# Design of High Speed I/O Interfaces for High Performance Microprocessors

A dissertation presented

by

Ankur Agrawal

to

The School 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

October 2010

Thesis advisor

**Gu-Yeon Wei**

Author

**Ankur Agrawal**

**Design of High Speed I/O Interfaces for High Performance Microprocessors**

# Abstract

Advances in CMOS process technology have enabled high performance microprocessors that run multiple threads in parallel at multi-gigahertz clock frequencies. The off-chip input/output (I/O) bandwidth of these chips should scale along with the on-chip computation capacity in order for the entire system to reap performance benefits. However, scaling of off-chip I/O bandwidth is constrained by limited physical pin resources, legacy interconnect technology and increasingly noisy on-chip environment. Limited power budgets and process/voltage/temperature (PVT) variations present additional challenges to the design of I/O circuits.

This thesis focuses on the need to improve timing margin at the data samplers in the receivers, to enable higher symbol-rates per channel. The first part of this thesis describes a technique to reclaim timing margin lost to jitter both in the transmitted data and sampling clock. The second part discusses two techniques to correct for static phase errors in the sampling clocks that can degrade timing margin. Two test-chips, designed and fabricated in 0.13$\mu$m CMOS technology, demonstrate the efficacy of these techniques.

An 8-channel, 5 Gb/s per channel receiver demonstrates a collaborative timing recovery architecture. The receiver architecture exploits synchrony in transmitted

data streams in a parallel interface and combines error information from multiple phase detectors in the receiver to produce one global synthesized clock. Experimental results from the prototype test-chip confirm the enhanced jitter tracking bandwidth and lower dithering jitter on the recovered clock. This chip also enables measurements that demonstrate the advantages and disadvantages of employing delay-locked loops (DLL) in the receivers. Two techniques to condition the clock signals entering the DLL are proposed that reduce the errors in phase-spacing matching between adjacent phases of the DLL and improve receiver timing margins.

A digital calibration technique takes a more general and inclusive approach towards correcting phase-spacing mismatches in multi-phase clock generators. A shared-DAC scheme reduces the area consumption of phase-correction circuits by more than 60%. This technique is implemented to correct phase-spacing mismatches in a 8-phase 1.6 GHz DLL. Experiments performed on the test-chip demonstrate reduction in peak differential non-linearity (DNL) from 37 ps to 0.45 ps, while avoiding any additional jitter penalties from the shared-DAC scheme.

# Contents

# List of Figures

# List of Tables

# Acknowledgments

Graduate school has been a rich and rewarding experience, and I have many people to thank.

First and foremost, I'd like to thank my advisor, Professor Gu-Yeon Wei. He was more fair, understanding, and patient than I could have ever hoped for. He challenged my intellect in every meeting, and encouraged me to think deeply and creatively. He also taught me how to effectively communicate my research in papers and presentations. He will always be a role model for me.

Professor Pavan Kumar Hanumolu was my co-advisor, and I am grateful to him for his friendship and guidance. My designs benefited greatly from his deep knowledge of circuits, and I would look forward to his comments and suggestions in our weekly meetings. He was generous with his time and advice, and was always available to answer my queries. I look forward to continued friendship.

I would also like to thank Professor David Brooks and Professor Paul Horowitz for reading this dissertation, and serving on my qualifying exam and defense committees. Professor Donhee Ham was my CHD advisor, and always showed a keen interest in my progress.

My internships at IBM Research, under the mentorship of John Bulzacchelli, helped me immensely to mature as a circuit designer. John is a brilliant researcher and extremely unselfish with his time, and I consider myself very lucky to have worked him. I am indebted to him — not only for sharing his vast technical expertise, but also for taking me to baseball games at Yankee stadium!

I would also like to take this opportunity to thank all my teachers in school, college and graduate school. This thesis is the result of the knowledge that they imparted to

me and the work ethic that they instilled in me.

Glenn Holloway managed our lab's computing infrastructure, and I can't thank him enough for going out of his way to help me with all my software-related issues. I will also cherish all our middle-of-the-night conversations in the pantry room — they kept my sanity intact while I was scrambling to meet tape-out and paper deadlines. Jim MacArthur and Al Takeda from the Electronics Design Laboratory, and Curtis Mead were indispensable as I rigged up my test setups.

My tenure as a graduate student would not have been as enjoyable and fulfilling without the support of my friends and colleagues. I learnt a lot from the members of the VLSI and Computer Architecture Research groups at Harvard, both past and present. Andrew Liu and Hayun Chung were generous with their help and ideas on technical matters. Mark Hempstead, Michael Lyons, Meeta Gupta and Wonyoung Kim were my regular lunch companions and I found our lunch-time conversations both stimulating and relaxing. I would also like to thank Alex Liang, Benjamin Lee, VJ Reddi, Krishna Rangan, Kevin Brownell, Jud Porter, Sophia Shao, Ruwan Ratnayake and Amber Tan for their friendship.

Outside of lab, my room-mates Anand Sampath, Arun Kancharla, Narayanan Ramachandran and Ayan Chakrabarti were almost like family to me. I will always fondly remember the time spent with them and I would like to thank them for standing by me through thick and thin, and inspiring me to be a better person. I always fell back upon my friends from college, Siddharth Garg, Siddharth Tata, and Shivaram Kalyanakrishnan in times of need. I'd also like to thank my other friends in the Boston area for their friendship and support.

Finally, I'd like to acknowledge my family. I'd like to thank my uncles, aunts and cousins, both in India and the US, for their support and encouragement. Dilip Mama has always been like a third parent to me, and is a perennial source of inspiration for me. I can never thank my mother enough for her love, prayer, sacrifice, and support. In the last stretch, I was lucky to have very caring parents-in-law and would like to thank them for their constant encouragement. My brother, Kushal, has been a source of strength for me throughout.

Last, but not the least, I'd like to thank my dearest Anshu. She brings joy everyday into my life. No challenge seems too difficult after having her by my side. She is the greatest blessing I could have asked for. Thank you!

*To he*
*who would have been proudest*
*to see this — Papa*

# Chapter 1

# Introduction

The relentless pursuit of Moore's Law has enabled exponential growth in the semiconductor industry. Modern high performance microprocessors integrate more than a billion transistors on a single die [30, 17] and have several cores that provide an aggregate performance of 100's of gigaflops (floating point operations per second). For the entire multi-chip digital system to benefit from this increased computation throughput, the off-chip input/output(I/O) bandwidth should also scale. This has led to the widespread use of parallel links, where interfaces employ tens to hundreds of I/O links in parallel to achieve their aggregate bandwidth targets. Because the number of pins do not scale as rapidly as the bandwidth requirements, each of these links needs to operate at higher data-rates. This thesis focuses on generating precisely positioned clocks to ensure correct operation of these interfaces at these increased data-rates.

## 1.1 Area of Focus

To set the context of the challenges that high-speed parallel interface designers face, Figure 1.1 shows the die-photo of the next generation Sun *Sparc* processor — a sixteen-core processor capable of running up to 128 threads in parallel [25]. All along the periphery of the die, shown in the boxes, are the high speed I/O circuitry. The I/O circuits are used for communication with memory and other processors. They are also used to provide interfaces like PCI Express and XAUI. Overall, the aggregate I/O bandwidth of all the I/O circuits is 2.4 Tb/s. In comparison, the current generation Sun *Sparc* has a total I/O bandwidth of 800 Gb/s — a 3x increase in aggregate I/O

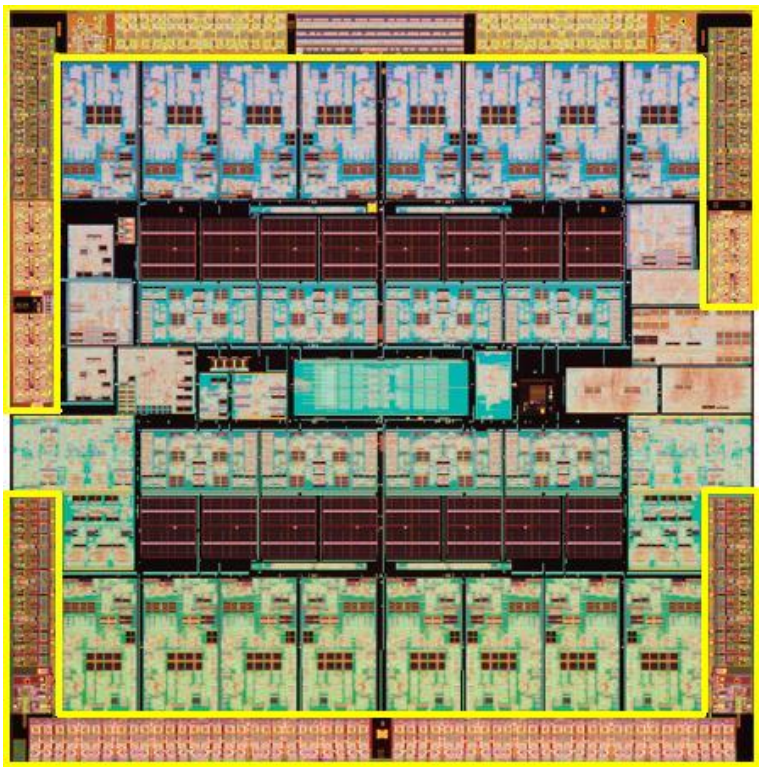

Figure 1.1: Sun Sparc *Rainbow Falls* die-photo

bandwidth in just one generation [16]. While this was just one specific example, the same trend can be observed for other high performance microprocessors. Meeting these bandwidth targets requires innovations in the design of the basic building block: the I/O link.

A representative block diagram of a typical serial link is shown in Figure 1.2. It consists of a transmitter, a channel, and a receiver. The transmitter converts digital binary data into electrical signals and launches them on the channel. This channel is normally modeled as a transmission line and can consist of traces on a printed circuit board (PCB), traces within packages, cables, and connectors that join these various parts together. The receiver converts the incoming signal back to digital data and requires a timing recovery block to determine the optimal timing instants for sampling the signal. To enable high bandwidth communication between two chips, a parallel transceiver consists of multiple high-speed links operating in parallel. Different architectures are available for timing recovery, and factors that govern this choice are discussed in detail in Chapter 2.

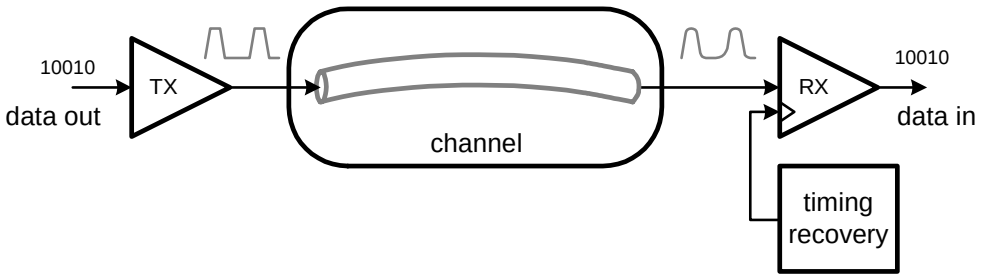


Figure 1.2: Typical high speed serial link block diagram

As the aggregate I/O bandwidth requirements increase rapidly without a corresponding increase in the pin-count dedicated to high speed I/O, each serial link must

operate at higher data-rates. As data-rates increase, both channel-related and circuit-related challenges emerge. First, the channel behaves as a lossy transmission line, introducing inter-symbol interference and severely degrading the transmitted symbol. To compensate for this frequency-dependent channel loss, increasingly complex equalization solutions need to be implemented in the transmitter and receiver [13]. Second, as bit-periods shrink, circuit related issues such transmitter and receiver bandwidth and clock jitter limit the performance of the link. The clock generator circuits in the transmitters and the timing recovery circuits in the receiver need to produce accurate low-jitter clocks to maximize timing margin at the receivers.

Meeting these challenges is further complicated by limited power budgets for I/O, and increasing on-chip process, voltage, and temperature (PVT) variations. The focus of this dissertation is on meeting some of the circuit-level challenges — in particular, techniques to improve the timing margin at the receiver. Before proceeding to a high-level overview of this work, it is instructive to understand the concept of timing margin.

### 1.1.1 Timing Margin

Figure 1.3 shows the noisy receive signal and its eye diagram, formed by wrapping the receive signal waveform around one bit period in the time domain, with the center of the eye at the ideal sampling position. The timing margin is the tolerable margin of error in the position of the sampling clock (RxClk), when the sampler decision threshold is at the center of the vertical eye opening. The timing margin should be ideally equal to the bit period, but various sources of timing noise (transmitter

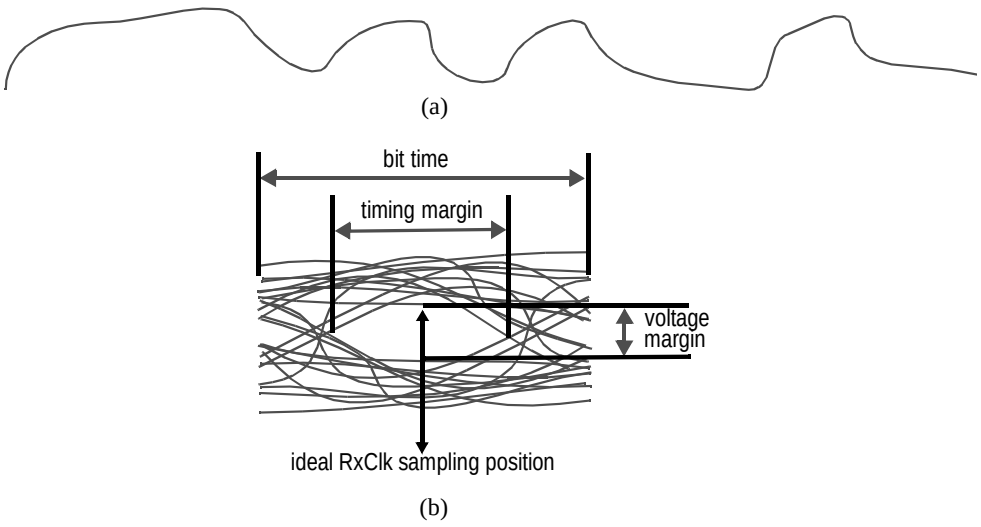


Figure 1.3: (a) Received noisy data signal (b) Folded signal eye diagram (from [33])

clock jitter, inter-symbol interference, cross-talk and sampling clock jitter) reduce the timing margin and "close the eye". Similarly, in the vertical direction, voltage noise reduces the voltage margin to less than the nominal signal swing at the receiver inputs. The next section shall provide an overview of the techniques proposed in this thesis to achieve timing margin improvements in the receiver.

## 1.2 Primary Contributions

This dissertation describes two techniques to improve the timing margin at the receiver — targeting both dynamic and static sources of timing margin degradation.

Dynamic timing margin improvements are obtained by increasing the clock jitter correlation in the transmitter and receiver. The sampling clock produced by the timing recovery circuit in the receiver tracks the dominant sources of jitter in the transmitter. It exploits the redundancy in the timing error information in multiple

data-streams in a parallel link to perform more effective timing recovery — we call this scheme *collaborative timing recovery*. It tracks the transmitter clock jitter over a wide bandwidth while itself generates less jitter when compared to other schemes.

Static phase mismatches can also eat into the timing margin in receivers. Both transmitters and receivers employ multiple equally-spaced clocks to enable signaling at frequencies greater than the on-chip clock frequencies, as depicted by Figure 1.4. Any error in the phase-spacings in the transmitter or receiver degrades the timing margins at the receivers by two mechanisms. First, the position of the sampling clock is offset from the ideal sampling instant, directly affecting the timing margin. Second, the feedback loop in the timing recovery block, used to position the sampling clock, relies on sampled data to measure timing error. Using data sampled at non-ideal instants increases the jitter on the synthesized clock and further degrades receiver timing margin. We propose a scheme to perform low-overhead clock calibration of multi-phase clock generators to reduce these penalties.

Two test-chips fabricated in 130nm CMOS technology, the details of which shall be described in this dissertation, demonstrate these techniques experimentally. Power consumption considerations drive key design choices in both these test-chips. For these techniques to be robust against PVT variations, digital CMOS circuits substitute precise analog circuits in a number of circuit blocks.

Overall, this work demonstrates techniques well suited to high speed I/O interfaces for future high performance microprocessors.

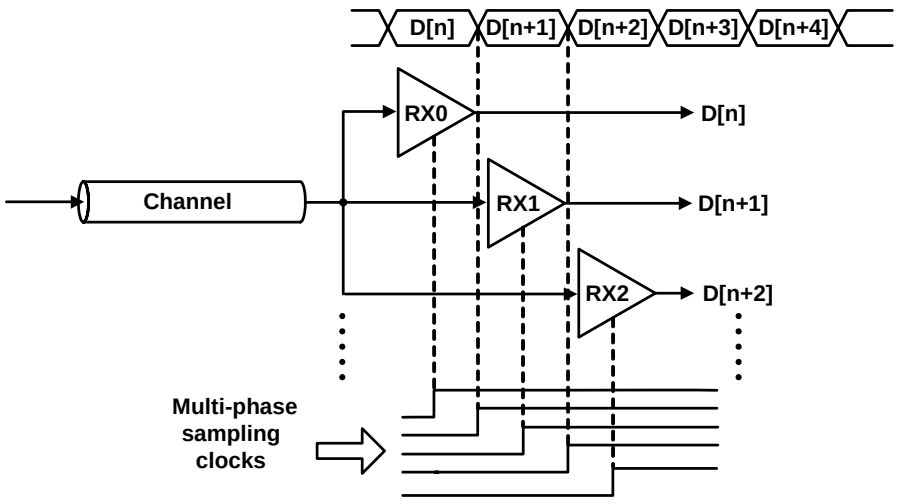


Figure 1.4: High speed signaling using multi-phase clocks

## 1.3 Overview of the thesis

This thesis is divided into two main parts; the first describes the proposed collaborative timing recovery architecture in Chapters 2 and 3, and the second describes two techniques for performing phase calibration of multi-phase clock generating circuits in Chapter 4. For both sections, we will discuss the motivations for the work, describe the prototype implementation, and present measurement results to substantiate our claims. Chapter 5 concludes this thesis with a few general remarks.

Chapter 2 begins with an overview of existing architectures for parallel interfaces. To better appreciate the shortcomings of these architectures, it is important to understand the sources of jitter in high-speed links operating in the noisy environment of a microprocessor and also some key performance metrics that are relevant to these interfaces. To accomplish this, we first review some of the basics of clock and data recovery circuits, and then provide a case study where we analyze a very popular

receiver architecture.

Chapter 3 describes the collaborative timing recovery architecture and the prototype test chip. System architecture, circuit design details, description of our experimental setup, and measurement results form the core of this chapter. It concludes by discussing the potential and limitations of this architecture. This also wraps up the first part of this thesis.

Chapter 4 begins by contrasting phase-locked loops and delay-locked loops in the context of multi-phase clock generation in parallel receivers. It argues in favor of delay-locked loops while acknowledging their short-coming: their susceptibility to various sources of phase-spacing mismatch. It provides a detailed description of the sources of this mismatch before describing two solutions that reduce this mismatch. The first solution is mostly analog, and was implemented in the receivers in the collaborative timing recovery test chip. The second solution is a low-overhead digital solution that results in very accurate multi-phase clocks, as evidenced by experimental results obtained from another prototype chip.

# Chapter 2

# Background

The first part of this thesis focuses on a new architecture for parallel receivers that enables the receiver to track transmitter clock jitter over a wide bandwidth, while itself generating lower jitter, to improve the overall timing margin at the receiver. This chapter provides a tutorial-style overview of high-speed receivers. It revisits the basic serial link introduced in Chapter 1, and begins by providing an understanding of the limitations to achieving high-speed operation. It then provides an overview of the various choices available for clocking the receivers and the trade-offs involved. Link performance is severely affected by clock and data timing uncertainty. This chapter describes both the sources of the these timing uncertainties, and how the link's resilience to these uncertainties is characterized. To tie these concepts together, a popular receiver architecture, semi-digital dual-loop receiver, is then discussed. We conclude this chapter by making a case for a new receiver architecture for parallel receivers.

## 2.1 High Speed Link Systems

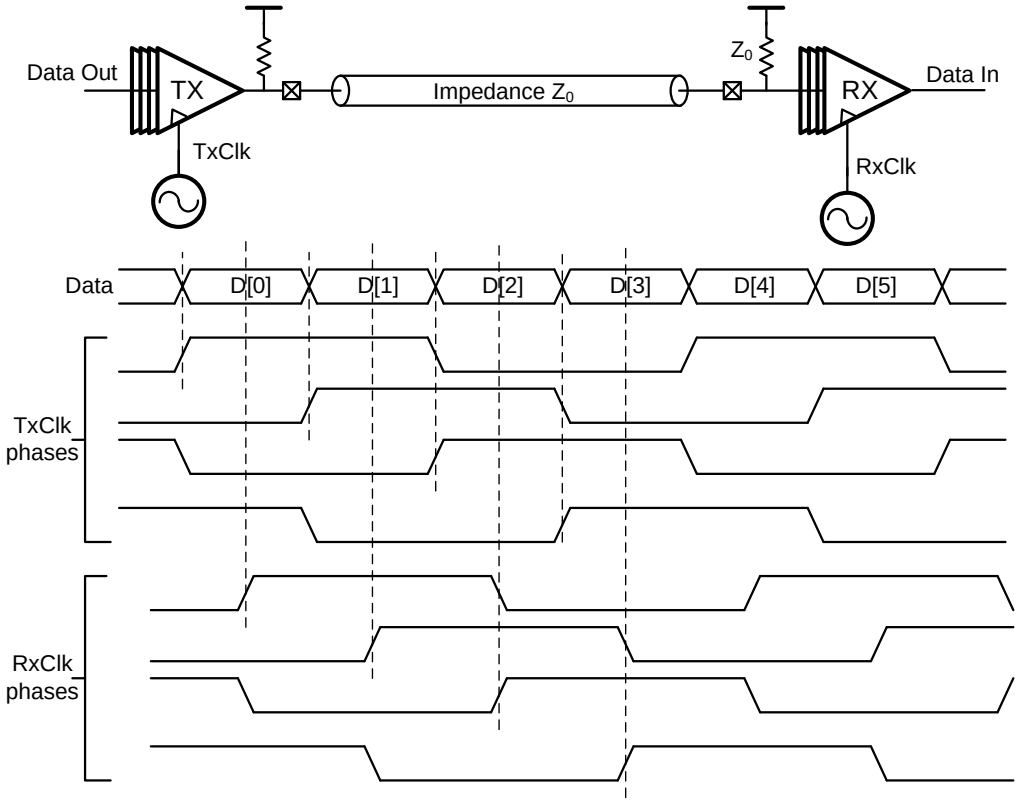


Figure 2.1: Typical serial link block diagram

We begin by revisiting the typical serial link, introduced in Chapter 1, in greater detail. Figure 2.1 shows the block diagram of a typical serial link, consisting of a transmitter, channel, and receiver. The transmitter (TX) converts the digital data into an electrical signal and launches it on the channel. Termination resistors on both ends of the channel, matched with its characteristic impedance ($Z_0$), avoid any reflections introduced by impedance discontinuities. With proper termination, new data bits can be launched on the channel without having to wait for the reflections to decay. This enables high-speed operation. The receiver (RX) at the other end of

the channel recovers the digital data bits from the incoming analog signal, usually by sampling and quantizing the signal.

One key factor that limits the maximum achievable data-rate in a given high-speed link is the speed of the process technology, often characterized by the fan-out-of-four (FO4) delay of the technology [31]. On-chip full-swing clock signals need to have a clock period of at least 6-8 FO4 delays for reliable operation. The FO4 delays can be roughly approximated as 500ps per $\mu$m of minimum drawn gate length in CMOS technologies. In the 0.13$\mu$m technology, the maximum achievable on-chip clock frequency is around 2-2.5 GHz. If the TX transmits 1 symbol per clock-period, this limits the link speed to 2-2.5 Gb/s. To achieve higher data-rates, multiple data-bits can be transmitted in one clock cycle by interleaving data bits using multi-phase clocks.

As shown in Figure 2.1, the TX employs a multi-phase clock generator like a phase-locked loop (PLL) or delay-locked loop (DLL) to transmit data at well-defined time intervals that are less than a clock period apart. To minimize bit-error rate (BER) the RX needs to sample the incoming data at the center of the symbol interval. Thus, the RX also requires multi-phase clocks that are frequency- and phase-aligned with the incoming signal. A precise timing generator is used to synchronize the sampling clocks with respect to the incoming data. Two choices are available for generating the sampling clocks in the RX:

- The TX forwards a clock to the RX, ensuring that the TX and RX operate synchronously. In this case, only the RX clock phases need to be aligned to the incoming data.

- The TX and RX have their own independent clock sources. While the two clock sources are designed to have the same frequency, in practice they are never *exactly* equal and the TX and RX operate plesiochronously. The frequency offset is usually measured in parts-per-million and translates to a phase-ramp in the time-domain (Figure 2.2). Additional circuitry is required in the RX to derive a sampling clock that is nominally at the center of the symbol interval.

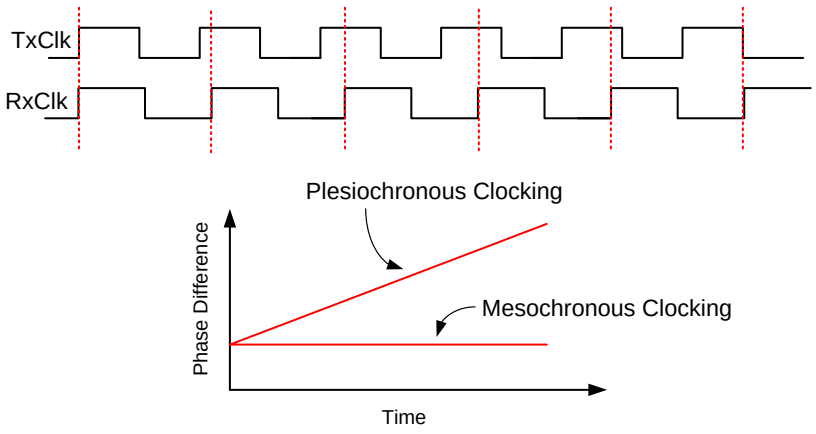


Figure 2.2: Plesiochronous and mesochronous clocking

These receiver clocking choices are the primary difference between the two most popular topologies used for implementing parallel receivers: source-synchronous links and ensemble of serial links.

In source-synchronous links (Figure 2.3), the TX transmits its clock on a separate channel along with multiple data channels. The RX uses this forwarded clock as a frequency reference. However, at high data-rates, the inter-signal skew can be a significant percentage of the symbol interval and thus these links need to perform per-pin skew compensation [34] to ensure that data is optimally sampled. The receivers

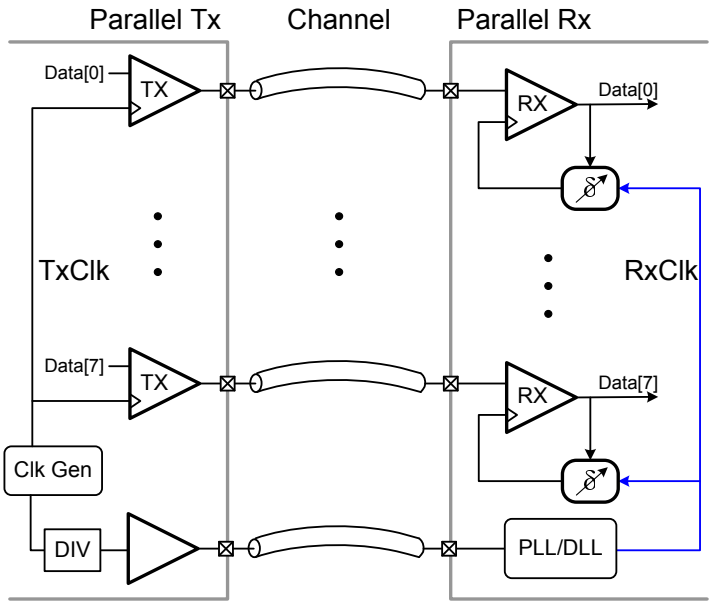


Figure 2.3: Block diagram of source-synchronous links.

also need to produce multiple phase-shifted clocks to sample the interleaved data. An optional PLL or DLL receives the forwarded clock and performs jitter-filtering and/or clock frequency multiplication.

In contrast, an ensemble of serial links, shown in Figure 2.4 uses independent clock sources in the TX and RX. The TX does not forward a clock and the RX performs its own clock recovery i.e., it uses the timing information embedded in the incoming data to position the sampling clock. It needs to track both the frequency and the phase of the incoming clock. As the links do not share any circuit resources, each link generates its own recovered clock. This makes these receivers area and power inefficient. Its biggest advantage is its flexibility — it can be scaled easily to meet the aggregate bandwidth requirement by employing more channels and replicating the TX and RX circuits.

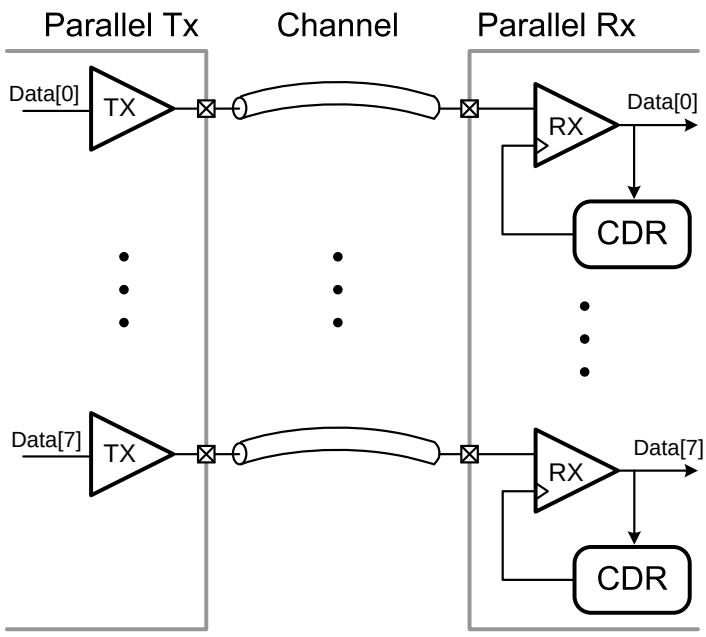


Figure 2.4: Ensemble of serial links.

There are other differences between forwarded clock (source-synchronous) and embedded clock (serial) links, mostly with respect to their jitter performance. The next section describes some of the sources of jitter in high speed I/O links.

