A CDR with a Digital Threshold Decision Technique and a Cyclic Reference Injected DLL Frequency Multiplier with a Period Error Compensation Loop

By
Qingjin Du

A thesis submitted to

The Faculty of Graduate Studies and Research

In partial fulfillment of

The requirements for the degree of

Doctor of Philosophy

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Electronics

Carleton University

Ottawa, Ontario

Canada

December 2007

Your file Votre référence
ISBN: 978-0-494-36781-0
Our file Notre référence
ISBN: 978-0-494-36781-0

The undersigned recommend to

the Faculty of Graduate Studies and Research

acceptance of this thesis

"A CDR with a Digital Threshold Decision Technique and a Cyclic Reference Injected DLL Frequency Multiplier with a Period Error Compensation Loop"

Submitted by Qingjin Du

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Thesis Supervisor

Prof. Tad Kwasniewski

Chair, Department of Electronics

Carleton University

December 2007

The information used in this thesis comes in part from the research program of Dr. Tad Kwasniewski and his associates in the VLSI in Communications group. The research results appearing in this thesis represent an integral part of the ongoing research program. All research results in this thesis including tables, graphs and figures but excluding the narrative portions of the thesis are effectively incorporated into the reach program and can be used by Dr. Kwasniewski and his associates for educational and research purpose, including publication in open literature with appropriate credits. The matters intellectual property may be pursued cooperatively with Carleton University and Dr. Kwasniewski and where and when appropriate.

Abstract

This dissertation proposes a CDR with a digital-threshold decision technique which enables high jitter tolerance performance, fast acquisition, low complexity and low power consumption, and a cyclic reference-injected, programmable DLL based frequency multiplier with a novel period error compensation loop to reduce the output spur as well as with a new switching scheme to avoid harmonic locking without initialization or extra locking detection circuitry.

First, the recently reported CDR circuits and DLL based frequency multipliers are reviewed according to the given classification. Performances are compared against each type of CDR or DLL frequency multiplier, and the advantages and disadvantages are discussed.

Then the proposed digital threshold decision technique and the CDR circuit implementation as well as the measured results are presented. The digital threshold decision technique in the general cases is first given, and with the chosen parameters, the CDR with the proposed technique is followed. The CDR functionality is verified with Matlab model simulation, and an event-driven simulation verifies the high jitter tolerance performance of the proposed CDR decision technique. The CDR was implemented in CMOS 90nm technology with new circuit ideas to reduce the circuit complexity and the power consumption. The total transistor count of the CDR excluding the output buffers is approximately 900. With the input data from an on-chip 7-bit PRBS data generator, and the sampling clocks from an on-chip DLL multi-phase clock generator, the CDR circuit was measured with an on-chip BER test circuit, and the expected correct phase tracking was observed from 2.0Gbps to 3.5Gbps. By measuring the maximum difference between the baud rate and the sampling clock rate, the jitter tolerance performance obtained from the measurements is close to the theoretically analyzed one, which is well comparable or even better than the reported ones. The measured power consumption of the core CDR circuit is 4mW at 2.5Gbps and 5.3mW at 3.0Gbps at a 1.2V power supply. All these verify the proposed CDR with the digital threshold decision technique and its CMOS implementation.

In the DLL based frequency multiplier design, the in-lock error from all error sources including the re-alignment error caused by the cyclic reference edge injection contributes to the spurious power level. So a low-bandwidth auxiliary loop, which was first verified in Matlab model simulation, is employed to compensate the output period error caused by the in-lock errors from various sources for spurious power reduction. By employing a novel switching control scheme, the circuit is capable of locking to frequencies either above or below the start-up frequency without initialization. With a dynamic frequency divider, programmable multiplication ratios from 13 to 20 are achieved with an output frequency range of 900MHz to 2.9GHz. The circuit was implemented in TSMC 0.18 μm CMOS technology and measured with the reference from an RF generator, and the measured results show a significant output spur improvement from -23dB to -46.5dB at 1.216GHz. The measured cycle-to-cycle timing jitter at 2.16GHz is 1.6ps (rms) and 12.9ps (peak-to-peak), and the phase noise is -110dBc/Hz at 100kHz offset with a power consumption of 19.8mW at a 1.8V power supply. The measurements also verify the proposed period error compensation method as well as the new switching scheme and their transistor-level implementation.

Table of Contents

List of Abbreviations

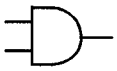
BER	Bit Error Rate
CDR	Clock and Data Recovery
CID	Consecutive Identical Digit
CMFC	Common Mode Feedback Circuit
CMU	Clock Multiplying Unit
CP	Charge Pump
DFE	Decision Feedback Equalizer
DLL	Delay Locked Loop
DETFF	Double Edge Triggered Flip Flop
DPLL	Digital PLL
FD	Frequency Detector
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
ISI	Inter Symbol Interference
JT	Jitter Tolerance
LF	Loop Filter
LFSR	Linear Feedback Shift Register
PD	Phase Detector
PDF	Probability Density Function
PISO	Parallel Input Serial Output
PLL	Phase Locked Loop

PRBS	Pseudo-Random Bit Stream
PVT	Process, Voltage, Temperature
QPD	Quadrature Phase Detector
RMS	Root Mean Square
RxP/RxN	Receiver Positive/Negative input
SCL	Source Coupled Logic
SIPO	Serial Input Parallel Output
SFI-5	Voltage-controlled oscillator
SNR	Signal to Noise Ratio
SONET	Synchronous Optical Network
TxD	Transmitter Data
TxP/TxN	Transmitter Positive/Negative output
UI	Unit Interval
VCDL	Voltage-Controlled Delay Line

List of Tables

List of Figures

List of Symbols


AND gate


Capacitor

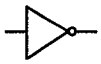

Crystal


D Flip Flop


Delay cell in VCDL

Buffer

Ground


Inverter


Resistor


Switch

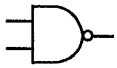

Inductor


Input port in Matlab models


Output port in Matlab models


Low-pass filter


Sum


NAND gate


OR gate


Voltage-controlled oscillator


VDD

NMOS transistor

PMOS transistor

Current source

Multiplexer

XOR gate

CHAPTER 1 *Introduction*

1.1 Motivation

The parallel data links in communication systems are being rapidly replaced by serial data communication systems such as synchronous optical networks (SONET), back-plane and chip-to-chip interconnects with the representative standards such as Serial ATA, PCI Express. Figure 1 shows a typical multi-gigabit transceiver block diagram. Normally

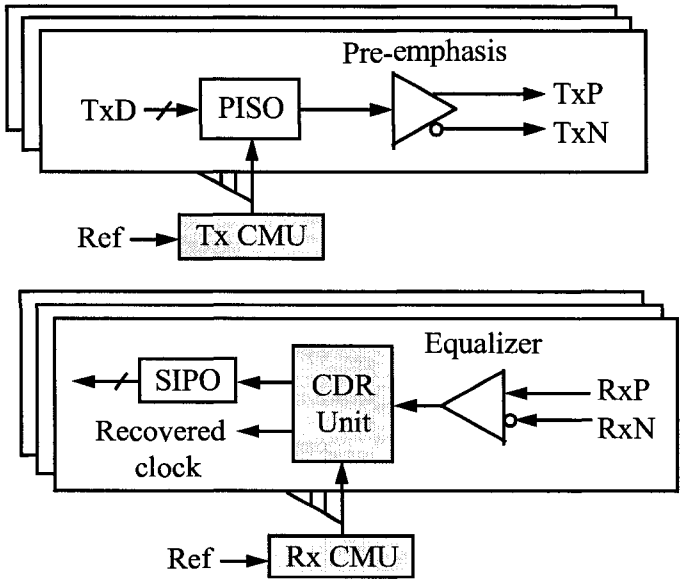


Figure 1.1 Typical Multi-Gigabit Transceiver Block

it consists of a transmitter part which is comprised of a pre-emphasis block, a parallel input to serial output converter (PISO), a transmitter clock multiplier unit (CMU), and a

receiver part consisting of an equalizer, a clock and data recovery circuit (CDR), a receiver CMU and a serial input to parallel output converter (SIPO). Both CMUs can provide clocks for multiple-channel transmitters or receivers, and in some applications, the transmitters and receivers share clocks from the same CMU.

The main functions of each block are briefly described as follows. The parallel data is serialized in the PISO and amplified in the pre-emphasis block. This transmitter pre-emphasis block enables the transceiver to drive long backplane or cables at high data rates. At high baud rates, the signal power degrades and the eye opening closes as the further the signal travels away from the transmitter side due to the dispersive channels that produce dramatic inter symbol interference (ISI). The pre-emphasis block compensates for the signal attenuation from the transmission line at high frequencies by pre-distorting these high-frequency components in advance so that the signal can be recovered at the receiver side. After a relatively long transmission line where the distortion and noise arise from the channel loss, reflections, and the high-frequency crosstalk, the signal is distorted and these impairments would be beyond the ability of the pre-emphasis block alone, so an equalizer at the receiver part is necessary. Normally a decision feedback equalizer (DFE) is used to subtract the ISI from the previously detected data symbols from the symbol that is being received and to recover the severely degraded data. It maximizes the data eye opening to facilitate the data and clock recovery in the CDR block.

Both of the transmitter and the receiver sides need a clock multiplier unit to provide clocks for the incoming data streams with different baud rates. With the reference clock from a low-frequency and low-noise crystal oscillator, the clock multiplier generates a high-frequency output with the frequency an integer times the reference clock. In some cases, the Tx clock and the Rx clock can share the same CMU.

The essential part of the receiver is the CDR block. It extracts clock information from the received data, and the recovered clock samples the incoming data streams to

recover the data and drive the SIPO, where the serial data stream is converted to parallel data to be processed in the DSP followed the SIPO.

As the critical component in the receiver, the CDR circuit has four main figures of merit, the jitter tolerance, the jitter transfer, the jitter generation, and the acquisition time, and they significantly affect the performance of the receiver. To date, the widely used CDRs are analog CDRs, which are mainly phase-locked loop (PLL) based and delay-locked loop (DLL) based CDRs. Those analog circuits are difficult to be ported and require analog charge pumps and filters, which consume a large area and a large power. Due to the limited bandwidth which has to be carefully set to meet the specifications including the jitter tolerance, the jitter transfer and the jitter generation, the acquisition time is usually long, resulting in more preamble data bits, which is undesired in high speed systems.

With the rapid development of the CMOS process technology, evolving from μm technology to *nm* technology, it is the trend for some of the analog circuits including the analog transceiver to be implemented in the low cost digital process, thus they can be easily ported across multiple technologies and targeted speeds with ultra low power.

Accordingly, the all-digital CDR is under exploration. It is considered to be a good choice for high speed links where the jitter generation specifications are more relaxed. The reported all-digital CDRs are mainly synchronous oversampling CDRs and asynchronous oversampling CDRs according to whether the data sampling clock phase is synchronized to the incoming data or not. In the asynchronous sampling CDRs, the post-sampling logic extracts the data information from the data samples by means of majority voting or averaging. These circuits are complex, and they normally have larger power consumption and area than the PLL based CDRs. There are two reported synchronous CDRs [27] [28], which take relatively long time to acquire lock due to the long feedback delay caused by the digitized loop and the updating period respectively, and the power consumption as

well as the area can be further reduced. In this dissertation, a CDR with a digital threshold decision technique is proposed for high jitter tolerance performance, fast acquisition and ultra low power consumption.

In the oversampling digital CDR, multiple-phase sampling clocks are required for sampling and processing data, and a low phase noise frequency multiplier providing a high-frequency signal which can be developed into multiple-phase signals if needed. The phase noise and timing jitter performance are among the critical parameters in the frequency synthesis design. One dominant and mature technique used in the design is the PLL. Due to the phase noise accumulation in the ring oscillator (or the large area of an LC oscillator) in PLLs, frequency multipliers based on the DLL are of interest recently. There are many types of DLL based frequency multipliers reported in literature.

