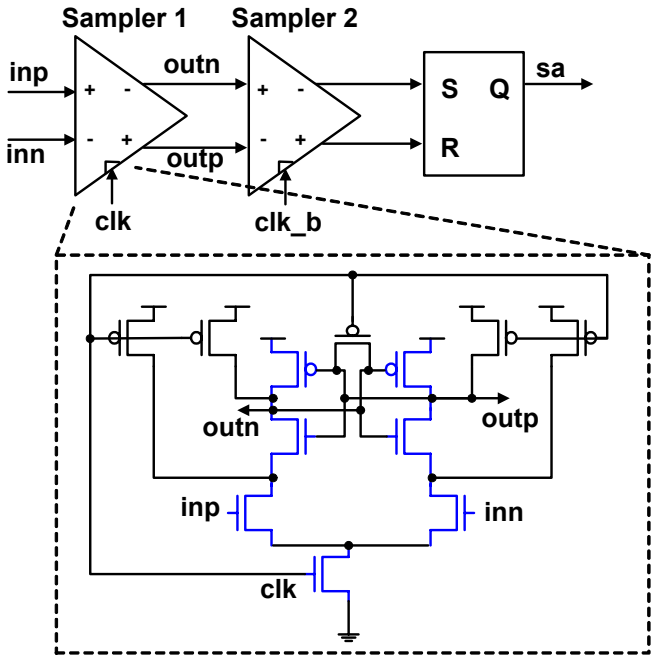


Figure 5.3: Schematic of the data sampler.

5.1.2 Phase Detection XOR Array

In order to facilitate mismatch compensation, an XOR-based phase detection circuitry as shown in Figure 5.4 is proposed. The basic idea is to measure phase

mismatch by sampling an external clock signal (*sam_clk*), which has a frequency that is very close (but not equal) to sampling clock frequency. The *sam_clk* edge transitions (both rising and falling) can be determined by taking two adjacent sampler outputs and

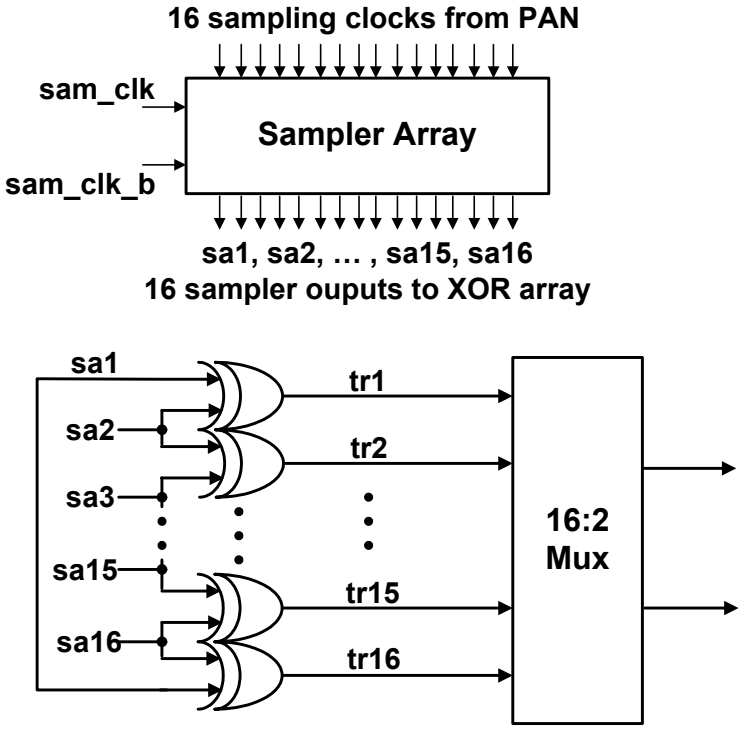


Figure 5.4: Schematic of the XOR-based phase detection circuit.

feeding them into an XOR gate. This is essentially a simple implementation of a bang-bang phase detector (!!PD). The pulse duration of the XOR output (*tr1*/…/*tr16*) is proportional to the phase difference between two adjacent sampling clocks. Since we can choose *sam_clk* frequency very close to the sampling clock frequency, the pulse duration can be made long enough to provide very high measuring resolution. A 16:2 multiplexer (mux) selects any two of the 16 pulses at one time to reduce I/O pin count.

Given the ability to detect the static phase mismatch, we can calibrate the static phase mismatch between *rclk* and *bclk* with the CP compensator, which is presented in Section 5.2. This detector can also measure the phase spacing mismatch reduction performance of PAN that is described in Section 5.3.

5.2 Charge Pump Compensator

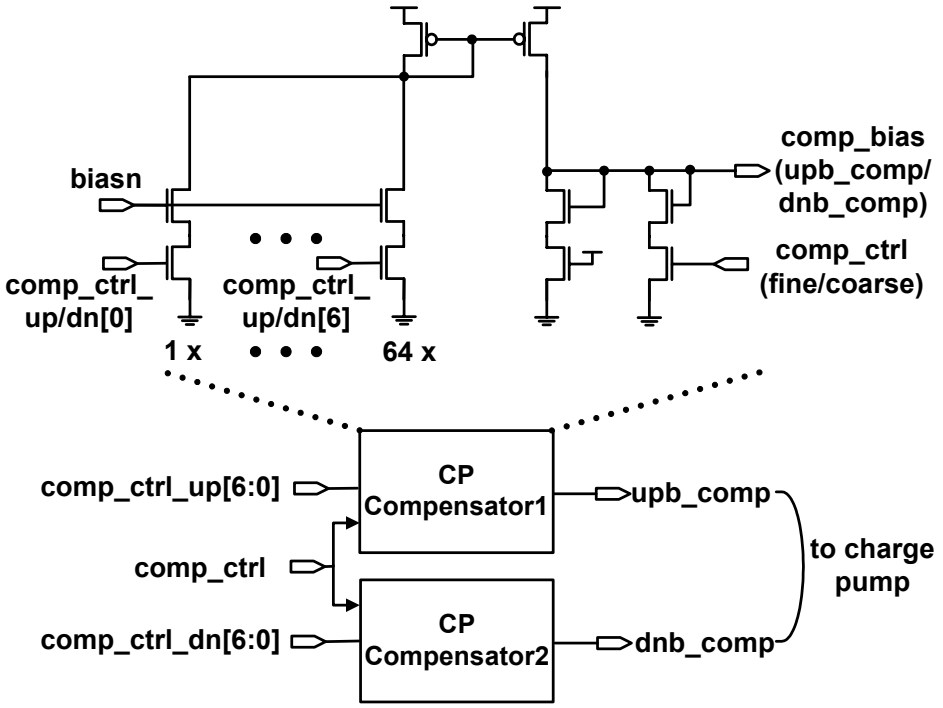


Figure 5.5: Schematic of the CP compensator.