## 2.2 Jitter in High Speed Links

Jitter refers to the uncertainty in the position of clock and data edges. It affects both the transmitted data and the recovered clock and strongly governs the bit-error rate (BER) of the link. Achieving good jitter performance is key to reliable link operation, and thus it is important to understand the dominant sources of jitter in high speed links. Figure 2.5 shows a sample eye diagram of a data signal. The jitter histogram shows a bimodal distribution. The two peaks can be attributed to system-

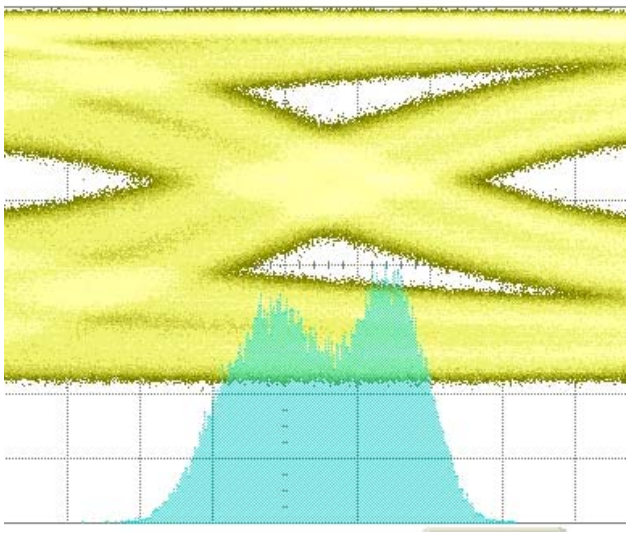

Figure 2.5: Sample data eye diagram with jitter histogram superimposed

atic sources of jitter, while the Gaussian jitter is caused mostly by random sources of noise in the system. In this section, we shall describe some of the phenomena that cause jitter in high speed links. This section will first focus on power supply noise, a big contributor to the total jitter in the system. Then we shall look at some of the other sources of jitter in the system. A discussion on jitter amplification wraps up this section.

### 2.2.1 Power Supply Noise

Power supply noise (PSN) in a microprocessor is the deviation of its supply voltage from its nominal value. Ideally, the supply voltage is steady at its expected value. However, perturbations occur when there are sudden changes in the current consumption by the processor. The power delivery network (PDN) of the processor can have substantial parasitic inductance, and this current variation produces voltage ripple on

the chip's supply lines. This is significant because the delay of the underlying circuits are a function of the supply voltage and PSN translates into jitter on clock and data edges.

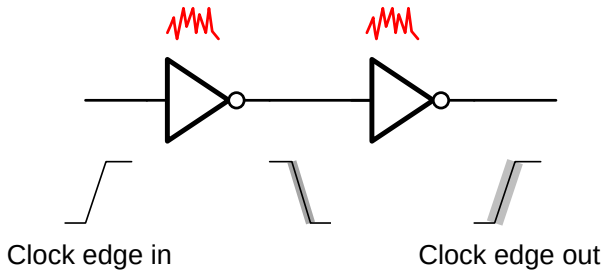


Figure 2.6: Power supply noise in buffers adds jitter to clock edges

As shown in Figure 2.6, as signals propagate through buffers, power supply noise adds jitter to the signals. In the context of high speed links, PSN can manifest itself as jitter in a few different ways:

- **Clock Generator Jitter** — Both transmitters and receivers employ a PLL- or DLL-based clock generator, and PSN can add jitter to the clock produced by these circuits. In both PLLs and DLLs, PSN modulates the delays of the buffers in the voltage-controlled oscillator (in the case of PLL) or voltage-controlled delay line (in the cases of DLL). The negative feedback loops compensate for the low-frequency noise up to the $-3$ dB bandwidth of the loop, but the rest of the noise translates to clock jitter.

- **Clock Distribution Jitter** — As the clock travels from the clock generators to the drivers (TX) or samplers (RX) through the clock distribution buffers, power supply noise adds more jitter to the clocks.

- **Driver Jitter** — The delays of the drivers also get modulated as a result of power supply noise.

Figure 2.7 summarizes the effects of PSN in the transmitter.

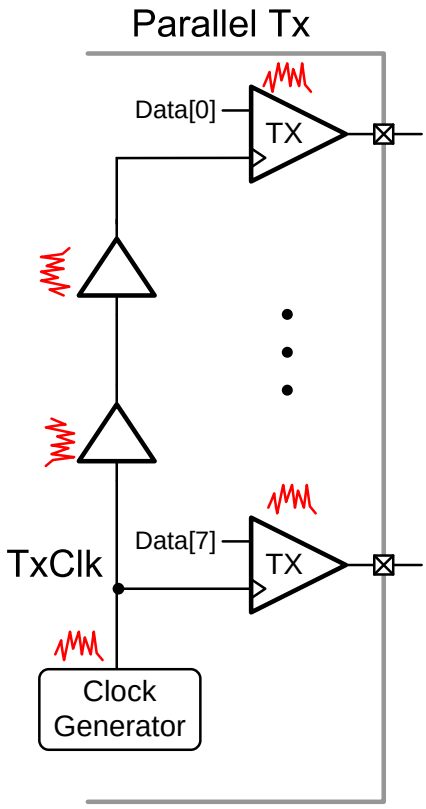


Figure 2.7: PSN vulnerabilities in a parallel transmitter

### PSN Frequency Dependence and Trends

Decoupling capacitors are used both off-chip and on-chip to suppress PSN. However, these capacitors are expensive — they occupy valuable real-estate on the silicon die or on the motherboard — and all of the PSN cannot be suppressed. Thus, the impedance profile of the PDN exhibits some peaking. Figure 2.8 shows the impedance

of the PDN as a function of frequency for the Intel Pentium 4 package [12]. It peaks at around 100 MHz, and consequently the PSN spectrum peaks at that same frequency too. PLLs/DLLs are usually designed to have bandwidths lower than 100 MHz and thus they are unable to suppress the resulting jitter. Overall PSN leads to a significant amount of "middle frequency" jitter in the 10's of MHz range. To maximize timing margin in the receivers, we must pay attention to this jitter and employ techniques to ensure that the receiver tracks this jitter.

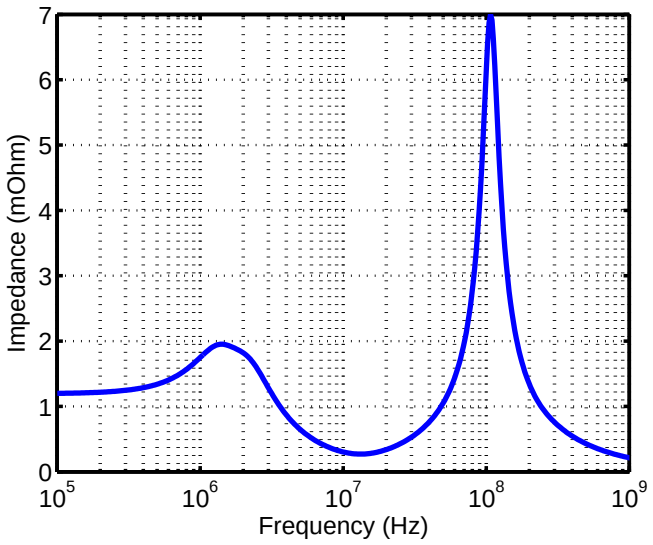


Figure 2.8: Impedance vs. frequency plot for Intel Pentium 4 power distribution network

The supply noise frequency range has remained roughly constant in the 100 MHz range in the last few process technology nodes since chip packages have not changed significantly and the reduction in the parasitic inductance of the power and ground C4 bumps has been matched to first order by an increase in the on-chip decoupling capacitance. Circuit solutions to tackle power-supply noise that benefit from technology scaling are thus likely to become more effective in future process technology

nodes.

### 2.2.2 Other Sources of Jitter

Besides power supply noise, there are other mechanisms that add jitter to the data and clock signals:

**Reference clock noise:** Transmitters and receivers usually require a reference clock that usually comes from an off-chip crystal oscillator. The jitter on this clock may propagate all the way to the transmit clock (in the TX) or the sampling clock (in the RX).

**Channel frequency response:** The channel transfer function exhibits significant frequency-dependent loss at higher frequencies. This causes the transmitted data edges to be attenuated and smeared, and inter-symbol interference (ISI) results. Most links employ some form of equalization to flatten the frequency response of the channel. However, all of the ISI cannot be canceled out, and the residual ISI manifests itself as jitter on the data. This jitter has little correlation from bit to bit, and thus is high-frequency jitter. High-frequency jitter is filtered out by the receiver to improve the performance of the clock recovery. The channel frequency response also results in a phenomenon called *jitter amplification* and we shall look into it in greater detail in the next subsection.

**Clock recovery dithering noise:** In the receiver, clock data and recovery (CDR) circuits use a phase detector to detect whether the sampling clock is ahead or behind the ideal sampling point. Usually a bang-bang phase detector is

employed, whose output is proportional to only the sign and not the magnitude of the error. Thus even when the CDR is in lock, the bang-bang phase detector produces full scale outputs for small errors, and the sampling clock dithers about its ideal position. The resulting jitter is called dithering jitter, and minimizing this jitter is important for good receiver performance.

**Device and layout mismatches:** Variations in circuit and device characteristics, caused either by random process variations or systematic layout mismatches, can also result in jitter on the clock and data. For example, a P-N skew in the clock distribution buffers can distort the duty-cycle of the TX clock at the drivers. Consequently, the data bit will have different bit-periods and the data eye will look similar to the one in Figure 2.5 with two peaks in the location of the edge, corresponding to two different bit-periods.

**Crosstalk:** In parallel links, flux coupling between nearby signals leads to cross-talk. The magnitude of the coupling depends on the magnitude of the mutual inductance and capacitance — both a function of the layout geometry. Inter-signal cross-talk may also be induced via a shared power supply or ground. Cross-talk manifests itself as voltage noise, but can translate into high-frequency jitter if the noise event occurs close to the data transition edge. Differential signaling is less susceptible to this as the induced voltage is mostly common-mode to the signal pair.

**Device Noise:** Finally device noise (thermal and flicker) adds jitter to the clock and data. In high speed link applications, this is not a dominant source of jitter.

Overall, when designing parallel links, it is important to understand the dominant sources of jitter and their impact on link performance. In the remainder of this section, we shall discuss jitter amplification where the lossy channel amplifies the jitter present on transmitted data and clock edges in certain cases.

### 2.2.3 Jitter Amplification

As we saw earlier in this section, power supply noise in the transmitter gets transferred to the transmitted clock, and manifests itself as phase modulation. Ignoring the higher order harmonics, we can express the clock waveform as:

$$c(t) = A\cos(2\pi f_c t + \beta \sin 2\pi f_m t) \tag{2.1}$$

where $f_c$ is the clock frequency and $f_m$ is the frequency of phase modulation. If the amplitude of phase modulation $\beta$ is small, the spectrum of $c(t)$ can be approximated as :

$$C(f) \approx \frac{A}{2}\delta(f - f_c) - \frac{\beta A}{4}[\delta(f - f_L) - \delta(f - f_H)] \tag{2.2}$$

where $f_L = f_c - f_m$ and $f_H = f_c + f_m$. This clock spectrum is shaped by the channel frequency response before reaching the receiver. Assuming a linear phase channel $H(f)$ and denote $H(f_c)$, $H(f_L)$, and $H(f_H)$ by $\alpha_c$, $\alpha_L$, and $\alpha_H$ respectively, the receive clock spectrum is given by:

$$S(f) = \frac{\alpha_c A}{2}\delta(f - f_c) - \frac{\beta A}{4}[\alpha_L\delta(f - f_L) - \alpha_H\delta(f - f_H)] \tag{2.3}$$

In the time domain,

$$s(t) = \alpha_c A\cos(2\pi f_c t + \beta_r \sin 2\pi f_m t) \tag{2.4}$$

where,

$$\beta_r \approx \left(\frac{\alpha_L + \alpha_H}{2\alpha_c}\right)\beta \tag{2.5}$$

Jitter amplification results if:

$$\left(\frac{\alpha_L + \alpha_H}{2\alpha_c}\right) > 1 \tag{2.6}$$

This can happen if the channel response has a strong roll-off near the clock frequency $f_c$. As most channels have an exponential decay in the voltage transfer function, there can be significant jitter amplification in the clock signal [4].

Jitter amplification increases with increasing clock frequency and also with increasing phase modulation frequency (jitter frequency). At higher clock frequencies, the channel is likely to roll-off very steeply, increasing the jitter amplification. Also, at high phase modulation frequency, $\left(\frac{\alpha_L+\alpha_H}{2\alpha_c}\right)$ will be higher, implying higher jitter amplification. This phenomenon affects clock signals more than data signals because the energy of the data signal is spread from DC to $f_c$, while the clock energy is concentrated at $f_c$ and its odd harmonics. Therefore, in source-synchronous links, where clock and data signals are both sent to the receiver, the jitter on the clock signal may be higher than the jitter on the data signals.

This section focused on understanding jitter. The next section provides an overview of some of the metrics used to characterize the performance of high speed interfaces. These directly influence the design considerations while building high speed interfaces.

## 2.3 High Speed Links Performance Metrics

We begin this section by defining some of the key metrics used to describe high speed links. We shall then focus on the jitter performance of these links.

**Data-rate:** The data-rate for a high-speed link is the number of data bits transferred per second from the TX to the RX. It is usually measured in gigabits per second (Gb/s) and expressed *per channel*.

**Power:** The power consumed by the interface is a function of the data-rate, the signaling scheme, channel characteristics, and circuit design. It is usually measured in milliwatts per Gb/s (mW/Gb/s). For microprocessor applications, this is sometimes the most important design consideration.

**Bit error rate (BER):** BER is the ratio of the number of bits incorrectly received to the total number of bits received in a given time interval. It is a measure of the correctness of operation of the interface and most interfaces require BER to be below $10^{-12}$. Minimizing BER requires maximizing voltage and timing margins at the receiver.

**Frequency tolerance:** Clock recovery circuits have a finite slew rate in terms of phase tracking. This slew-rate dictates the maximum frequency offset between the transmitter and receiver reference clocks that the receiver can tolerate before it is unable to keep up with the transmit clock. Most modern interfaces track at least 5000 ppm (0.5%) frequency offsets to be suitable for use in interfaces that employ spread-spectrum clocking for reducing electro-magnetic interference.

### 2.3.1 Jitter Tolerance and Jitter Tracking

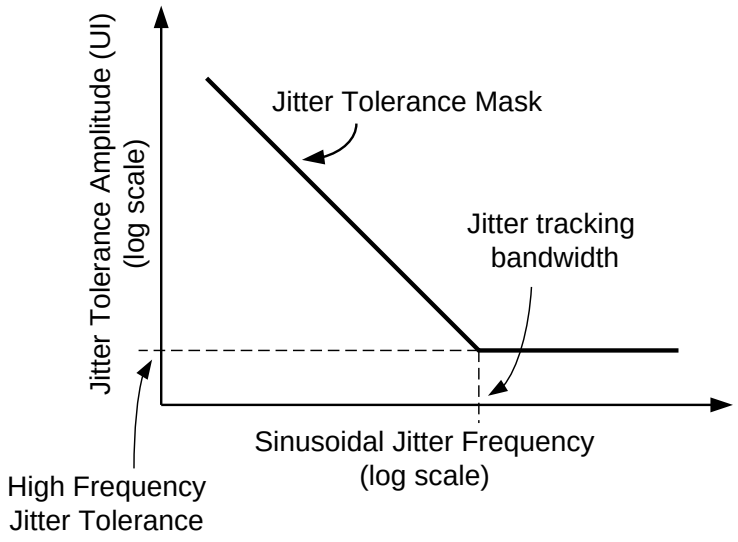


Figure 2.9: Sample jitter tolerance mask

Jitter tolerance is a measure of the jitter performance of receivers. It is defined as the maximum peak-to-peak amplitude of sinusoidal jitter that can be present on the data before the receiver BER exceeds a given threshold. Figure 2.9 shows a sample jitter tolerance mask. It is a two-dimensional plot with jitter frequency on the X-axis and jitter amplitude on the Y-axis. The BER of the receiver is within acceptable limits in the region of the plot to the left of the jitter tolerance mask. Since the jitter tracking of the CDR is limited by its slew-rate, at lower frequencies it is able to tolerate large jitter amplitudes and jitter tolerance decreases as frequency increases. At high jitter frequencies, the CDR circuit does not track the jitter on the data and the jitter eats into the timing margin of the receiver. Thus, the high-frequency jitter tolerance is a measure of the timing margin at the receivers. Higher jitter tracking

bandwidth of the CDR pushes the elbow of the curve to the right, while low-jitter sampling clocks pushes the curve upwards.

A related metric for CDR jitter performance is jitter tracking. Jitter tracking refers to the transfer function from the input data jitter to the sampling clock jitter. It measures how closely the clock recovery is able to track the jitter on the incoming data. Jitter tolerance and jitter tracking are closely related [27].

If $L(e^{-j\omega})$ is the open loop gain of the clock recovery feedback loop from input jitter $\Phi_{in}$ to the sampling clock phase $\Phi_{samp}$, the jitter transfer function is given by:

$$\frac{\Phi_{samp}}{\Phi_{in}} = \frac{L(e^{-j\omega})}{1 + L(e^{-j\omega})} \tag{2.7}$$

The jitter tolerance curve is given by:

$$\text{Jitter tolerance} = \left(1 - \frac{n \cdot \sigma_j}{T_{bit}}\right)\left(1 + L(e^{-j\omega})\right) \tag{2.8}$$

where, $\sigma_j$ is the rms jitter on the sampling clock and $T_{bit}$ is the bit period. The first parenthetical term in Eq. 2.8 is the timing margin for a given BER. Assuming Gaussian statistics for the sampling clock jitter $\sigma_j$, $n$ sets the horizontal eye opening and typically n=12 for a BER of $10^{-12}$. It is important to note that this analytical expression for jitter tolerance is derived from a linear model of the clock recovery loop, and is optimistic at low frequencies. At low frequencies, the large-signal slew-rate limited dynamics govern the jitter tolerance of the clock recovery loop.

The next section describes a very popular architecture for clock and data recovery: the semi-digital dual-loop CDR, proposed by Sidiropoulos et al. [26]. This will serve two purposes: First, the discussion will enhance our understanding of CDR performance metrics and the trade-offs between them; second, the clock recovery ar-

chitecture proposed in Chapter 3 of this thesis is directly inspired from the dual-loop CDR and this discussion provides a good background for the next chapter.

## 2.4 Case Study: Semi-Digital Dual-loop CDR

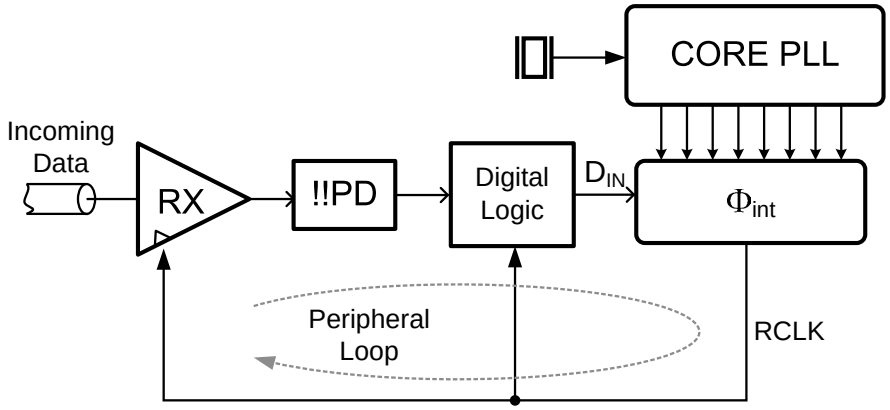


Figure 2.10: Semi-digital dual-loop receiver block diagram

Figure 2.10 presents a block diagram of the semi-digital dual-loop CDR. It consists of a cascade of two loops, namely, a core phase-locked loop (PLL) and a peripheral CDR loop. The PLL generates the clock phases that feed into the phase interpolator ($\Phi_{int}$) of the peripheral loop to control the phase of the output clock (RCLK) and align it with the incoming data. The bang-bang phase detector (!!PD) uses the sampled data to generate the sign of the phase error between RCLK and incoming data. The digital logic generates the control word $D_{IN}$ for the interpolator based on this phase error information. The negative feedback action forces the phase of the sampling clock (RCLK) to be at the center of the received data eye to maximize timing margin. To enable plesiochronous clocking between the transmitter and the receiver, the digital

control word rolls over to provide *infinite phase shift* capability.

This architecture is very simple and allows for independent tuning of the bandwidths in the two loops. Depending on the dominant sources of noise in the operating environment, the PLL bandwidth is set to minimize jitter on the output clock phases while the CDR bandwidth is tuned to meet jitter tolerance specifications. Also, the use of digital control in the CDR loop makes it less susceptible to process, voltage and temperature variations. Even as these merits have led to the widespread adoption of this architecture, it suffers from a few critical drawbacks.

First, there are conflicting bandwidth requirements for achieving optimal jitter performance from this CDR. For superior jitter tolerance in the receiver, the jitter tracking bandwidth of the CDR should be high while it should add little dithering jitter to the sampling clock.

The phase resolution of the phase interpolator is at the heart of this bandwidth conflict. The steady-state of this bang-bang controlled CDR is a limit cycle. This oscillatory steady-state manifests itself as a recovered clock dithering jitter that depends on the phase resolution of the interpolator and the latency in the feedback loop. A small phase step is needed to minimize dithering jitter. Ultimately, the largest phase step in every update interval of the CDR will dictate the magnitude of the dithering jitter in the recovered clock.

The maximum frequency tolerance and jitter tracking bandwidth of the CDR also depend on the phase step of the interpolator [14]. The maximum frequency tolerance is calculated to be:

$$\frac{\Delta f}{f} = \alpha_d \cdot \frac{\Delta T}{DF \cdot T_{update}} \tag{2.9}$$

where $\Delta T$ is the resolution of the phase interpolator, $T_{update}$ is the update period of the digital control word, DF is any decimation that may be applied to the control word, and $\alpha_d$ is the transition density on the data. To achieve good frequency tolerance, it is important that the phase step be large at every update of the control word, as this improves the slew-rate of the CDR. Also, as evidenced by Eq 2.8, the jitter tolerance bandwidth improves with increasing loop gain $L(e^{-j\omega})$. The loop gain is directly proportional to the phase step in the interpolator, and this also makes a case for high gain in the phase interpolator. This requirement is contrary to the need for a smaller phase step to minimize dithering jitter. In other words, this architecture suffers from a direct trade-off between jitter generation and jitter tolerance.

The frequency tolerance of this CDR is also a function of the data transition density. Low transition density limits the tracking ability of the CDR and can result in significant phase deviations of the recovered clock from the ideal sampling instant. Coding schemes like 8b/10b encoding guarantee minimum transition densities to overcome this issue, but they make inefficient use of the available channel bandwidth.

Finally, to use this architecture for receiving time-interleaved data signals, another multi-phase clock generator needs to be inserted between the phase interpolator and the samplers. This is an additional overhead, both in terms of power and area.

## 2.5 Conclusion: Motivation for a New Architecture

This chapter provided a high-level overview of various design choices available for implementing high speed parallel receivers, the performance limiters and the key metrics used to characterize these receivers.

To summarize these choices, in an ensemble of serial links (Figure 2.4), each transceiver has its own dynamic clock phase- and frequency-tracking loop, but this replication consumes excess area and power. These loops also suffer from conflicting loop bandwidth requirements to maximize their timing margins. Finally, low edge-transition density in the data streams limits the tracking bandwidth of these links. While encoding schemes like 8b/10b can ameliorate this issue, it comes with throughput and power overheads.

Source-synchronous links (Figure 2.3) are more area and power efficient as they do not require clock-recovery hardware. The dynamic phase-tracking bandwidth of these links depends on the correlation between the clock and data jitter, and is limited by the mismatch in path length between the sampling clock and data. However, to save clock power, the frequency of the forwarded clock is often stepped down and a multiplying PLL or DLL is used in the receiver to step the frequency back up. This can filter out some of the jitter and add additional latency, reducing the correlation in the jitter between the clock and data and degrading phase-tracking bandwidth. Moreover, limited channel bandwidth causes jitter amplification at high data rates [4], further reducing the correlation between the clock and data jitter.

Finally, source-synchronous links require forwarding a clock to the receiver and this consumes additional pin resources.

In the next chapter, we shall propose a new architecture that we believe is better suited for high performance microprocessor applications. It aims to maximize timing margin at the receivers by tracking data jitter over a wide bandwidth, while limiting the amount of dithering jitter. It neither requires a forwarded clock, nor the replication of clock recovery circuits per channel.

# Chapter 3

# A Collaborative Timing Recovery Architecture for Parallel Receivers

## 3.1 Introduction

This chapter describes a *collaborative timing recovery* architecture that seeks to merge the desirable attributes of source-synchronous links and ensemble of serial links to enhance jitter tracking in high-speed parallel links that reside in noisy digital systems. It aims to track wideband jitter on incoming data-streams without requiring either a forwarded clock or replication of clock recovery hardware. Given synchrony between parallel data channels, per-channel clock recovery is replaced by a single global timing recovery (TR).

Figure 3.1 shows the block diagram of a typical parallel transmitter. Multiple transmitter drivers share one clock generator to save power and area. As a result, the jitter on TxClk propagates to all transmitted data streams. Also, power supply noise

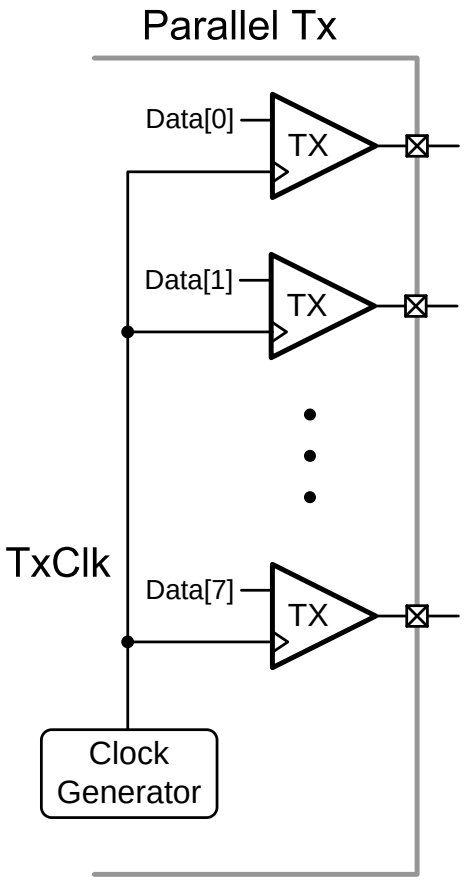


Figure 3.1: Typical parallel transmitter block diagram

exhibits strong spatial correlation and adds common-mode jitter to data-streams from transmitter drivers located close by. Thus, there is strong correlation in the low- and mid-frequency jitter on transmitted data-streams. By using just one global TR block that combines error information from multiple channels, the collaborative receiver tracks this correlated jitter. This collaborative architecture incurs approximately 10-15% area and power overhead when compared to source-synchronous links and is more power efficient than an ensemble of serial links. Collaborative timing recovery also greatly enhances the effective edge transition density, which enables higher bandwidth

in the TR loop—without increasing recovered clock jitter or susceptibility to long sequences of 1's and 0's—to track mid-frequency supply noise that plague large digital systems.

The remainder of this chapter is organized as follows. Section 3.2 presents the proposed receiver architecture, and Section 3.3 demonstrates its superior jitter performance in the presence of power-supply noise using numerical models. Implementation details of the main blocks follow in Section 3.4. The performance advantages of the proposed collaborative architecture are discussed in Section 3.5 based on experimentally measured results from the prototype chip. Lastly, Section 3.7 summarizes the chapter.

## 3.2 Proposed Architecture

The block diagram of the proposed receiver architecture is shown in Figure 3.2 [1]. It consists of multiple local receiver slices, one for each data channel of the parallel receiver, and one global timing recovery (TR) block. Each local receiver slice uses Alexander-type bang-bang phase detectors, and early/late timing error information from each of these detectors is sent to the global TR block. The global TR aggregates the error information and produces the recovered clock (Global RxClk) that tracks the frequency and the correlated jitter of the data signals. No synchrony is assumed between the reference clock and the clock on the transmitter, and the burden of frequency synthesis is placed on the global TR block. This recovered clock is distributed to the local receivers. The local receivers manipulate only the phase of the incoming global RxClk to compensate for static inter-channel skew. They also generate local

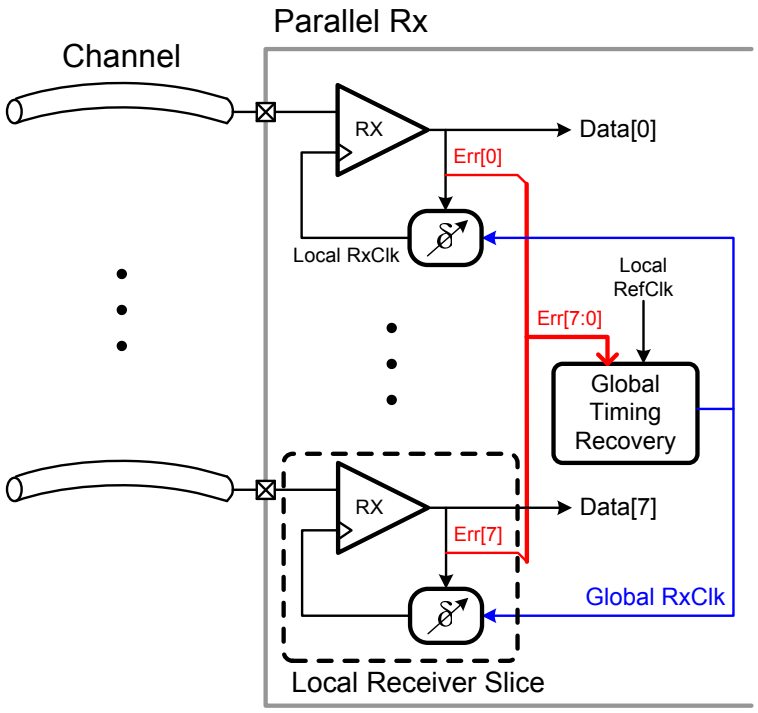


Figure 3.2: Receiver architecture.

multiphase clocks (Local RxClk) that drive local time-interleaved receiver samplers.

Figure 4.6 illustrates the details of the global TR block with respect to the design parameters used in the prototype chip. The popular dual-loop architecture [26] is employed for timing recovery. A phase-locked loop generates eight clocks evenly spaced by 45°, which are then used by the clock recovery loop to generate the synthesized clock. In the clock recovery loop, the early/late timing error signals from eight data channels are first aligned to a common clock, and then summed up in the phase error summer. The error information arrives once every two clock cycles to alleviate speed requirements of downstream digital circuitry at the expense of tracking bandwidth. The error sum is passed to a 2nd-order digital filter having both proportional ($K_p$) and integral ($K_i$) paths. Using 2nd-order control enables the loop

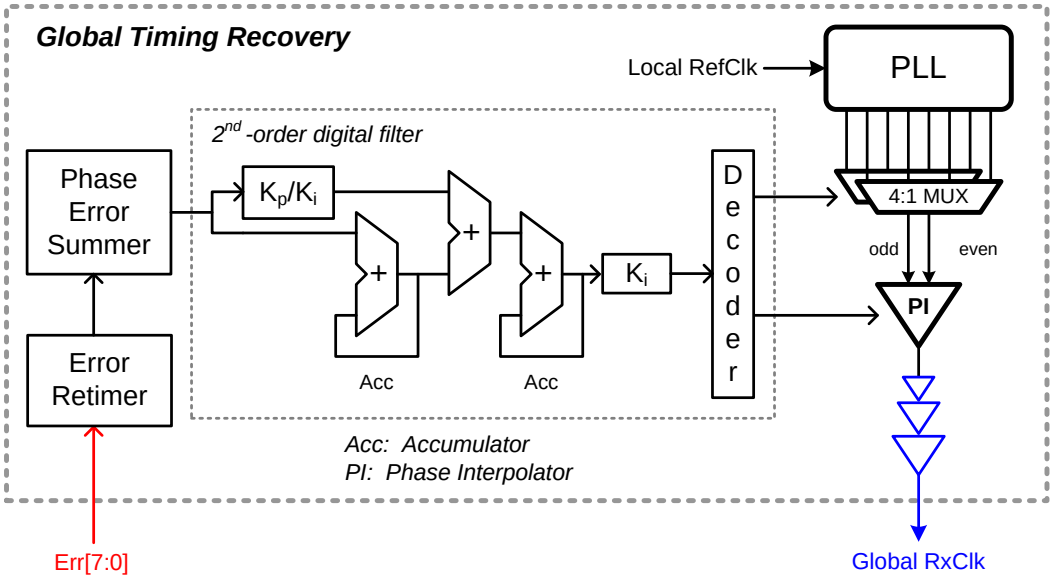


Figure 3.3: Global timing recovery block diagram.

to track frequency offsets between reference clocks in the transmitter and receiver, and extend the frequency-tracking range of the timing recovery loop [14]. Decoded bits out of the filter drive a clock synthesizer consisting of two 4:1 MUXes and a 5-bit phase interpolator (PI) to generate the Global RxClk.