Recent publications show that the DLL based frequency multipliers have some drawbacks including large output spurs caused by the systematic errors and harmonic locking. Two main techniques used in the frequency multiplying in the DLL are the edge combining technique and the cyclic reference injection technique. The former one generates the high frequency output by combining the delayed reference edges from a voltage controlled delay line (VCDL), in which the in-lock error and any mismatch among the delay stages/the edge combining circuits result in spur at the output. The cyclic reference injected frequency multiplier eliminates the noise and spur from the mismatch, while introducing an extra injection error or phase re-alignment error which results in extra spurious power at the output. Another problem that often remains ignored is that due to the nature of the DLL, harmonic locking might happen, which is normally resolved by initializing the control voltage of the VCDL at circuit power-on or by using an extra locking detection circuitry which adds more complexity to the circuits. In this dissertation, a cyclic reference injected DLL frequency multiplier with a period error compensation loop and a new switching logic scheme is proposed to overcome the above limitations.

1.2 Contributions to knowledge

In this dissertation, two main components in a transceiver, the CDR and the DLL based frequency multiplier are reported, and the main contributions to knowledge are summarized as follows,

1. An *N*X oversampling CDR with a digital threshold decision technique was proposed. Ideally, it can achieve the maximum possible jitter tolerance of the oversampling CDR with an oversampling ratio of *N*. Furthermore, the proposed CDR is capable of acquiring lock within one baud period.
2. A 5X oversampling CDR based on the digital threshold decision technique is proposed. The ideal high-frequency jitter tolerance is 0.8UI, which is verified with an event-driven jitter tolerance simulation methodology.
3. A low-complexity circuit implementation of the proposed CDR with the oversampling ratio of 5 in CMOS 90nm was given. The estimated jitter tolerance based on the measured maximum tolerable frequency difference is very close to the theoretical values, with a low power consumption, confirming the proposed digital threshold decision technique and its CMOS implementation.
4. A programmable, cyclic reference injected, DLL based frequency multiplier with a period error compensation loop and a novel switching scheme was proposed, and a Matlab model of an equivalent DLL with the proposed compensation loop verified its efficiency in reducing the in-lock error.
5. Some novel circuit implementation techniques were employed in the proposed DLL frequency multiplier circuit design, especially in the period error compensation loop and the switching logic circuit. The measured results confirm its improved phase noise, output spur and timing jitter performance. By disabling and enabling the compensation loop, a significant improvement in spur and tim-

ing jitter performance was observed, which verified the proposed techniques.

1.3 Document organization

This document is organized as follows. Chapter 2 reviews the most recently reported CDR circuits in literature, where different types of CDR are described with respect to their architecture and operation. Chapter 3 reviews the most recently reported DLL frequency multipliers mainly focused on two reported frequency multiplying techniques which are the edge-combiner based and the cyclic reference injection based.

A low-power, fast acquisition, 5X oversampling CDR with high jitter tolerance performance is proposed in Chapter 4, and the general case, the proposed *N*X oversampling CDR with a digital threshold decision technique is first given, then design parameters are chosen for the 5X oversampling CDR and its functionality is verified in Matlab model simulations. The jitter tolerance is obtained with an event-driven simulation.

In Chapter 5, the in-lock error in the DLL is confirmed to result in spurious power at the output, and a compensation loop in a cyclic reference injected DLL frequency multiplier is proposed to reduce the in-lock error, which is verified in Matlab model simulation with respect to its functionality and efficiency.

Chapter 6 presents the implemented all-digital CDR circuit and the programmable DLL frequency multiplier in detail. The implementation considerations and circuit simulations are also given. In Chapter 7, layout considerations and test setups are first given, followed by the measurement results of both chips with explanation and analysis.

Chapter 8 summarizes the previous chapters and the main contributions to knowledge in this dissertation.

CHAPTER 2

Overview of the Clock and Data Recovery Circuits

2.1 Introduction

In this section, selected CDR circuits as well as their main operation principles reported in the recent years will be reviewed. The CDRs are first classified into three main types, the analog one, the semi-analog one and the all digital ones, and a few of figures of merit in the CDR design are followed. In the analog CDRs, they will be reviewed based on whether the data recovery is data edge tracked or data eye tracked. Then the semi-analog and all-digital CDR circuits are followed. The all-digital CDRs are reviewed according to whether the data sampling clock is synchronized to the data or not. Finally performance comparisons for both the analog CDR and digital CDR circuits are given in Table 2.1 and Table 2.2 respectively in the summary.

2.2 Classification of Clock and Data Recovery Circuits

There are many ways to classify the CDR circuits. Basically it can be classified to analog, semi-analog or semi-digital, and digital types as shown in Figure 2.1. The analog CDR can be implemented with PLL or DLL, and according to the type of phase detector employed, both PLL based CDRs and DLL based CDRs can be classified to binary CDRs if a bang-bang PD such as an Alexander PD is used, multi-level CDRs and linear CDRs if an Hogge phase detector is used.

The semi-analog CDRs normally use multiple clock phases to over-sample the data with a large amount of discrete phases which are comparable to the phase tuning of the analog VCO in the analog CDR.

In the digital CDR, according to whether the sampling clock phases are synchronous to the phase of the data eye center or not, it can be classified to synchronous oversampling CDRs and asynchronous oversampling CDRs. In the synchronous CDRs, at the initial state, the sampling clock phases are not aligned to the data eye center, and the phase error between them is detected and reduced until they are aligned by the feedback loop. The data sampling clock phase which is aligned to the data eye center samples the data as the recovered data. In the asynchronous oversampling CDRs, the data is sampled by the clock phases which are asynchronous to the data eye center and the loop does not try to align them by reducing the phase error, instead it recovers the data by picking one of the sampling clock phases which is most aligned to the data eye center, and recovered data is the data sample that is sampled by the most aligned sampling clock phase. In some reported work, it is called as blind oversampling CDRs [1] by blindly sampling the data, and the digital processing logic followed detects one of the clock phases that is most aligned with the data phase. Obviously, in the asynchronous oversampling CDR, there is also loop tracking behavior. In other words, all the CDR circuits are tracking loops, and the correct clock phase is tracked to recover the data. Detailed overview of each type of

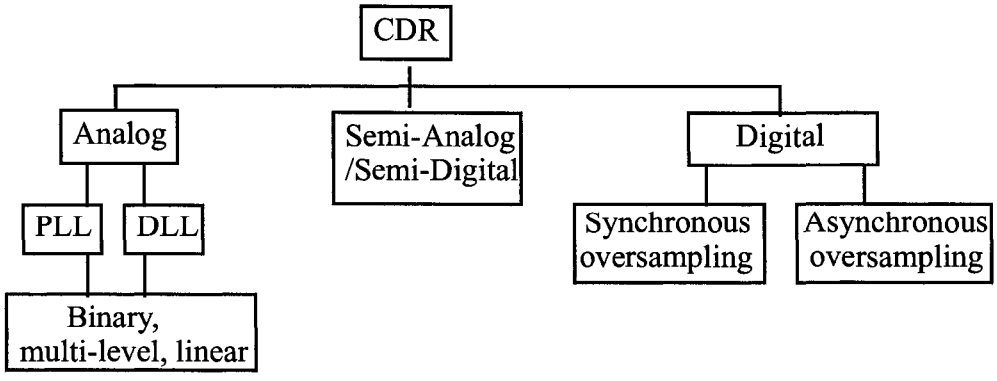


Figure 2.1 Classification of the CDR circuit architecture

CDRs will be given in the following sections.

2.3 CDR Figures of Merit

There are several figures of merit to estimate the performance of the CDRs in the specified applications, which include the jitter tolerance, the jitter transfer characteristics, the jitter generation, and acquisition time. Some other characteristics are common to other circuits such as the power consumption, the operation frequency, the chip area and so on. This section only focuses on the figures of merit that are specific to the CDR circuits.

2.3.1 Jitter Tolerance

The jitter tolerance is the item to measure the capability of a CDR circuit to achieve a specified bit error rate (BER) under the worst-case jitter conditions especially in the presence of incoming jitter from the data. Typically, the jitter tolerance is the maximum sinusoid jitter amplitude which is usually the peak-to-peak phase amplitude modulated by the jitter. The jitter tolerance is normally given as a function of the worst-case jitter amplitude over the corresponding jitter frequencies. The principle and characteristics are illustrated in Figure 2.3. From the figure, the corner frequency is at jitter frequency of f_i MHz. The flat line beyond this point is the jitter tolerance at high jitter frequencies, where the CDR loop can not track and the jitter amplitude is A unit interval (UI). At the jitter frequencies lower than f_i MHz, the loop can track the jitter, and due to the loop filter characteristics, the slope is -20dB/decade. A jitter tolerance which is beyond the required jitter tolerance mask is acceptable. For different applications, the jitter tolerance mask varies.

The jitter tolerance at low jitter frequencies is dependent on how fast the CDR circuits can update its decision, while the JT at high jitter frequencies usually can be improved by averaging the updated decision results over a certain number of the data periods. The longer the averaging period is, the higher the JT at high jitter frequencies is,

which would definitely degrade the JT at the low jitter frequencies or the jitter frequencies lower than the CDR bandwidth. Besides, the oversampling ratio is also a critical parameter determining the jitter tolerance performance at high jitter frequencies in the oversampling CDRs.

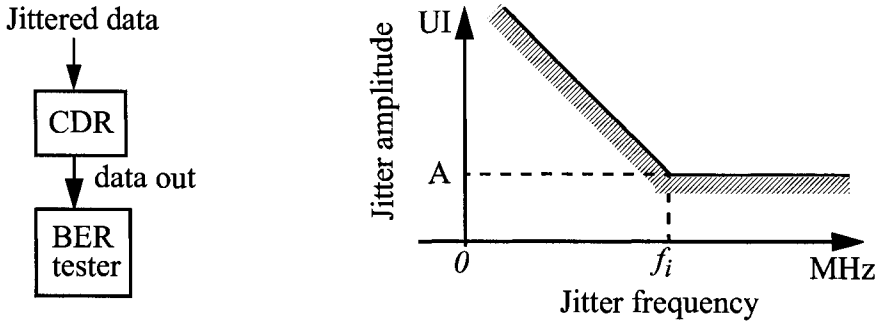


Figure 2.2 Jitter tolerance of CDRs

2.3.2 Jitter Transfer

Except for the jitter tolerance, the other specification of the frequency response of the CDR circuits is the jitter transfer. It is the ratio of the output jitter from the input jitter over jitter frequencies, mainly describing the gain of the CDR to attenuate or amplify the input jitter. For example, a negative dB gain means the loop removed the jitter, and zero dB gain indicates the loop has no effect on the jitter. The jitter transfer specification limits how much jitter may be passed to the new or the repeated data stream. The example jitter transfer characteristics is shown in Figure 2.3. The flat part with the value of G is the DC gain of the CDRs, and the cutoff frequency f_c is the CDR bandwidth. The roll-off beyond the cutoff frequency is also 20dB/decade. For a given required specifications, the values beyond the mask is not allowed. Again, the jitter transfer specification varies over different standards.

Generally speaking, both the jitter tolerance and the jitter transfer are related to the CDR bandwidth. Increasing the bandwidth can speed up the loop tracking behavior and can increase the jitter tolerance, while limiting the bandwidth can reduce more high fre-

quency jitter components. To meet both of the requirements, careful design is needed for the loop bandwidth to satisfy both specifications.

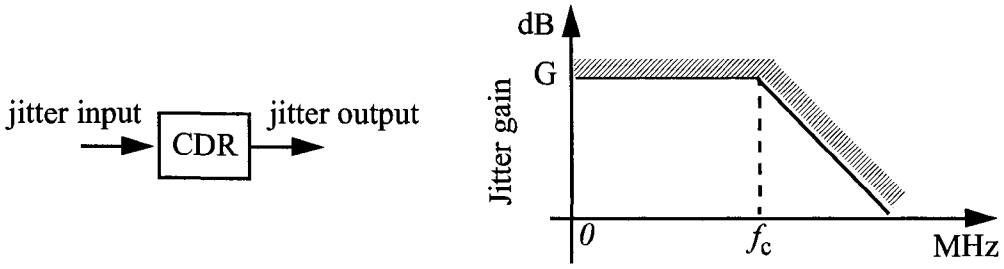


Figure 2.3 Jitter transfer of CDRs

2.3.3 Jitter Generation

The jitter generation is defined for the recovered clock and the recovered data. It is the amount of jitter added to the data signal by the CDR circuit with the input data stream from a clean pseudo-random bit stream (PRBS) patterns. The PRBS patterns are generated from a linear feedback shift register (LFSR) with the length of 7, 23 or 31. Longer length of LFSR often proves harder to maintain low jitter, so to ensure the CDR performance, LFSR with length of 31 can be used, however, in most applications, the jitter generation specifications are usually limited to length of 7 and 23 PRBS data patterns.