One static phase mismatch source is the static phase offset between *rclk* and *bclk*. This static phase offset can manifest as uneven phase spacing between the multiphase clocks out of a multiphase clock generator. Consider that the MX-PDLL operates as a

DLL. If there is a phase offset at the DLL input (between *rclk* and *bclk*) while in lock, a phase spacing mismatch between the *bclk* and the output of the first delay cell will exist in the MX-VCO/VCDL because of a fresh *rclk* edge that drives the VCDL for every reference cycle. As a result, this static phase offset can lead to uneven phase spacing in the multiphase clock generator, when operating in DLL mode. However we do not have this concern for the PLL-mode operation. A charge pump-based compensation scheme is proposed to correct this offset. The CP compensator can purposely skew the differential charge pump currents to compensate for the static phase offset between *rclk* and *bclk*.

The charge pump compensator involves a pair of 7-bit DACs (Digital-to-Analog Converter) to generate the compensation bias currents that feed into the CP through gate control voltages, *upb_comp* and *dnb_comp* as shown in Figure 5.5. Also the schematic of the differential CP is shown again in Figure 5.6 in order to demonstrate how the circuit operates. A control signal, *comp_ctrl*, is used to set the resolution of the digitally-controlled output. When *comp_ctrl* is set high, the output voltage has a higher resolution but a smaller compensation range. Two charge pump compensators are used to generate *upb_comp* and *dnb_comp* separately according to the 7 control bits. The differential CP has two auxiliary NMOS current sinking devices that are controlled by CP compensator in addition to the main pull-down current paths that are controlled by Common-Mode Feedback (CMFB). Therefore, the differential CP current output can be skewed via digital compensation codes to cancel the static phase offset between *rclk* and *bclk*.

The CP compensator can remove the large phase spacing mismatch between *bclk* and the output of the first delay cell in the MX-VCO/VCDL. However, there is still delay mismatch between the delay cells caused by process variations and systematic

imbalances in the physical design in both the PLL and DLL modes. A passive phase-averaging network can be used to mitigate the residual mismatch after removing the big phase spacing mismatch in the first delay cell.

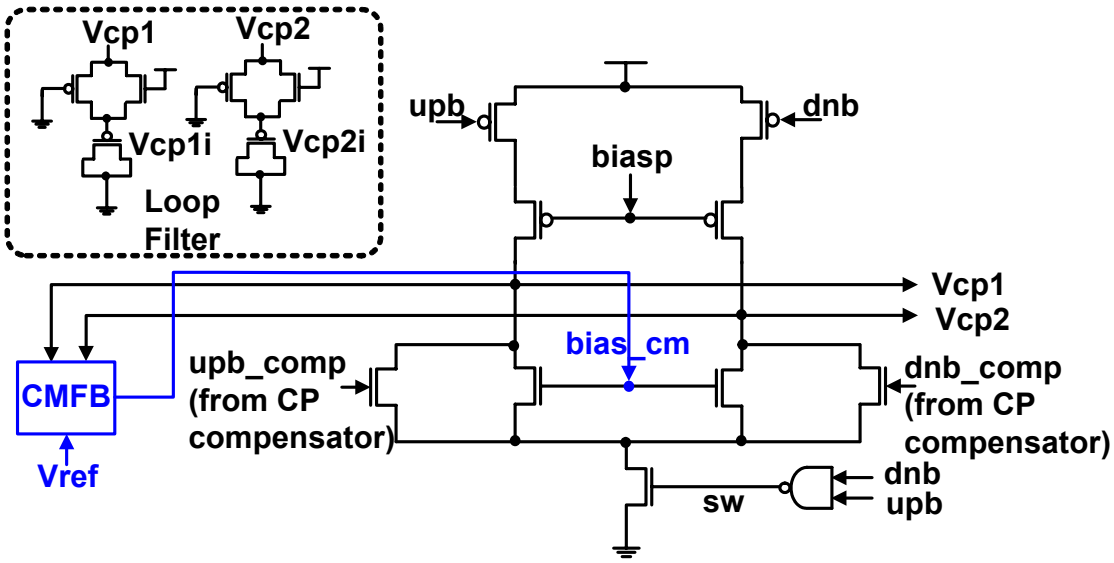


Figure 5.6: Schematic of the differential CP.

5.3 Passive Phase-Averaging Network

To mitigate uneven phase spacing among multiphase clocks in the MX-PDLL, a PAN as presented in Figure 5.7 is used in this prototype chip. The 16 clock phases out of the MX-PDLL are uniformly connected to two layers of interconnected resistors (R-ring) [8]. The R-ring has the benefit of smearing or averaging the voltage transitions, which reduces the phase errors caused by mismatch along different clock paths. Phase averaging is achieved through the RC low-pass filtering between all of the clock phases, reducing

phase spacing offsets. The resistor is implemented using the transmission gate, which can be enabled by two complementary signals, *ph_ctrl* and *ph_ctrl_b*. The filtering capacitor comes from the parasitic capacitor at the internal nodes.