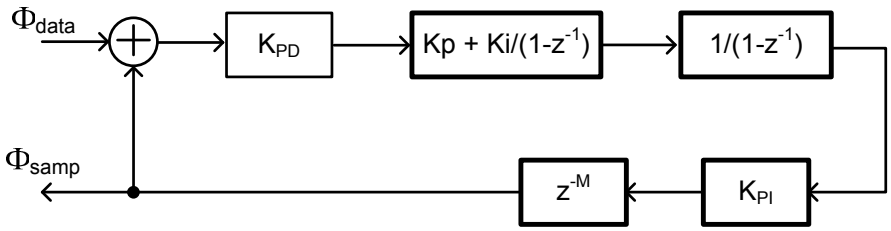


Figure 3.4: Linearized CDR model.

A linear analysis of the global timing recovery loop was used to set the $K_p$ and $K_i$ values, and ensure stability of the system. A number of recent works have developed

small-signal models for digital clock and data recovery (CDR) loops, as shown in Figure 3.4 [27, 14]. In this model, $K_{PD}$ is the linearized gain of the bang-bang phase detector (PD), $K_p$ is the proportional gain of the filter, $K_i$ is the integral gain of the filter, $K_{PI}$ is the gain of the phase interpolator, and M is the latency in the loop. The overall open-loop transfer function is:

$$L(z^{-1}) = \left(\frac{K_{PD} \cdot K_{PI}}{1 - z^{-1}}\right)\left(K_p + \frac{K_i}{1 - z^{-1}}\right) z^{-M} \tag{3.1}$$

The phase transfer function is given by:

$$\frac{\Phi_{samp}}{\Phi_{data}} = \frac{L(z^{-1})}{(1 + L(z^{-1}))} \tag{3.2}$$

where $\Phi_{samp}$ and $\Phi_{data}$ are the phase of the sampling clock and the data, respectively.

The collaborative timing recovery loop can also be modeled using Eq. 3.2, with the only difference being the representation of the PD. The models described in [27, 14] linearize the grossly non-linear transfer function of the PD by calculating its gain in the presence of sampling clock jitter. It has been shown that if the rms value of the clock jitter is $\sigma_j$, the linearized gain $K_{PD}$ is equal to $\frac{1}{\sqrt{2\pi}\sigma_j}$ [20]. The $K_{PD}$ for the collaborative loop is the effective gain for the ensemble of PDs. The original $K_{PD}$ is scaled by a factor of 8 since error from eight receivers are summed together. Also, the interleaved front-end samplers have small voltage offsets caused by inherent device mismatch in deep sub-micron process technologies. The spread in sampler offsets further linearizes the transfer function of the phase detector. This linearization does not come at the expense of the slew-rate of the timing recovery loop. This model is used to set the phase tracking bandwidth of the collaborative timing recovery loop.

Figure 3.5 presents details of each receiver slice, which consists of two cascaded

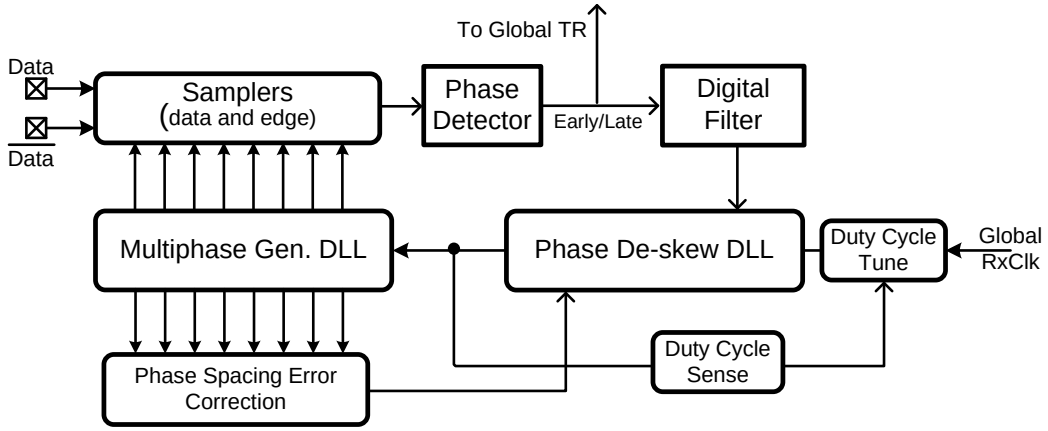


Figure 3.5: Local receiver slice block diagram.

DLLs, interleaved data and edge samplers, skew-compensation logic, and phase spacing error correction circuits. The global RxClk first goes through a duty cycle corrector (DCC) to compensate for distortion from the clock distribution buffers. The first DLL, called the phase de-skew DLL, adds a variable amount of delay to the clock path such that its output clock (Local RxClk) is phase aligned to the incoming data. The second DLL produces the eight evenly-spaced clock phases that drive eight interleaved edge and data samplers. The phase detector logic generates early/late information sent to the global TR block and to the digital filter in the local phase de-skew DLL. The filter controls a simple 4-bit thermometer encoded DAC to add offsets between the up and down currents of the charge pump, which translates to skewing the DLL output clock [32]. The digital filter uses $\Delta$-$\Sigma$ modulation to reduce quantization noise in this loop with phase filtering provided by the low-pass transfer function through the DLL ($I_{CP}$ to $\Phi_{out}$). The tuning range of the de-skew circuit is greater than $\pm$0.5 UI. Also shown in the figure is the phase spacing error correction

loop that corrects imbalances in the delays through each stage of the delay line due to any nonidealities in the reference clock entering the second DLL.

The cascaded-DLL architecture was chosen for a variety of reasons, although other architectures can also provide similar functionality. It only requires one phase of the global RxClk to generate the de-skewed local RxClk, which avoids having to distribute multiple clock phases with phase mixers at each receiver slice. DLLs exhibit substantially all-pass phase transfer characteristics and, hence, does not limit the global TR bandwidth. No extra preshaping delay buffers are required in the clock path before the second DLL used for multiphase clock generation. Lastly, this topology offers a convenient location for placing the duty cycle correction circuit. The duty cycle of the clock entering the second DLL is sensed and tuned to 50% via correction circuitry placed before the first DLL to not disturb the shape of the clock signal into the second DLL.

Given the large number of loops in this design, care must be taken to ensure that loops do not adversely interact with one another. Since DLLs in the local receiver slices do not introduce phase filtering in the clock path of the local receiver, they do not impose additional constraints on the phase-tracking bandwidth of the global TR loop. With a digital filter, local de-skew loop bandwidth can be programmable, and is set 2 orders of magnitude lower than the bandwidth of the global TR loop. Also, the local de-skew loop only operates periodically to compensate for static skew and temperature drift. This periodic operation minimizes additional dithering jitter on the sampling clock, which would otherwise occur as a result of the global TR loop trying to track the combined dithering of the parallel de-skew loops.

We must also ensure that the global timing recovery loop does not interfere with the functioning of the local skew-compensation circuits. If the global loop and all of the local loops are active, there needs to be a system-level mechanism that ensures that the global loop does not push the local de-skew control words to their limits, preventing them from further tracking the inter-channel skew. The global loop initialization steps achieve this without incurring any additional circuit overhead. First, the collaborative link is programmed as a 1-channel CDR and only one receiver slice contributes timing error information to the global TR loop. For this slice, the local skew-compensation loop is disabled and the control-word (i.e. output of the accumulator) is reset to 0. Once the global loop acquires the phase and frequency of the data on this channel, the other receiver slices perform their skew-compensation. After this is complete, the receiver slices start contributing phase error information to the global TR loop. The skew-compensation logic in the first slice remains disabled, and can be thought of as the "master slice" that pins the phase of the global recovered clock, and prevents phase wander with respect to the input data phases.

## 3.3 Noise Analysis

A simple noise analysis can highlight the differences in phase-tracking performance of serial links, source-synchronous links, and the proposed collaborative timing recovery. Time-domain simulations, based on a quad-tree model, shown in Figure 3.6, elucidate link performance trade-offs across different noise scenarios given supply-induced jitter on the transmitted clock and data streams. In the model, total jitter is divided into three components: low-frequency jitter ($N_{LF}$) centered around 1 MHz;

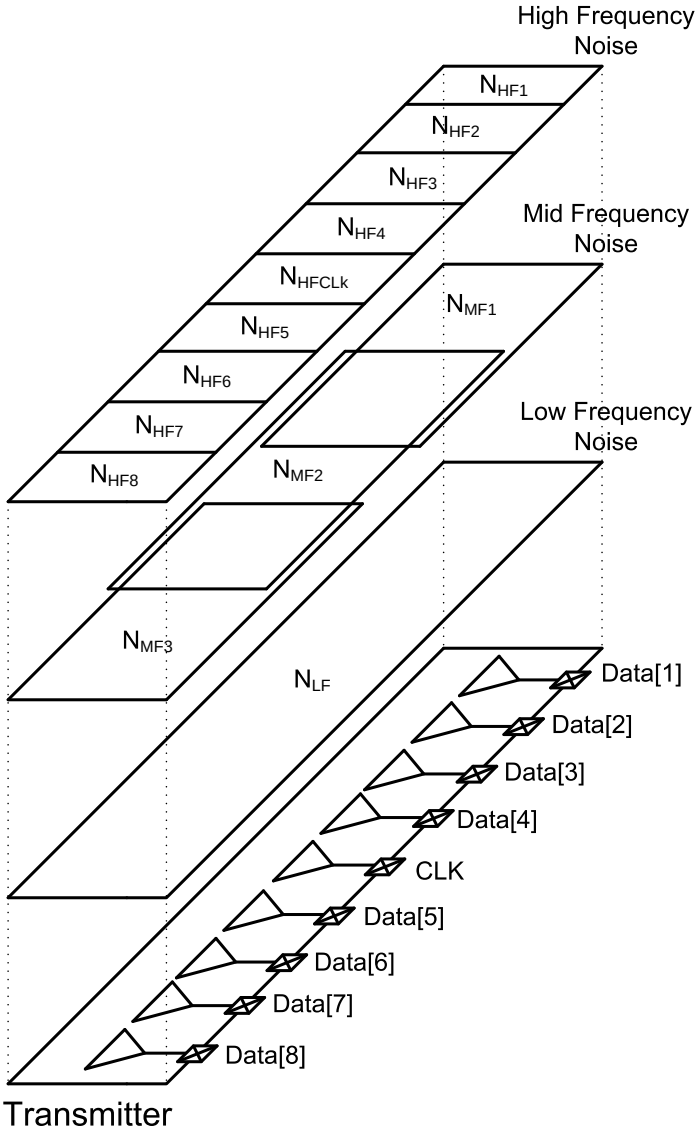


Figure 3.6: Quad-tree model for jitter injection in a parallel transmitter.

mid-frequency jitter ($N_{MF}$) centered around 30 MHz; and high-frequency jitter ($N_{HF}$) centered around 1 GHz. The model assumes low-frequency jitter is correlated across all channels, because it stems from a shared transmit-side PLL. Mid-frequency jitter incorporates the effects of spatially-correlated noise across adjacent groups of channels due to mid-frequency resonance in the impedance of the power-delivery network, derived from analytical models of processor on-chip power-delivery networks that show mid-frequency noise exhibits spatial correlations [12]. High-frequency jitter is not correlated across channels. We evaluate three scenarios with low-, mid- and high-frequency jitter dominating the total jitter, respectively. Figure 3.7 plots the simulated rms values of phase-tracking error for four different link configurations and three noise scenarios, assuming a data rate of 5 Gb/s and 0.2 UI p-p jitter added to the data streams.

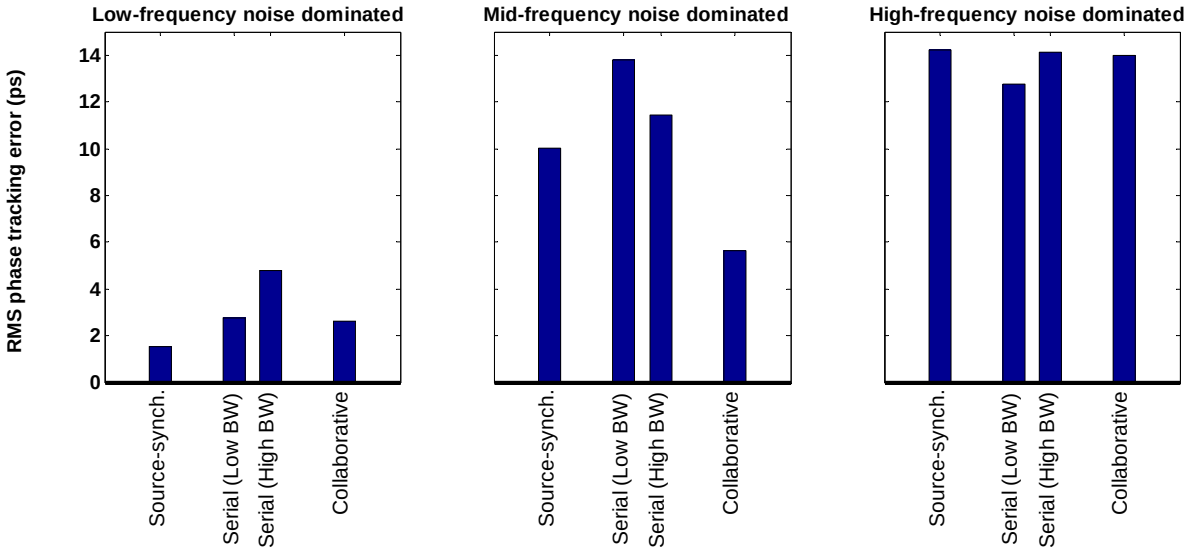


Figure 3.7: RMS phase-tracking error for different parallel link configurations across three noise scenarios.

For the low-frequency jitter dominated case, we find that all links have acceptable

phase-tracking performance. The higher BW serial link, whose bandwidth is the same as that of the collaborative link, has higher dithering jitter. The links also have comparable performance in the high-frequency jitter dominated case, as the receivers filter out most of the jitter on the incoming data. The mid-frequency noise dominated scenario is interesting. The source-synchronous link exhibits high phase-tracking error because mid-frequency jitter on the clock and data channels is not strongly correlated for data channels not located close to the clock channel. The low bandwidth serial link filters out the mid-frequency noise and the high bandwidth serial link has prohibitively high dithering jitter, leading to conflicting trade-offs. In comparison, the collaborative link tracks the average mid-frequency jitter across all of the channels with low dithering jitter and exhibits the lowest amount of phase-tracking error. While the collaborative architecture requires 10-15% area and power overhead when compared to source-synchronous receivers, it is more power efficient than serial links with high phase tracking bandwidth. Trends for on-chip power supply noise over the past few years show that the mid-frequency component of the noise is growing [3], which in turn leads to higher amounts of mid-frequency jitter in the transmitted data signals. Thus, it is important for high-speed receivers to have the ability to track mid-frequency jitter and motivates this collaborative architecture.

Although the analysis above only uses discrete values of jitter frequencies, it provides insight into the behavior of the collaborative architecture. The phase tracking error is low for jitter frequencies within the tracking bandwidth of the loop and when noise across the channels are correlated. However, when the jitter is uncorrelated, or beyond the tracking bandwidth of the loop, performance degrades. This suggests de-

signing the tracking bandwidth of the collaborative timing recovery loop to be greater than the mid-frequency noise frequency, which has been in the 100-300 MHz range for multiple generations of modern microprocessors. While the test-chip prototype, implemented in a 130nm technology, exhibits much lower bandwidth (1-10MHz), it ought to scale well with technology. Higher clock rates, lower-latency clock buffers, and lower degrees of interleaving can enable the collaborative architecture, implemented in aggressively-scaled technologies, to track mid-frequency jitter with low tracking error. We shall revisit this argument again towards the end of this chapter, after analyzing the measurement results.

## 3.4 Circuit Design

This section describes the implementation details of the building blocks for both the global timing recovery loop and the local receiver slices.

### 3.4.1 Global Timing Recovery Loop Components

We begin with the main components of the global TR loop. The error summer, digital filter, and decoder all rely on digital circuitry, implemented with a standard digital CAD flow. We used an existing PLL design in the same technology to generate the eight phases needed in the clock synthesizer. This section shall describe the design of the digital filter, phase interpolator, and error-retimer.

The error summer produces a 7-bit word every 8 UIs. Since we add early/late error information from 8 channels over 8 UIs, a total of 7 bits are required to represent the error word. Figure 3.8 presents a block diagram of the digital filter. In the

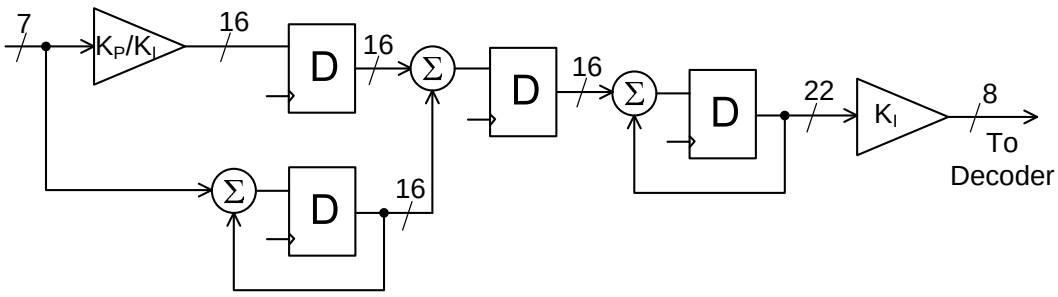


Figure 3.8: Digital filter design

proportional path, the error word is first scaled by $\frac{K_P}{K_I}$. The core accumulation and addition functions are then performed with adder sizes chosen to prevent overflows. Truncation of the LSB's of the final accumulator implements the final multiplication by $K_I$. This truncation could have been implemented in two stages without any loss of accuracy, by throwing away some LSBs after the first accumulation and then some more after the second. This would have resulted in a less complex digital filter. Also, the delay inserted in proportional path adds latency in the loop without providing any benefits.

The 5-bit phase interpolator, combined with the eight VCO clock phases, provides a total of 256 phase steps per clock period. The unmodified digital interpolator shown in Figure 3.9 uses weighted inverters to mix two phases and generate a full-swing clock. Thermometer-weighted inverters are used for the MSBs to minimize the impact of process mismatch in order to achieve a monotonic phase-versus-code characteristic. One drawback of this digital interpolator is that large RC time constants at nodes A, B, and C are desirable such that the clock transition times are roughly 2-3x the VCO phase spacing to provide good linearity. However, this reduces the signal swing

at node C. Also, due to mismatched drive strengths between the PMOS and NMOS devices in the inverters and process variations, the DC voltage at node C can shift away from the switching threshold of inverter Z. The shifted DC voltage and reduced signal swings can cause severe duty-cycle distortion out of inverter Z or failure to switch at all. A remedy is to add a series capacitor before inverter Z and a feedback resistor to self-bias its input voltage to precisely the switching threshold. With the series capacitor, even if node C sees a large DC shift with small signal swing, node D will be biased to the optimal voltage for inverter Z to amplify the signal with minimal duty-cycle distortion. The series capacitance is large (100fF) with respect to the gate capacitance of inverter Z (2fF) to minimize signal attenuation, and the feedback resistance is large ($90k\Omega$) to minimize static current draw. Interleaved metal-fingers comprise the series capacitor, which introduce little parasitic capacitance and negligible amounts of additional power.

During the operation of the clock recovery loop, in steady-state the LSB of the control word to the PI exhibits dithering, that leads to additional jitter in the output clock of the PI. In this implementation where we use a multiplexer for the 3 MSB's there is the possibility that the control word dithers between two words that dithers the MSB bits and toggles the PI control word from one extreme to another (for e.g. 10011111 and 10100000). A large phase step may occur at this interval, resulting from the PI operating in a different operating region, and can lead to excessive dithering jitter on the output clock. This can be avoided by a simple code-remapping scheme as illustrated in Figure 3.10. The key observation is that when the code is dithering between such words, the output clock phase is weighted almost completely towards

one of the input clocks. Thus the code-remapping leaves that input clock and the PI control word untouched, while changing the other input to the PI.

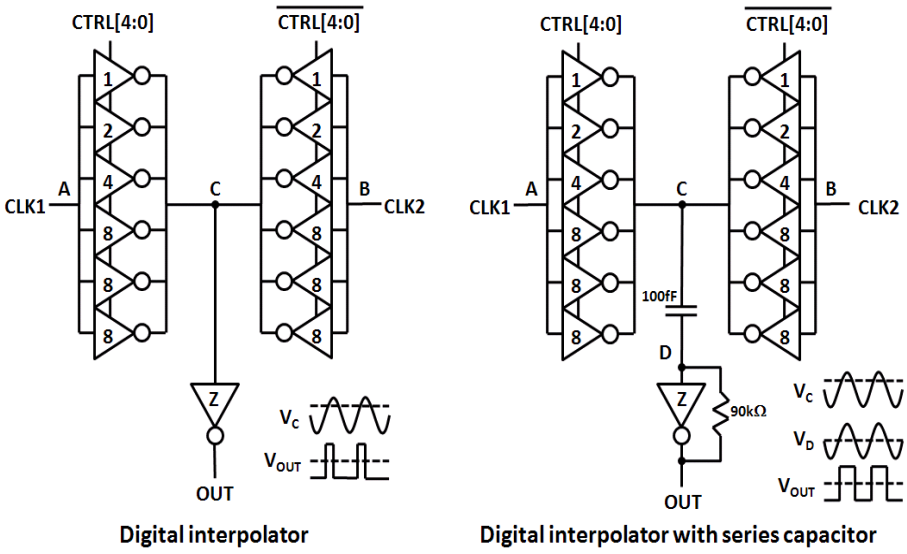


Figure 3.9: Phase interpolator.

Figure 3.11 illustrates the error-retimer circuit in the global TR block to align error information from multiple receiver slices to its local clock domain. This circuit is required since the arrival times of the E/L signal to the global TR block depends on the timing in each local Rx slice and the propagation delay from the local slice to the global TR block. Since frequency is matched, this circuit samples the incoming E/L[n] signal with a pair of rising-edge clock signals separated by more than the setup-and-hold window of the flip-flops. A mismatch between the two samples tells the XOR based logic to select the output data sampled on the falling clock edge.

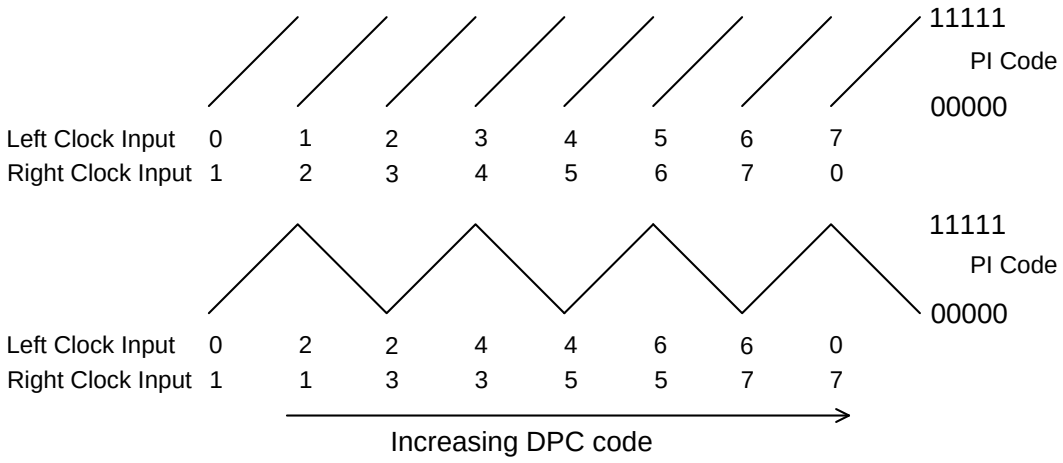


Figure 3.10: Phase interpolator code re-mapping.

### 3.4.2 Local Receiver Slice Components

The details of the local receiver building blocks are discussed next. The digital logic for the phase detectors and loop filter in the local receivers were created using a standard digital CAD flow. First order $\Delta$-$\Sigma$ modulation is used to shape the quantization noise in the digital control word to the phase de-skew DLL DAC.

Nearly identical DLLs are used for phase de-skew and multi-phase clock generation. Figure 3.12 shows the details of the DLLs.

In the multi-phase clock generating DLL, the incoming clock goes through a delay line consisting of 4 differential buffers. Each buffer produces differential clocks, resulting in a total of 8 phases to drive the 8 interleaved samplers. The phase detector (PD) in the loop locks the delay of the delay line to 180°. The PD is designed to have only 2 states, to avoid the false-lock problem associated with using phase-frequency detectors in DLLs [29]. When the loop is in lock, the PD generates narrow UP and DN pulses of equal widths to prevent a dead zone in its transfer function. An active

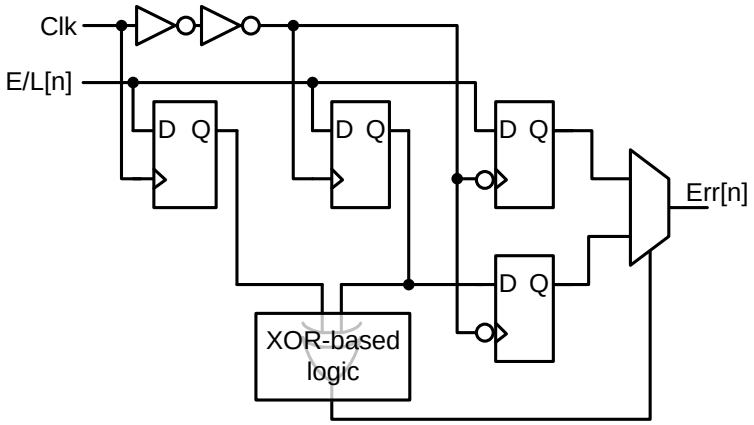


Figure 3.11: Error retimer.

loop filter with a feedback amplifier biases the output node of the charge pump to $V_{REF}$, irrespective of the delay of the delay line. Changing $V_{REF}$ changes the UP and DOWN currents of the charge pump, due to the finite output impedances of the current sources. Thus, small mismatches in the charge pump UP and DOWN currents can be compensated by adjusting $V_{REF}$. In this DLL, the DAC connected to the UP current is always set to its mid-value. We had implemented a digital feedback loop to adjust the DAC output current to correct for static charge pump current mismatches, but this circuit failed to operate correctly in our test chips.

The phase de-skew DLL uses the same topology, with a few subtle differences. The mismatch in the charge pump currents and PD offset do not matter since this DLL's function is to add skew to the input clock, and these offsets can be absorbed by the phase-deskew loop. The 4-bit thermometer-coded DAC in the UP current of the charge pump is used to add the variable skew. This DLL can add up to 1.5 UI's of skew when operating at 5 Gb/s. To improve the timing resolution of the phase-deskew beyond 4 bits and achieve better timing margins, $\Delta$-$\Sigma$ modulation is employed

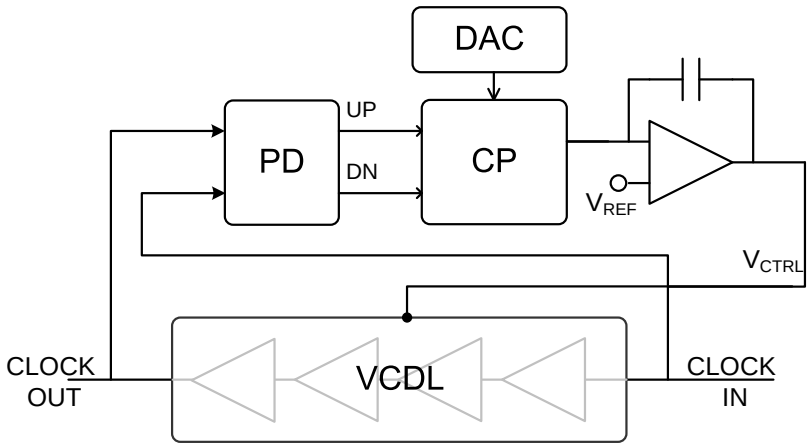


Figure 3.12: Delay-locked loop block diagram

to effectively achieve more than 7 bits of resolution. The filtering is provided by the pole in the DLL feedback loop as shown in Figure 3.13. The location of dominant pole is given by Eq. 3.3:

$$\omega_n = \frac{I_{CP} K_{VCDL} F_{REF}}{sC} \tag{3.3}$$

where $I_{CP}$ is the nominal charge pump current, $K_{VCDL}$ is the gain of the delay line, $F_{REF}$ is the frequency of the reference clock and C is the loop capacitor. The dominant pole is designed to be at around 4 MHz, providing an over-sampling ratio $\approx 80$. The higher-order poles introduced by the amplifier further suppress the shaped quantization noise, and reduce residual jitter.

The delay cell uses pseudo-differential current starved inverters with rail-to-rail swing, and is shown in Figure 3.14. The current source is split into a 4X device and a 1X device that are controlled by the coarse and fine voltage control signals, respectively. In this test chip, the coarse control voltage is set manually and the feedback loops determines the fine control voltage. In a real system, a peripheral

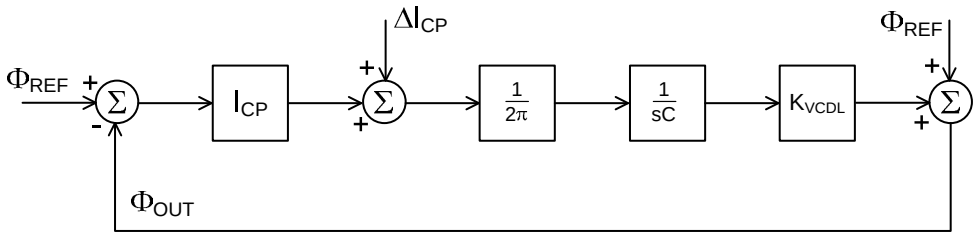


Figure 3.13: Frequency-domain model of DLL illustrating transfer function from $I_{CP}$ to $\Phi_{OUT}$

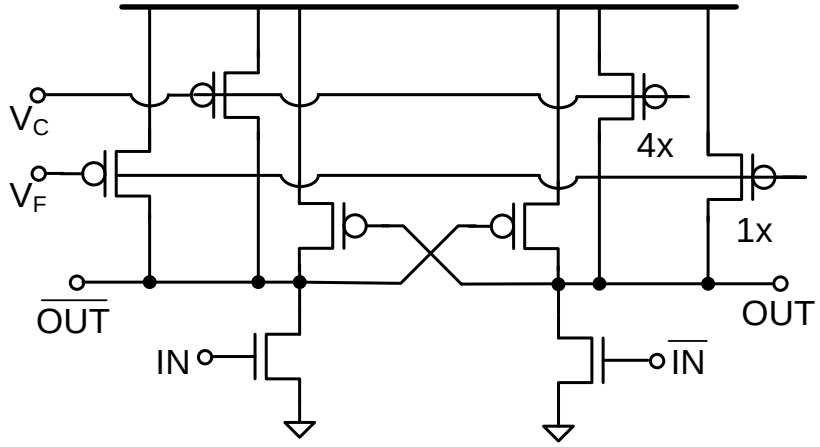


Figure 3.14: Delay cell schematic

loop can be designed to set the coarse control voltage during the initialization of the receiver. The simulated coarse and fine gain of the delay line are 800 ps/V and 200 ps/V, respectively. While these delay cells produce nearly full-swing clock signals, they have asymmetric rise and fall times due to the difference in strengths of the pull-up and pull-down paths. To correct for this, duty-cycle restoring buffers similar to those described in [14] are used at the output of the delay buffers to drive the samplers.