Various jitter transfer specifications are needed for different applications. For example, Telcordia specifies limits of 0.1UI peak-to-peak and 0.01UI rms (root mean square), and in the OC 192 SONET specification where the data rate is 10Gb/s, the peak-to-peak jitter is less than 100mUI or less than 10 ps, and the rms jitter is less than 10mUI or less than 1ps [2]. Both the recovered clock and recovered data must meet the above specifications. The peak-to-peak jitter is measured as the width of the crossing of the rise and fall signals on the eye diagram and the RMS jitter measures the distribution of the crossing pattern.

The main jitter sources generated in an analog CDR include the phase noise from the VCO and the phase detector and charge pump combination, the ripple on the control

line, and noise from the power supply and the substrate. In the semi-analog CDRs with the VCO, except for the noise similar in analog CDRs, there is additional noise from discrete phase steps in the recovered clock.

The CDR bandwidth is a critical parameter with respect to the jitter tolerance, jitter transfer and the jitter generation of the CDR circuits. A low bandwidth helps to achieve better jitter transfer and jitter generation, while high bandwidths increase the jitter tolerance performance at low jitter frequencies. Design trade-off must be considered in balancing the bandwidth and the three specifications.

In high speed links, CDRs used in the chip-to-chip communication usually do not need to meet a jitter transfer specification, instead they mainly target at a certain low bit error rate. The jitter generation is also more relaxed in high speed links, which makes the digital CDR or the semi-analog CDR a good choice for those applications.

2.3.4 Acquisition Time

In high-speed CDR circuits, fast acquisition is required for networks with fast switching between nodes, and short acquisition time reduces the number of preamble bits to achieve high efficiency.

In the analog CDR circuits using a linear PLL, the acquisition time usually refers to the PLL acquisition time. It is the time which the PLL takes to achieve lock with a given initial frequency and phase error without cycle slips [5]. Figure 2.4 shows an exam-

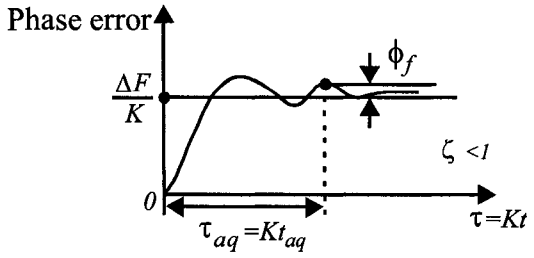


Figure 2.4 Acquisition time in PLL CDR

ple of the acquisition time of the PLL based CDR circuits, where ΔF is the initial frequency error (the phase error is assumed to be zero), K is the open-loop gain, ϕ_f is the maximum peak-to-peak oscillation amplitude after the loop acquires lock. The explicit acquisition time is the time that the loop takes to reduce the initial frequency error to the value which is expressed as

$$\phi(\tau_{aq}) = \frac{\Delta F}{K} \pm |\phi_f|$$

In all-digital CDR circuits, the acquisition time usually is the time the loop takes to correctly make its decisions on incoming data at the initial phase error. Fast acquisition time normally means short preamble data bits, and there is no such explicit expression for the acquisition time due to the nature of the non-linear system except for the all-digital PLL based CDRs which is similar to the analog PLL based ones.

2.4 PLL Based analog CDR Circuits

The conventional PLL based CDR circuit is shown in Figure 2.5. It consists of a

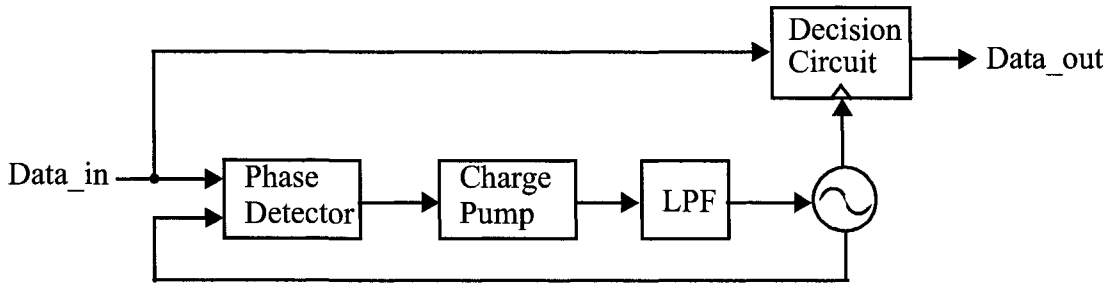


Figure 2.5 Conventional clock and data recovery circuit

phase detector (PD), a charge pump (CP), a low-pass filter (LPF), a VCO and a decision circuit. The essential part of the loop is a PLL. The PD detects the phase error that is proportional to the phase difference between the data edges and the clock from the VCO which is integrated and low-pass filtered to tune the VCO frequency and phase that are compared with those of the data in the PD. After the loop is locked, the recovered clock

samples the data in the decision circuit to output the recovered data.

Usually the PD, especially the bang-bang type PD, has three sampling clock phases, among which there are two data edge sampling clocks and one data eye sampling phase. The two data edge sampling clock phases detect the data transition, and the data eye sampling clock phase samples the data 0.5UI after the data edge detected, which is the fixed interval oversampling of the three clock phases. To overcome this drawback, instead of tracking the data edge, an eye tracking method is reported in the literature. Based on the two types of tracking, the analog PLL CDR can be categorized to data edge tracking CDR and data eye tracking CDR to facilitate the overview process. The main advantages of employing a PLL in the CDRs are that its ease to be implemented as a monolithic IC, and its dynamic behavior can be easily adjusted by choosing the loop filter parameters. In addition, its relative small bandwidth can increase the recovered clock jitter performance.

2.4.1 Full-rate data edge tracking CDR circuits

Figure 2.6 shows the basic concept of data edge tracking CDR operation in the

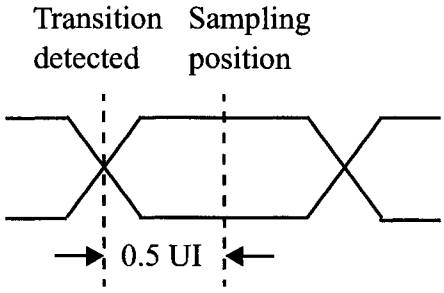


Figure 2.6 Data edge tracking

example of an ideal linear or binary CDR. The CDR loop tracks the data edges or transitions, and after the data transitions are detected, the data is recovered 0.5 UI after. Most of the analog CDRs utilize this data edge tracking technique, and based on the sampling clock rate, they can be classified as full-rate CDRs, and non-full-rate CDRs such as the half-rate CDRs and the 1/8 rate CDRs.

The full-rate PLL based CDR has of a simple structure and reliable operation with input data of various data patterns [6] [7] [8] [9] [10] [11]. Compared with non-full-rate CDRs such as, half-rate, 1/4 rate CDRs, or even 1/8 rate CDRs, full rate CDRs require higher clock rate, while the non-full-rate CDRs need multiple clock phases which require precise matching to avoid clock jitter degradation.

Figure 2.7 shows a full-rate CDR circuit reported in [11]. The CDR employs a

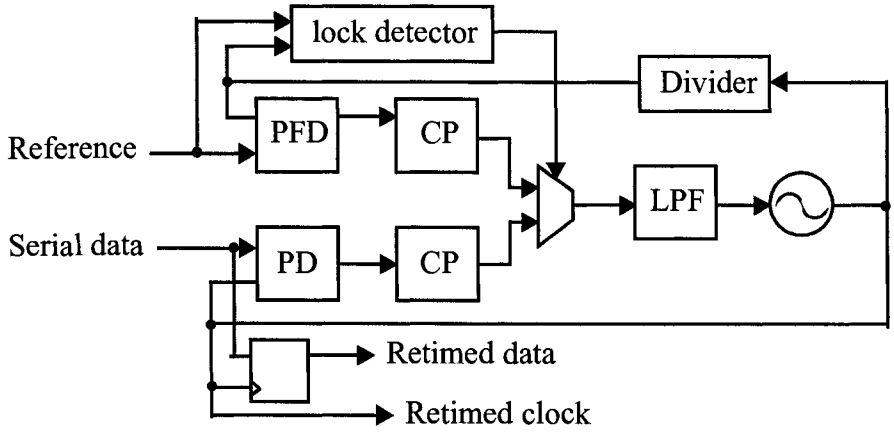


Figure 2.7 Conventional clock and data recovery circuit

dual-loop structure including a phase acquisition loop and a frequency acquisition loop. The frequency locked loop takes control of the CDR when the clock frequency of the LC VCO is off the reference by a certain value to aid the CDR with frequency lock, and it is disengaged after the loop is locked to the reference frequency. This helps to achieve a large frequency acquisition range. The lock detector controls the switching between the phase locked loop and the frequency locked loop. The LC VCO output is fed to the PD after it is divided down by the divider, and its phase is compared with the phase of the data edge to acquire phase lock in the phase acquisition loop. Finally a DFF re-times the serial input data with the extracted clock from the VCO output.

The main advantages of this architecture are its simple, straightforward implementation, the reduced high frequency jitter due to the relaxed loop bandwidth resulting from the separate regeneration of data and high operation frequency. However, there are

some disadvantages. For example, the high-speed LC VCO consumes large power and chip area, and the requirement of an external reference and the frequency locked loop adds more hardware cost. In the PLL based CDR, a small noise bandwidth and large pull-in range cannot be satisfied simultaneously by employing the PLL. Thus an acquisition aid is indispensable.

The frequency locked loop helping the large pull-in range without reducing the noise bandwidth in the previous example can be further reduced to a dual-path loop structure with less circuit complexity as reported in [6] [7] [8]. Figure 2.8 gives the use of

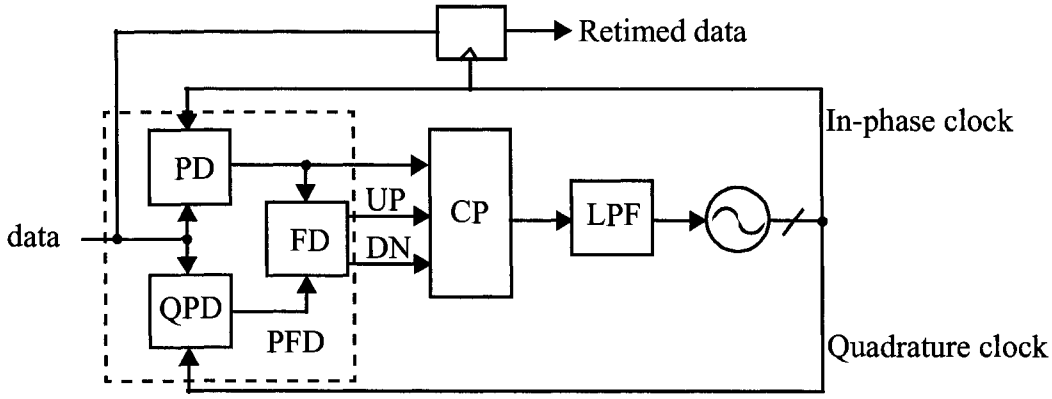


Figure 2.8 The CDR circuit with a binary quadrature PFD

binary PFD based full-rate CDR architecture [6]. The binary PFD consists of a PD, a quadrature PD(QPD), a frequency detector. Driven by an in-phase clock and quadrature phase clock respectively, the PD and the QPD generate two phase difference signals at each data transition by sampling the VCO output and its delayed output. When the bit rate is lower than the VCO output, the PD output leads the QPD output, and when the bit rate is faster than the VCO output frequency, the beat note of the PD lags the one of the QPD. The two error signals are compared in the FD to generate two outputs to charge up or down the currents in the charge pump during the frequency acquisition. After the frequency locking is acquired, the phase error is tuned by the PD outputs and the two outputs of the FD are both high. The main advantages of this CDR architecture in [6] are no refer-

ence frequency required in frequency acquisition, the dual path structure and the fully differential structures in the CDR blocks.