Sizing the transmission gate can be tricky. If the transistors are too small, the corresponding resistance is too large to provide sufficient averaging such that the reduction of phase spacing offsets is negligible. On the other hand, if the transmission

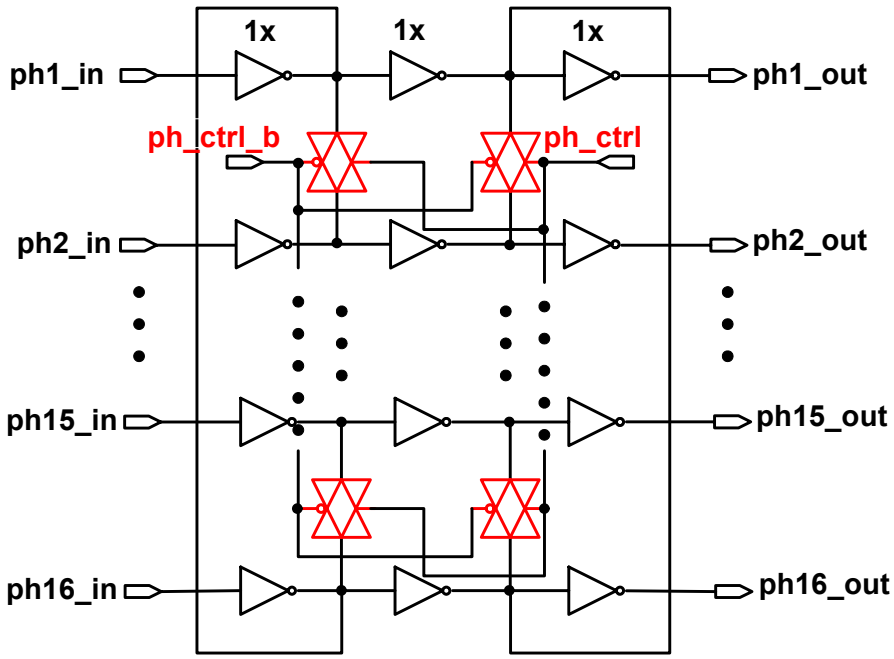


Figure 5.7: Schematic of phase-averaging network.

gate sizes are too large, the corresponding small resistance causes too much active current significantly reducing the voltage swing and increasing power consumption. Hence there is a tradeoff between the amount of reduction in phase spacing mismatch and power consumption with respect to sizing of the transmission gates. Figure 5.8 plots the ratio of

phase mismatch reduction over power consumption vs. transistor size for three different P/N ratios. A P/N ratio of 3 was chosen to achieve the maximum reduction rate without significant power compromises.

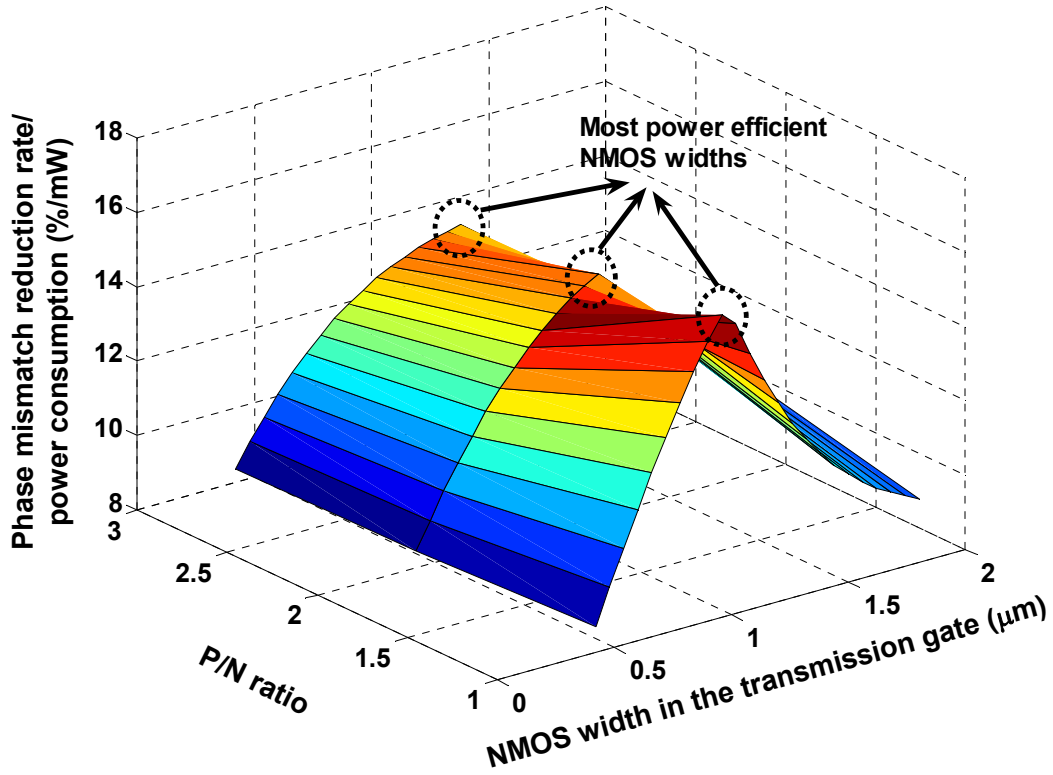


Figure 5.8: Power efficiency vs. transmission gate sizing.

A resistor-ring-based PAN can reduce the uneven phase spacing offsets caused by device mismatch and systematic imbalances in the physical design along different clock paths. The 16 averaged clock outputs after PAN can be used to clock the parallel data samplers to retrieve the data and edge information from the incoming data. In this prototype chip, the 16 sampling clocks are used to sample an external clock to provide static phase mismatch information in the phase mismatch detector.