Series-connected sense amplifier based samplers, shown in Figure 3.15, are em-

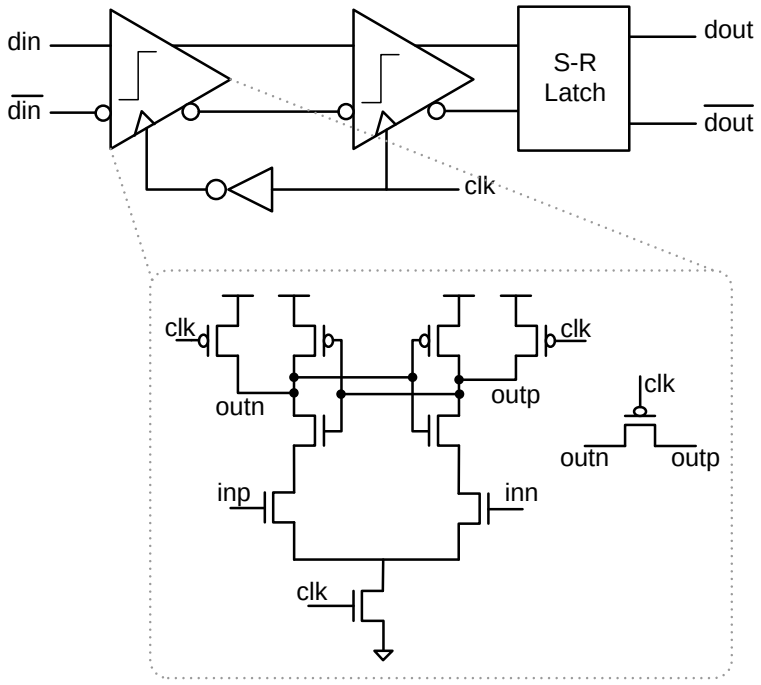


Figure 3.15: Sampler schematic

ployed in this design [19]. The sense amplifier operates in two phases. In the reset phase, when CLK is low, the tail transistor is off, and the *outp* and *outn* nodes are charged to *Vdd*. Another PMOS device shorts the two output nodes and ensures that the differential voltage at the output node collapses to 0 during reset phase. In the evaluation phase, the tail transistor turns on and the output nodes start getting discharged. A differential voltage develops between outp and outn depending on the input voltage. One of the node discharges faster than the other, triggering regeneration in the sense amplifier. The evaluation phase of the second sense amplifier overlaps that of the first by an inverter delay. This increases the overall evaluation time to 4UIs, and also improves the voltage gain of the sampler. An SR latch ensures that the output is held steady even when the second amplifier is in the reset phase.

## 3.5 Measurements and Results

A test chip was fabricated in the UMC 130nm CMOS logic process and the die photo is shown in Figure 3.16. The floor plan of each local receiver slice is shown in Figure 3.17 for additional detail. The chip is 5mm x 1.25mm and occupies a total area of 6.25 $mm^2$, and an active area of 3.2 $mm^2$. Layout considerations for the 8 local receiver slices dictate the choice of aspect ratio for the test-chip.

The die is bonded directly to a 4-layer test board without using a package to minimize parasitics. This is critical as no equalization is included in this receiver test chip and maximum signal integrity of the data signals is desirable at the sampler inputs. To enable direct chip-to-board bonding, the landing pads on the PCB are plated with soft gold. The die sits on a landing-pad that grounds the substrate and spreads the heat out of the chip. The die and the bond wires are encapsulated in a non-conducting epoxy to prevent mechanical damage to the fragile assembly.

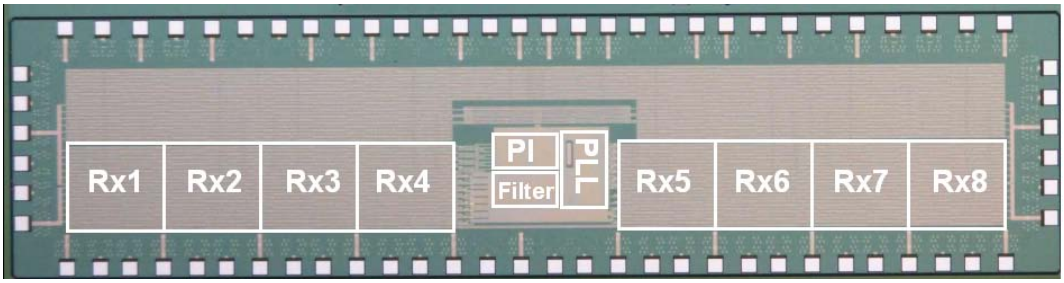


Figure 3.16: Die micrograph with floor plan overlay.

To test this chip, up to 8 synchronous PRBS data signals are required with the ability to add controlled amounts of correlated jitter to the data signals. While the simplest way to achieve this is to use a high-speed parallel bit error rate tester

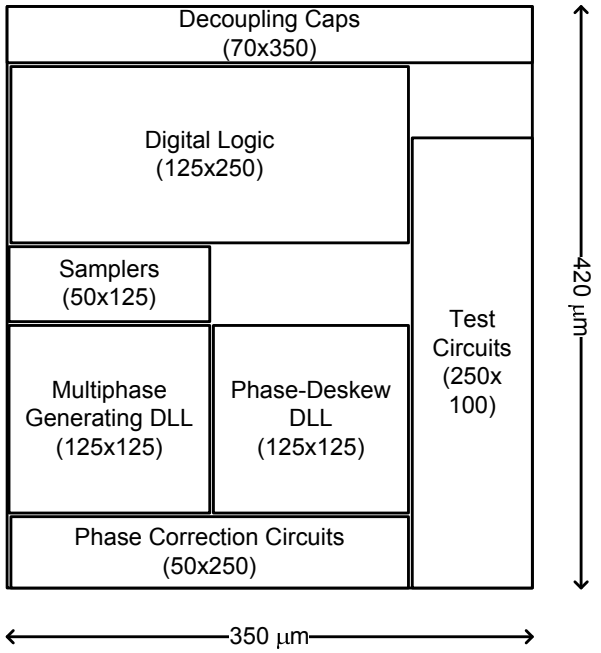


Figure 3.17: Local Rx slice floor plan.

(parBERT), we rely on two lower-cost options available to us.

For initial tests, a Tektronix 5334 parallel data generator was used. It consists of a mainframe with four slots that can produce up to 8 lanes of data. The mainframe allows users to program the data on each of the channels, and we used PRBS sequences of 7- and 15-bit coefficients for our tests. It allows for the use of an external clock to clock the data generators and we produce a jittered clock to inject correlated jitter on all 8 data-lanes. Its maximum data rate is 3.35 Gb/s and we performed a comprehensive set of tests at 3.2 Gb/s to verify functionality and performance of the test-chip at lower data rates.

To test the chip at higher data rates, we rely on an FPGA-based parallel transmitter with the ability to add correlated jitter on multiple data streams as shown in

Figure 3.18, consisting of a Xilinx Virtex4 FPGA with *RocketIO* transceivers that can generate multiple parallel data streams. Correlated jitter is added by combining voltage noise to one of the differential clock input signals using a power combiner/splitter. Inside the FPGA, the output of the differential to single-ended converting buffer is a clock with jitter. This clock is then frequency multiplied by a PLL and distributed to the transceiver slices. The -3 dB bandwidth of this PLL is in the 20-30 MHz range, facilitating the injection of relatively wideband jitter to the transmitted data. We used an evaluation board (ML423) for the FPGA to provide us easy access to the *RocketIO* pins through SMA connectors.

The local reference clock for the test chip is generated from a separate clock generator. This allows the addition of a frequency offset between the clock and data. The recovered clock is driven to a sampling scope to measure the jitter histogram. An on-chip PRBS verifier is used to perform BER measurements. Due to a small bug in the verilog code for the PRBS verifier – the bit-error counter was non-saturating – we were unable to count the number of bit-errors; we could only detect the presence or absence of bit-errors. Thus, we could not plot bath-tub curves for the receiver timing-margins and we only report jitter-tolerance measurements to demonstrate reliable operation of the receiver.

### 3.5.1 System Results

Several parameters can be swept in the test-chip to perform a comprehensive set of experiments. The receiver can be configured to combine error information from different numbers of channels and have different settings for $K_p$ and $K_i$ in the digital

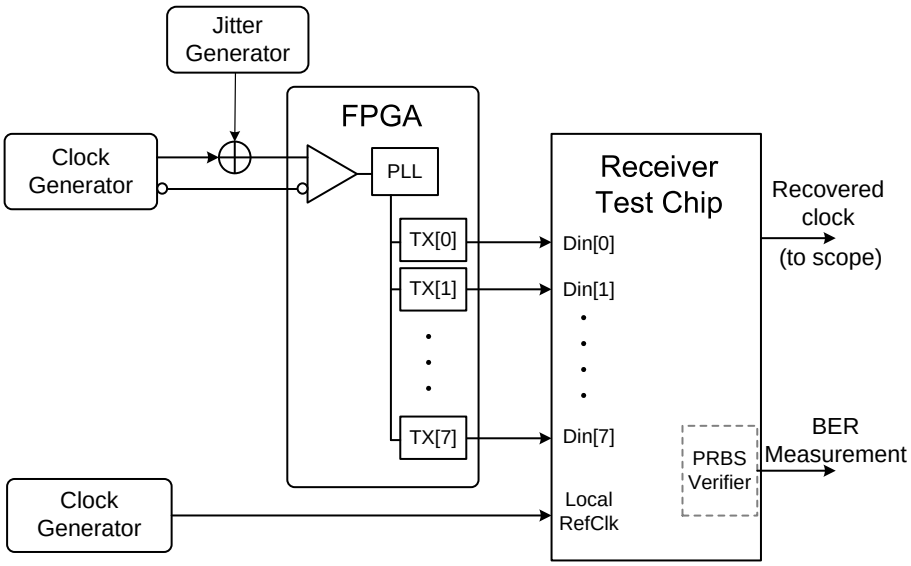


Figure 3.18: Experimental Setup.

filter of the global TR. Different frequency offsets can be added to the local reference clock of the receiver with respect to the transmitter clock, and PRBS data of different run-lengths can be transmitted to the receiver. Figure 3.19 plots the jitter histogram of the global recovered clock (triggered by the transmitter clock) when the global TR is combining error information from 7 channels, $K_p = 2^{-5}$, $K_i = 2^{-11}$, 7 bit PRBS data, and no frequency offset. Access to the 8th channel is unfortunately blocked by insufficient connector spacing on the board. When operating at 5 Gb/s, The rms value of the recovered clock jitter is 5.24 ps.

### Bandwidth Measurements

To demonstrate the merits of the collaborative timing recovery architecture, we first performed a series of jitter tolerance and loop bandwidth measurement experiments. These results demonstrate the ability to easily increase the jitter tracking

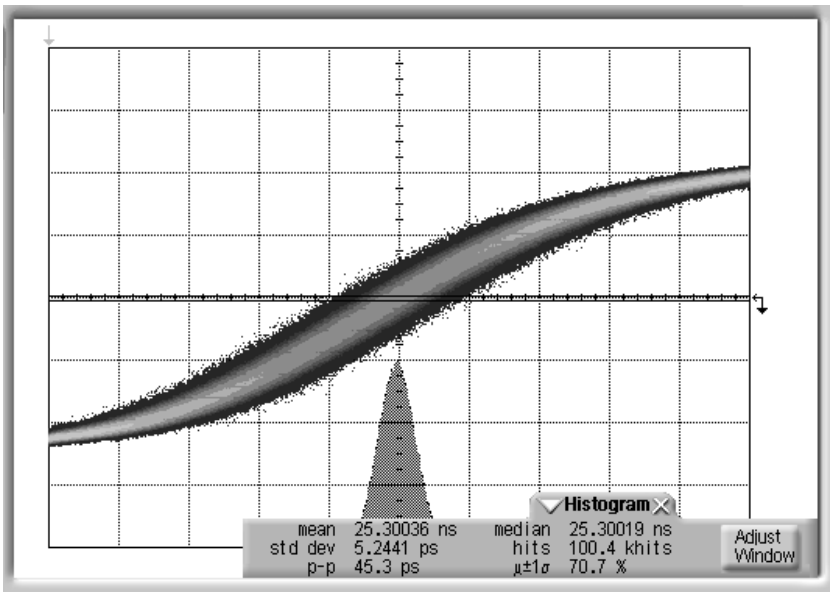


Figure 3.19: Recovered clock jitter measurement.

bandwidth of the timing recovery loop.

The low-speed test setup (at 3.2 Gb/s) allowed us to perform jitter tolerance experiments. Figure 3.20 plots the jitter tolerance vs. loop proportional and integral gain coefficients, for the receiver configured as a 1 channel CDR. The $K_p/K_i$ ratio is kept fixed at $2^6$. As expected, the loop bandwidth increases as we increase the gain coefficients. We also observe some jitter peaking in the transfer function, evidenced by degradation in the jitter tolerance near the elbow of the curves. This jitter peaking can be attributed to additional latency added to the global TR loop in the latter stages of the design, and our choice of $K_p/K_i$ ratio.

The same increase in jitter tolerance bandwidth can be obtained by keeping the loop coefficients fixed and increasing the number of channels sharing timing information, as shown in Figure 3.21 Adding timing error information from multiple channels

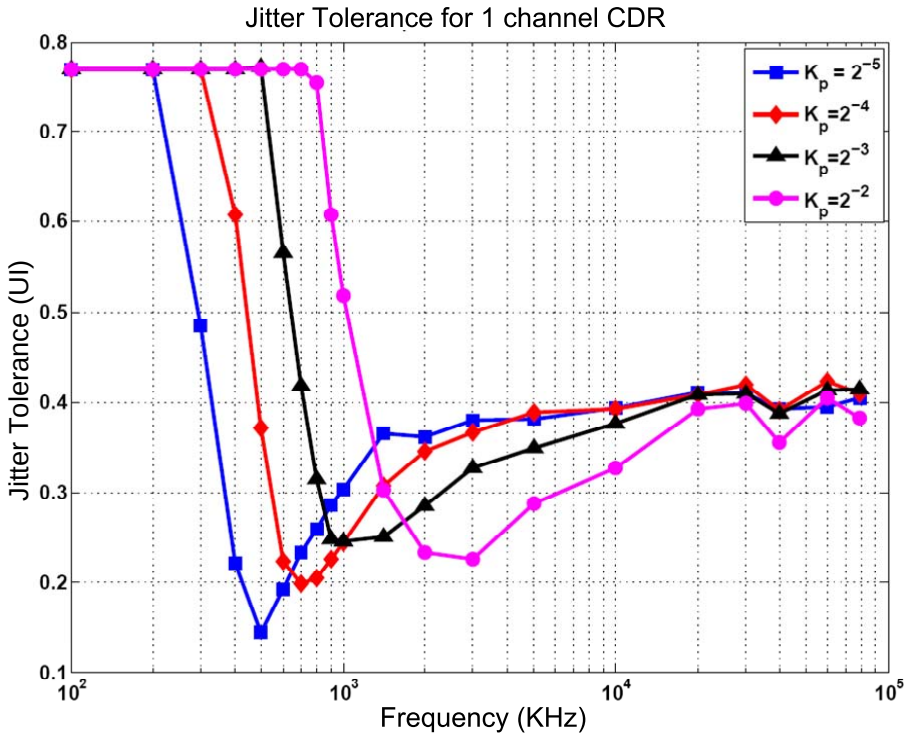


Figure 3.20: Jitter tolerance vs loop coefficients for 1 channel CDR

is equivalent to increasing the effective gain of the phase detector.

Instead of simply summing up error information, we can perform averaging to keep the effective gain of the phase detector constant, as we increase the number of channels sharing timing information. This holds the jitter tracking bandwidth of TR loop roughly constant. Figure 3.22 shows the jitter tolerance for this experiment. The jitter tolerance plots match our intuition, and we observe that the loop bandwidth is almost identical in all 4 cases. However we see that loops with higher degrees of collaboration perform better at middle and higher frequencies. At middle frequencies (1- 10 MHz) the jitter tolerance improves as the number of channels sharing error information increases. This improvement in jitter tolerance suggests lowering jitter

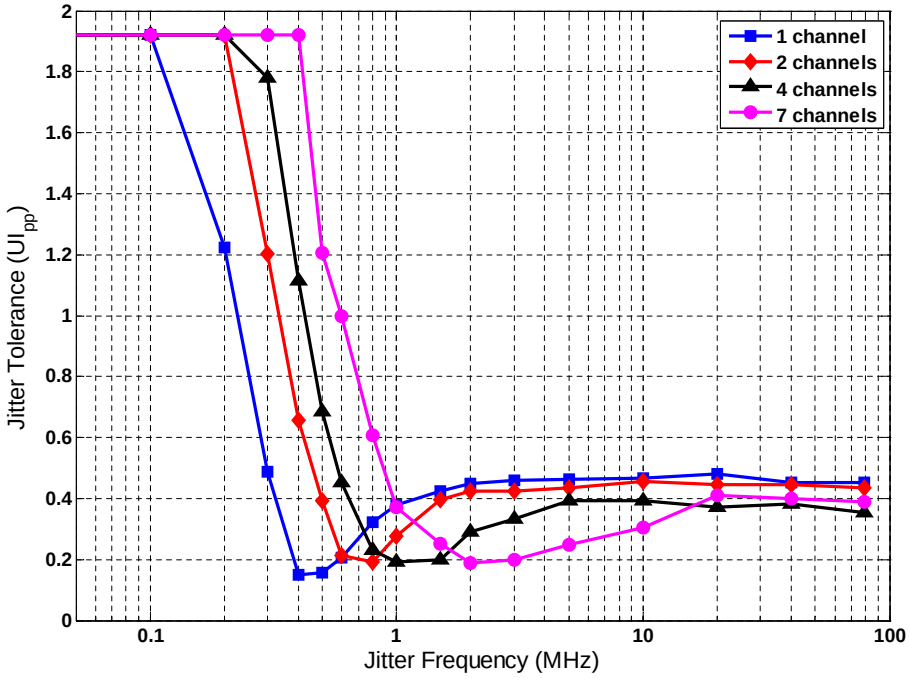


Figure 3.21: Jitter tolerance vs no. of channels for collaborative CDR

on the recovered clock, demonstrating a key merit of this architecture. This trend is not consistent at higher frequencies, and this can be attributed to imperfections in our test setup.

At 5 Gb/s, it was not possible to perform full jitter tolerance tests, because the outputs of the transmitters in the FPGA had bit errors if the input clock jitter amplitude exceeded half of a UI. Thus, to measure loop bandwidth, Figure 3.23 plots the sideband in the spectrum of the recovered clock, relative to a clean clock, when correlated wide band jitter is added to the transmitted data. The number of channels sharing timing information is swept while the $K_p$ and $K_i$ settings of the global TR block are fixed to $2^{-5}$ and $2^{-11}$, respectively. Shown in the plot is the spectrum of the free-running PLL, and the recovered clock for different degrees of collaboration. The

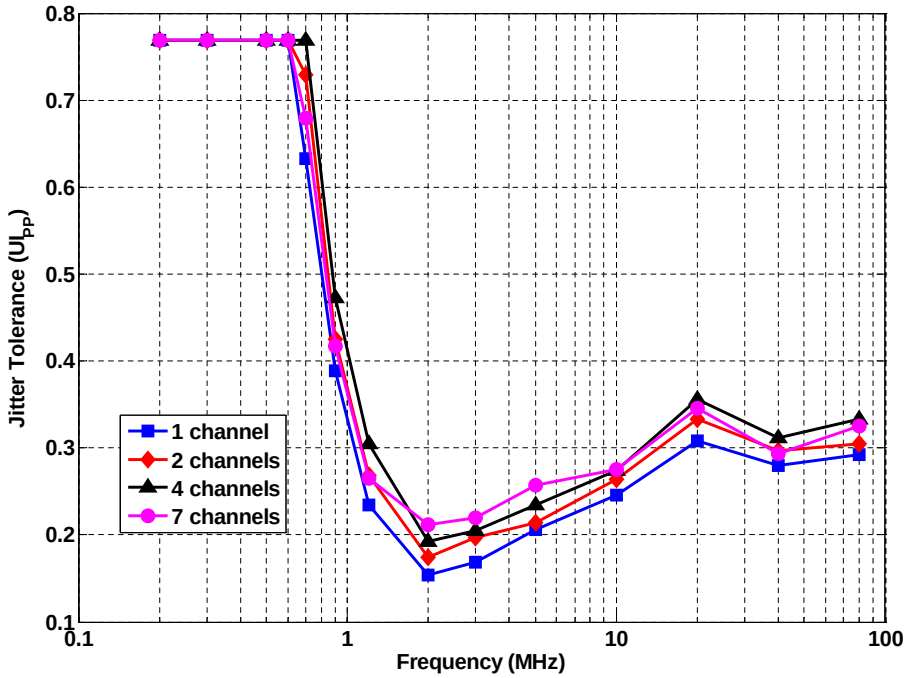


Figure 3.22: Jitter tolerance vs no. of channels for collaborative CDR

plot shows that the jitter tracking bandwidth of the receiver improves as the number of channels that contribute timing error information increases from 1 to 6. Results for 7 or 8 channels could not be obtained due to limitations of the FPGA-based test setup.

### Jitter Measurements

As mentioned above, the jitter tracking bandwidth of the global TR loop can be increased by increasing the gain coefficients in its digital filter, while fixing the number of channels that share timing information. Figure 3.24(a) plots the dithering jitter on the recovered clock for the two scenarios. Shown in the dark line is the case where the number of channels is fixed to 1 and the proportional and integral

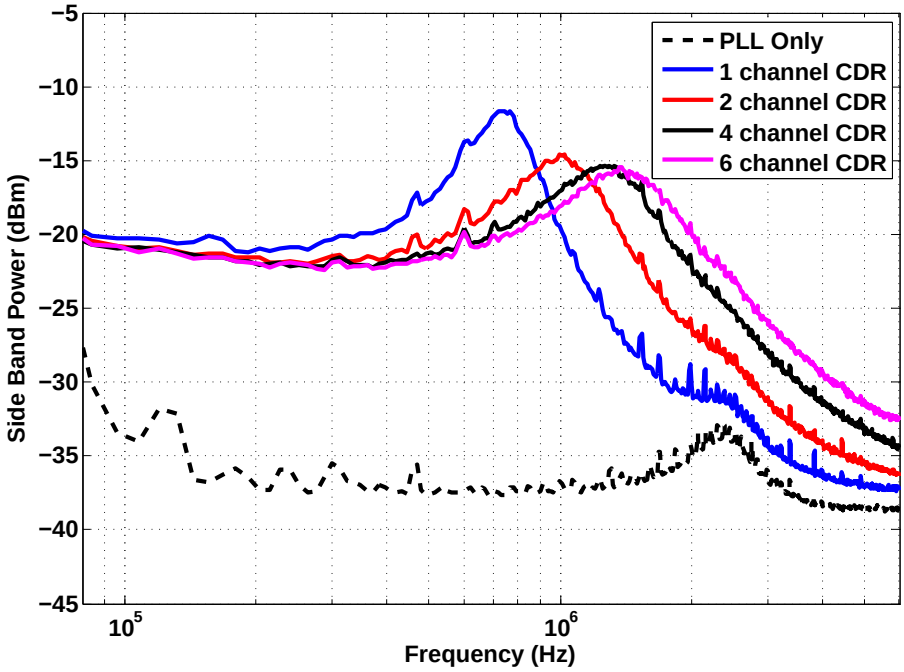


Figure 3.23: Sideband of the spectrum of recovered clock when wideband jitter is added to the data-streams for different number of channels sharing timing information.

gains are increased to increase the loop bandwidth. The $K_p/K_i$ ratio is fixed to $2^{-6}$. As expected, we see that as the coefficients are increased, the dithering jitter also increases from 2.7 ps to 3.8 ps. The other scenario, plotted using the red line, shows the dithering jitter for the collaborative TR. We sweep the number of channels that share timing information while keeping the gain coefficients fixed. In this case, we see that as the bandwidth of the loop increases, the dithering jitter reduces from 2.7 ps to 2.2 ps. The numbers reported in this plot indicate only the dithering jitter that is obtained by subtracting the baseline PLL jitter of 4.7 ps from the total clock jitter. This plot confirms that increasing the effective edge transition density via collaboration enables the timing recovery loop to operate with higher tracking bandwidth settings while reducing dithering jitter. We also observe that increasing

the degree of collaboration beyond a certain number of channels yields diminishing returns. To further demonstrate this, Figure 3.24(b) shows dithering jitter measured on the recovered clock reduces as the number of channels increases while tracking bandwidth is held roughly constant by modifying gain coefficients.

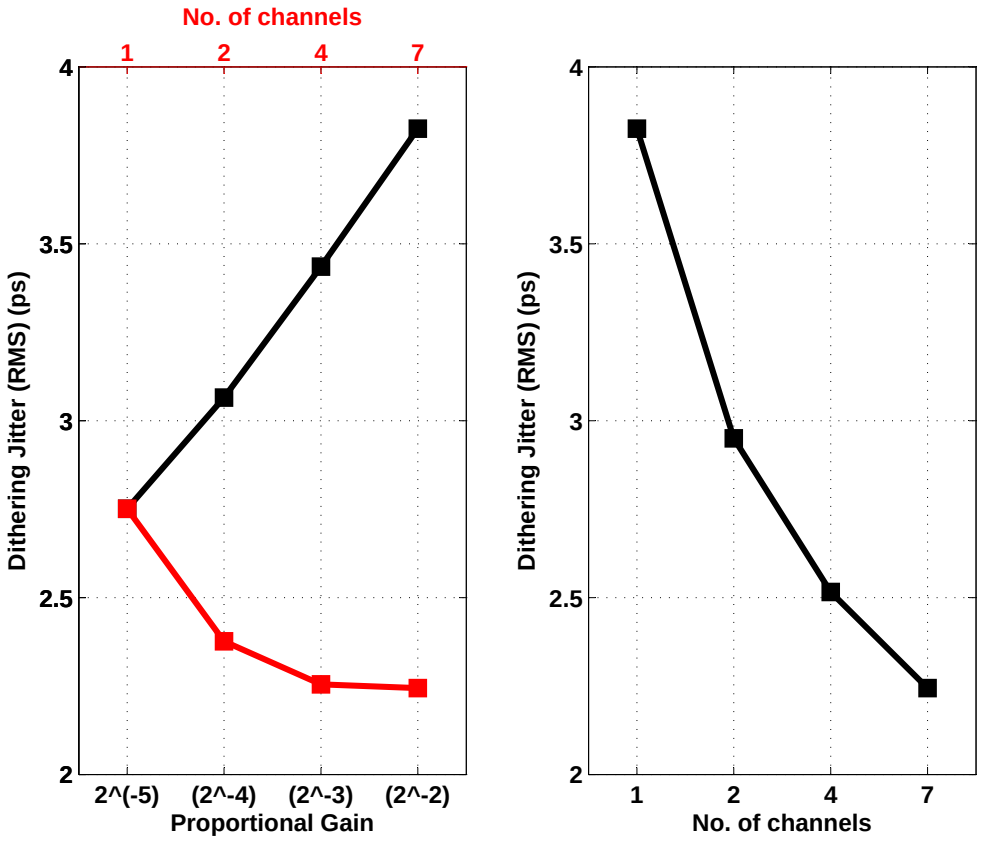


Figure 3.24: (a) Measured rms jitter vs. $K_p, K_i$ settings, and vs. number of channels sharing timing information, (b)Measured rms jitter for different degrees of collaboration while keeping loop tracking bandwidth fixed.

The increased edge transition density also helps to improve the robustness of the timing recovery in the face of long strings of 1's and 0's. Figure 3.25 plots the dithering jitter in the recovered clock for 7-bit PRBS case and 15-bit PRBS case, when 200-ppm frequency offset is present between the local reference clock and the data.

For the 1 channel case, dithering jitter is higher when the longer run-length PRBS sequence is used. As the number of channels sharing timing information increases, the difference in dithering jitter for the two PRBS lengths reduces, demonstrating the resilience of the collaborative architecture to long strings of 1's and 0's. A 200-ppm frequency offset is used in this experiment, as the CDR becomes more sensitive to the run-length when it is functioning as a frequency synthesizer. This also explains the higher dithering jitter than those reported in Figure 3.24.

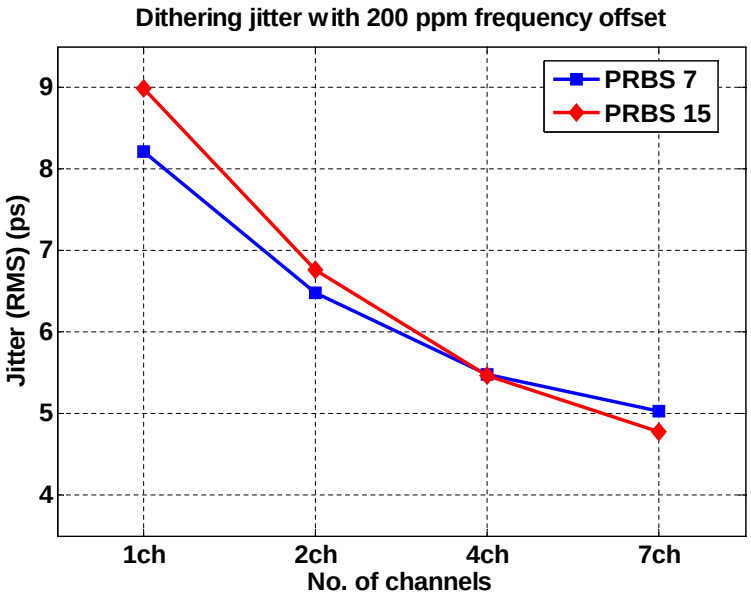


Figure 3.25: Dithering jitter on the recovered clock vs. number of channels sharing timing information vs. PRBS run-length.

We also performed measurements to evaluate the suitability of the collaborative timing recovery architecture to accommodate spread-spectrum clocking. In spread-spectrum clocking, the reference clock for the transmitter drivers is frequency modulated by ±0.5% using a 33 KHz triangular wave. This spreads out the transmitted energy uniformly over a region of the spectrum, to meet established specifications

on the maximum electro-magnetic energy radiation permissible by electronic devices. The receiver operated without any bit-errors at 3.2 Gb/s. Figure 3.26 shows the spectrum of the recovered clock for the receiver configured with different degrees of collaboration, and loop bandwidth held constant. In all cases, the recovered clock exhibits the expected spectrum. We observe that the noise in the side-bands is slightly lower for the 7-channel CDR compared to the 1-channel CDR, again suggesting lowering dithering jitter.