2.4.2 Non-full-rate edge tracking analog CDR

The bottleneck of high-speed CDRs is the VCO frequency in the CDR design. To increase the operation speed, the VCO frequency can be reduced by sampling the data at half of the data rate, and it can be further reduced by sampling the data at 1/4 data rate or even 1/8 data rate[17][18][19]. So the VCO can provide sufficient tuning range and tolerate the process variation, voltage, and temperature (PVT) effectively, helping suppressing the substrate noise [20] and increasing the data rate. However, more sampling clock phases often result in more complex circuitry and larger power consumption, and the mismatch among the delayed multiple phases normally causes large clock jitter.

The review on non-full-rate analog CDRs mainly focuses on half-rate CDRs. Half-rate CDRs sample and process the incoming data by using both the rising and the falling clock edges. This also helps reducing the power consumption with the VCO running at half of the data rate. Half-rate CDRs can be implemented in the similar way as the full-rate dual loop CDRs, however, the half-rate PD is among those mature techniques, while the half-rate FD is still under exploration. Consequently, most of the reported half rate CDR circuits focuses on the frequency detector design [13] [14] [15]. Figure 2.9 shows one example reported recently [13]. The circuit employs a half-rate Hogge PD to extract the phase information, and a shifted averaging VCO to mitigate the phase errors among the four clock phases caused by the mismatch among delay stages to increase the clock phases accuracy. The re-timed clock and recovered data are from the phase detector.

The linear PD produces an error pulse with the width equal to the phase difference between the data and the clock whenever there is a data transition, and a reference signal is thus produced at each data transition with a pulse width half of the clock period. When the clock transition is in the middle of the data eye, by scaling up the error signal by 2, the dif-

ference between the average error and the average reference signal will drop to zero. Consequently, the phase error with respect to this point is linearly proportional to the difference. In other words, ideally the data is sampled 0.5UI after the data transition is detected. The half-rate FD achieves frequency acquisition by detecting the state transition changes. Table 2.1 gives the measured performance for comparison.

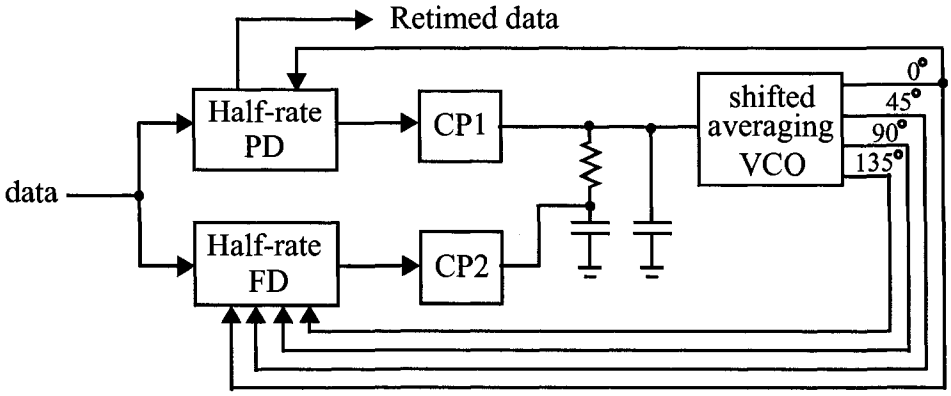


Figure 2.9 A half-rate CDR architecture with dual loops

2.4.3 Data eye tracking CDR circuits

In the edge-tracking CDR as described in the previous sections, theoretically the CDR recovers the data 0.5UI after the data transition is detected. In case of severe jitter environment where the data eye opening portion is small and asymmetrical, the CDR based on the data edge tracking method is difficult to maintain the optimum BER performance. Instead of tracking the data transitions, by tracking the data eye, the CDR performance can be improved. An example is shown in Figure 2.10. *Lclk* and *Rclk* are the data

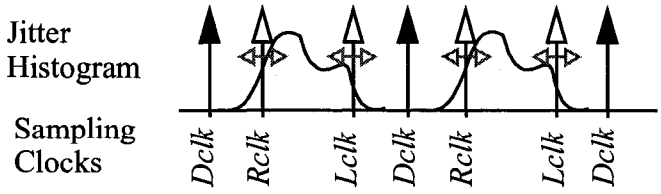


Figure 2.10 Data eye tracking

edge sampling clocks adjusted to track data transitions with variable interval, and signal *Dclk* is the data sampling clock that is placed in the middle of the two edge sampling

clocks. So in spite of the asymmetric data eye opening, *Dclk* can always track the data eye center.

One example CDR architecture using this data eye tracking technique is shown in Figure 2.11 [21]. It comprises of one reference loop and one data loop. The reference loop is used to lock the sampling clock frequency to that of a local reference clock, then the data loop adjusts the clock to track the data. The data loop consists of two loops, one

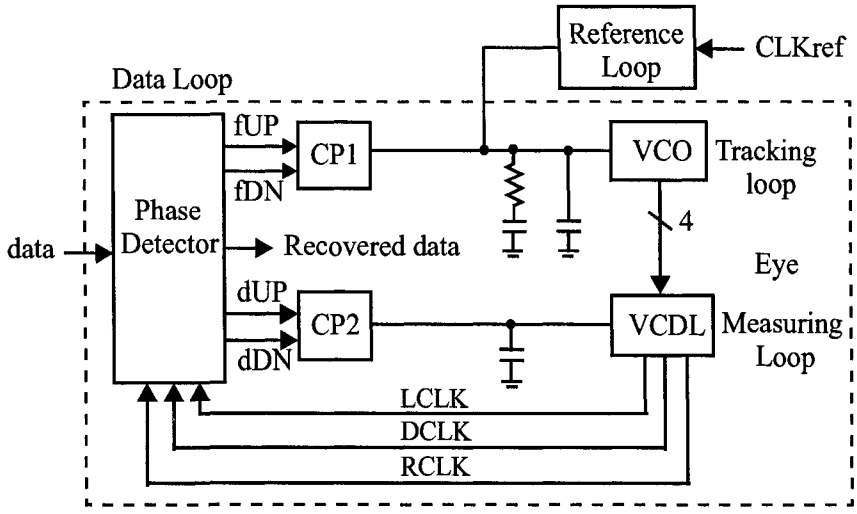


Figure 2.11 Data eye tracking based CDR architecture

tracking loop and one eye measuring loop. The PD compares the phases of the edge sampling clocks with that of *DCLK*, and the error signals *fUP* and *fDN* are generated for the tracking loop to adjust the VCO frequency. The phase detector also outputs another two signals *dUP* and *dDN* for the eye measuring loop to adjust the VCDL delay of the VCDL, where multiple phase clocks are generated to sample the data. For stable operation, the bandwidth of the tracking loop can be far larger than that of the eye measuring loop, thus the phase of the data sampling clock *DCLK* is adjusted by the tracking loop and will not be affected by the operation of the eye measuring loop.

The loop acquisition can be illustrated in Figure 2.12. After the VCO frequency is locked to the local frequency, the circuit is switched to data acquisition mode, and the procedure can be divided into three steps as shown in the figure, where the arrows show the

movement direction of the edge and data sampling clocks. Phase (a) shows the acquisition of the tracking loop. The VCO frequency is tuned by the control voltage from the charge pump to the direction to reduce the phase error between the phases of the recovered data and the VCO output, so all the sampling clocks are moved in one direction so the data sampling clock better approaches the data eye center. Phase (b) shows the acquisition process of the eye measuring loop. The VCDL delay is tuned to reduce the time interval between the two sampling clocks according to the data eye width. As *LCLK* and *RCLK* approach the data edge, *DCLK* is finally located in the data eye center as shown in phase (c). At this point, both the tracking loop and the eye measuring loop are in lock.

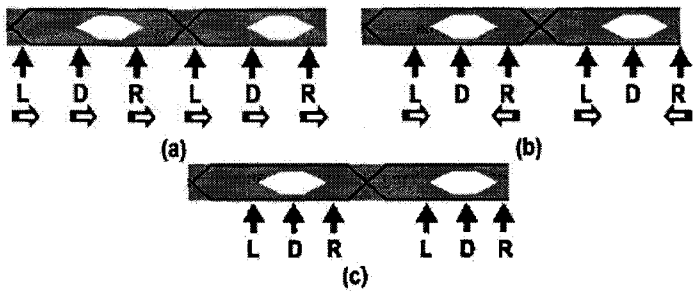


Figure 2.12 Loop operation: (a) phase 1 (b) phase 2 (c) phase 3

2.5 Semi-Analog CDR circuits

The previous sections reviewed the analog CDR circuits based on the PLL, which consists of analog components in the circuits. In this section, the typical semi-analog CDR circuits will be reviewed. Due to the use of digital filters and phase selection from multiple clock phases to recover the clock, these CDR circuits are named as semi-analog CDRs.

Figure 2.13 shows the diagram of a semi-analog CDR architecture [23]. It combines the DLL and the PLL characteristics by using both phase selection technique and a PLL. There are two loops in the circuit. Loop A is the phase selection loop, mainly consisting of a DLL, which can be locked to the data within a few of clock cycles. The phase detector in loop A compares the data with the clock phase selected from the multiple clock

phases, and the phase error signal is applied to a digital filter to generate a selecting signal that enables a correct phase selection. The loop B is essentially a PLL, which guarantees the correct VCO frequency. With a low bandwidth, low jitter is obtained after the loop is in lock. Fast acquisition is obtained by feeding back the sign and amplitude of the phase error from loop A and loop B respectively to the multi-phase VCO. The drawback of this architecture is that the recovered clock phase is chosen from several clock phases, so finite phase step error is resulted in the recovered clock, while in the conventional VCO based PLL CDR, the clock phase error is continuously compensated, resulting in less timing jitter. To reduce the timing step, more than 100 clock phase steps can be employed to mimic the analog PLL CDR.

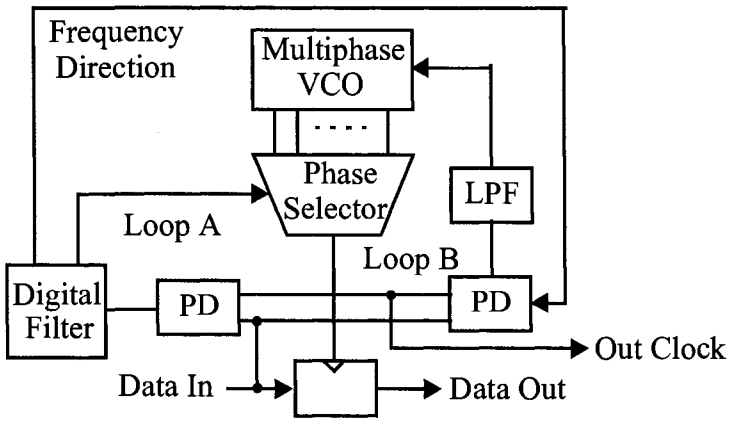


Figure 2.13 A semi-analog CDR circuit with combined the PLL and DLL

To overcome the discrete phase error in the recovered clock, a CDR circuit with a large number of clock phases to remedy the jitter caused by the discrete phase step is presented in [24] as shown in Figure 2.14. The phase difference between the data eye center and clock *Ck* is detected by loop B, then the control signal at the output of the digital filter is changed and a new clock phase is selected which is divided by *N*. A phase difference between *fref* and the new clock phase is detected by the PFD and converted to the control voltage by the CP and the filter in loop A, so the phase step caused by the finite number of phases are smoothed by loop A. Consequently the phase of the recovered clock *Ck* is

slowly tuned to the phase of the data eye center, and the timing jitter is spread out over many clock cycles. This averaging effect is achieved by employing an average phase

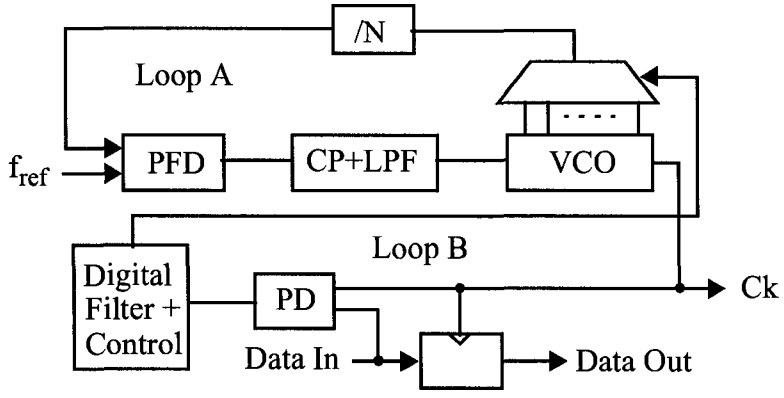


Figure 2.14 Semi-analog CDR with a large number of clock phases

interpolation. With both the phase interpolation and the phase selection technique, as many as 128 clock phases are generated, and the large number of selectable clock phases makes the recovered clock cleaner. However, the acquisition time is larger than the conventional semi-analog CDRs due to the clock recovery involvement of an extra loop. The main measured performance is also given in Table 2.1 for comparison.