5.4 Charge Pump Compensator and Phase-Averaging Network Measurements Results

The static phase offset between *bclk* and *rclk* caused by delay mismatch through the PFD and current mismatch in the differential CP can be compensated by purposely

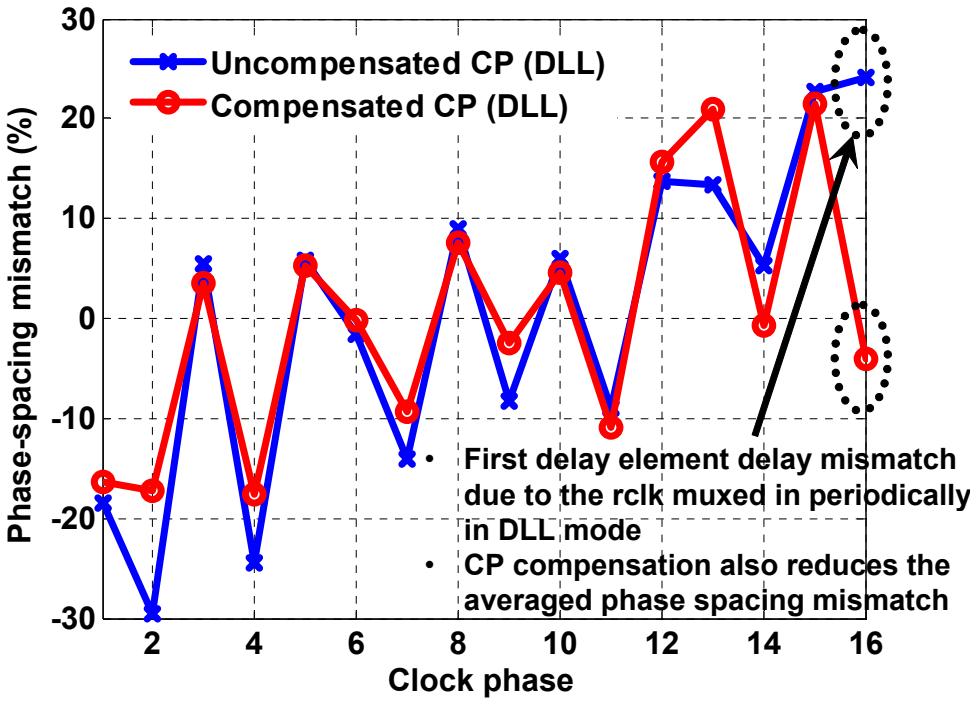


Figure 5.9: Measured static phase spacing mismatch in DLL mode at 750MHz.

skewing the CP currents. The static phase mismatch is particularly heinous in DLL mode due to the fresh *rclk* edge that drives the VCDL every reference cycle. As a result, a phase spacing mismatch between the *bclk* and the output of the first delay cell in the MX-VCO/VCDL will exist. This is exactly what shows in Figure 5.9. The uncompensated CP

in DLL mode shows a large delay mismatch of the first delay cell (clock phase 16) at 750MHz due to the static phase offset issue previously discussed. After CP compensation, this big delay difference is removed, and the overall phase spacing

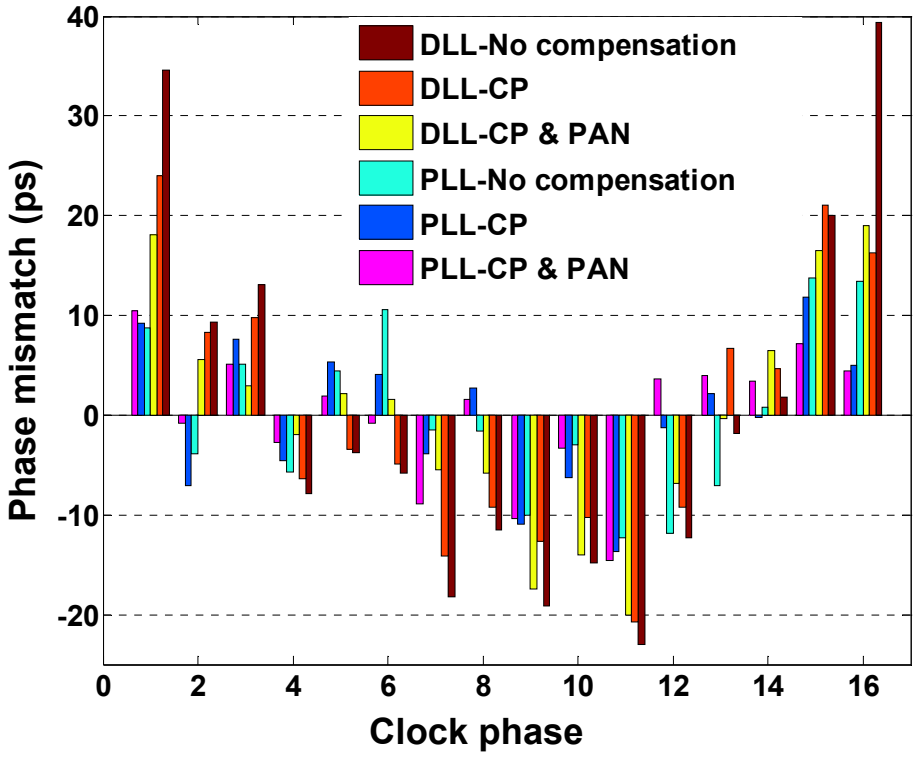


Figure 5.10: Measured CP compensator and PAN static phase spacing mismatch results in PLL and DLL modes at 750MHz (ideal phase spacing is 83.33ps).

mismatch is also slightly reduced. This measurement was made by using the XOR-based phase detector to measure the relative phase spacing in the MX-PDLL by sampling a clock (*sam_clk*) that has a frequency very close to the *bclk* frequency. The high duration of phase detector output is comprised of over 3,000 sampling clock edges, which provide very high transition resolution.

Next, Figure 5.10 presents the benefits of using the PAN along with the CP compensator. CP compensation can remove the big phase mismatch at clock phase 16 and therefore reduces the standard deviation (std) of phase spacing mismatch among all 16 clock phases from 18.72ps (22.464% of ideal phase spacing) to 13.31ps (15.972% of ideal phase spacing) when operating in DLL mode. PAN further helps reduce the standard deviation from 13.31ps to 11.74ps (14.088% of ideal phase spacing) when operating in DLL mode and from 7.28ps (8.736% of ideal phase spacing) to 6.71ps (8.052% of ideal phase spacing) in PLL mode.