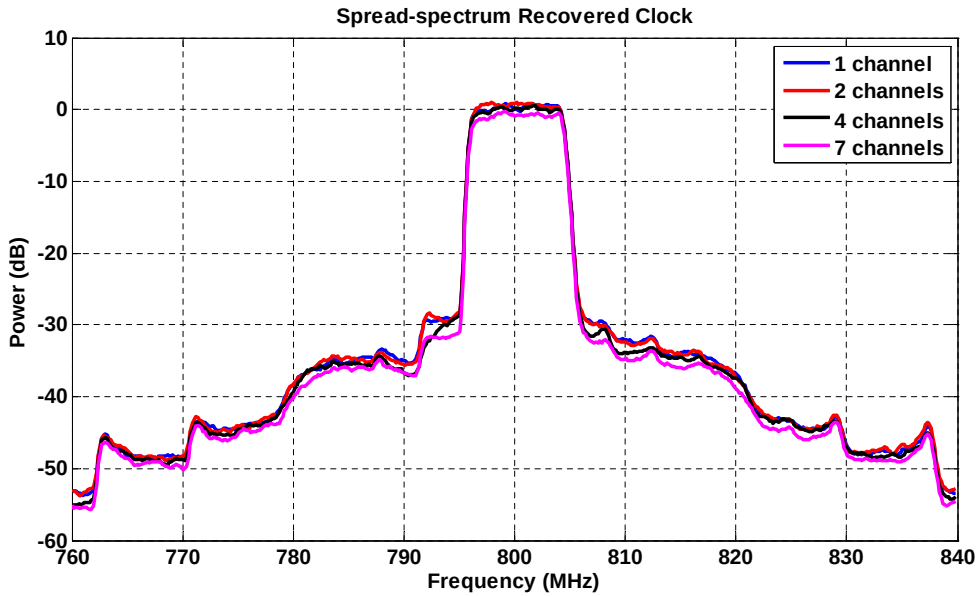


Figure 3.26: Spectrum of the recovered clock vs. degree of collaboration when spread-spectrum clocking is employed in the transmitter

Finally, the measured linearity of the phase interpolator is shown in Figure 3.27. The worst-case INL and DNL of the interpolator are 5.6 LSBs and 0.8 LSBs, respectively.

Overall, the 8x5Gb/s parallel receiver consumes 310 mA of current from a 1.45 V supply (including all output drivers), which translates to a power efficiency better

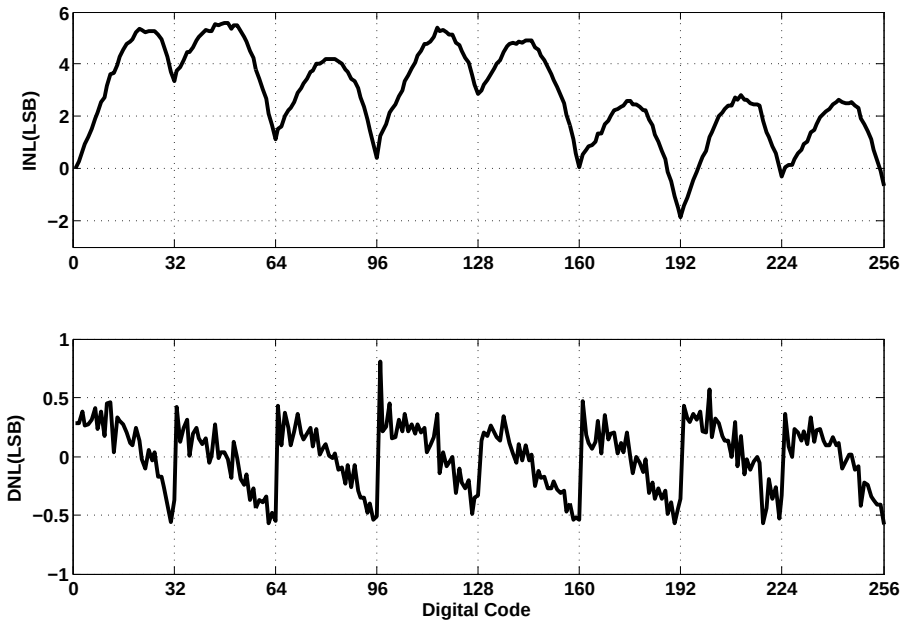


Figure 3.27: Phase interpolator INL and DNL.

than 11.2 mW/Gb/s. The performance of the test chip is summarized in Table 4.1.

Table 3.1: Performance summary

| | |
|---|---|
| Technology | 0.13$\mu$m CMOS |
| Supply voltage | 1.45V |
| Active Area | 3.2 mm$^2$ |
| Throughput | 8×5 Gb/s |
| Power consumption | 450 mW |
| Tracking range | ±5000 ppm |
| BER | $\leq 10^{-12}$ |

## 3.6 Discussion

### 3.6.1 Performance Limits

While this receiver architecture aims to track power-supply noise induced jitter that peaks at around 100 MHz, the prototype test-chip achieved a much smaller jitter tracking bandwidth. This can be attributed to the overall latency in the feedback loop — approaching 100 UIs in the worst case. This high loop latency limited the achievable jitter tracking bandwidth, and drove the loop to instability if we attempted to use aggressive filter coefficients.

Table 3.2 presents a breakdown of the loop latency:

Table 3.2: Loop latency breakdown

| | |
|---|---|
| Error Summer and Global Digital Filter | 60% |
| Phase Interpolator | 4% |
| Clock Distribution Buffers | 6% |
| DLLs and DCC | 7% |
| Samplers and Phase Detector | 18% |
| Data Buffers (error signals) | 5% |

The total latency is dominated by the latency in the digital filter. This can be attributed to a few design choices: First, for this test-chip, the filter is programmable i.e. the $K_P$ and $K_I$ coefficients can be chosen among 4 different values each. This forced us to design wider adders than required, and add multiplexers in the data-path. Second, truncation of bits to implement division is performed at the end after wide additions, requiring wide adders. An approach similar to that implemented by Stonick et al. [27] would have resulted in a less complex digital filter. Finally, the proportional path had an additional 1 clock cycle (8 UIs) latency to align with the

integral path that has an additional integration. This alignment is not necessary for correct operation of the timing recovery loop. Using CAD-tools to design the phase detector and the global digital filter also required us to use conservative timing in this critical path.

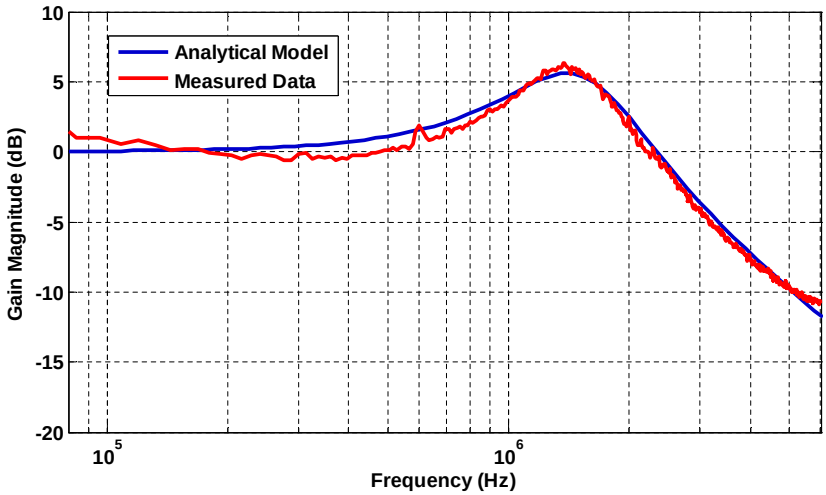


Figure 3.28: Fitting frequency-domain model to measured transfer function

To predict the performance of this architecture at current technology nodes, we fitted the frequency domain model described in Section 3.2 to measured loop characteristics. We obtained the values of parameters like loop latency, update rate, coefficients from the schematics and simulations. The small-signal gain of the phase detector was used to fit the models. The small signal gain depends on the jitter on the recovered clock and the variation in the sampler offsets and thus cannot be obtained analytically. Figure 3.28 shows the experimentally obtained loop transfer function with the fitted curve overlaid. Using this model, we predict the limits of performance of this architecture. Figure 3.29 plots the loop transfer function for three scenarios. The line in blue shows the highest bandwidth achievable with optimized loop

coefficients from the current chip. We can obtain a bandwidth of 7 MHz with less than 2 dB of peaking. If we were to port this design to 45 nm technology that has one-third the FO4-delay compared to the 130 nm technology used to implement this chip, we can assume that it will support 3x the data-rate and the digital logic and buffers will be 3x faster. This translates into a design with 3x the update rate, with the same loop latency (when measured in terms of numbers of UIs). Thus we can expect the design to have 3x the loop tracking bandwidth with 2 dB peaking (shown using the red line in Figure 3.29). Finally, if we were to redesign this chip such that the overall loop latency was reduced to 75 UI and re-optimize filter coefficients, a loop bandwidth of 50 MHz is achievable with 2 dB of peaking (shown using the black line in Figure 3.29). This brings us close to tracking mid-frequency jitter.

### 3.6.2 Scalability

The test-chip described in this chapter consisted of 8 receiver slices that contribute error information to global TR. One might ask: How much more can this system scale? Can we envision a scheme where 16, 32 or even 100 channels collaborate to perform timing recovery? While the exact number of channels depends on the technology and the operating environment, there are a number of limitations on the scalability of this architecture. There are three considerations why we believe this system does not scale beyond a certain point.

- As observed in Figure 3.24, the improvement in dithering shows diminishing improvements as we increase the number of channels sharing timing information. This is to be expected, as effect of increased edge transition density and richer

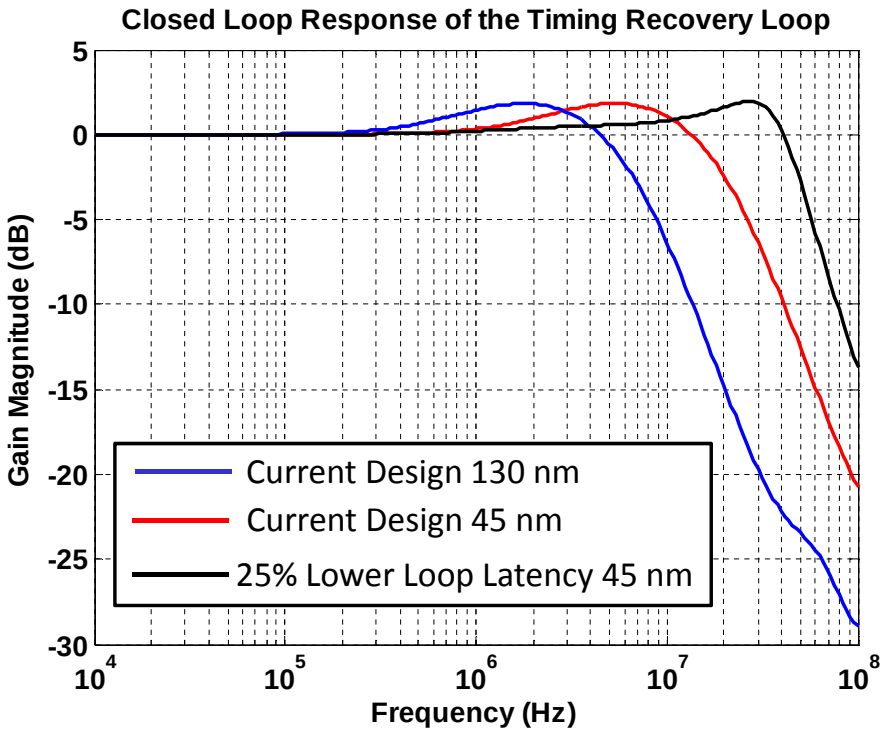


Figure 3.29: Future outlook for collaborative TR

timing information starts to level off.

- One of the basic assumptions made in proposing the collaborative timing recovery architecture is the correlation in the power-supply noise induced jitter in parallel data streams. Since this correlation has a spatial dependence, only transmitters located close together have strongly correlated jitter in their output data streams. Thus, the scalability of this architecture is limited by the number of transmitters that can be packed in a region exhibiting power-supply noise correlation.

- Finally, this architecture trades off computation power and area to perform per-pin clock recovery for communication power to send the error information and

receive the global recovered clock. If a small number of channels share timing information, the power cost of on-chip communication of error and clock signals is relatively low. However, as the number of channels increases and consequently on-chip distances increase, one must pay close attention to when the trade-off is no longer favorable for a low-power design.

In very wide parallel receivers like those employed in memory interfaces, we envision a system where we divide the wide bus into smaller clusters. Each cluster has its own global TR block and operates independently from other clusters.

### 3.6.3 Power and Area Overheads

While the collaborative timing recovery architecture enables wideband jitter tracking, some attention must be paid to the area and power overheads of this design. Table 3.3 provides a power breakdown of this loop, calculated from post-layout schematic simulations. The power overhead of implementing this architecture over traditional source-synchronous receivers is around 8.5%. The area overhead of the phase interpolator, global digital loop filter, error summer, and routing is around 15% per receiver. The power and area costs of the PLL has not been taken into account while calculating the overheads.

The additional power and area cost is used to achieve the increased jitter tracking bandwidth while limiting the dithering jitter. As was shown in Section 3.3, source-synchronous receivers fail to track the spatially varying components of the jitter on incoming data. If one were to design a serial receiver with large tracking bandwidth, the update rate of the control logic in the receiver would have to be very fast, making

each receiver consume too much power. Moreover, some techniques to control the higher dithering jitter would have to be implemented.

Table 3.3: Detailed power breakdown

| | |
|---|---|
| Local Rx Slices | 81.9% |
| PLL | 3.6% |
| Clock Distribution | 6.0% |
| Phase Error Signal Routing | 2.7% |
| Phase Interpolator and Global Control Logic | 5.8% |

### 3.6.4 Comparison with Other Schemes

Recently, O'Mahony et al. and Hossain et al. proposed injection-locking based parallel receivers that achieves wide-band jitter tracking while having very good power efficiency [23, 15]. However, both these schemes require a forwarded clock. Also, Ahmadi et al. proposed a pilot-based scheme for clock and data recovery [2]. This scheme relies on embedding a low amplitude clock along with the data on the same channel, and performing injection locking at the receiver. While this scheme also achieves wide-band jitter tracking without requiring a forwarded clock, it requires changes to the transmitter design.

## 3.7 Summary

A parallel receiver with a collaborative timing recovery architecture is presented. Sharing timing information from several synchronous data streams enables wideband jitter tracking by reducing dithering jitter on the received clock. The proposed design incorporates several techniques: a global digital clock synthesizer that sums up timing

error information from multiple receiver slices and filters it using a 2nd-order digital filter; and a dual DLL based receiver architecture that performs skew compensation and samples 4-way interleaved data without introducing phase filtering in the path of the sampling clock. The design techniques are validated by a test-chip fabricated in a 130nm CMOS process. An FPGA-based parallel transmitter is developed to test the receiver test-chip at high data rates. Experimental results confirm that the collaborative architecture improves the overall jitter performance of the timing recovery loop.

# Chapter 4

# Phase Calibration in Delay Locked Loops

## 4.1 Introduction

In the previous chapter, the local receiver slice in the collaborative parallel receiver architecture employed a delay-locked loop (DLL) to generate eight clock phases. These clock phases drive eight interleaved samplers that sample 4-way interleaved data both at the center and edge of the data eye. The eight sampled data values are used to extract early-late timing information using an Alexander phase detector. Thus, any deviations from the ideal phase-spacings in the output clocks from the DLL will result in additional jitter on the recovered clock, and reduced timing margin for sampling the data. Both random transistor mismatches and systematic sources of phase error can cause phase-spacing mismatches in DLLs. Correction circuits are required to assist the DLL in producing equally-spaced clock phases. This chapter

shall describe two different solutions to correct for static phase mismatches in DLLs.

The remainder of this chapter is organized as follows: Before we describe the phase correction circuits, it is instructive to understand some of the trade-offs between phase-locked loops (PLL) and DLLs when used for multi-phase clock generation in clock and data recovery circuits, and Section 4.2 describes PLLs and DLLs and argues in favor of DLLs. Section 4.3 presents some of the sources of phase-spacing errors in DLLs. We then propose two circuits to reduce these errors: an analog-intensive solution in Section 4.4 and a digital-intensive phase calibration in Section 4.5.

## 4.2 PLL vs. DLL

There are two popular choices for producing multiple equally-spaced clock phases from a differential clock input: PLL and DLL. Both are negative feedback loops that are somewhat similar in topology, but have very different properties. In this section, we shall briefly describe key aspects of both these loops and contrast some of their properties that are relevant in the context of our application.

### 4.2.1 PLL

PLLs are widely used as clock generators for numerous applications – microprocessor clocking, frequency synthesizers, clock and data recovery and so on. In high speed I/O circuitry, PLLs find use both in the transmitter and receivers. In transmitters, they provide accurate reference clocks while in receivers they are used to extract the clock from incoming data [24], provide sampling clocks for the receivers [18], or to perform *phase filtering* [18].

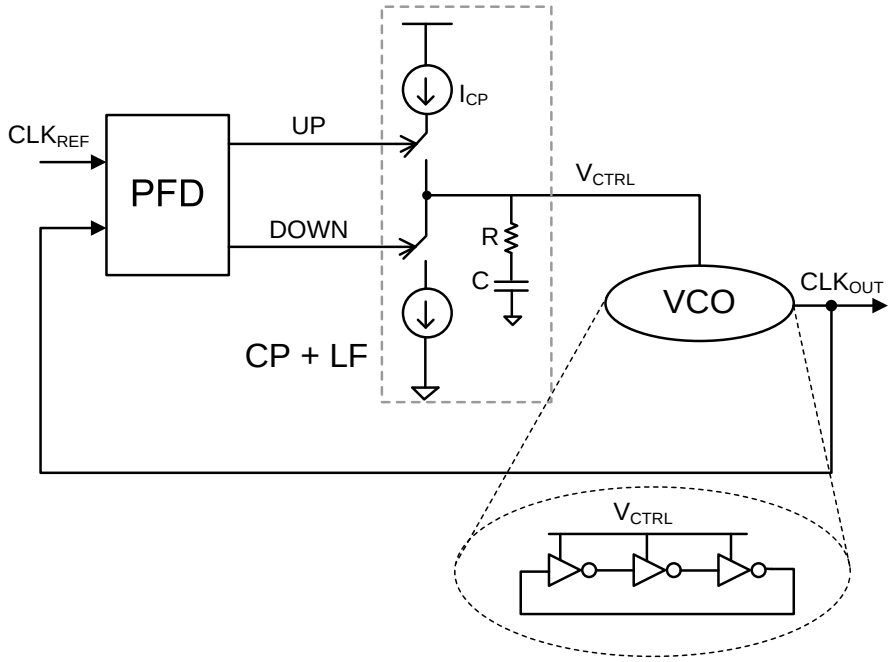


Figure 4.1: Block diagram of a charge-pump PLL

Figure 4.1 shows the block diagram of a common implementation of a PLL. It is a feedback loop that produces an output clock whose phase and frequency is aligned with an input reference clock. It comprises of a voltage-controlled oscillator (VCO), phase-frequency detector (PFD), charge-pump (CP), and loop-filter (LF). The heart of the PLL is the VCO, whose control voltage is adjusted until its output clock has correct phase and frequency. The PFD, CP, and LF perform this control. The PFD compares the input reference clock ($\text{CLK}_{REF}$) with the output clock ($\text{CLK}_{OUT}$) to determine whether it leads or lags the output clock, and generates UP and DOWN pulses. The CP consists of two current sources that are controlled by the UP and DOWN pulses, to dump charge to the loop filter and adjust the voltage $V_{ctrl}$ until the loop achieves phase and frequency lock. Once the loop achieves lock, there is no

phase error and $V_{ctrl}$ remains stable.

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

### S-Domain Modeling

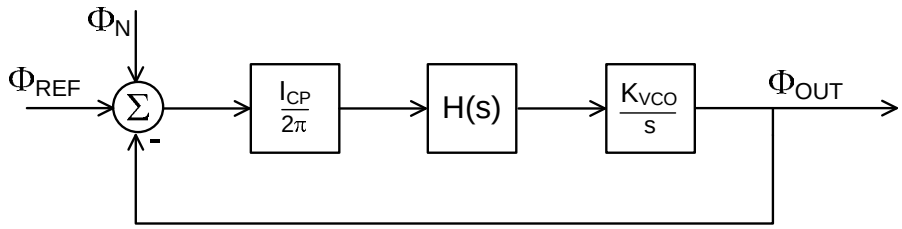


Figure 4.2: S-domain model of a charge-pump PLL

A linear model of the PLL described above is presented in Figure 4.2. $\Phi_{REF}$ represents the phase of the input clock, $\Phi_{OUT}$ represents the phase of the output clock and $\Phi_N$ represents the input phase noise (jitter). The open loop transfer function is:

$$L(s) = \frac{I_{CP}}{2\pi} \cdot H(s) \cdot \frac{K_{VCO}}{s} \tag{4.1}$$

where $I_{CP}$ is the charge pump current, $H(s)$ is the loop filter transfer function and $K_{VCO}$ is the voltage-to-frequency gain of the VCO (expressed in Hz/V). For simplicity, $H(s)$ is assumed to be a simple series RC filter, with transfer function $\left(R + \frac{1}{sC}\right)$, though in real implementations another capacitor is added in parallel. Overall the closed loop transfer function can be written as

$$\frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{2s\zeta\omega_n + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{4.2}$$

where:

$$\omega_n = \sqrt{\frac{I_{CP}K_{VCO}}{2\pi C}} \tag{4.3}$$

$$\zeta = \frac{R}{2}\sqrt{\frac{I_{CP}CK_{VCO}}{2\pi}} \tag{4.4}$$

The reference phase noise transfer function $\left(\frac{\Phi_{OUT}}{\Phi_N}\right)$ is exactly the same as the input phase transfer function described above by Eq. 4.2. The transfer function is a second-order low-pass transfer function, suggesting that $\Phi_{OUT}$ cannot track the high frequency components of $\Phi_N$. This implies the PLL can be thought of as a *phase filter*. The output clock can track jitter on the input clock within the bandwidth of the PLL. This property of PLLs is exploited in applications where a noisy input clock needs to be cleaned-up to produce a low-jitter clock. This phase filtering property makes the PLL unattractive for use in the local receiver slices of the collaborative receiver architecture, as we shall see later in this section.

**PLLs for De-skew and Multi-phase Clock Generation**

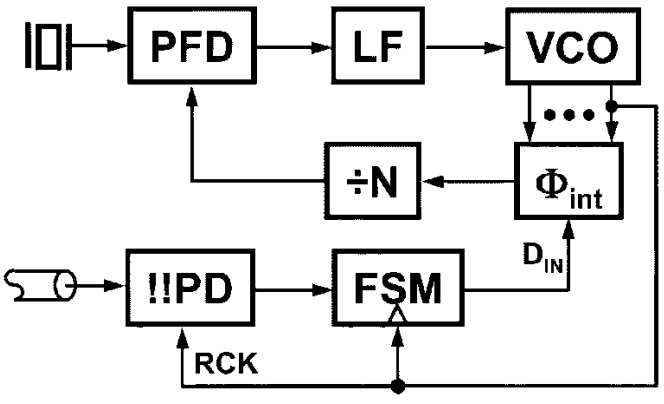


Figure 4.3: Phase averaging clock and data recovery circuit

PLLs are widely employed in clock and data recovery circuits [24]. In the context of collaborative timing recovery architecture and source-synchronous receivers, PLLs can perform the de-skew and multi-phase clock generation functions that are required in each local receiver slice. Fig 4.3 shows the block diagram of a phase-averaging CDR, proposed by Larsson [18]. In this architecture, the multiple clock phases produced by the VCO feed into a phase interpolator ($\Phi_{int}$) that is placed inside the core PLL loop. A separate bang-bang phase detector (!!PD) produces digital error signals to denote whether the sampling clock (RCK) phase is early or late compared to the incoming data. This digital error signal is used by a digital finite state machine (FSM) to produce the control word for a phase interpolator. The output clock from the phase interpolator is fed back to the PFD. This architecture offers three important benefits. First, it is well suited for multi-phase clock recovery since the VCO provides multiple equally-spaced clock phases, avoiding multiple power-hungry and area-inefficient phase interpolators. Second, phase quantization errors at the output of the phase interpolator are suppressed by the low-pass transfer function of the PLL. This reduces the jitter on RCK. Third, just one PLL is sufficient to perform both the de-skew and multi-phase clock generation. VCO-based multi-phase clock generators are also attractive because the only sources of phase-spacing mismatch in VCOs are layout-related or random transistor mismatches. However, one critical drawback of this architecture is the low-pass filtering of the input clock jitter as it goes through the PLL. The input clock to the local receivers in the collaborative architecture (and also in source-synchronous receivers) contain the correlated tracking jitter. Thus it is important to preserve the jitter on the clock and avoid the phase filtering introduced

by the PLL. Said another way, the closed-loop transfer function of the PLL will introduce additional poles in the path of the recovered clock and limit the jitter-tracking bandwidth of the timing recovery loop.

### 4.2.2 DLL

A delay-locked loop (DLL) is another way to produce an output clock with a known phase relationship to an input clock. It is a slight modification of the PLL. A DLL contains a voltage-controlled delay line (VCDL) instead of a VCO, and the feedback loop sets the control voltage of this delay-line to enforce a condition between the input and output clock phases of a DLL. Figure 4.4 shows a block diagram of a commonly-used DLL topology. It consists of a phase detector (PD), charge pump (CP), a capacitor as loop filter, and a VCDL. Since the VCDL only manipulates the phase of the input clock, it only requires a PD instead of a PFD. In most applications, the delay of the delay-line is locked to a full clock period. The PD compares the phase of the input clock and the feedback clock and produces UP and DOWN pulses with widths proportional to the relative phase difference between the two clocks. The CP, consisting of two current sources that are controlled by the UP and DOWN signals, integrates the phase error on the capacitor to generate $V_{CTRL}$ and consequently controls the delay-line delay to close the feedback loop. It is important to note that the delay line only adds a controlled amount of delay to the input clock. It does not synthesize a new clock. If the delay-line consists of multiple voltage-controlled buffers, the output of each of these buffers can be tapped to generate multi-phase clocks.

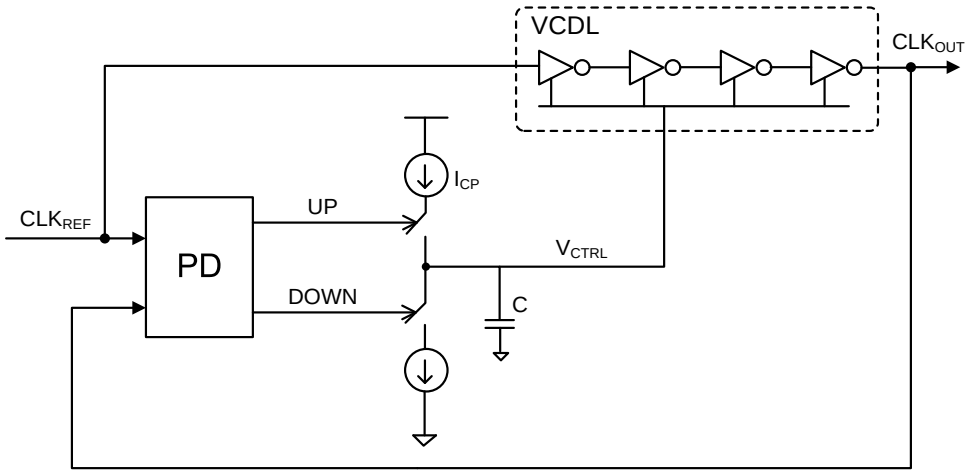


Figure 4.4: Block diagram of a charge-pump DLL

### Z-domain Model

Since the VCDL does not perform any integration, a DLL is a simple first-order loop with one pole introduced by the capacitor. Thus, it is inherently stable and simple to design. Figure 4.5 presents a z-domain model of a DLL [21]. A z-domain model represents the DLL better as the DLL adds one clock cycle delay to the reference clock before comparing it with the reference clock, and this one cycle latency can be represented easily in the z-domain.

The overall loop transfer function is:

$$\frac{\Phi_{OUT}}{\Phi_{REF}} = \frac{(1+\alpha)z - 1}{z - (1-\alpha)} \tag{4.5}$$

$$\alpha = K_{VCDL} \cdot \frac{I_{CP} T_{REF}}{2\pi C} \tag{4.6}$$

where $T_{REF}$ is the period of the reference clock and $C$ is the capacitor in the loop. The $\frac{\Phi_{OUT}}{\Phi_{REF}}$ transfer function is also the jitter transfer function. It is all-pass with a

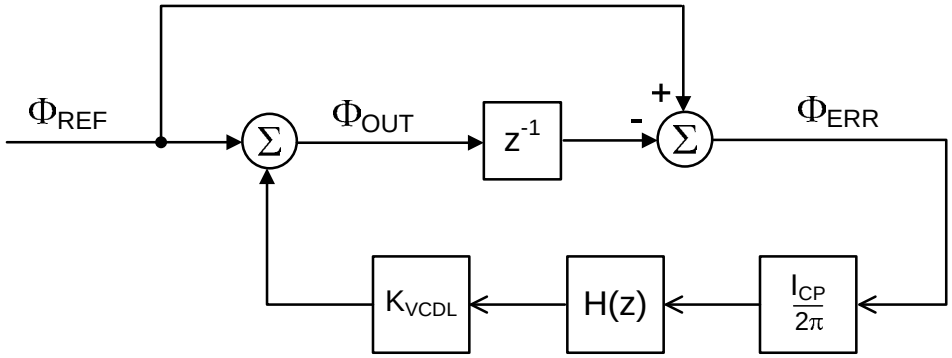


Figure 4.5: z-domain model of a charge-pump DLL

slight peaking at high frequencies. This peaking is caused because the PD is unable to distinguish between high frequency jitter on the reference clock and the feedback clock. To keep the output clock phase unaffected by high-frequency cycle-to-cycle jitter on either clock, the feedback loop should instantaneously react in opposite directions depending on whether it is the input clock or the feedback clock that has jitter. However, because of its inability to make this distinction, it always reacts in one direction, causing jitter amplification when the reference clock is jittered. This jitter peaking is usually small and the transfer function is considered *nominally* all-pass.

### DLLs for De-skew and Multi-phase Clock Generation

The DLL does not filter the jitter on the input clock and allows it to pass unattenuated to the output. Thus it is suitable for applications where it is important to preserve the jitter on the input clock. In the collaborative architecture, DLLs are a natural fit since we require the local receiver to de-skew the input clock and generate multiple clock-phases while preserving the jitter on the input clock. One point to

note is that two DLLs are required to perform the de-skew and the multi-phase clock generation, respectively. Since the delays from clock buffers in the delay line add up to produce the total delay, the same set of clock buffers cannot be used to provide variable-delay (for de-skew) and clocks spaced equally over one period of the reference clock. Another key difference is that the multiple clock phases produced by a DLL are susceptible to multiple sources of phase-spacing mismatch — both random and systematic.