2.6 Digital CDR

In this section, two reported types of digital CDRs will be reviewed. According to whether the data sampling clock phase and the data are in phase or not, it is categorized into asynchronous oversampling CDRs (it is also called as blind oversampling CDRs in some reported work) and synchronous oversampling CDRs. In the asynchronous oversampling CDR, the phases of the sampling clocks are not adjusted to the track the data eye center, instead from the sampling clocks, the most aligned clock phase is picked up to recover the data. The following paragraph gives more detailed review and analysis of each type of oversampling CDRs.

2.6.1 Asynchronous Oversampling CDR

The asynchronous oversampling CDR shown in Figure 2.15 is a feed-forward system using phase-picking technique instead of a feed-back one as in the analog or the semi-

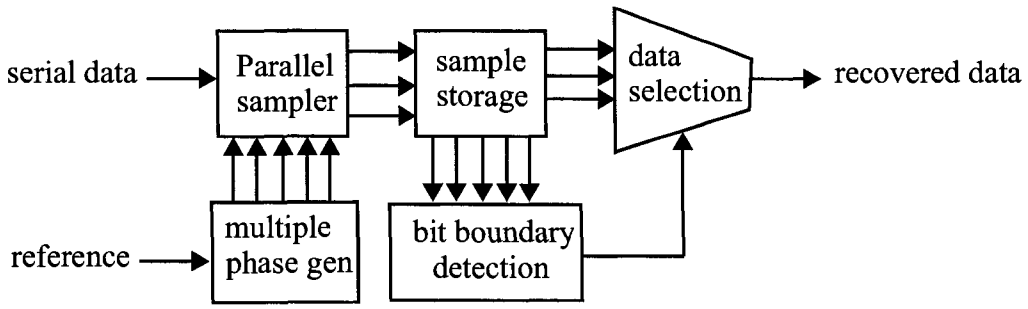


Figure 2.15 Typical block diagram of an asynchronous oversampling CDR

analog CDRs. The data is first sampled by the parallel samplers by multiple phase clocks, and the data samples are applied to the phase detection logic consisting of sample storage, bit boundary detection and data selection to select the most reliable data samples. The oversampling rate is equal to the number of data samples within one baud period. The bit boundary is detected according to the data samples within one decision window which is a number of data samples and the data transition information is extracted based on those samples. Then according to the data transition information, the most reliable data samples which are farthest from the bit boundary are selected by the data selection logic.

Due to the feed forward characteristics, there is no intrinsic loop bandwidth constraints, and the tracking speed depends on how fast the phase selection decision can be updated, which is relatively faster than the PLL based analog CDRs. However, the noise performance is impacted by the maximum static phase error from quantization determined by the number of sampling phases. Although this quantization error causes an SNR penalty, there is far less jitter generated within the clock and data recovery process than the analog CDR does due to the noise-immune digital processing units such as the phase detection logic after the data is sampled.

The power consumption due to the logic following the samplers is higher than the analog CDRs, and increasing the oversampling rate will reduce the static phase error but also will reduce the receiver bandwidth. So the typical oversampling ratio is limited to 3 or 5.

Figure 2.16 shows one reported asynchronous 3X oversampling CDR [25]. First

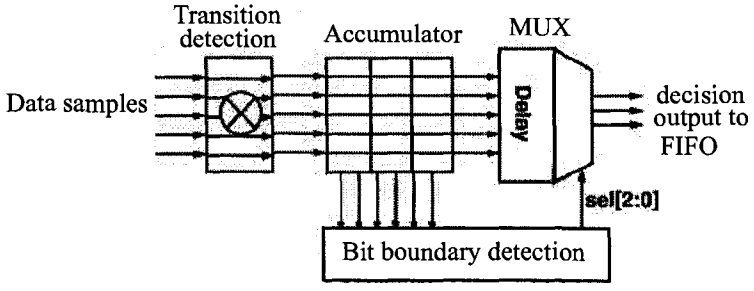


Figure 2.16 Asynchronous CDR with center phase picking technique

the decision logic detects the transitions with XOR gates and the transition could be one of three possible positions. The transitions that happen at the same bit boundary positions are tallied and the number of the tallies are accumulated in the accumulator, and the bit boundary is determined by the largest number of tallies. Due to the high frequency noise which is near the baud rate, enough transitions should be examined to average the possible bit-to-bit variations. In this reported work, the tally is counted from a sliding decision window of 3 bytes, i.e. the transitions are accumulated from the current byte, the previous one, and the next one. When two transition positions have equal number of tallies, either of the middle samples will give the same performance, so arbitration is needed in this case. The middle samples are selected to be the recovered data once the bit boundary is detected.

Another example asynchronous CDR is shown in Figure 2.17 [26]. By averaging a large number of the edge phase information from the data samples, the circuit detects the data phase. In other words, this phase averaging operation is like a repetitive self-convolution of the jitter probability density function (PDF) which suppresses the incorrect phase selection probabilities and increase the correct phase detection probability. Accordingly,

the larger number of samples taken by the data samplers, the better performance the circuit could achieve. This phase averaging function mimics a PLL based analog CDR by tracking the low frequency jitter while filtering out the high frequency jitter.

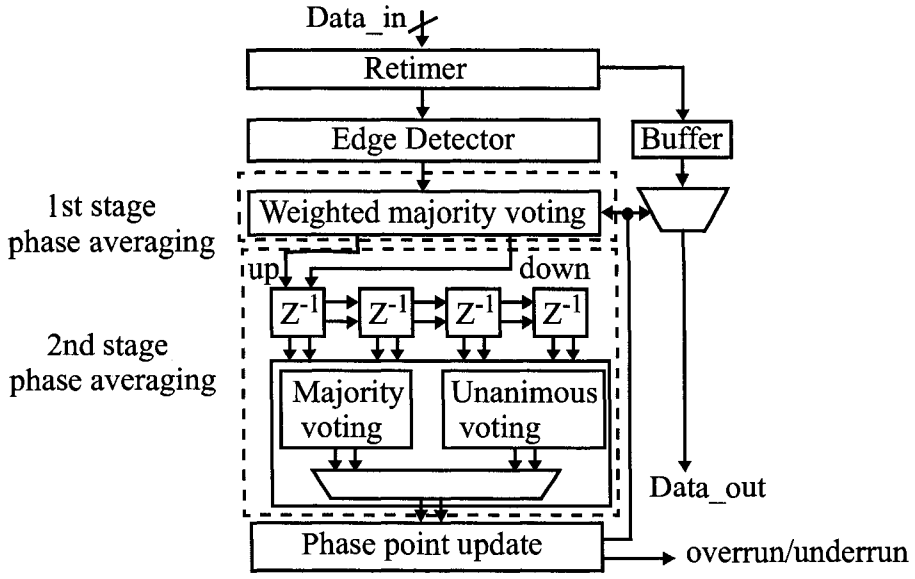


Figure 2.17 Asynchronous CDR with majority voting technique

The functional block diagram explains the detailed operation principle. The retimed data is applied to the data path where it is buffered and the phase information is extracted in the control path. The phase averaging is performed in the control path in two sequential stages, the weighted majority voting and unanimous voting which both function as low-pass filters. The phase information is first calculated and compared with the previous phase information to generate *up* or *down* signals that are stored first and then used in the second phase averaging stage. The first-stage averaging filters out most of the high-frequency jitters, and the unanimous voting reduces the data phase error more effectively than the majority voting, so more accurate phase detection can be achieved with less phase samples and hence less hardware.

Timing is of great importance in a digital circuit. In this functional implementation, a forward path is added in the unanimous voting stage, and it passes the latest block

phase information directly to the phase pointer, which makes the latency of 1 cycle, and this latency in the control path can be matched by simply adding a buffer with the same latency in the data path. Otherwise, the phase averaging logic including the two averaging stages inserts great latency into the control path, and the phase pointer lags the data phase, which might cause instability at the intermediate frequencies in the loop.

2.6.2 Synchronous Oversampling CDR

The previous section reviews asynchronous oversampling CDR with edge tracking methodology to recover the data, and the main drawbacks include that the circuit area and power consumption are large, and not easy to be integrated in multi-channel transceiver on a single chip. To reduce the area and power, recently a synchronous oversampling CDR with data eye tracking technique to recover the data is of interest.

The main circuit block diagram is shown in Figure 2.18 [28]. It mimics the PLL based CDR in a digital way by shifting the clock phase step by step in a cyclic shift type machine. It consists of 4 blocks, the phase comparator, the Up/Down decision circuit, the clock phase pointer and a clock phase interpolator. The phase detector is a bang-bang type, and it generates *UP* or *DOWN* signal when the edge of the data leads or lags the one of the recovered clock respectively. The incoming data is delayed twice to generate a window width with the value of $+TW$ or $-TW$ to detect the data transitions. The up-down decision circuit functions as a low-pass filter. When either *UP* or *DOWN* is collected, *INC* or *DEC* signal is generated respectively, and when both *UP* and *DOWN* are collected, neither *INC* nor *DEC* is outputted, so the selected phase that recovers the data does not change. The third block is a cyclic clock phase pointer. On receiving an *INC* signal, the count is increased to delay the clock phase to recover the data, and on arriving of a *DEC* signal, it decreases the count to advance the clock phase for proper data recovery. Its cyclic structure makes the phase shifting exceed one unit interval for excessive data wander, and it is clocked by a divided clock with the division ratio of $2/ND$, where *ND* defines the decision

updating period. In this way, the total power consumption is reduced. The last block is the clock interpolator and selector, where multiple phases (*NC* phases) are generated, and two of them (*CLK_P* and *CLK_N*) are selected as the realigned clock to pick up the data sample. The recovered data is the output of the second DFF in the PD. This simple structure

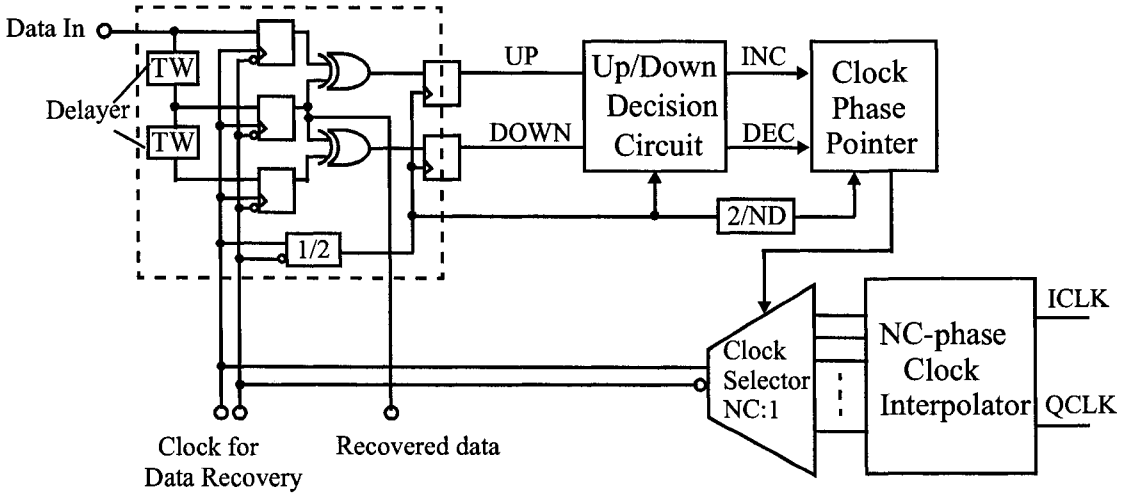


Figure 2.18 Circuit blocks of the digital CDR

reduces the power consumption, and by tracking the data eye, the jitter tolerance performance is high. However, there is still analog parts in the phase detector, the delay elements which consume additional power, and the fixed delay value cannot accommodate various data rates, and the acquisition time as well as the power can be further reduced.