Precise generation of evenly spaced clocks is needed for high-performance, high-speed links. The static delay mismatch between multiphase sampling clocks caused by static phase offset between *rclk* and *bclk* and mismatches along different clock paths can reduce the timing margin of the time-interleaved receiver; thus it can also degrade the link performance. Measured results show that by using an auxiliary charge pump compensator, over 28% reduction of phase spacing mismatch can be achieved. The combined reduction by using both the CP compensator and simple passive resistor-ring-based PAN is over 37%.

5.5 Duty-Cycle Corrector

A duty-cycle correction (DCC) circuit is implemented in conjunction with the divide-by-2 circuit (÷ 2) to compensate for any duty-cycle mismatch on *xclk* to create evenly spaced quadrature clock phases (*i*, *q*, *i_b*, and *q_b*). Figure 5.11 shows the DCC block, which consists of a duty-cycle tuning block (DCC tune) that drives a ÷ 2 block,

which is a simple modulus-2 counter. A set of XOR and XNOR gates generates output pulses that correspond to the phase spacing between *i* and *q*, and between *q* and *i_b*. A passive RC circuit filters the pulses to generate a pair of pseudo-differential duty-cycle

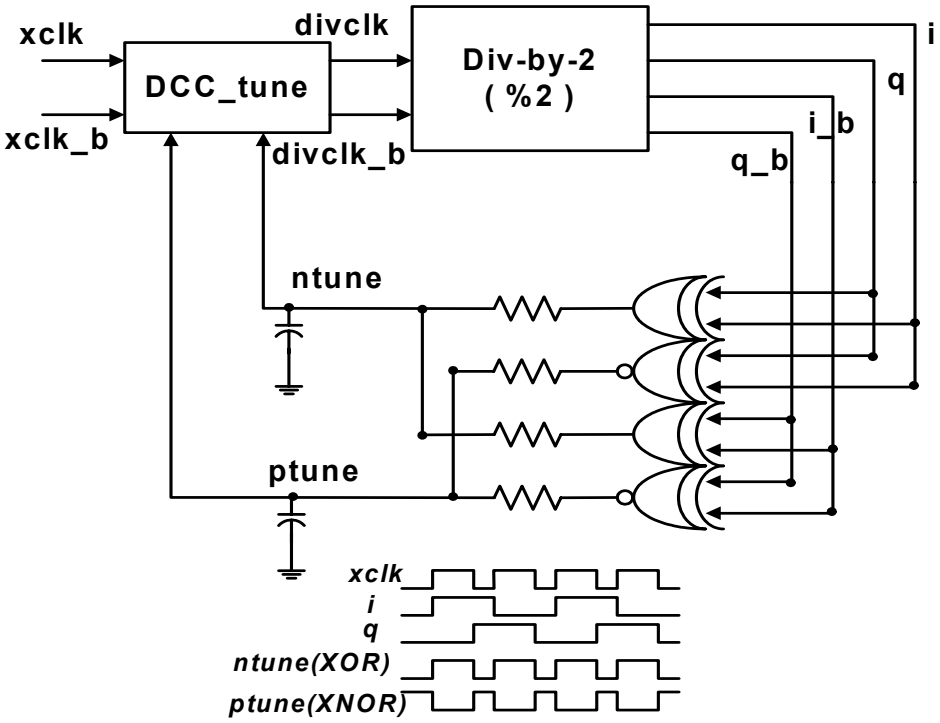


Figure 5.11: Duty-cycle corrector circuit block diagram.

tuning signals, *ptune* and *ntune*. Through negative feedback, the tuning signals are fed into DCC_tune block to compensate for any duty-cycle mismatch on *xclk*. Therefore the ÷ 2 block outputs are evenly spaced quadrature clock phases. In the RC filter, a small capacitor can be used by implementing a large resistor to reduce layout area. The drawback of this simple phase detection and filter circuitry is the small non-zero static-state offset resulting from a finite DC gain.

The DCC_tune block consists of two tuning stages with four nMOS tuning devices, shown in Figure 5.12. The nMOS tuning devices can delay either the rising or falling transition of *xclk* according to *ptune* and *ntune*. Since only nMOS devices are

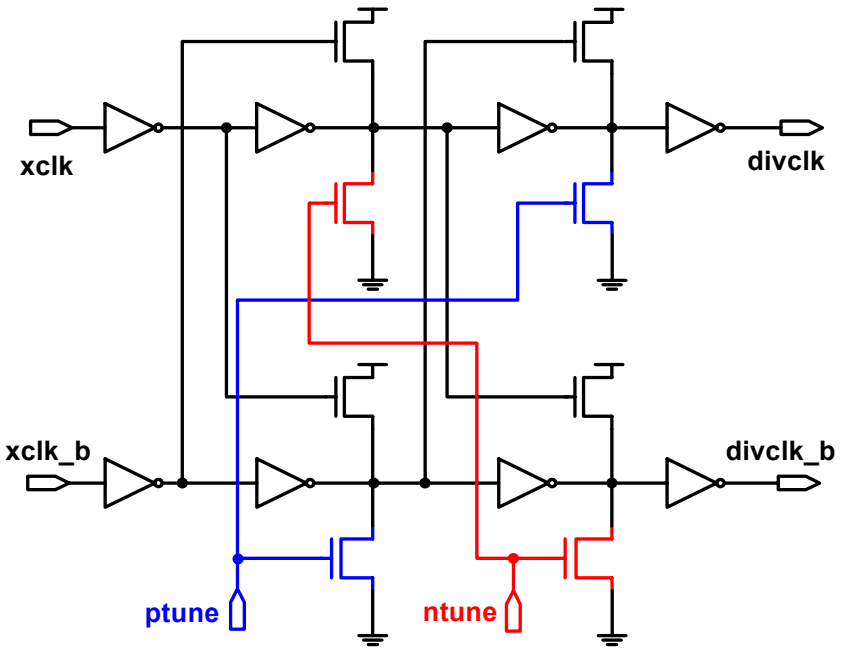


Figure 5.12: Schematic of DCC_tune.

controlled by the tuning signal, only the rising edges at the internal nodes are delayed. The four additional feed-forward NMOS pull-up devices are added to balance the discharging currents from the tuning devices and also to provide some duty-cycle correction. When the DCC feedback loop is locked, *ptune* and *ntune* settle to some values which guarantee that the phase difference between *i* and *q* (detected by XOR) and the phase difference between *q* and *i_b* (detected by XNOR) are identical. In fact, this corresponds to a 50% duty-cycle on *divclk*.