The next subsection shall describe experimental results to demonstrate the system-level performance implications of choosing PLLs vs DLLs in the local receiver slice.

### 4.2.3 Comparison between PLL and DLL

A single PLL can be used for multi-phase clock generation and clock de-skew [18]. Alternatively, two DLLs can be used to perform the two tasks independent of each other. PLLs have a low-pass transfer function from the phase of the reference clock to the output clock and, thus, filter out the middle and higher frequency jitter on the received clock. On the other hand, DLLs have a nearly all-pass phase transfer function, and are able to preserve the correlation between the jitter on the incoming data and received clock, resulting in good jitter tolerance over a wide frequency range. Recently reported designs [7, 22] have avoided the use of PLLs in the path of the received clock to achieve wide jitter tracking bandwidth.

Our test chip, the details of which were discussed in the previous chapter, enables a direct comparison between the jitter tracking bandwidths of PLL- and DLL-based local receiver slices. The receiver is configured to operate as a source-synchronous

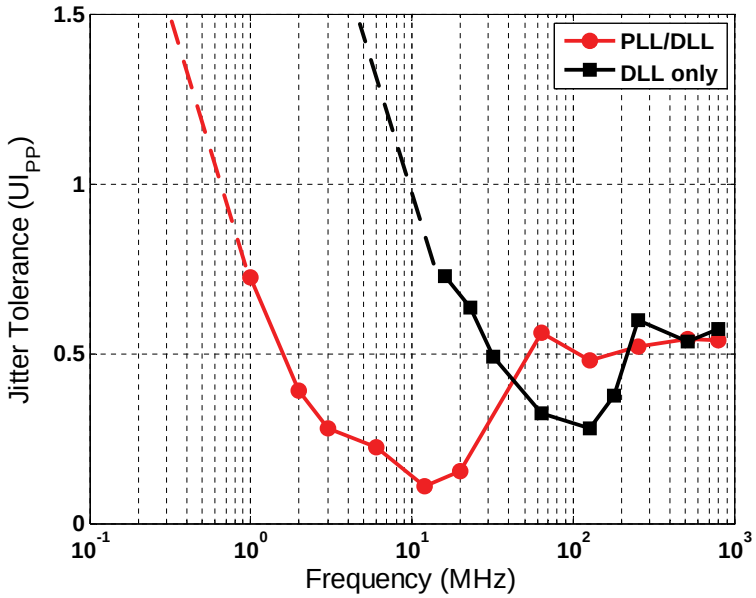


Figure 4.6: Jitter tolerance plots for the DLL only case and PLL/DLL case measured at 3.2 Gb/s.

receiver to highlight the difference in performance in the two cases. Figure 4.6 plots the jitter tolerance curves for BER $< 10^{-9}$ for two different configurations of the test chip. In the "DLL-only" case, a quarter rate clock is directly fed to the local receivers. In the "PLL/DLL" case, a sub-rate clock is first multiplied on-chip using a PLL to the desired frequency. The jitter tolerance bandwidth for the DLL-only case is >100 MHz and is limited by the difference in the on-chip path length between the data and clock signals. In contrast, a jitter tolerance bandwidth of around 10 MHz is obtained for the PLL/DLL case. The PLL bandwidth limits the maximum achievable jitter tolerance bandwidth in this case. Due to test equipment limitations, a maximum jitter of 0.72 UI can be introduced. These measurements confirm that phase filtering should be avoided in the path of the received clock in local receiver slices to improve jitter tracking.

This motivates our choice of using cascaded DLLs in the local receiver slice. However, it is important to understand and address the sources of phase-spacing mismatch in DLLs. In the next section we shall look at some of the sources of mismatches present in DLLs.

## 4.3 Sources of Phase-Spacing Mismatch in DLLs

There are multiple sources of phase-spacing mismatches in DLLs: (1) A PD offset or a small difference between the charge pump up and down currents can result in static phase spacing error at the end of the delay line in a DLL, (2) DLLs that lock the delay of the delay line to half a reference clock cycle are very sensitive to the duty cycle of the reference clock signal. (3) The shape of the reference clock entering the delay line can exacerbate phase spacing mismatches in DLLs. By carefully addressing each of these issues, DLL-based de-skew and multi-phase generation can be made attractive. First, it is instructive to understand each of these sources of error.

### 4.3.1 Charge Pump and Phase Detector Offsets

Due to transistor mismatches and layout non-idealities, a small offset might be present in the phase detector, i.e., the PD output might be non-zero when the input clock phases are perfectly matched. In the charge pump, current-source mismatches between the UP and DOWN currents might be present. Also, the finite output impedance of the current sources may imply that depending on the voltage at the output of the charge pump, a mismatch in the two currents might be present. Both these errors result in steady-state settling errors, much like the input-referred offset

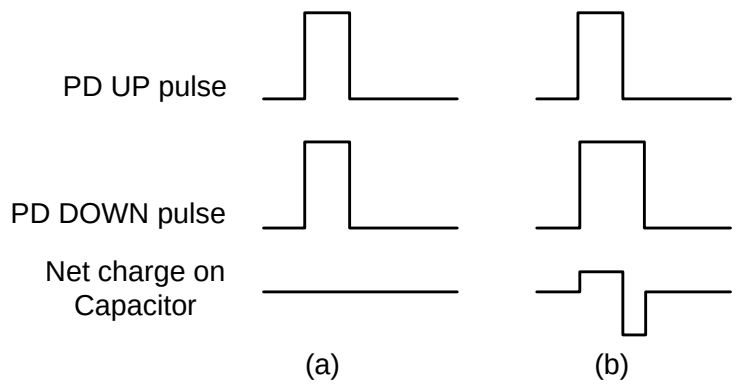


Figure 4.7: Effect of CP offset on DLL steady-state operation

in an op-amp.

Figure 4.7(a) illustrates the case where there is no mismatch, and the PD produces finite width UP and DOWN pulses to prevent a dead-zone in the feedback loop [29]. In this case, no charge is dumped on the capacitor. However, when there is an offset, the output clock phase settles where the UP and DOWN pulse-widths are such that there is no net charge dumped on the capacitor (Figure 4.7(b)). Figure 4.8 plots the integral non-linearity (INL) in a half-cycle DLL that has some charge-pump current mismatch. The mismatch manifests itself as a large phase jump at the end of the delay-line caused by another edge entering the delay-line. In contrast, the output phases from a multi-phase generating PLL do not exhibit any phase-spacing mismatch when CP/PD offsets are present. Both of these errors can be corrected by adjusting the difference between the charge-pump UP and DOWN currents.

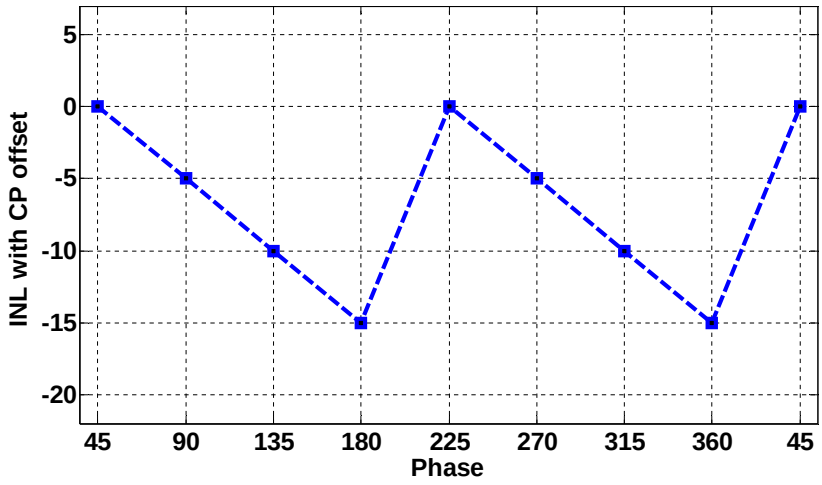


Figure 4.8: INL in clock phases in the presence of PD/CP offsets

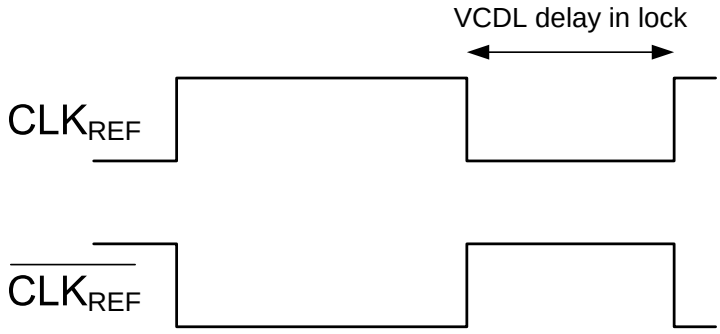


Figure 4.9: Illustration of duty-cycle error consequences

### 4.3.2 Duty Cycle Error

In half-cycle DLLs, where the loop locks the delay of the delay-line to half the reference clock period, any duty-cycle distortion in the reference clock leads to phase-spacing mismatches. Figure 4.9 illustrates the error-mechanism. The loop tries to align the rising edge of the reference clock with the rising edge of the complementary clock. If this time-interval is not exactly equal to half the reference clock period,

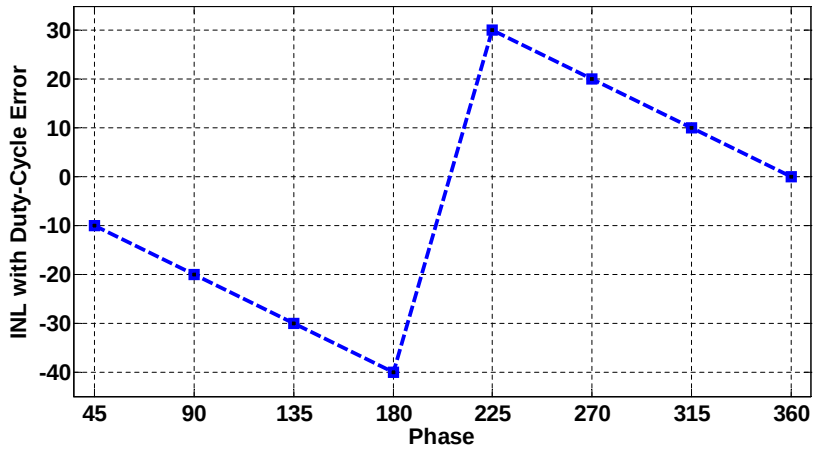


Figure 4.10: INL in clock phases in the presence of reference clock duty-cycle error

phase-spacing mismatches result. Figure 4.10 shows the resulting INL in the clock phases when 10% duty-cycle is present on the reference clock. Again we observe a large jump when a new edge enters the delay-line. This source of phase-spacing mismatch is not present in PLLs or traditional full-cycle DLLs. A duty-cycle corrector can be employed to reduce this error.

### 4.3.3 Shape of the Reference Clock

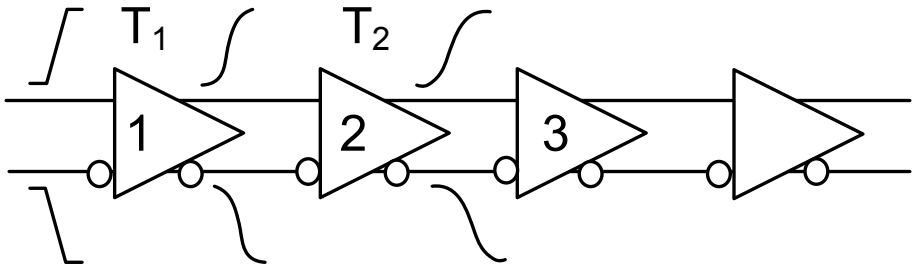


Figure 4.11: Shaping of reference clock as it flows through the VCDL

A VCO generates its own clock, but a VCDL takes a clock input. The shape of

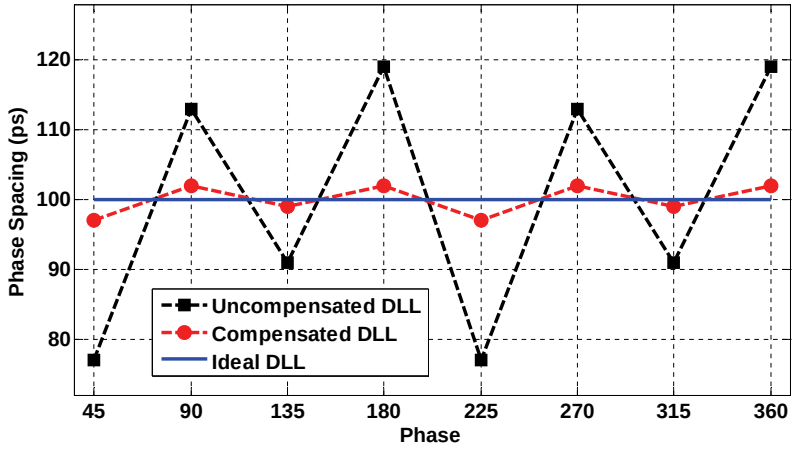


Figure 4.12: Phase-spacing error caused by the shape of reference clock.

the clock, characterized by its voltage swing and the slope of its edges, entering the first delay cell in the VCDL can be different from those entering subsequent stages, as illustrated in Figure 4.11. This results in adjacent delay stages having different delays. Preshaping the clock signal by adding identical delay cells before the delay line helps, but does not eliminate the problem. The optimal clock shape is one that would be produced by a voltage-controlled oscillator composed of the same delay cells and having the same control voltage as the delay line. Figure 4.12 shows the simulated nonlinearity in the phase spacings for a DLL consisting of a 4 stage differential delay line after preshaping using an identical delay line. It also shows the simulated nonlinearity achieved after an edge-conditioning technique to shape the reference clock signal is used. This technique, which we call phase spacing error correction (PSEC), is described in greater detail in Section 4.4.

### 4.3.4 Random Mismatches

Finally, process variations in deep-submicron technologies can result in delay mismatches between adjacent delay cells. Figure 4.13 plots the histogram from 1000 Monte-Carlo simulations to estimate the impact of the process variations on delay cells [8]. The authors estimated that in 0.18 $\mu$m technology, up to $\pm 20\%$ phase-spacing mismatch might be present when there is 0.1 V threshold voltage mismatch in the transistors. Correction of this kind of error requires a phase averaging network [9] or static phase calibration.

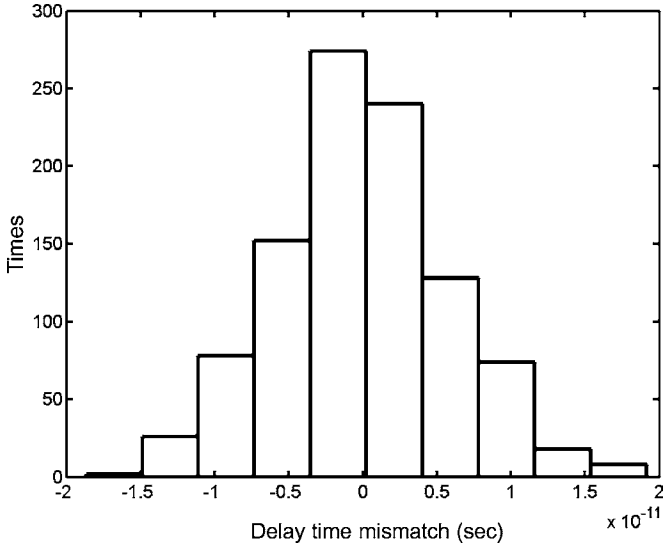


Figure 4.13: Distribution of phase-spacing mismatches in the presence of threshold voltage variations (Monte-Carlo simulations)

In the next two sections, we shall describe two solutions to reduce phase-spacing mismatch in DLLs. The first is an analog solution that addresses each of these sources of errors separately. The second is a digital solution, that lumps all of the error sources together, and tries to minimize the differential non-linearity in the phase-spacings.

## 4.4 Analog Phase Spacing Error Correction

This section describes the enhancements made to the local receiver slice to ensure that the phases produced by the multi-phase generating DLL are equally spaced. We begin with a block diagram of the receiver itself to explain some design choices related to the placement of the correction circuitry.

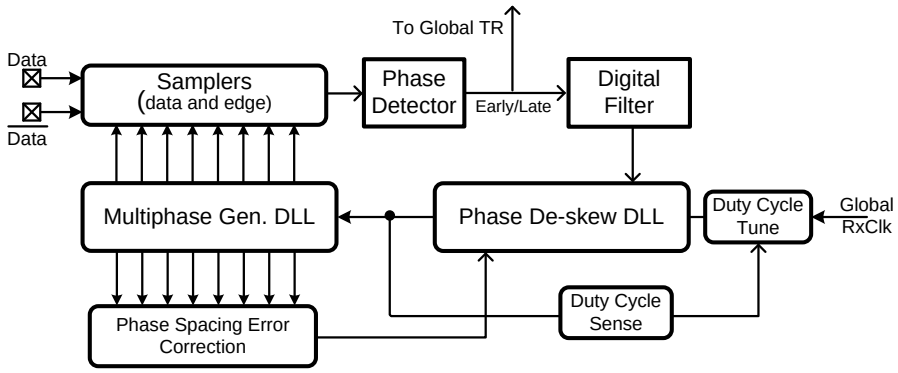


Figure 4.14: Local receiver block diagram

Figure 4.14 presents details of the receiver architecture. Each receiver slice consists of two DLLs, samplers, skew-compensation logic, and phase spacing error correction circuits. The global RxClk first goes through a duty cycle corrector (DCC) to compensate for any distortion from the clock distribution buffers. The first DLL, called the phase-deskew DLL, adds a variable amount of delay to the clock path such that its output clock is phase aligned to the incoming data. The second DLL produces the eight clock phases that drive eight interleaved edge and data samplers. Also shown in the figure is the phase spacing error correction loop that balances the delays of the delay cells by adjusting the slope of the clock edges.

The cascaded DLL architecture is chosen for a number of reasons. No extra

preshaping delay buffers are required in the clock path before the DLL used for multiphase clock generation. It also provides a convenient location for placing the duty cycle correction circuit. The duty cycle of the clock entering the second DLL is sensed and tuned to 50%, but the correction circuit is placed before the first DLL in order to not disturb the shape of the clock signal for the second DLL. The offsets in the phase-deskew DLL do not matter as they get absorbed into the overall delay of its delay line.

Adjusting $V_{REF}$ — the output voltage of the CP — to set the output voltage provides a simple knob to compensate for small CP and PD offsets. Based on phase-offset measurements at the end of the delay-line, $V_{REF}$ is set to minimize phase-spacing mismatches due to CP and PD offsets. This chip does not contain any special circuits to reduce random phase-mismatches introduced by process variations and layout imbalances. The rest of this section will describe the design and performance of the phase-spacing error corrector and the duty-cycle corrector circuits.

### 4.4.1 Phase Spacing Error Corrector (PSEC)

As described in Section 4.3, the shape of the clock signal entering the DLL affects the delays through the delay cells. To correct for this effect, the rise and fall times of the input clock to the second DLL are adjusted by sinking or sourcing small amounts of current from the penultimate delay stage of the first DLL.

A feedback loop, shown in Figure 4.15, consists of XOR based delay detectors, charge pump, loop filter, and a V-to-I converter. As the error is such that all odd delay stages are either faster or slower than all even delay stages, the charge pump dumps

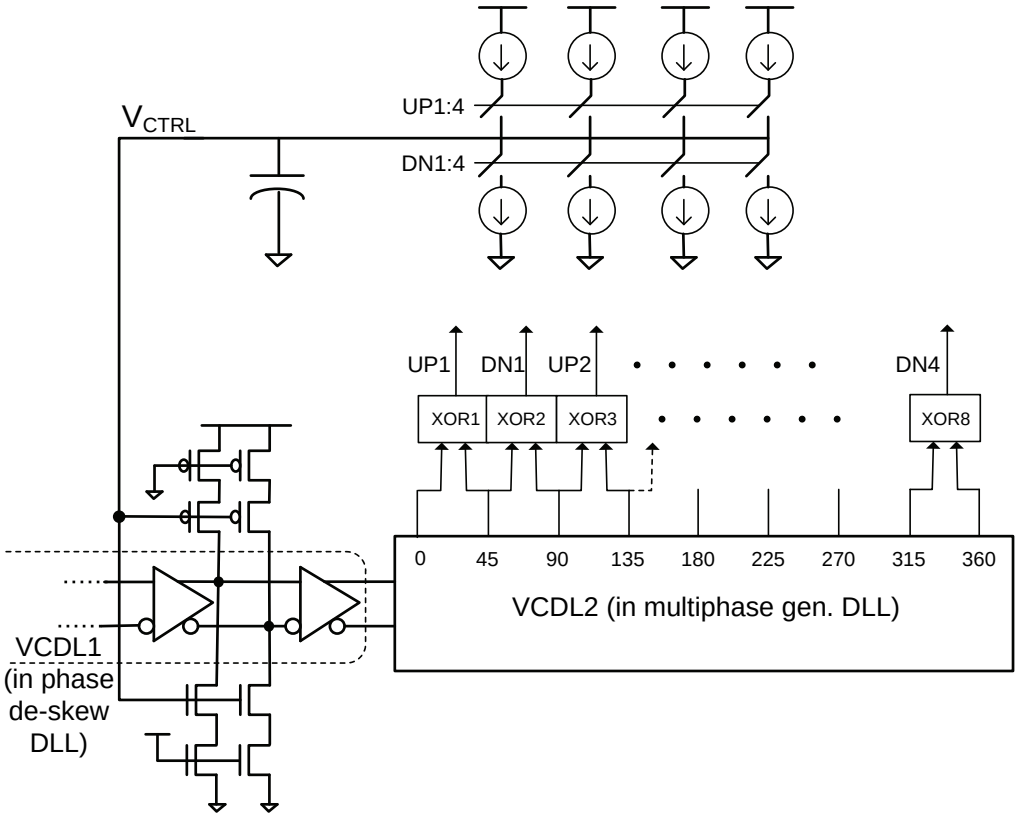


Figure 4.15: Phase spacing error corrector loop

or removes charge proportional to the difference between the sum of odd and even delays. The error is integrated on the capacitor and the V-to-I converter translates $V_{CTRL}$ into a current that conditions the edge of the clock entering the multi-phase generating DLL.

The error detectors are implemented as half-XOR gates that produce pulses whose widths are equal to the phase-spacing between adjacent clock phases. Figure 4.16 shows a schematic of the error-detector. Multiple error detectors are used to average out the effects of random mismatches between the adjacent stages. The charge pump, depicted in Figure 4.17 employs feedback biasing to ensure that static and dynamic

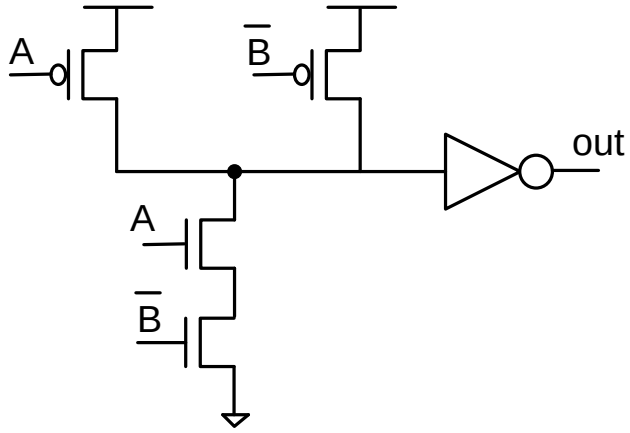


Figure 4.16: Half-XOR gate schematic

mismatches in the up and down currents do not introduce offsets that can result in residual delay mismatches. The V-to-I converter consists of linearized current sources that are biased from the same control voltage. The sizes of the sourcing and sinking current sources are chosen carefully to ensure symmetrical control.

This feedback loop, while performing the edge-conditioning, also has an impact on the overall skew of the clock. Close attention must be paid to the gain of the V-to-I-to-skew conversion (expressed in ps/V) and the bandwidth of this loop. The small additional skew that this circuit introduces, coupled with the low bandwidth of the feedback, ensures that there is no interaction with the local de-skew. Simulation results (Figure 4.12) indicate that the RMS DNL reduces from 16.9 ps to 2.1 ps. The efficacy of this method is limited by random mismatches in the delay cells and XOR gates.

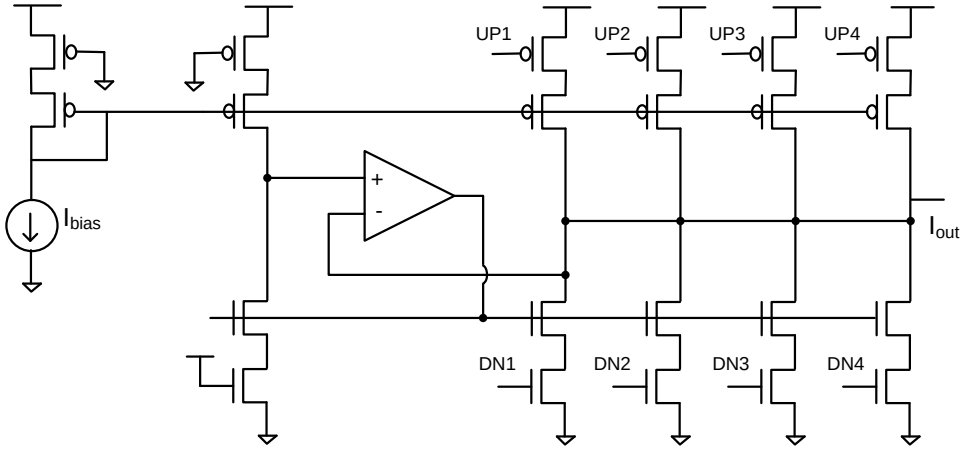


Figure 4.17: Charge pump with feedback-biasing schematic

### 4.4.2 Duty Cycle Corrector

The duty cycle corrector (DCC) shown in Figure 4.18, consists of a duty cycle sensing block (DC_sense) placed before the second DLL, and a duty cycle tuning block (DC_tune) placed before the first DLL.

The DC_sense circuit is a differential charge-pump that produces two voltages, *ntune* and *ptune* depending on the magnitude of duty cycle distortion in the clock signal. Common-mode feedback ensures *ntune* and *ptune* are differential with a common-mode of $V_{DD}/2$. Figure 4.19 shows a schematic of the common-mode feedback biasing circuit. Intuitively, it compares the sum of *ntune* and *ptune* to twice the *vref* value and adjusts the pull-up current in the charge-pump to set-up negative feedback. The DC_tune circuit, similar to that in [28], uses these voltages to slow down or speed up one of the clock edges. It consists of two tuning stages, where each stage consists of an inverter with weak pull-down and a parallel pull-down path whose strength depends on *ntune* or *ptune*. Consequently, each stage can adjust the

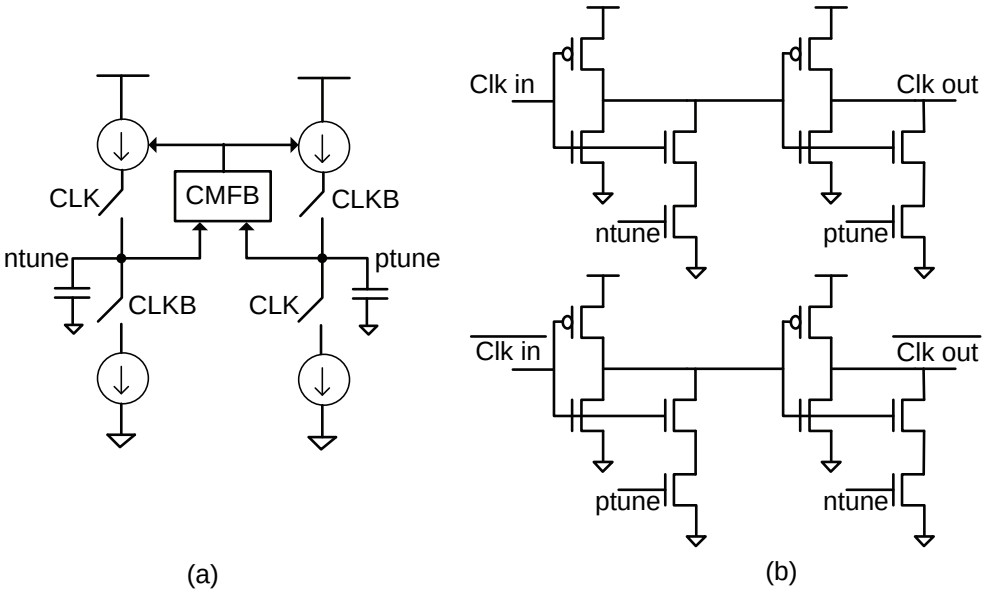


Figure 4.18: (a) Duty-cycle sensor (DC_sense), (b) Duty-cycle corrector (DC_tune)

falling transition of the clock signal. Post-extraction simulation results (Figure 4.20) show that the circuit suppresses duty cycle distortion by more than 5x, when the input duty-cycle error is within ±8%. Beyond that, the value of *ntune* and *ptune* start approaching the voltage supply rails, reducing the efficacy of the duty-cycle correction.

### 4.4.3 Experimental Results

The test chip described in Chapter 3 contains all of the correction loops described thus far. Figure 4.21 shows a block diagram of our measurement setup. A clock generator is used to create two clocks, 1.25 GHz and 2.5 GHz respectively, with programmable skew between them. The 1.25 GHz clock goes through the multi-phase DLL and samples the 2.5 GHz clock. As we experimentally sweep the skew

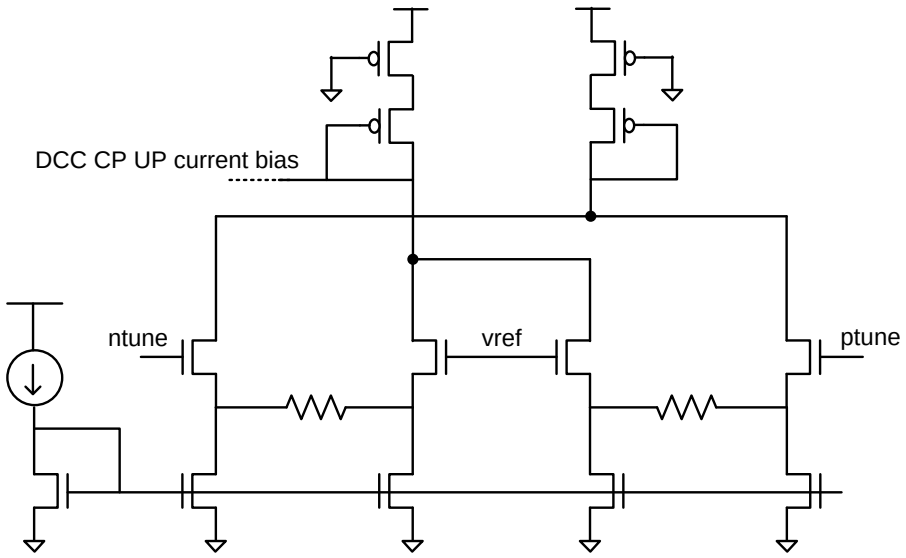


Figure 4.19: Common-mode feedback circuit employed in duty-cycle sensor

between the two clocks, we infer the skew at which the data and clock edge coincide by observing the received data. By looking at the output of each of the samplers and recording the relative skews at which the clock edges cross each other, the phase-spacings in the DLL clock phases can be obtained.