Another reported synchronous CDR is based on the digital PLL as shown in Figure 2.19. It mainly replaces each analog component with a digital equivalent. The main blocks include a bang-bang phase detector, a decimation block, a phase and frequency integrator, and a digital to phase converter.

The phase detector is also a bang-bang type, and its input is the sampling clock phase *phn*, the data sampled by the previous sampling phase, d_{n-1}, and the data sampled after the sampling phase, d_n. The word width of the PD is *W*, each of which produces a 2-bit output. The PD output is decimated in the decimation block, which is used to reduce the baud rate to a rate compatible with high-resolution digital signal processing. By deci-

mation, the circuit operates at lower frequency, so the power and the chip area can be reduced. A finite impulse response (FIR) boxcar filter was used for this decimation operation, and the *w* de-serialized 2-bit phase error samples are added to generate a multi-bit output per clock cycle, and for faster implementations, the decimation is performed by voting across *w/2* of the phase error samples at the cost of latency and increasing the input related noise.

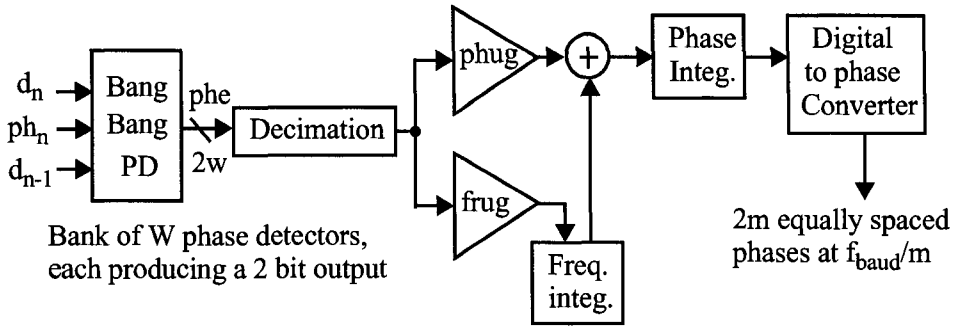


Figure 2.19 Digital PLL used in the digital CDR

The output of the decimation is integrated by the phase update gain (*phug*) and the frequency update gain (*frug*), which models the proportional path gain in the analog charge pump and loop filter (CPLF) combination and models the integral path gain in the CPLF respectively. By summing the output of the proportional path and the integral path, an equivalent control voltage is generated from the phase integrator to tune the digital to phase convertor with the gain that models the gain of an analog VCO. The *2m* equally spaced phases at f_{baud}/m are tuned by the digital output of the phase integrator to recover the data.

One of the advantages of the digital PLL based CDR is that it is easy to analyze the circuit with the linear transfer function, thus easy to choose the design parameters such as *phug* and *frug*. The power consumption is relative high though no detailed results were given in the report due to the large digital process including the decimation block. The acquisition time is also affected by the loop response time and thus is relatively long.

2.7 Summary

In this section, several types of CDR circuits with different clock and data recovery techniques were reviewed, and their main operation principles were given. To better compare their performances, Table 2.1 gives the comparison between the selected analog and semi-analog CDRs reported in the recent years. From the table, the analog CDRs exhibit better jitter on the recovered clock while larger area and power consumption than the semi-analog CDRs. Theoretically the analog and PLL based semi-analog CDR have similar jitter tolerance, which can be observed from the measurements listed.

TABLE 2.1: Performance comparison of selected CMOS analog CDR circuits

Items	[6]	[13]	[14]	[21]	[22]	[24]
Technology	0.35 μm	0.18 μm	0.18 μm	0.25 μm	0.15 μm	0.25 μm
Supply (V)	3.3	1.8	1.8	2.5	1.2	0.9-2.5
Power (mW)	200	83(wo buffer)	91(wo buffer)	153 m@5 G	60 m	18 @250 M
f(Hz)	622-933 M	3.125G	10G	5G	2.5-3.125G	2-1600 M
Loop bandwidth	450 kHz	2 M	5.2 M	N/A	1 MHz	2 MHz
Jitter tolerance	<=0.5 UI	meet specs	N/A	0.605UI	0.5UI	N/A
Recoverd clk Jitters (p-p)	12.5 ps @933	16 ps	9.9 ps	54 ps @5 Gbps	30 ps @3.12 5G	68 ps @1.6 GHz
Area (mm2)	0.84 x 0.7	0.6x0.8	1.75x1.55	0.7x0.5	0.125x1.05	0.036

Table 2.2 compares the performance of recent digital CDR publications. The asynchronous sampling CDR [25] has higher power consumption and larger silicon area than the synchronous one [28] due to complex hardware, and the jitter tolerance are similar. It can also be observed that in the asynchronous CDRs, the larger the averaging window, the better the jitter tolerance, however the acquisition time is sacrificed. Due to the massive digital process logic, the power consumption is huge for the asynchronous CDRs and possible including the digital PLL (DPLL) based one. The synchronous CDR with phase rotator has the best jitter tolerance, and smallest power consumption, chip area, and acquisition time, however, its power consumption can be further reduced and jitter perfor-

mance can be improved at a higher data rate, and the acquisition can be further reduced to meet more stringent high-speed systems requirements with a novel decision technique and architecture which will be detailed in followed chapter.

TABLE 2.2: Performance comparison of selected CMOS digital CDR circuits

Items	[25]	[26]	[27]	[28]
Technology	0.5 μm	0.13 μm	0.13 μm	0.18 μm
Supply (V)	2.7-4.0	1.2	1.2	1.8
Power (mW)	~195	50 per channel	moderate (N/A)	50
Baud rate (bps)	4 G	3.125 G	5 G	2.5 G
Jitter tolerance	N/A	0.67UI	<=0.5UI	0.7UI
BER	$<10^{-9}$ with Δf= 1 MHz	$<10^{-12}$	10^{-9}	N/A
Acquisition time	24 data bits	40 or 80 data bits	comparable to DPLL	16 data bits
Area (mm^2)	3 x 3	1.5 x 1.3	N/A	0.02

CHAPTER 3

Overview of DLL Frequency Multipliers

3.1 Introduction

The frequency multiplier is one important building block in the transceiver. It provides clocks for both the transmitter and the receiver, and its performance, especially the phase noise and timing jitter performance greatly affects the speed and the sensitivity of the transceiver. The frequency multiplier normally takes a low-frequency signal from a clean source to synthesize a high-frequency clock signal for the bit stream, and any phase noise from the process of synthesizing directly contributes phase noise to the data eye. Figure 3.1 shows the timing components of the data eye diagram, where t_u is the peak-to-peak timing noise, and t_r is the rising time, or the time required to switch stage. t_a is the receiver aperture time or the time duration that the signal must be above the voltage mar-

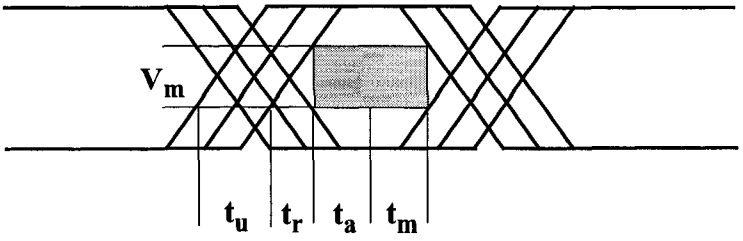


Figure 3.1 Timing components of an eye diagram

gin V_m. t_m is the timing margin of the system. The bit period must be at least the sum of t_u, t_r and t_a, where t_u is the total timing noise in the system. One major contributor of t_u is the

frequency multiplier in the system.

So the phase noise and timing jitter performance of a frequency multiplier are critical parameters in frequency synthesis design. One dominant and mature technique used in the design is phase locked loop (PLL), however due to the phase noise accumulation in the ring oscillator, or the large area an LC oscillator may consume, recently many frequency multipliers based on delay locked loop (DLL) are reported. Compared with its counterpart, the PLL based frequency multiplier, the DLL is a first order system, so it is unconditionally stable. With a capacitor as the loop filter in the DLL based frequency multiplier, it is easier than PLL based ones to be integrated. The main drawbacks include that it is difficult to achieve multiple frequencies, and the locking range is limited. Besides, unlike the PLL which functions as a low-pass filter by suppressing the reference phase noise, the jitter performance of DLL based ones also depends on the reference. However, this drawback can be avoided by deriving the frequency output from a clean reference. Due to the advantages over the PLL based ones, the DLL based frequency multipliers are under exploration in the recent years with efforts to overcome the drawbacks.

3.2 Types of reported DLL frequency multipliers

The reported publications in literature can be categorized into two types basically, the edge combining circuit based DLL frequency multipliers and the cyclic reference injection based frequency multipliers as shown in Figure 3.2. Among the edge combiner

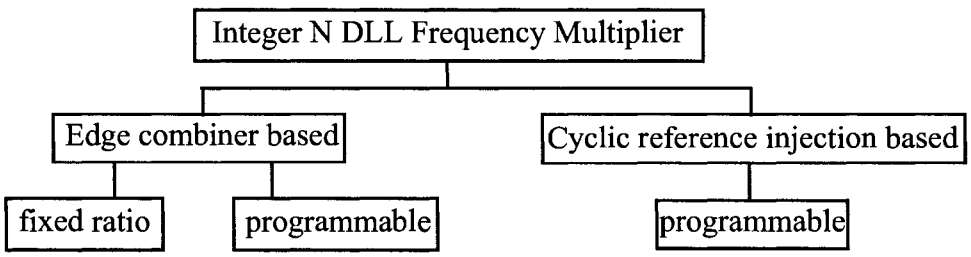


Figure 3.2 Types of DLL frequency multipliers

based ones, there are fixed-ratio integer N frequency multipliers and programmable inte-

ger N frequency multipliers. The reported cyclic reference injection based ones are all programmable.