The divide-by-2 circuit is a simple D Flip-Flop (DFF) with an inverted output that feeds back to the input. Two DFFs with control circuitry to properly start up the division are clocked by *divclk* and *divclk_b* respectively to generate *i*/*i_b* and *q*/*q_b* outputs.

This section presents a duty-cycle corrector using a simple negative feedback loop to compensate for the duty-cycle mismatch on the external reference clock to guarantee evenly spaced quadrature clock phases for the phase rotator. The measurement results of the duty-cycle correction block for generating evenly spaced quadrature clock phases are presented in the next section.

5.6 Duty-Cycle Corrector Measurement Results

Any duty-cycle mismatch on *xclk* will result in quadrant spacing mismatch in the phase rotator and further aggravate linearity (i.e., DNL and INL). Therefore DCC performance is particularly critical in order to minimize receiver clock jitter in the CDR while it is tracking phase and frequency of the data according to the conclusions drawn in Chapter 4.

Figure 5.13 plots the measured phase range out of the phase rotator for quadrants I and II given a 36%-duty-cycle *xclk* at the input. The DCC reduces the quadrant spacing mismatch from ±1.59% to ±0.1%. Notice that the 36%-duty-cycle was the reading from the instrument and might not be the actual duty-cycle of *xclk* on chip. In fact, as we see from the measured quadrant spacing mismatch results, it was clearly not 36%. Figure 5.14 plots the measured quadrant spacing with respect to the *xclk* duty-cycle swept from

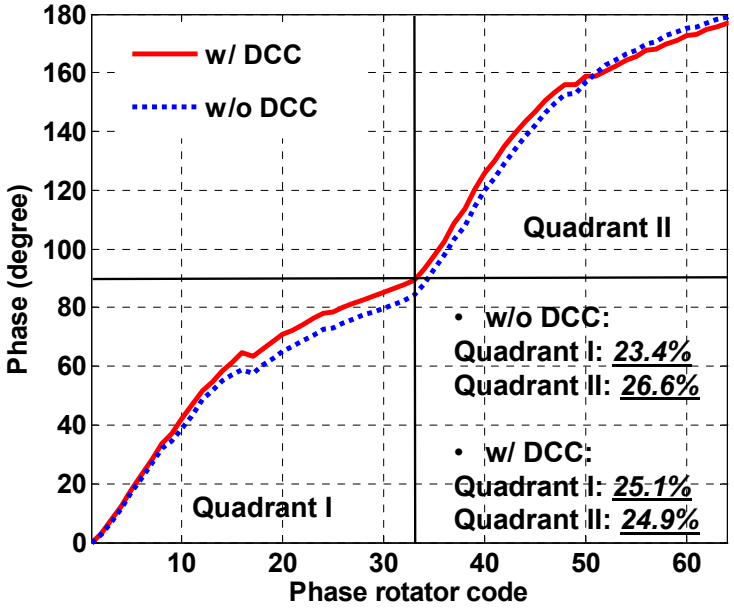


Figure 5.13: Measured phase range out of the phase rotator for quadrants I and II w/ DCC and w/o DCC.

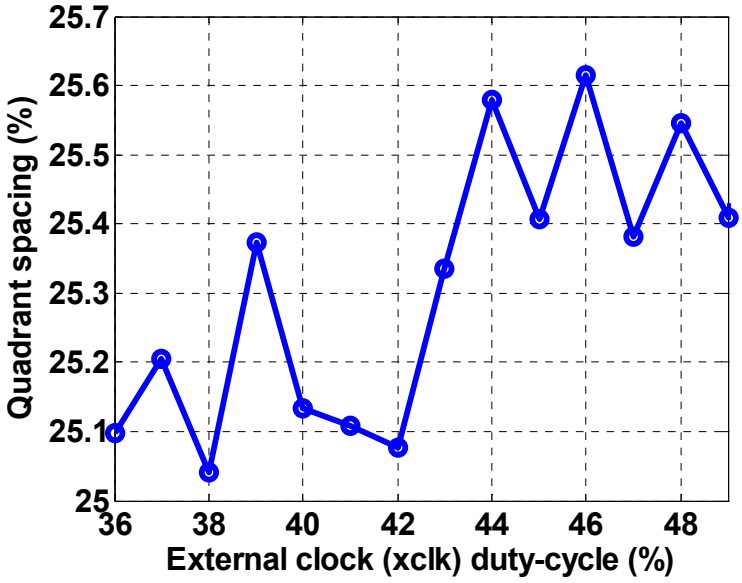


Figure 5.14: Measure quadrant spacing vs. *xclk* duty-cycle.

36% to 49%. The residual mismatch is less than 0.65% of the desired 25% spacing. Notice again that a 50%-duty-cycle *xclk* from the pulse generator does not guarantee a 50%-duty-cycle differential reference clock on-chip.

5.7 Summary

In this chapter, two static phase mismatch compensation schemes are presented to generate evenly spaced MX-PDLL multiphase clock outputs. The CP compensator can remove the static phase offset between *bclk* and *rclk*; therefore, it reduces the large phase spacing mismatch between *bclk* and the output of the first delay cell in the MX-VCO/VCDL in the DLL-mode operation. The passive PAN then further reduces the residual phase mismatch among all clock phases in both PLL and DLL-mode operations at a cost of higher power consumption and potential device mismatch due to extra buffers.

Then a divide-by-2 block with duty-cycle correction function is presented to guarantee evenly spaced quadrature clocks for the following phase rotator even if the input reference clock, *xclk*, does not have a 50% duty-cycle. The evenly-spaced quadrants in the phase rotator ensure that the differential non-linearity of the phase rotator will not be further aggravated.