Figure 4.22 plots the measured phase spacings with the PSEC loop turned on and off for two representative DLLs in our parallel receiver. The reference clock frequency is 1.25GHz and the nominal delay of each stage is 100ps. A reduction in the differential nonlinearity (DNL) is observed. The residual error can be attributed to random delay cell mismatches that our correction loops cannot correct. In Figure 4.23 we plot the rms value (standard deviation) of the DNL in phase spacings for 7 DLLs from 7 receiver slices, with the PSEC loop enabled and disabled. The eighth channel is inaccessible due to insufficient connector spacing on the board. On average, the RMS DNL reduces by 42%.

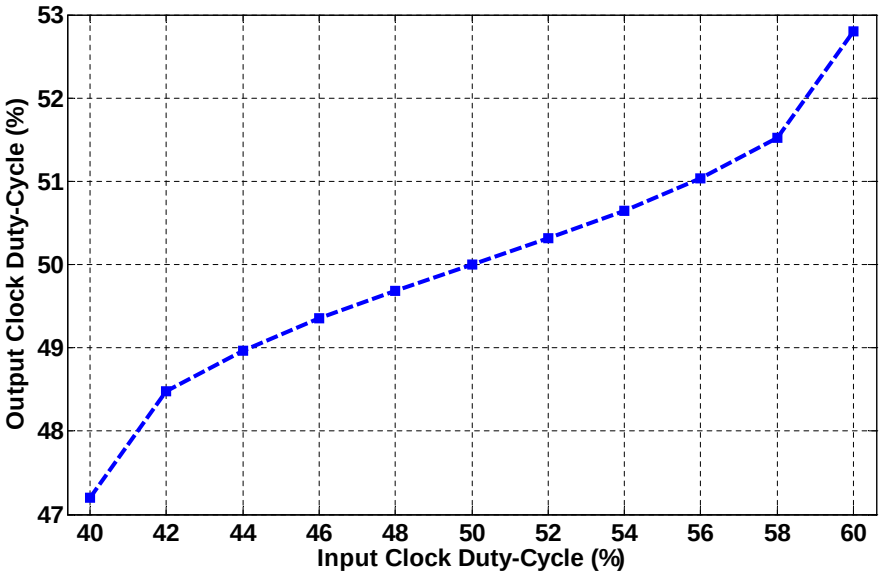


Figure 4.20: Efficacy of duty-cycle corrector (post-extraction simulations)

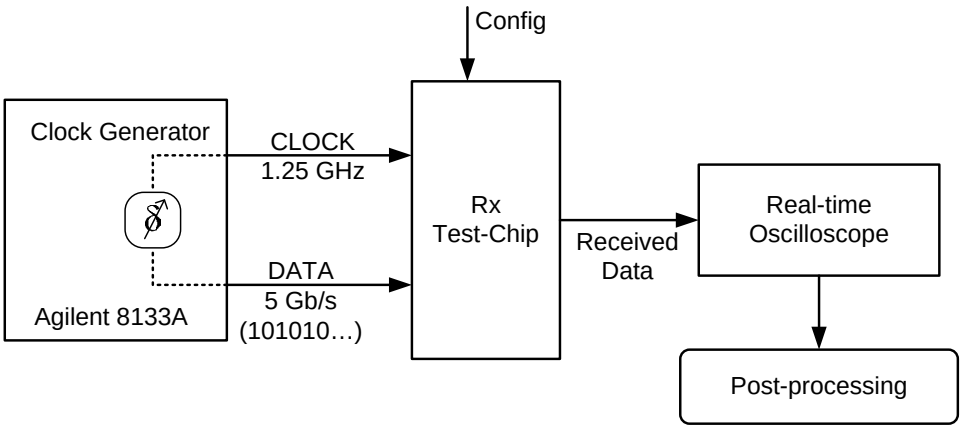


Figure 4.21: Measurement setup for phase-spacing measurements

Figure 4.24 presents the integral nonlinearity (INL) in the phase spacings with the DCC loop turned on and off. INL is defined as the deviation from the ideal location of the clock edge. We observe a reduction in the INL in phase spacings from 25.8ps to 7.8ps. The dotted line shows the theoretical INL for a 10% duty cycle error.

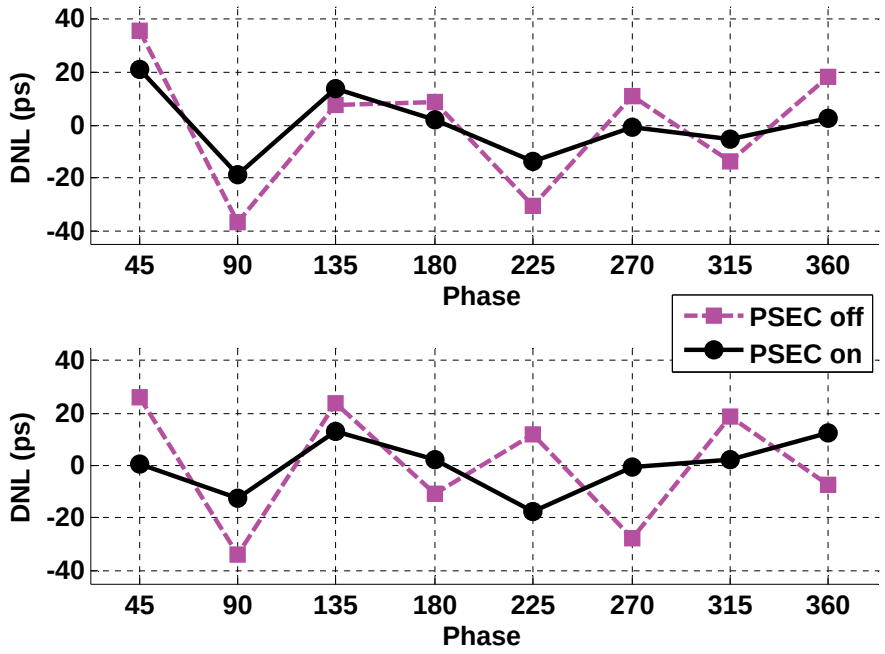


Figure 4.22: Differential non-linearity in phase spacings with the PSEC loop disabled and enabled for 2 representative DLLs

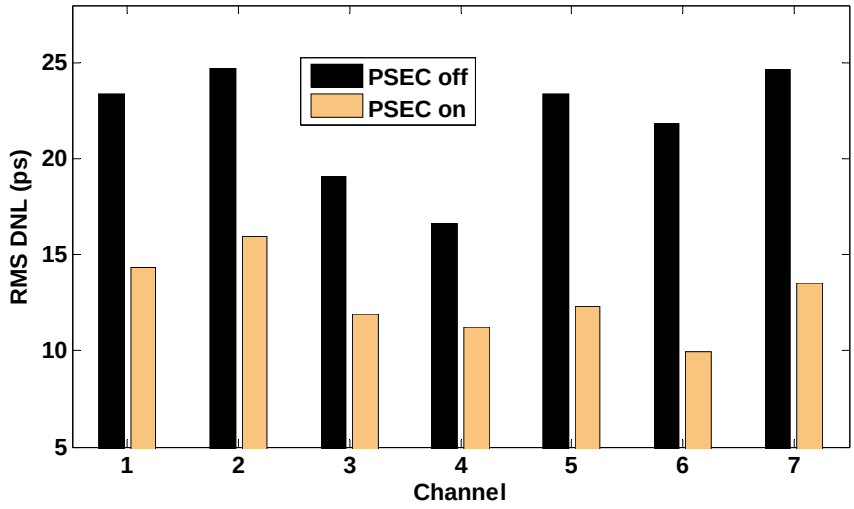


Figure 4.23: Comparison of the RMS DNLs for 7 DLLs with the PSEC loop disabled and enabled

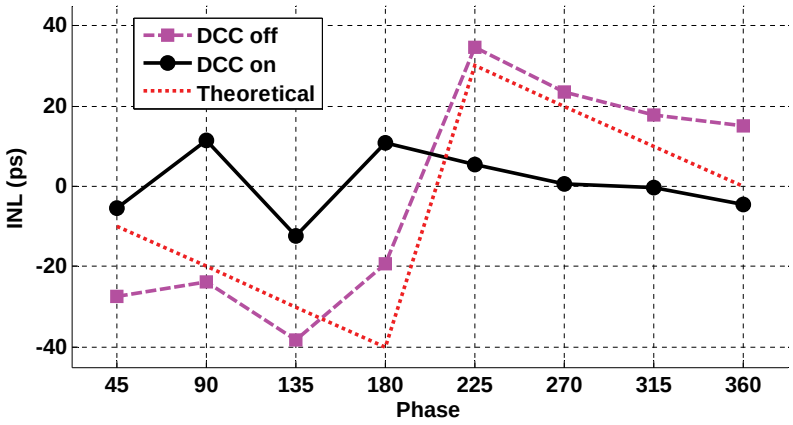


Figure 4.24: Integral non-linearity for the phase spacings with the duty-cycle correction loop disabled and enabled

### 4.4.4 Conclusion

This section describes techniques to correct for sources of phase spacing mismatches in multi-phase generating DLLs. A phase-spacing error correction circuit conditions the edges of the clock entering the DLL to reduce errors caused by the non-ideal slope of the clock edges. Another circuit corrects for any duty-cycle distortion in the input clock. Measurement results from our prototype test chip demonstrate the efficacy of these circuits in reducing DNL in the DLL's output clock phases. However, the improvements are limited by the mismatches in the correction circuits themselves. The next section shall discuss a digital scheme for performing phase calibration. Unlike the circuits described in this section, this technique is not DLL-specific. It allows for accurate, yet low-overhead phase-spacing mismatches in multi-phase clock generators.

## 4.5 Digital Phase-Calibration for Multi-Phase Clock Generating DLL

### 4.5.1 Introduction

In the previous section, we described analog design techniques to reduce phase-spacing mismatches in DLLs. These techniques are sensitive to process, voltage and temperature (PVT) variations that plague circuits designed in modern CMOS technologies. While experimental results in the previous section demonstrate some reduction in the phase-spacing DNL, the residual errors are significant. In this section, we shall present an accurate digital phase calibration technique that (a) is mostly digital and thus has much higher resilience to PVT variations and (b) is not specific to DLLs and applicable to all multi-phase clock generators. This also makes this technique useful in other systems that require multi-phase clocks, namely, time-interleaved A/D converters, clock frequency multiplication circuits, time-to-digital converters.

Other digital calibration techniques for multi-phase clock generators have been proposed in the literature [11, 6, 5, 8]. Some make assumptions about the overlying system where the multi-phase clocks are employed [11]. Other schemes come with high area overhead to perform digital calibration [5]. Sharing of calibration logic to perform the calibration serially reduces some of the area overhead [6, 8]. However, these schemes still require multiple digital-to-phase converters to translate the digital word produced by the calibration logic into phase-skew to perform the correction.

This work demonstrates a digital phase-calibration technique that addresses the need to reduce the area overhead associated with the multiple digital-to-phase con-

verter circuits required in the calibration of multi-phase clock generators. Our prototype test-chip employs the DLL described in Chapter 3 and replaces DCC and PSEC circuits with the proposed calibration technique. In experiments, the DLL operates at 1.6 GHz and produces 8 clock phases with maximum phase-spacing error of 450 fs.

### 4.5.2 Digital Phase Calibration Architecture

Figure 4.25 shows a high level block diagram of the DLL with phase calibration. The 8 uncalibrated clock phases out of the DLL feed into a bank of 8 digitally-controlled clock buffers (DCCB) that add variable delay to the clock phases and perform the calibration. The phase-spacing between adjacent clock phases out of these buffers are first measured by sub-sampling [10] with a clock having a small frequency-offset with respect to the DLL reference clock. Digital logic uses these phase-spacing measurements to compute the phase-spacing error and depending on the sign of the error, increment or decrement the control words that set the delay of the DCCBs. This is a simple first-order negative feedback loop that operates independently from the DLL and the calibration process does not interfere with the overlying circuitry. Since our prototype DLL is intended for CDR-type applications, the calibrated clock phases ($\Phi_{c1...8}$) also drive the data-path samplers.

One source of area inefficiency in this calibration architecture is the DCCBs. To have enough range to cover worst-case phase mismatches while having high resolution to reduce steady state errors, DCCBs need to have multi-bit D/A converters (DAC) — that convert the digital word either directly to phase or to an analog voltage that in-turn controls a variable-delay buffer. These DACs have high area cost, and we pro-

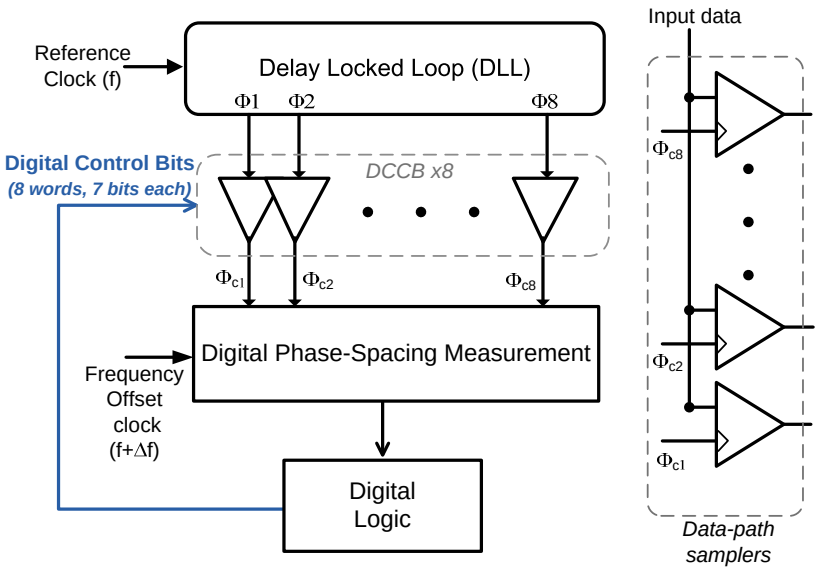


Figure 4.25: DLL with digital phase calibration

pose modifying the calibration loop shown in Figure 4.25 to share one DAC among the 8 digital-to-phase conversion buffers. Figure 4.26 presents a block diagram of the modified architecture. The DCCBs are replaced with analog variable delay buffers, and each of these buffers uses the same DAC to generate its control voltage. A multiplexer and de-multiplexer enable sharing of the DAC in a round-robin fashion, similar to time-division multiplexing. As the DAC dominates the overall area consumed by each DCCB, the shared-DAC topology makes the phase-calibration circuit area efficient.

The three main components of this calibration circuit are the phase-spacing error measurement circuit, the digital calibration logic and the digital-to-phase translation circuit that performs the error correction. Each of these will be discussed in detail in the rest of this section.

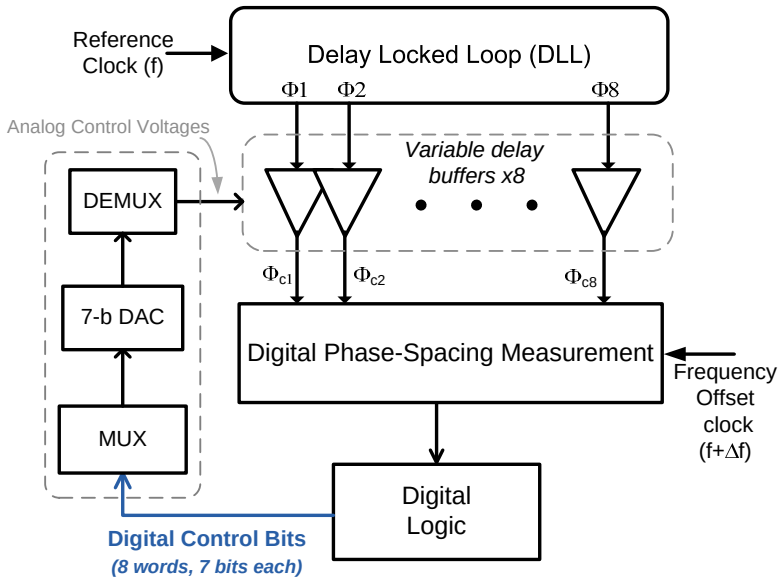


Figure 4.26: DLL with shared D/A converter based phase calibration

### 4.5.3 Phase-Spacing Error Measurement

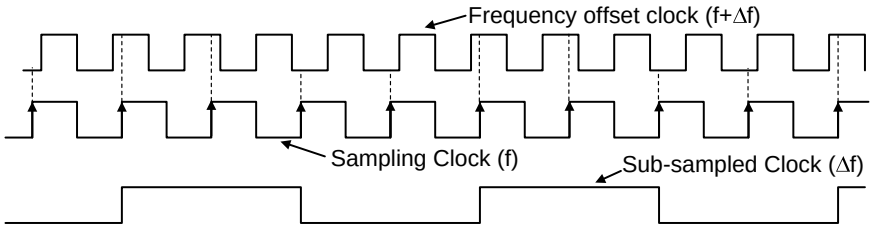


Figure 4.27: Sampling a clock with small frequency offset produces a low frequency clock

In digital phase calibration schemes, the resolution for phase error detection sets a lower limit the maximum steady-state error after calibration. We employ sub-sampling [10] to perform high-resolution yet simple and flexible digital measurement. In this technique, the DLL output clock phases sample another clock with a small (<1000 ppm) frequency offset with respect to the reference clock frequency. As illus-

trated in Figure 4.27, the output signal from each of these samplers is a sub-sampled clock whose frequency is equal to the frequency offset between the two clocks. Since the 8 samplers sample the same frequency-offset clock, the phase relationship between the output clocks of the DLL — the sampling clocks — are reflected between the phases of the sub-sampled clocks. This is effectively a stretching out of the time-axis by a large factor, possibly exceeding 1000x, enabling high-resolution digital measurement of the phase-spacing mismatches by counting the number of reference clock cycles between the adjacent sub-sampled clock rising edges. Figure 4.28 presents a block diagram of the error-measurement circuit. Multiple clock phases out of the DLL sample the frequency offset clock. Two back-to-back flip-flops (MH) are employed for meta-stability hardening of the outputs from the samplers. Also, the sub-sampled clocks do not have a clean 1-to-0 and 0-to-1 transitions because the cycle-to-cycle jitter on the two clocks can be more than the relative phase step in one clock cycle. A digital debouncing scheme, described later in this section, cleans up the sub-sampled clock edges. The rising edges of the debounced clocks are the start and stop signals to an 8-bit saturating counter, that counts the number of reference clock cycles in the interval. The counter output is a measure of the phase-spacing between the two input clocks.

The samplers are identical to those described in Chapter 3. The flip-flops for meta-stability hardening are from a standard cell library. A simple finite-state machine (FSM) implements the debouncing for the clock edges and controls the counter that measures the phase-spacings. The debouncing is not explicit; instead, the reference

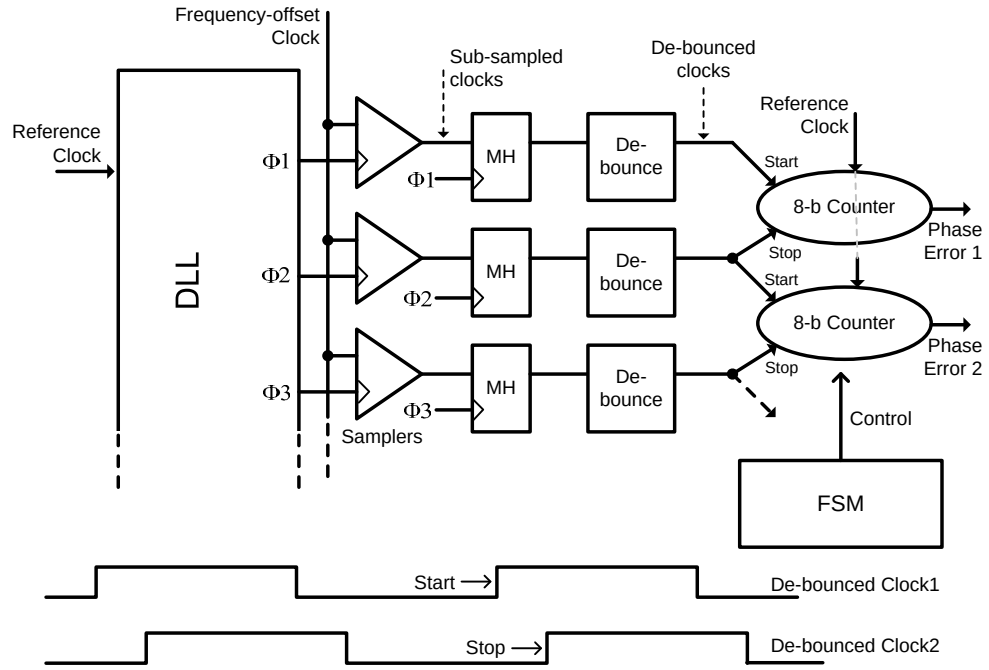


Figure 4.28: Digital phase-spacing measurement scheme

clock for the counter is gated according to the equation:

$$\text{Counter Clock} = S_{early} \cdot \overline{S_{late}} \cdot \text{CK}_{ref} \tag{4.7}$$

where the rising edge of $S_{early}$ starts the counter, the rising edge of $S_{late}$ stops the counter and $\text{CK}_{ref}$ is the reference clock. Using this gated clock for the counter is logically equivalent to positioning the rising edges of $S_{early}$ and $S_{late}$ at the mean location of the unclean (bouncy) edge. The FSM reads the counter value and resets the counter after $S_{early}$ reliably transitions back to 0.

A 8-b counter is employed to provide a measurement resolution $\leq$0.5 ps, well below the expected jitter on the output clock phases. It operates at the frequency of the reference clock making it difficult for a standard-cell based counter to meet the timing requirement. Instead, a ripple-counter is implemented, as shown in Figure 4.29.

Typically, digital designers favor synchronous counters over ripple counters since the output bits of the ripple counter do not change at the same time. In this design, there is plenty of time (100's of nanoseconds) available for the counter to reliably settle to its final value before it is read and reset. A 10-b counter combined with logic in the FSM implements saturation in the counter: the FSM monitors 2 extra MSB's at the end of the counter and raises a flag if either of them toggles.

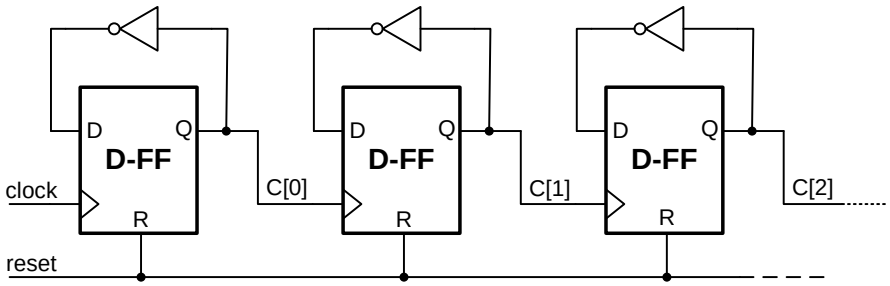


Figure 4.29: Ripple counter schematic

### Sampler Offsets

Offsets in the samplers add errors to the phase-spacing measurements, and if not properly dealt with, can add static offsets to the output phases. In the digital measurement circuit described above, a small positive offset in the sampler can delay the rising edge of the sub-sampled clock and advance the falling edge, as shown in Figure 4.30. Since the implemented algorithm only relies on rising edges of the sub-sampled clock, it is susceptible to sampler offsets. However, as the input clocks are full-swing with sharp edges, the residual errors are expected to be small.

It is possible to make a simple change to the error-measurement algorithm to make it immune to sampler offsets: performing two phase-spacing measurements at the

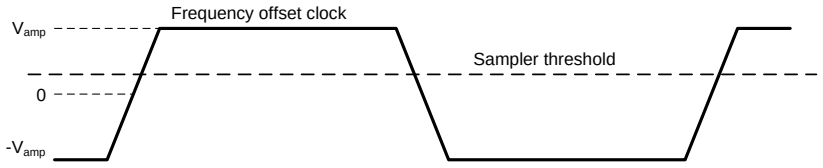


Figure 4.30: Sampler offsets can add delays to sub-sampled clock waveforms

rising edges and falling edges of the sub-sampled clocks, respectively, and averaging these measurements will cancel out the sampler offsets.

### 4.5.4 Digital Calibration Logic

The on-chip digital calibration logic is straight-forward: it compares the counter output with the ideal phase-spacing count and depending on the sign of the error increments or decrements the control word. The counter output is updated at the sub-sampled clock frequency and a clock derived from the sub-sampled clock times the digital calibration logic. Under typical operating conditions, the sub-sampled clock frequency is around 1 MHz and the digital logic consumes very little power. Figure 4.31 shows simulation results from a Matlab model of the calibration loop. It being a digital loop, the control word dithers about its ideal value in steady-state. In our on-chip implementation of the calibration, the steady state word dithers with a small bias in one direction. This is an artifact of a small bug in the digital logic — when the digital phase-spacing error measurement is 0, the control word should ideally not change. Instead, the on-chip logic treats this as equivalent to the clock phase being too early, adding a small (0.5 LSB) average skew to the output clock phase.

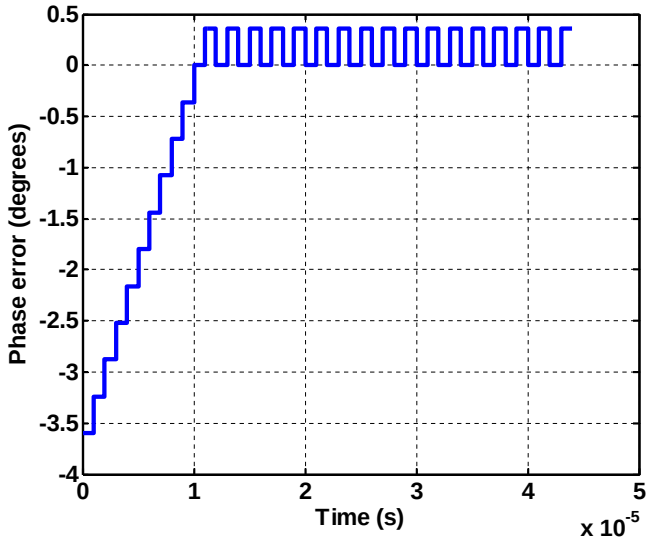


Figure 4.31: Matlab simulation result illustrating digital logic functioning to perform calibration

### 4.5.5 Digitally Controlled Clock Buffers with Shared D/A Converter

Figure 4.32 presents details of the proposed digital-to-phase conversion circuit with a shared DAC. Current-starved inverters (Figure 4.33) function as variable-delay clock buffers with a common 7-b DAC to translate digital bits into control currents. 7-bits of control ensure that the buffers have enough resolution to meet the worst-case phase offset while providing a resolution comparable to the error-measurement circuit (0.5 ps). The DAC connects to each of the 8 digital registers cyclically using 8 clocks (regck[1:8]), only one of which is high at any given time. Switches connect the DAC output voltage to the corresponding current-starved inverter bias node ($V_{bias}$). The pulses controlling this set of switches (bufck[1:8]) are narrower than regck[1:8] to account for the settling time of the DAC. When these bufck[1:8] pulses are low, the

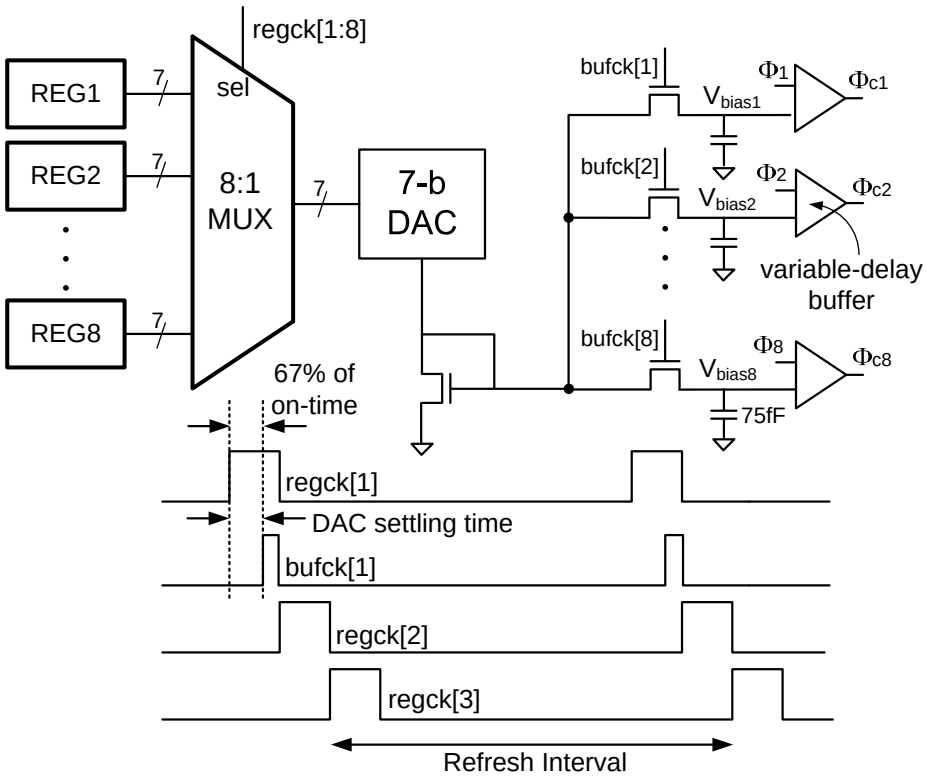


Figure 4.32: DAC sharing and cyclic-bias refresh for current-starved inverters

DAC does not actively drive the $V_{bias}$ node of the corresponding inverters. A 75fF metal-metal capacitor, constructed from 6 metal layers in this 8-metal layer CMOS process technology, acts as an analog memory element and holds the $V_{bias}$ node steady. However, the capacitor cannot hold the node voltage indefinitely and the bias voltage needs to be refreshed periodically to ensure that voltage does not wander too much. In steady state operation, each periodic refresh resets the wander on the $V_{bias}$ node to 0. The refresh interval is chosen such that the wander does not add appreciable jitter on the calibrated clock phases.

Since the $V_{bias}$ node is high-impedance for a majority of the time, care must be taken to minimize capacitive coupling from other nodes. As shown in Figure 4.33,

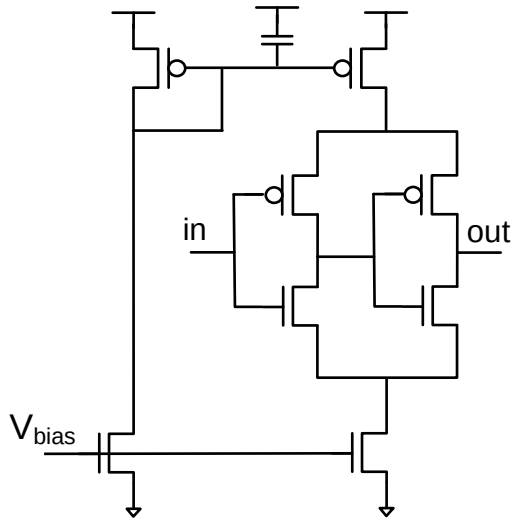


Figure 4.33: Current starved inverter schematic

two inverters connect to the tail current source, ensuring that the coupling from the two inverters cancel each other to first order.

The 7-b current-mode DAC used in this design is a hybrid of binary-coded and thermometer-coded current branches. Thermometer-coded branches for the 3 MSBs ensure monotonicity in the code vs. current transfer function. The use of cascoded current sources improves the output impedance of the current sources and linearize the DAC's transfer function. In post-extraction simulations, the systematic DNL for this DAC was 0.15 LSBs, and its settling time was better than 15 ns. The transistor size for the current source in the LSB branch limits the $3\sigma$ variation in current to less than 50%.