3.3 DLL based frequency multiplier with edge combining

The basic idea of the edge combining technique is illustrated in Figure 3.3 [30]. The low phase noise signal f_{ref} is applied to a VCDL, and the delayed signals from delay stages I to V are gathered in the edge combining circuit. The delayed signal from the last stage is applied to the feedback circuitry where a control signal is generated to tune the VCDL delay value to exactly one reference period. The final output signal is V_{out} which is

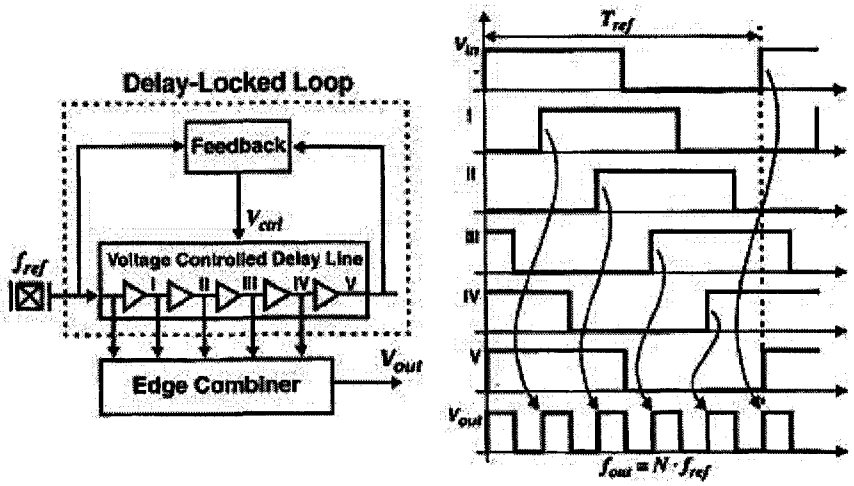


Figure 3.3 The fundamental idea of edge combining DLL frequency multiplier

the combined high-frequency signal. The delayed signals are shown in the figure on the right from stage I to V as an example. Within one reference period T_{ref}, the reference signal is delayed by N stages with equal delay values, and each rising edge of the delayed signal is taken as the rising edge of the output to obtain the final output with the frequency N times the f_{ref}, where N is 5. So the more delay stages, the higher combined frequencies, and by selecting different delay signals to combine, different multiplication ratio can be obtained. For example, combing the edges of stage I, III and V, an output frequency of 3 times the reference is generated. Based on the multiplications ratio, the frequency multi-

pliers are categorized into edge combing frequency multiplier with a fixed ratio and programmable ratio.

3.3.1 Fixed-ratio edge combining frequency multiplier

Figure 3.4 shows a block diagram of the DLL based frequency multiplier with an fixed multiplication ratio [29]. The reference signal is from a crystal oscillator, and it is first amplified for better driving the delay line. The delay line consists of nine delay stages with the delay value locked to half of the reference period. The phase detector, charge

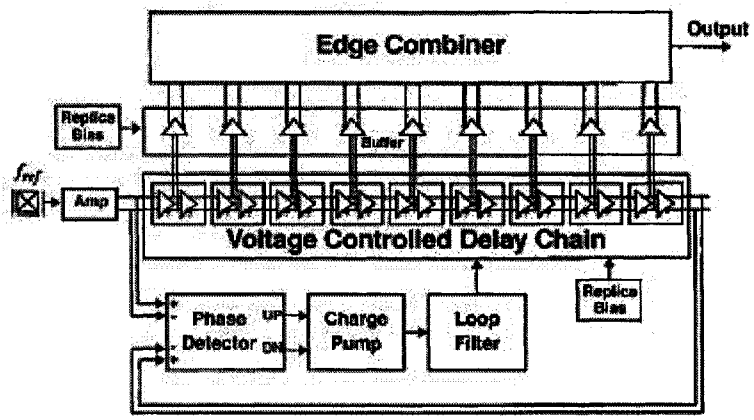


Figure 3.4 Edge combiner based frequency multiplier with LC tanks

pump and loop filter are used to create a control voltage on the delay line to tune the delay. The delayed signals are buffered before they are applied to the edge combiner, where multiplied high frequency output is generated. In order to generate separate I/Q signals, each delay stage consists of two delay cells, and the additional delayed signals are combined with an additional off-chip edge combiner to generate the quadrature signal. The delay cell uses source coupled logic (SCL) to reject the common-mode noise mainly from the supply and the substrate. The delay is tuned by adjusting the tail current in the SCL circuit. The edge combiner consists of nine NMOS input differential pairs and one pair of LC-tanks as shown in Figure 3.5. The tail current from the odd number of the delay stages is modulated between the LC tanks and multiplied high frequency is produced. The LC tanks at

the output is mainly to enhance the load impedance at the resonance frequency, and the close-in phase noise will not be affected since the quality factor of the inductor is low.

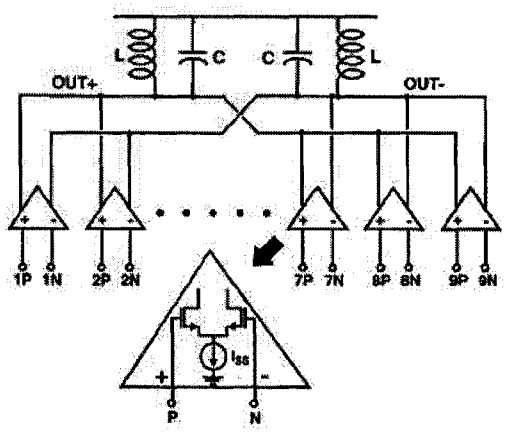


Figure 3.5 The fundamental idea of edge combining DLL frequency multiplier

The drawback of the circuit includes that the LC tanks used in the edge combiner consumes large area and power, and the output frequency has fixed single ratio. The possible false locking due to the DLL nature is not addressed with the possible solution that the control voltage needs to be set at the circuit power-on. The frequency multiplier generates one single frequency of 900MHz with 130mW power dissipation. Table 3.1 gives more detailed performance.

Another DLL edge combining frequency multiplier with fixed ratio is shown in Figure 3.6 [30]. With an additional lock detector, this self-correcting DLL overcomes the false locking problem in the conventional DLL while maintains a large tuning range. The lock detect helps to detect wether the DLL is locked or attempts to be locked to a wrong delay value, and brings the DLL to the correct locking direction, thus the circuit initialization is avoided.

The detailed circuit operation is as follows. Multiple phases from the propagated reference signal in the VCDL are fed to the lock detect where three outputs, *release*, *under* and *over* are generated. When the *under* signal is active, which means the delay value of

the VCDL is less than 0.75 clock periods, and the DLL might lock in a direction to lock the reference to itself, so the lock detect takes control of the circuit and brings down the v_{cntl} to increase the delay value. When the *over* signal is active, it means the VCDL delay exceeds 1.5 reference clock periods, and the DLL might lock to the delayed reference signal with delay value equal to 2 or more clock periods. Similarly, the lock detect controls the circuit by outputting *over* signal to increase *vcntrl* by activating the *UP* in the PD to decrease the VCDL delay. When the VCDL value reaches 1.25 clock periods, the output is *release* which clears the both the *over* and the *under* controls signals and release the PD from reset condition, and the PD regains the control to finely tune the VCDL delay value.

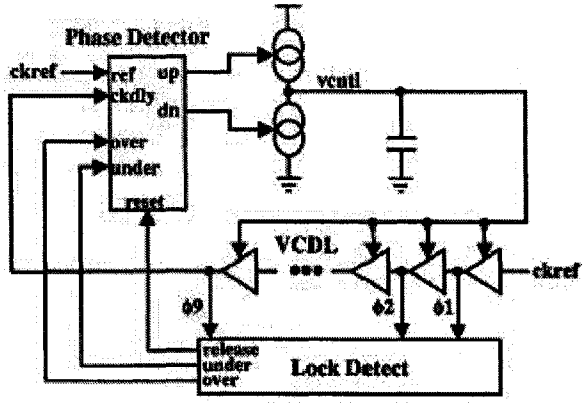


Figure 3.6 The edge combining DLL frequency multiplier with self correction

Nine phases are generated in the VCDL and those phases are fed into the edge combiner to obtain the multiplied frequency as shown in Figure 3.7. It comprises of 4 sets of AND-OR logic structure, and in the last set of AND-OR gate, an analog OR is used as an I/O buffer with the pull-up resisters to set the output swing and match the output impedance to that of the test equipment.

False locking problem is solved in this DLL frequency multiplier with the lock detect, while similar to the edge combining DLL frequency multiplier with LC tanks at the output in the previous paragraph, this DLL frequency multiplier also has fixed multiplication ratio, and the straightforward edge combining technique is also very similar, which is

vulnerable to the mismatch in both the VCDL and the edge combining circuitry. Without using LC, power consumption and active area is dramatically reduced as shown in Table 3.1.

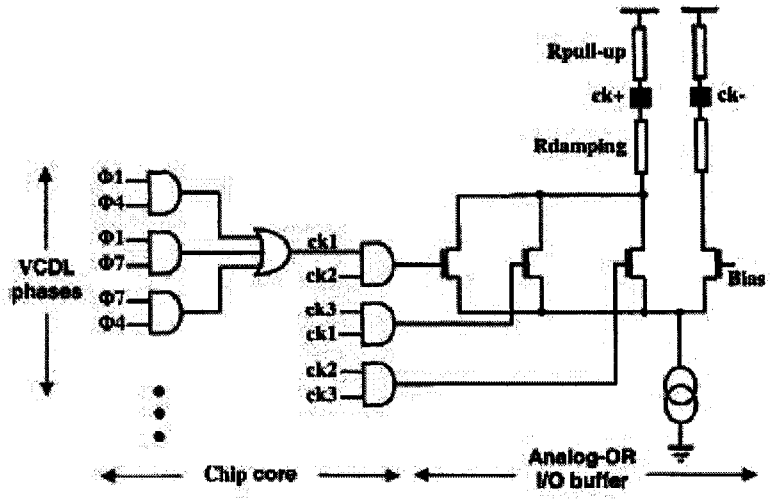


Figure 3.7 The edge combining DLL frequency multiplier with self correction

The digital AND-OR gate based edge combiner introduces phase noise to the output frequency due to the mismatch between stages as well as the power and supply noise. Figure 3.8 [31] shows the reported DLL frequency multiplier with an edge combining circuit which is powered by a regulated voltage source to avoid phase noise caused by the power supply. To increase the locking range as well as to ensure correct locking, a reset circuitry is added to the PD, however, a control voltage on the delay line has to be initialized to make the VCDL delay within the locking range. The outputs of the phase detector are filtered and applied to a voltage regulator to generate the control voltage for the VCDL. Multiple phases are generated and buffered before being applied to the frequency multiplier to generate the combined high-frequency clock, and both of the buffer and the frequency multiplier are supplied by a regulated voltage from the on-chip voltage regula-

tor.

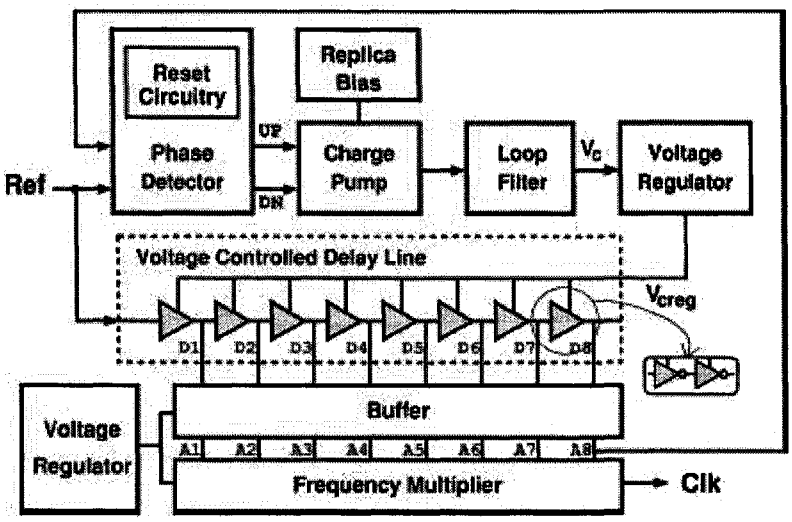


Figure 3.8 The edge combining DLL frequency multiplier with self correction

The operation principle and loop behavior is similar to the circuit reviewed previously. The main new point lies on the edge combining circuitry named frequency multiplier as shown in Figure 3.9. The VCDL outputs *A1* to *AN* are applied to the NMOS transistors with shared node *X* at the drain and shared node *Y* at the source. The node *X* is applied to transistor *P2* and *Y* is applied to *N4* to generate the combined signal *Clk* and one feedback signal *Qbd* for the cyclic toggling. Transistor *P1* and transistor *N3* are connected to node *X* and *Y* respectively to control charging and discharging nodes *X* and *Y*. For example, when *Qbd* is high, node *X* keeps its previous *Hi* data value while node *Y* is discharged to the low level, and *Clk* keeps its previous value too. On a rising edge of *Ai*, data transfers from node *X* to *Y*, and node *X* is discharged to the low level, which turns on transistor P2 and after three inverter delay, *Qbd* turns to low, and *P1* is turned on. At the next rising edge of *Ai*, *Y* node is charged to high since *P1* is already on, which turns on *N4* and discharges node *Q*, consequently the state of *Clk* is inverted. Therefore, the toggling process generates the multiplied frequency *Clk*, and its state transition is triggered at each rising

edge of the *Ai* signals, so the multiplication ratio is *N/2*.