These three compensation circuitry provide very important functions to make sure that the average positions of the sampling clocks are as close to their ideal positions as possible in spite of various mismatch sources.

Chapter 6

Conclusions

The demand for a low-power, low-cost, and small-footprint high-speed link design (both serial and parallel) from applications for communication equipments, consumer electronics, and computing systems is increasingly stronger. The performance of the link system greatly depends on how well the "jitter" is controlled. Therefore a noise-robust design becomes extremely useful to be used in today's highly-integrated mixed-mode System on Chip (SoC) and System in Package (SiP). Moreover the receiver is usually noisier than the transmitter, because it has to integrate more circuit blocks. A noise-robust receiver and clock and data recovery (CDR) macros are desirable for use in different applications across various noise conditions. This dissertation presents an adaptive-bandwidth mixed phase-locked loop (PLL)/delay-locked loop (DLL)–based multiphase clock generator to achieve the best timing accuracy across various noise conditions by tuning the clock generator's bandwidth. Then a noise-robust clock and data recovery (CDR) architecture incorporating this mixed PLL/DLL (MX-PDLL) is proposed. This CDR architecture has the flexibility to adapt the bandwidth of its clock generator under different noise conditions in order to achieve a reliable performance. Lastly, a static phase offset detection method and several compensation schemes to mitigate the static phase offset among different clock phases in the MX-PDLL are described. With all three of these measures, a flexible and reliable high-speed link system can be achieved.

Chapter 2 presented two popular options to generate accurate timing in the CDR, the PLL and DLL. Each of the clock generators has its suitable noise conditions to be operated in. Therefore a design tradeoff has to be made according to different system considerations. Several prior art examples have been demonstrated to adjust the bandwidth of the PLL loop in order to minimize the jitter under different noise conditions, but have their own limitations.

In Chapter 3, an adaptive-bandwidth mixed PLL/DLL (MX-PDLL) clock generator that can achieve the optimum jitter performance by having a wide-range bandwidth tuning capability under different noise conditions (power supply noise and reference clock noise in particular) is presented. The proposed MX-PDLL uses a phase mixing interpolator to mix the signal energy of the reference clock and output clock. By tuning the mixing weight, the loop can be configured as a PLL, a DLL, or a mixture of the two and achieves a wide bandwidth tuning range. This design does not have the same limitations present in the prior adaptive-bandwidth PLL designs. Most importantly, it enables a convenient method to tune the bandwidth over a wide range of frequencies and to achieve the optimum jitter performance.

In Chapter 4, a new dual-loop CDR architecture using the MX-PDLL as the core clock generator to take advantage of the bandwidth tuning capability of the MX-PDLL to minimize the sampling clock jitter across different noise conditions is proposed. This CDR utilizes a digitally-controlled phase rotator to shift the reference clock phase that feeds into the MX-PDLL in order to track the phase and frequency of the data. Moreover this CDR architecture also mitigates the digital control-induced quantization noise by filtering the noise with the MX-PDLL loop where its bandwidth is tunable.

All of the proposed solutions for dealing with uncertain noise conditions on-chip and off-chip have been verified with measurement results from the prototype chip fabricated in a 0.18μm logic CMOS technology. The tradeoffs between power supply noise suppressing, input reference noise filtering and quantization noise filtering have been clearly shown in the measurement results. The measurement results have demonstrated that by having the ability to tune the clock generator's bandwidth, the CDR system can find the optimum bandwidth setting to achieve the minimum jitter on the output clock. Therefore it can provide very accurate timing to retrieve the incoming data.

In Chapter 5, we have seen that in addition to the "dynamic" type timing uncertainty, also known as jitter, there is also the "static" type, phase mismatch, in the system due to process variations and systematic imbalances in the physic design (i.e., layout and cross talk on-chip). If the static phase mismatch exists in the sampling clock, it will exhibit a static "mean" offset from its ideal position to further degrade link performance. Since this mismatch is static in nature, it can be calibrated after the chip is fabricated. Therefore an auxiliary charge pump compensator is used to first remove the static phase offset between the reference clock (*rclk*) and the output clock (*bclk*) in order to remove a big phase spacing mismatch in the first delay cell in DLL mode. Then a passive resistor-ring-based phase averaging network is used to "smear" out the residual phase spacing mismatch among the rest of clock phases. Measurement results show that some percentage of the static phase mismatch can be removed by the above compensation schemes. However there is still residual mismatch that can be reduced by means of a careful layout. A divide-by-2 block with duty-cycle correction function to guarantee the even quadrant spacing in a 360-degree phase rotator is also presented. The

evenly spaced quadrant in the phase rotator can minimize the nonlinearity in the phase rotator thus reduce the output clock jitter.

This dissertation has presented a noise-robust CDR design using an adaptive-bandwidth MX-PDLL-based clock generator for a time-interleaved receiver. One of the potential future research directions can be to include an on-chip jitter monitoring or measurement circuitry to monitor the change in sampling clock jitter and send the information back to the interpolation controller in order to find the optimum mixing weight under time-varying noise condition. The appropriate control loop's bandwidth and efficient adaptation algorithm require further investigation. Finally with this adaptation control loop together with everything presented in this dissertation, a noise-robust clock and data recovery design that can be re-configured automatically in order to minimize the output clock jitter is achievable.

Bibliography

[1] K.-Y. K. Chang, et al., "A 0.4-4Gb/s CMOS quad transceiver cell using on-chip regulated dual-loop PLLs," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 747–754, May 2003.

[2] R. Farjad-Rad, et al., "A 33-mW 8-Gb/s CMOS clock multiplier and CDR for highly integrated I/Os," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1553–1561, September 2004.

[3] S. Sidiropoulos and M. Horowitz, "A semidigital dual delay-locked loop," *IEEE Journal of Solid-State Circuits*, November 1997.

[4] A. H.-Y. Tan and G.-Y. Wei, "Adaptive-bandwidth mixing PLL/DLL based multi-phase clock generator for optimal jitter performance," *IEEE Proceedings of Custom Integrated Circuits Conference (CICC)*, pp. 749-752, September 2006.