An on-chip pulse-generator generates the pulses required in the round-robin bias refresh circuit. In normal operation, a 100 MHz clock drives the counter-based pulse-generator. The CLK1,2...8 pulses span 3 periods of this clock, while the corresponding CK1,2...8 pulses are high only in the third clock period of this 3-clock period interval.

This gives the DAC 20 ns to settle.

### 4.5.6 Frequency Locked Loop

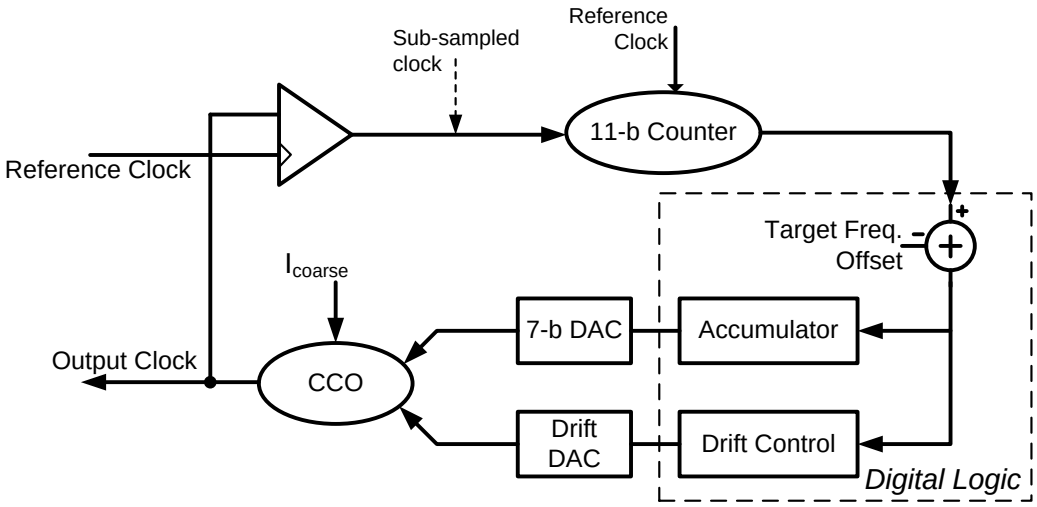


Figure 4.34: Frequency-locked loop block diagram

The phase-calibration scheme requires a frequency offset clock. While it is possible to drive the clock from off-chip, integrating the clock source on-chip makes for a self-contained calibration system. A simple frequency-locked loop (FLL) uses the reference clock to generate its frequency offset version. Figure 4.34 shows a block diagram of the FLL. This FLL also employs sub-sampling to determine the frequency error between the reference clock and its output clock. The reference clock samples the FLL's output clock, and the resulting sub-sampled clock frequency is the frequency-offset between the 2 input clocks to the sampler. The feedback loop tries to drive the frequency of the sub-sampled clock to the desired value by tuning the frequency of its output clock.

A counter counts the number of reference clock cycles in each sub-sampled clock period to produce a digital measure of the frequency offset. The counter has 11 bits of resolution to enable frequency offsets as low as 500ppm, and is similar in structure to one described earlier in this section. The digital logic compares this measurement against the target frequency offset, and uses the error to increment or decrement an accumulator that drives a DAC that controls the frequency of a current-controlled oscillator (CCO). The CCO has 3 control knobs: an analog coarse current control, set manually in this prototype-chip, that brings the CCO frequency close to the target frequency; a digital fine control that is driven by the feedback logic to achieve frequency lock; and a third digital control knob, drift compensation, to compensate for temperature-drift induced frequency drifts that can push the feedback loop out of range. Additional logic in the digital control uses the error measurements to sense if the FLL is out of range, and adjust the control word to a drift DAC.

The CCO delay cells topology are similar to those used in the delay-line in the DLL. From post-extraction simulations, the coarse gain of the CCO is 3 MHz/$\mu$A and the fine gain is 100 KHz/$\mu$A. Combined with a 7-b current-mode DAC, the resolution of this oscillator is around 80 KHz and the range is 10 MHz. Unfortunately, in our prototype test-chip the oscillator failed to oscillate and we were not able to characterize its performance experimentally.

### 4.5.7 Experimental Results

Figure 4.35 shows a micrograph of the phase calibration test chip, fabricated in a 0.13$\mu$m CMOS process. To test this chip, two clock sources drive in two clocks

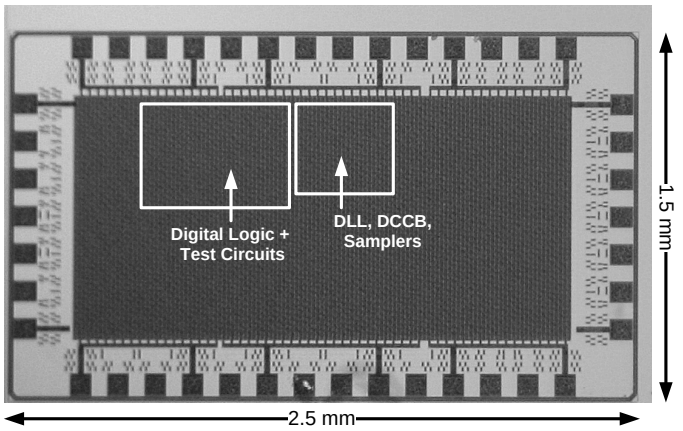


Figure 4.35: Chip micrograph with floor-plan overlay

with a small frequency offset, while a real-time oscilloscope stores the output sub-sampled clocks (Figure 4.36). The phase-spacing between the clock phases is inferred by processing the digitized data from the oscilloscope. As mentioned in the previous subsection, the on-chip FLL was dead-on-arrival and we had to rely on our alternative test-plan of using an external clock generator to characterize the performance of the phase calibration.

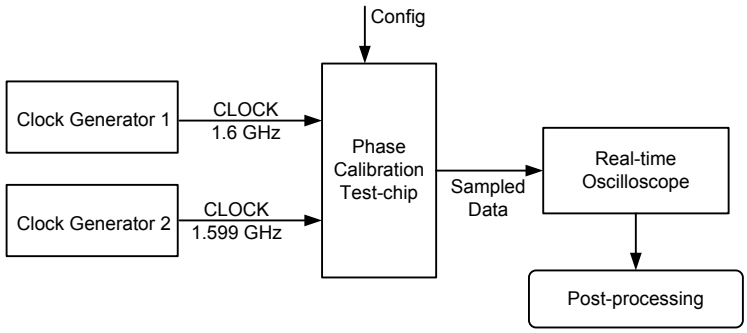


Figure 4.36: Experimental setup for DLL phase-spacing DNL measurement

### Calibration Measurements

The DLL runs at 1.6 GHz and produces 8 output phases, intended for 6.4 Gb/s applications. There is a frequency offset of 1 MHz (625 ppm) between the reference clock and calibration clock. The ideal phase spacing between the 8 clock phases is 78.125 ps. Figure 4.37 presents the differential non-linearity (DNL) in uncalibrated and calibrated clock phases from six chips: two typical corner and four slow corner chips . The maximum DNL value is 37 ps, which is more than 47% of the ideal phase spacing. In the slow corner chips, the non-linearities seem to have both a systematic and a random component. Duty-cycle distortion of the reference clock in the clock buffers that deliver the clock to the DLL are the dominant source of systematic offsets, as can be inferred from the shape of the DNL curves.

After calibration the maximum DNL in these six chips is reduced to 0.45 ps, less than 0.6% of the ideal phase spacing. The DNL after calibration has a mean value of 0.3 ps, and this nonzero mean error is an artifact of a small bug in our digital calibration logic. Also, these calibration results ignore the sampler offset induced residual phase errors — there is no direct visibility of the clock phases in the test chip and phase-spacings inferred using the sub-sampled clocks do not capture the impact of sampler offsets.

Figure 4.37 also plots the calibration results for a scenario with additional 15 ps rms jitter on the frequency offset clock to mimic a low-quality, low-overhead FLL, and demonstrates robust operation. While the jittered frequency-offset clock hardly affects the mean position of the calibrated clock phases, it does increase the jitter on these clock phases. The additional input jitter translates into noise in the phase-

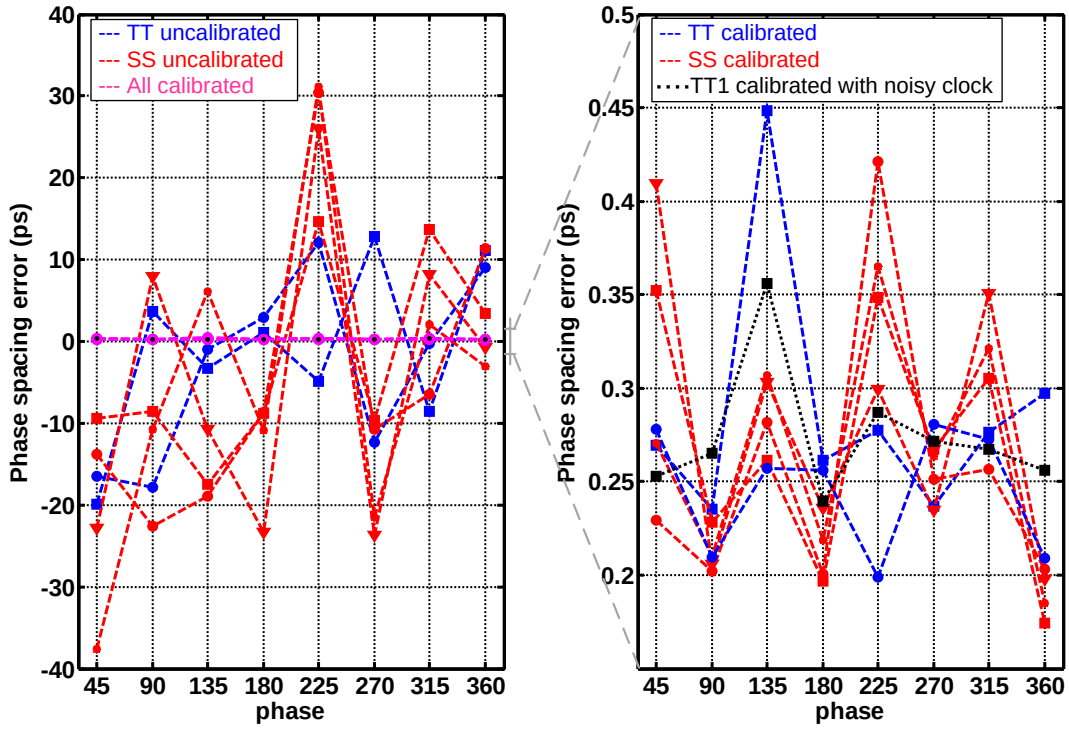


Figure 4.37: Uncalibrated and calibrated DLL DNL measurement from 6 chips

error measurement that, in turn, manifests as jitter on the calibrated clock phases. Figure 4.38 plots this increase in jitter when varying amounts of additional jitter is added to the frequency-offset clock.

Figure 4.39 explores what frequency offset to use in the calibration clocks. It plots the DNL in calibrated clock phases for 5 different values of frequency offset. The mean error is higher when the frequency offset is larger since the digital error measurement is coarser. If the digital logic was designed correctly and there was zero mean error, there is not a big difference in the DNL in these experiments. Figure 4.40 shows the delay vs. code transfer function of the variable delay clock buffers. The

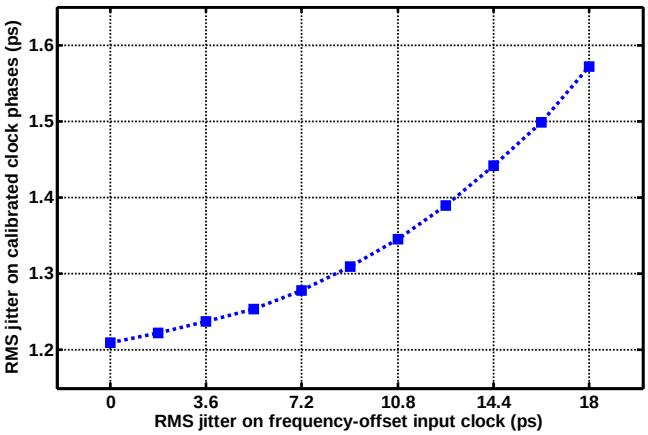


Figure 4.38: Calibrated clock rms jitter vs. frequency-offset clock rms jitter

measured range is higher than expected.

### DAC sharing scheme evaluation

To visualize the performance penalty that the DAC sharing scheme imposes, Figure 4.41 plots the histogram of the location of the rising edge of the sub-sampled clock for two different refresh intervals of the $V_{bias}$ node: 240ns and 240$\mu$s. Since this is a plot of the sub-sampled clock, the time-axis is stretched out by a factor of 1600. The peak-to-peak uncertainty in the position of the rising edge is the sum of the peak-to-peak jitters on the frequency-offset clock and the calibrated clock phase, respectively. Any wander in the voltage at the $V_{bias}$ node when it is being held by the capacitor, translates into peak-to-peak jitter on the calibrated clock phase. When the refresh rate is 240ns, and the $V_{bias}$ voltage is steady, the peak-to-peak uncertainty is normally disturbed with a peak-to-peak value of 20.6 ns that is equivalent to 12.9 ps of combined jitter. When the refresh rate is 240$\mu$s, we observe a long tail in the

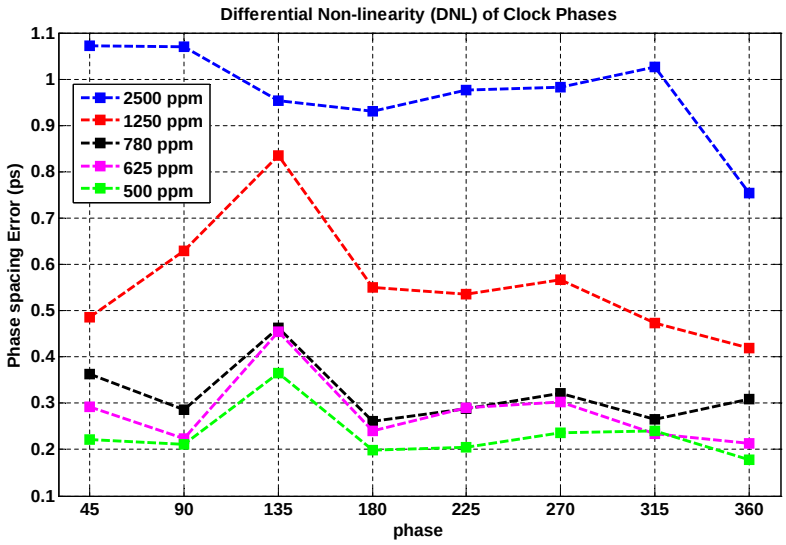


Figure 4.39: DLL phase-calibration results for different frequency offset values

histogram, and a combined jitter of 18.7 ps, indicating voltage wander. The shape of histograms indicates a spread in the sampling instant at the slower refresh rate. The histograms exhibit spikes because the measurement is equivalent to sampling the jitter at discrete intervals instead of making a continuous measurement.

Figure 4.42 plots the peak-to-peak jitter on the calibrated clock vs. the refresh interval. This measurement is made by sweeping a synchronous clock edge across the edge of the calibrated sampling clock and observing the sampled data. The amount of skew that needs to be added to one of the edges to make the sampled data reliably flip from 0 to 1 is the total peak-to-peak jitter. From this, the measured peak-to-peak jitter on the input synchronous clock is subtracted to arrive at the peak-to-peak jitter on the calibrated clock.

We observe that the peak-to-peak jitter does not increase appreciably until the refresh interval is in the 10's of $\mu$s range. Given that we use a refresh interval of 0.5 $\mu$s

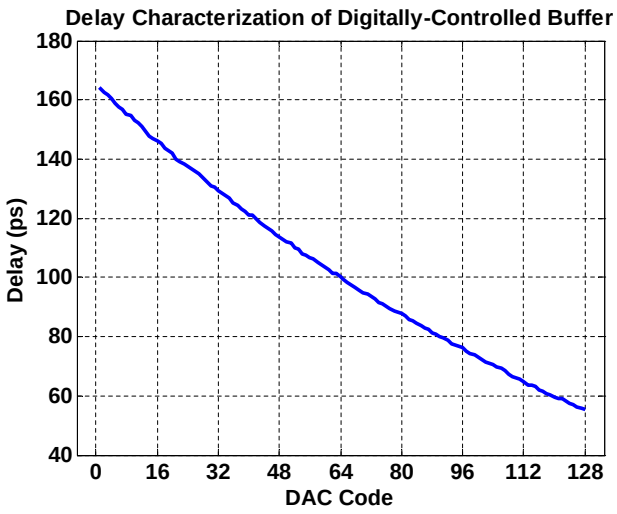


Figure 4.40: Delay vs. control code for variable delay clock buffers

under normal operating conditions, the cyclic-refresh scheme does not impose any performance penalties. We summarize the area savings of the DAC refresh scheme in Table 4.1. The circuits required to enable DAC sharing (switch arrays, cyclic refresh clock generators, capacitors) consume the area of equivalent of less than 2 DACs, while providing area savings of 7 DACs. Overall, we consume 38% of the area that would have been consumed by a similar scheme employing 8 separate DACs for the 8 buffers. The digital logic consumes $5000\mu m^2$. While this implementation employed separate logic blocks for the calibration each of the phases, it is possible to share the circuitry and perform the calibration serially [6].

The DLL, the calibration scheme and test circuits consume 20 mW of power from a 1.2V supply. Table 4.2 provides a breakdown of the power consumption of the various sub-blocks. Overall, the calibration scheme imposes a 27% power overhead.

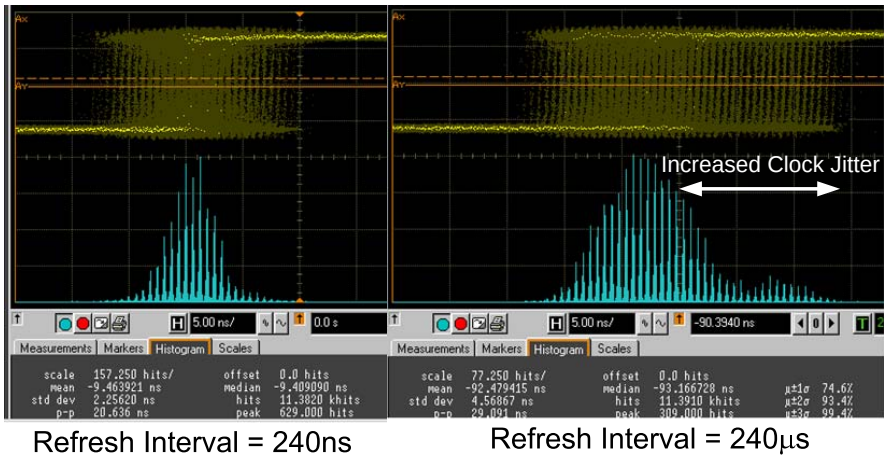


Figure 4.41: Histogram of location of rising edges of sub-sampled clock for 2 refresh-interval values

Table 4.1: Area consumption of various circuit sub-blocks (in $\mu m^2$)

| | |
|---|---|
| Current starved inverters (x8) | 680 |
| DAC | 1800 |
| Input switch array | 1400 |
| Output switch + capacitors (x8) | 360 |
| Cyclic refresh pulse generator | 1500 |
| Digital Logic | 5000 |

### 4.5.8 Summary

This section described a digital phase calibration technique for calibrating multi-phase clock generation circuits. The technique, applied to a DLL in our prototype test-chip, employs sub-sampling for phase error measurement and an area-efficient shared-DAC topology for error correction. This technique reduces the DNL in phase-spacings from 46% to less than 0.6%. The shared-DAC scheme reduced the area consumption of the digitally-controlled delay buffers by more than 60%.

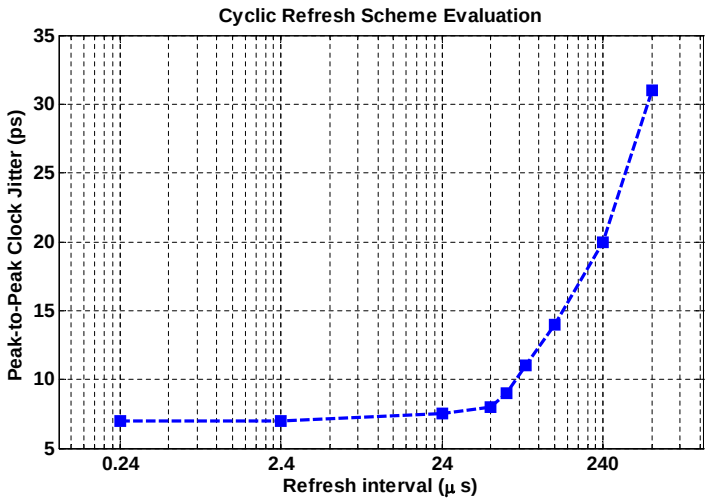


Figure 4.42: Calibrated clock jitter vs. refresh interval

Table 4.2: Power consumption of various circuit sub-blocks

| | |
|---|---|
| DLL | 11mA |
| Data-path samplers | 2mA |
| Calibration samplers | 2mA |
| Shared DAC + delay buffers | 1mA |
| Digital logic | 0.5mA |

## 4.6 Conclusions

This chapter focussed on reclaiming timing margin lost to static phase-offsets in the sampling clocks in receivers. It began by contrasting the use of PLLs and DLLs in the receivers, and making a case for DLLs. However, DLLs are more susceptible to phase-spacing mismatches. Two techniques were proposed for improving the phase-spacings in DLLs: the first was a technique specific to DLLs that conditions the clock entering the delay-line to reduce DNL; the second was a more widely-applicable techique that treats all phase-spacing errors as random, and uses a digital algorithm to position the clock edges close to their ideal locations.

# Chapter 5

# Conclusion

For parallel interfaces in high performances microprocessors to deliver ever increasing I/O bandwidths with limited pin resources, each serial-link must support higher data-rates. Further complicating the challenge, interfaces must operate in increasingly noisy on-chip environments, be power-aware and robust against PVT variations. In this thesis, we have introduced, analyzed, and demonstrated techniques that improve the timing margin in receivers to enable link operation at higher data-rates. The challenges outlined above drive key design choices.

The collaborative timing recovery architecture enables robust link operation in power-supply noise dominated environments. Power-supply noise induced jitter on the transmitted data and sampling clocks is difficult to track using traditional receiver designs. The proposed receiver employs collaboration between multiple channels in a parallel receiver to overcome this challenge. Sharing timing information from several synchronous data streams enables wide-band jitter tracking while reducing dithering jitter on received clock. Collaborative timing recovery also provides area and power

savings — it replaces multiple clock recovery circuits with one global digital clock synthesizer. The receiver performance was verified in a prototype implementation, fabricated in a $0.13\mu m$ CMOS technology, with multiple system-level measurements.

One feature of the collaborative receiver architecture is the use of cascaded DLLs to de-skew the clock and sample 4-way interleaved data, respectively, to avoid phase-filtering in the path of the sampling clock. We analyzed the sources of static skews in the clock phases out of the DLL, and demonstrated circuits that shape the clock entering the DLL to reduce these mismatches. In a second test-chip, we tested a more general approach to building accurate multi-phase clock generators. A digital calibration circuit compensates for all phase-spacing errors in a DLL. This technique borrows a periodic-refresh technique from DRAM design, to use just one area-hungry D/A converter to calibrate 8 clock phases thereby reducing the area-overhead of the calibration.

Overall, a few themes emerged from this work. Both system- and circuit-level challenges emerge when designing I/O interfaces with improved timing margins. Meeting these challenges not only requires innovation in the design of the basic transistor-level building blocks, but also in the system architecture design. Also, sharing and re-using of circuit blocks enables more efficient and effective designs, well-suited to power-critical applications. Finally, because analog device characteristics in future CMOS processes are not expected to improve, digital circuits replace or assist their analog counterparts. This trend is likely to become more pronounced, making this work attractive for interfaces for high performance microprocessors.

# Bibliography


[1] A. Agrawal, P.K. Hanumolu, and G-Y. Wei. A 8x3.2Gb/s Parallel Receiver with Collaborative Timing Recovery. *ISSCC Dig. Tech. Papers*, pages 468–469, 2008.

[2] MR Ahmadi, A. Amirkhany, and R. Harjani. A 5Gbps 0.13$\mu$m CMOS Pilot-Based Clock and Data Recovery Scheme for High-Speed Links. *Solid-State Circuits, IEEE Journal of*, 45(8):1533–1541, 2010.

[3] K. Aygun and others. Power Delivery for High Performance Microprocessors. *Intel Technology Journal*, November 2005.

[4] G. Balamurugan and N. Shanbhag. Modeling and mitigation of jitter in multi-Gbps source-synchronous I/O links. *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 254–260, 2003.

[5] V. Balan, J. Caroselli, and J. Chern. A 4. 8-6. 4-Gb/s serial link for backplane applications using decision feedback equalization. *IEEE Journal of Solid-State Circuits*, 40(9):1957–1967, 2005.

[6] F. Baronti, D. Lunardini, R. Roncella, and R. Saletti. A self-calibrating delay-locked delay line with shunt-capacitor circuit scheme. *IEEE Journal of Solid-State Circuits*, 39(2):385, 2004.

[7] B. Casper et al. A 20Gb/s Forward Clock Transceiver in 90nm CMOS. *ISSCC Dig. Tech. Papers*, pages 90–1, 2006.

[8] H.H. Chang, J.Y. Chang, C.Y. Kuo, and S.I. Liu. A 0.7–2-GHz Self-Calibrated Multiphase Delay-Locked Loop. *IEEE Journal of Solid-State Circuits*, 41(5):1051, 2006.

[9] J.M. Chou, Y.T. Hsieh, and J.T. Wu. A 125MHz 8b digital-to-phase converter. In *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC*, pages 436–505, 2003.

[10] P.K. Das, B. Amrutur, J. Sridhar, and V. Visvanathan. On-chip clock network skew measurement using sub-sampling. In *IEEE Asian Solid-State Circuits Conference, 2008. A-SSCC'08*, pages 401–404, 2008.

[11] M. El-Chammas and B. Murmann. A 12-GS/s 81-mW 5-bit Time-Interleaved Flash ADC with Background Timing Skew Calibration. In *VLSI Symposium*, pages 157–158, 2010.

[12] M.S. Gupta, J.L. Oatley, R. Joseph, G-Y. Wei, and D.M. Brooks. Understanding Voltage Variations in Chip Multiprocessors using a Distributed Power-Delivery Network . *Proceedings of the conference on Design, automation and test in Europe*, pages 624–629, 2007.

[13] P.K. Hanumolu, G. Wei, and U. Moon. Equalizers for high-speed serial links. *International Journal of High Speed Electronics and Systems*, 15(2):429–458, 2005.

[14] PK Hanumolu, G.Y. Wei, and U. Moon. A Wide-Tracking Range Clock and Data Recovery Circuit. *Solid-State Circuits, IEEE Journal of*, 43(2):425–439, 2008.

[15] M. Hossain and AC Carusone. A 6.8 mW 7.4 Gb/s clock-forwarded receiver with up to 300MHz jitter tracking in 65nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 158–159. IEEE, 2010.

[16] G. Konstadinidis, M. Rashid, PF Lai, Y. Otaguro, Y. Orginos, S. Parampalli, M. Steigerwald, S. Gundala, R. Pyapali, L. Rarick, et al. Implementation of a Third-Generation 16-Core 32-Thread Chip-Multithreading SPARC Processor. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 84–597. IEEE, 2008.

[17] NA Kurd, S. Bhamidipati, C. Mozak, JL Miller, TM Wilson, M. Nemani, and M. Chowdhury. Westmere: A family of 32nm IA processors. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 96–97. IEEE, 2010.

[18] P. Larsson. A 2-1600-MHz CMOS clock recovery PLL with low-Vdd capability. *Solid-State Circuits, IEEE Journal of*, 34(12):1951–1960, 1999.

[19] B.J. Lee, M.S. Hwang, S.H. Lee, and D.K. Jeong. A 2.5-10-Gb/s CMOS transceiver with alternating edge-sampling phase detection for loop characteristic stabilization. *Solid-State Circuits, IEEE Journal of*, 38(11):1821–1829, 2003.

[20] J. Lee, KS Kundert, and B. Razavi. Analysis and modeling of bang-bang clock and data recovery circuits. *Solid-State Circuits, IEEE Journal of*, 39(9):1571–1580, 2004.

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

[22] F. O'Mahony et al. A 27Gb/s Forwarded-Clock I/O Receiver using an Injection-Locked in 90nm CMOS. *ISSCC Dig. Tech. Papers*, pages 452–453, 2008.

[23] F. O'Mahony, S. Shekhar, M. Mansuri, G. Balamurugan, J.E. Jaussi, J. Kennedy, B. Casper, D.J. Allstot, R. Mooney, and H. Intel. A 27Gb/s Forwarded-Clock I/O Receiver Using an Injection-Locked LC-DCO in 45nm CMOS. *ISSCC Dig. Tech. Papers*, pages 452–453, 2008.

[24] B. Razavi. *Design of analog CMOS integrated circuits*. McGraw-Hill, Inc. New York, NY, USA, 2000.

[25] JL Shin, K. Tam, D. Huang, B. Petrick, H. Pham, C. Hwang, H. Li, A. Smith, T. Johnson, F. Schumacher, et al. A 40nm 16-core 128-thread CMT SPARC SoC processor. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 98–99. IEEE, 2010.

[26] S. Sidiropoulos and MA Horowitz. A semidigital dual delay-locked loop. *Solid-State Circuits, IEEE Journal of*, 32(11):1683–1692, 1997.

[27] J. Sonntag and J. Stonick. A digital clock and data recovery architecture for multi-gigabit/s binary links. *Conference 2005, IEEE Custom Integrated Circuits*, pages 532–539, 2005.

[28] A.H. Tan and G-Y. Wei. Phase Mismatch Detection and Compensation for PLL/DLL Based Multi-Phase Clock Generator. *Conference 2006, IEEE Custom Integrated Circuits*, pages 417–420, 2006.

[29] Gu-Yeon Wei. Enery Efficient I/O Interface Design With Adaptive Power-Supply Regulation. *Ph.D thesis, Stanford University*, 2001.

[30] D. Wendel, R. Kalla, R. Cargoni, J. Clables, J. Friedrich, R. Frech, J. Kahle, B. Sinharoy, W. Starke, S. Taylor, et al. The implementation of POWER7: A highly parallel and scalable multi-core high-end server processor. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 102–103. IEEE, 2010.

[31] N.H.E. Weste and D.F. Harris. *CMOS VLSI design: a circuits and systems perspective*. Addison-Wesley, 2005.

[32] K.L.J. Wong, H. Hatamkhani, M. Mansuri, and C.K.K. Yang. A 27-mW 3.6-gb/sI/O transceiver. *Solid-State Circuits, IEEE Journal of*, 39(4):602–612, 2004.

[33] E Yeung. *Design of High Performance and Low Cost Parallel Links*. PhD thesis, stanford University, 2002.

[34] E. Yeung and MA Horowitz. A 2.4 Gb/s/pin simultaneous bidirectional parallel link with per-pin skew compensation. *Solid-State Circuits, IEEE Journal of*, 35(11):1619–1628, 2000.