[5] G.-Y. Wei, et al., "A 500MHz MP/DLL clock generator for a 5Gb/s backplane transceiver in 0.25μm CMOS," *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, pp. 464-465, February 2003.

[6] J. Jaussi, et al., "A 20Gb/s embedded clock transceiver in 90nm CMOS," *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, pp. 340–341, February 2006.

[7] M. Meghelli, et al., "A 10Gb/s 5-tap-DFE/4-tap-FFE transceiver in 90nm CMOS," *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 80–81, February 2006.

[8] J.-M. Chou, Y.-T. Hsieh, and J.-T. Wu, "A 125 MHz 8b digital-to-phase converter," *IEEE International Solid-State Circuits Conference (ISSCC,) Digest of Technical Papers*, pp. 436–437, February 2003.

[9] P. Larsson, "A 2-1600-MHz CMOS clock recovery PLL with low-Vdd capability," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 1951–1960, December 1999.

[10] J. Montanaro, et al., "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 1703–1714, November 1996.

[11] H.-T. Ng, et al.; "A second-order semidigital clock recovery circuit based on injection locking," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 2101–2110, December 2003.

[12] M.-J. E. Lee, et al., "A second-order semi-digital clock recovery circuit based on injection locking," *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, pp. 74–75, February 2003.

[13] S. R. Norsworthy, R. Schreier, and G. C. Temes, editors. *Delta-Sigma data converters, theory, design, and simulation*, New York: IEEE Press, 1997.

[14] M. Horowitz, et al., "PLL design for a 500MB/s interface," *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, pp. 160–161, February 1993.

[15] T. H. Lee, et al., "A 2.5 V CMOS delay-locked loop for an 18 Mbit, 500Megabytes/s DRAM," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 1491–1496, December 1994.

[16] M. Horowitz, C.-K. K. Yang, and S. Sidiropoulos, "High-speed electrical signaling: overview and limitations," *IEEE Micro*, pp. 12-24, January/February 1998.

[17] C.-K. K. Yang and M. A. Horowitz, "A 0.8-µm CMOS 2.5 Gb/s oversampling receiver and transmitter for serial links," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 2015–2023, December 1996.

[18] C.-K. K. Yang, R. Farjad-Rad, and M. A. Horowitz, "A 0.5-µm CMOS 4.0-Gbit/s serial link transceiver with data recovery using oversampling," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 713–722, May 1998.

[19] R. Farjad-Rad, C.-K. K. Yang, M. A. Horowitz, and T. H. Lee, "A 0.3-µm CMOS 8-Gb/s 4-PAM serial link transceiver," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 757–764, May 2000.

[20] B. Razavi, *Design of integrated circuits for optical communications*, *1st ed.*, New York: McGraw-Hill, 2003.

[21] B. Kim, T. C. Weigandt, and P. R. Gray, "PLL/DLL system noise analysis for low jitter clock synthesizer design," *IEEE Proceedings of International Symposium on Circuits and Systems (ISCAS),* vol. 4, pp. 31-34, June 1994.

[22] T. H. Lee and J. F. Bulzacchelli, "A 155-MHz clock recovery delay- and phase-locked loop," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 1736–1746, December 1992.

[23] A. X. Widmer and P. A. Franaszek, "A DC-balanced, partitioned-block, 8B/10B transmission code," *IBM Journal of Research and Development*, vol. 27, no. 5, pp. 440-451, September 1983.

[24] Gigabit Ethernet, *IEEE 802.3z-1998*, July, 1998.

[25] L. DeVito, et al., "A 52MHz and 155MHz clock-recovery PLL," *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, pp. 142–143, February 1991.

[26] M. Mansuri and C.-K. K. Yang, "Jitter optimization based on phase-locked loop design parameters," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1375–1382, November 2002.

[27] V. Stojanovic and M. Horowitz, "Modeling and analysis of high-speed links," *IEEE Custom Integrated Circuits Conference (CICC),* September 2003.

[28] S. Sidiropoulos, et al., "An 800mW 10Gb Ethernet transceiver in 0.13μm CMOS," *IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, pp. 168–169, February 2004.

[29] M.-J. E. Lee, et al., "Jitter transfer characteristics of delay-locked loops-theories and design techniques," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 614–621, April 2003.

[30] P. Larsson, "Measurements and analysis of PLL jitter caused by digital switching noise," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 1113-1119, July 2001.

[31] F. M. Gardner, "Charge-pump phase-lock loops," *IEEE Transactions on Communications*, vol. COM-28, pp. 1849-1858, November 1980.

[32] S. Ye, L. Jansson, and I. Galton, "A multiple-crystal interface PLL with VCO realignment to reduce phase noise," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1795–1803, December 2002.

[33] S. Ye and I. Galton, "Techniques for phase noise suppression in recirculating DLLs," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1222–1230, August 2004.

[34] M. Mansuri and C.-K. K. Yang, "A low-power adaptive bandwidth PLL and clock buffer with supply-noise compensation," *IEEE Journal of Solid-State Circuits*, vol. 38, pp. 1804–1812, November 2003.

[35] E. Alon, V. Stojanovic, and M. A. Horowitz, "Circuits and techniques for high-resolution measurement of on-chip power supply noise," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 820–828, April 2005.

[36] PCI-SIG "PCI Express jitter modeling," revision 1.0RD, July 14, 2004."

[37] M. Aoyama, et al., "3Gbps, 5000ppm spread spectrum SerDes PHY with frequency tracking phase interpolator for serial ATA,"*IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, pp. 107-110, June 2003.

[38] Digital Display Working Group "Digital Visual Interface DVI," revision 1.0, April 2, 1999.

[39] Jaeha Kim. *Design of CMOS Adaptive-Supply Serial Links*. PhD thesis, Stanford University, CA, December 2002.

[40] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing, principles, algorithms, and applications*, *3rd ed.*, New Jersey: Prentice-Hall, Inc., 1996.