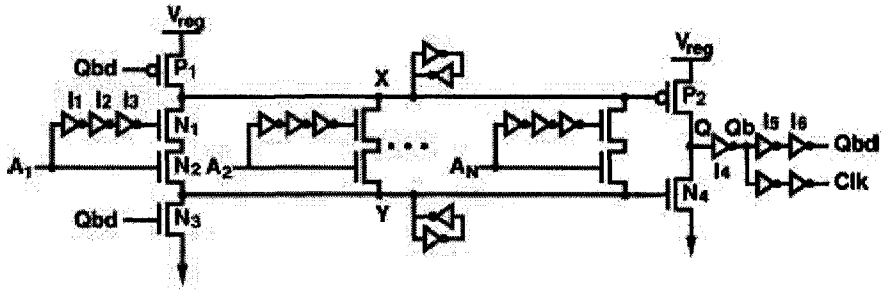


Figure 3.9 Schematic of the edge combining circuit--frequency multiplier

The advantages of this edge combing circuit include that it is less sensitive to the mismatch among the data path with shorter transmission path and less components. The main drawback is that the multiplication ratio is smaller than the previously reviewed publication. For example, to obtain the same multiplication ratio as of nine, eighteen delay stages have to be used in the VCDL, which cause more power consumption and add more phase noise since the more stages in the VCDL, the more phase noise is accumulated.

3.3.2 Programmable DLL frequency multiplier with edge combiner

The edge combining DLL frequency multipliers with fixed multiplication ratio are reviewed in the last section. However, for wide applications, it is more desirable to generate frequencies with multiple ratio. This section gives the reported DLL frequency multipliers based on edge combining technique with programmable multiplication ratio.

Figure 3.10 shows the block diagram of the programmable edge combining DLL frequency multiplier [32]. Similar to the previous reviewed architecture, it comprises of a PFD, a charge pump, a loop filter which is a capacitor as well, a voltage controlled delay tap line (VCDTL), a positive edge collector and a clock generator. The VCDTL is programmable with 10 delay stages, and each delay stage consists of two inverter-based delay

cells and two DEMUX as shown in the figure. The DEMUX at the output of the odd-num-

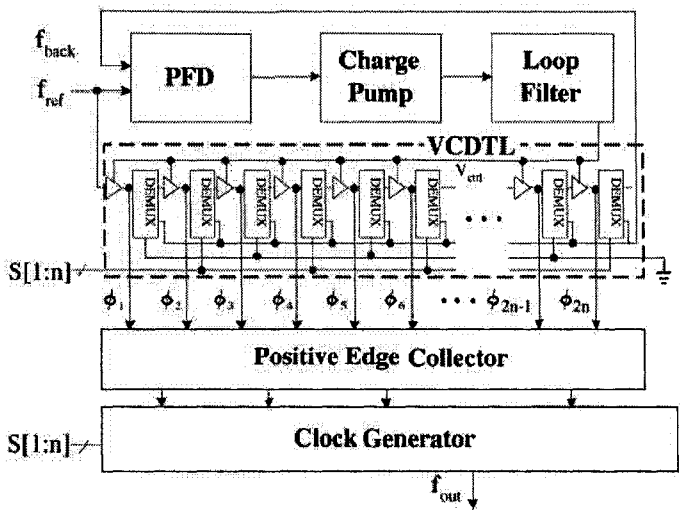


Figure 3.10 The programmable edge combining DLL frequency multiplier

bered stage is a dummy DEMUX for load balance with the control signal connected to the ground, and the other DEMUX are controlled by signal S1[1] to S1[n] to select which signals are chosen to be applied to the edge collector to generate the final output. The control voltage of the VCDTL *vctrl* is also initialized at power-on to the middle of the entire operating voltage for correct and fast locking.

The positive edge collector shown in Figure 3.11 is to detect the rising edges of the

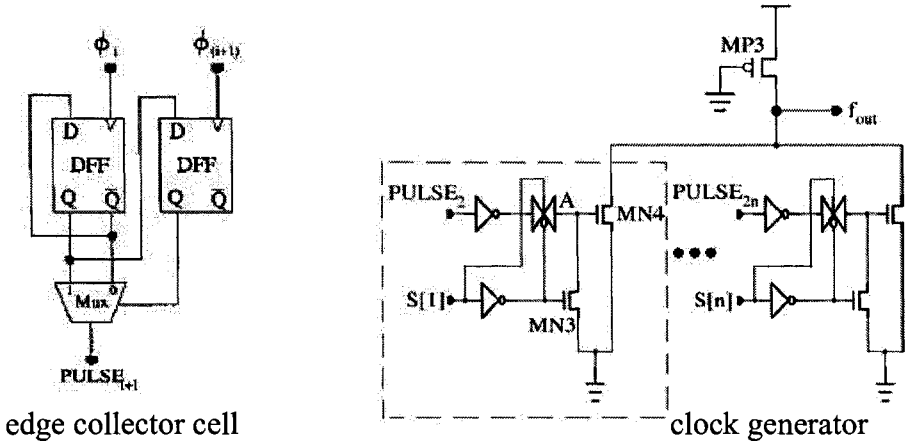


Figure 3.11 Schematic of edge collector and clock generator

multiple phases from the VCDTL and produce low pulse trains corresponding to the rising

edges. One cell consists of two DFFs and one MUX, and its function is independent of the initial conditions of the DFFs. The main advantage is that it is less sensitive to noise since it is purely digital. The resulting low pulse train is applied to the clock generator as shown in the figure to generate the final output. S2[1] to S2[n] is the external selection signal to select which pulse to be passed to drain of the NMOS transistor as the final output.

Except for the drive circuit in the VCDTL and the pseudo-N logic, the combining circuitry in this DLL frequency multiplier is digital and programmable, which is immune to noise, but the mismatch among different signal combining paths results in large spur at the output (-20dB at reference frequency with the carrier frequency of 1.2GHz), and the external control signals add more complexity to the circuit. Another external signal the circuit need is the VCDTL initialization voltage, so the circuit lacks the ability to lock to correct reference edges automatically. For the operation frequency, due to the delay in the DEMUX in VCDTL, the DFF and MUX in the pulse generator, the frequency of the combined signal is low with the measured maximum frequency of 1.2GHz.

Another example of programmable DLL frequency multiplier with the edge combining technique is shown in Figure 3.12 [33]. The architecture is very similar to the one shown in Figure 3.8 except that this one has a programmable feature. Similarly, the feedback circuit includes the PFD, the charge pump, a loop capacitor, and a voltage regulator to avoid the phase noise from the power supply. The reference signal propagates in the VCDL consisting of 8 delay stages, and multiple clock phases *D1* to *D8* are generated and applied to a transition detector to find the transitions of the multiple phase signal, then the transition signals are combined in the edge combiner to generate the high-frequency signal *CLK* with the maximum frequency value of 4 times the reference frequency. A multiplication factor controller generates control signals for both the MUX and the transition detector, and the proper phase for the DLL to lock the reference to is selected in the MUX. The control signals *S1* to *S8* in the transition detector select appropriate transitions which are

combined in the edge combiner as shown in the Figure 3.13. The transition detector con-

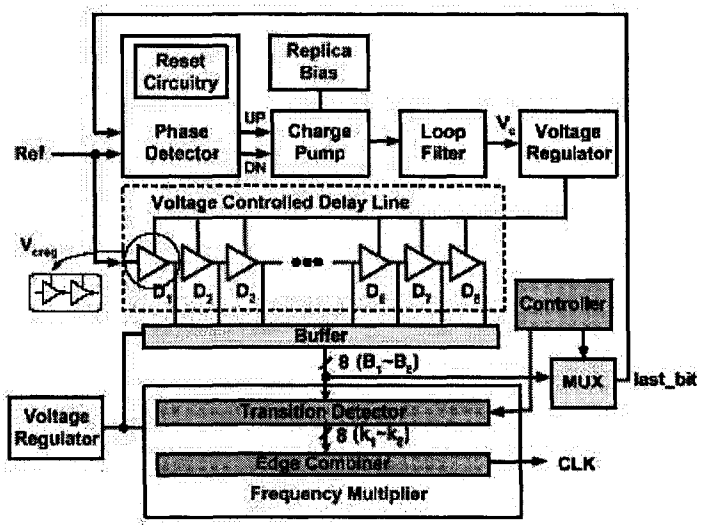


Figure 3.12 The programmable edge combining DLL frequency multiplier

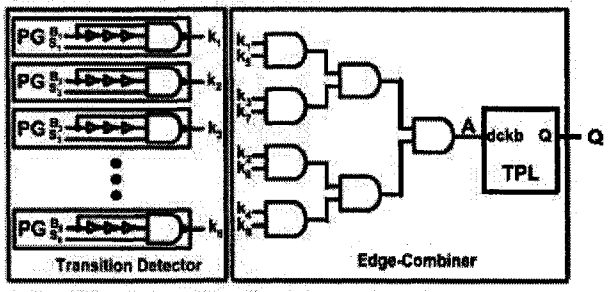


Figure 3.13 Schematic of the transition detector and the edge combiner

sists of 3 inverters and one three-input AND gate. Signals *S1* to *S8* are the enabling signals to adjust the multiplication factor, and a negative pulse is generated at any rising edge of *B1* to *B8*, and those negative rising edges are combined in the edge combiner which consists of AND gates and a toggle latch whose output is connected to the data input. First the pulses are combined at node A, then the latch toggles its output once there is negative pulse at node A. Except for the advantages same with the similar structure shown in previous paragraph, this DLL frequency multiplier can be programmable. However, the multiplication ratios are not continuous (the multiplication ratio are 0.5, 1, 15, 2, 2.5, 3, 3.5 and 4), and to achieve higher frequency, more hardware has to be added.

3.4 Programmable DLL frequency multiplier with cyclic injection

In the previous sections, DLL frequency multipliers with edge combining circuits are reviewed. With the edge combining technique, multiple multiplication ratio can only be achieved by combine the delayed reference signal from different delay stages, and it is difficult to obtain high frequency output because the multiplication process has to introduce several MUXes into the delay line.

Instead of using edge combining circuit, this section reviews the recently reported DLL frequency multipliers with cyclic reference injection and conventional dividers which are normally used in PLL based frequency multipliers to achieve programmable multiplication and high frequency output. Another significant advantage of the cyclic reference injection over the edge combiner is that the output frequency is less sensitive to the mismatch among the VCDL and the edge combining circuitry, which results in large spurious output power.

The block diagram of the programmable DLL frequency multiplier is shown in Figure 3.14 [34]. As shown in the figure, it comprises of a PD, a charge pump, a VCDL, and loop capacitors as a conventional DLL may have. Besides, it has a MUX, a divider and a select logic to help generate high frequency output with programmable multiple ratios. Similar architecture can also be found in [36] [37]. With the MUX, the DLL output is fed back to its input, and forms a ring oscillator that oscillates with a period twice of the VCDL and MUX delay. The high-frequency signal generated from the ring oscillator is divided by the divide-by-M divider which is essentially a counter, and signal *last* is generated at every M cycles of the ring oscillator. The *last* signal triggers the select logic that switches the MUX input to inject the clean reference signal to the delay line which now works as a voltage controlled delay line, and the clean reference edge resets the phase noise accumulated in the previous M-1 cycles. The function of periodically reference

injection into the ring oscillator removes the accumulated phase noise, and continuos integer N multiple ratio can be achieved with the conventional counter with the output frequency N times the reference.

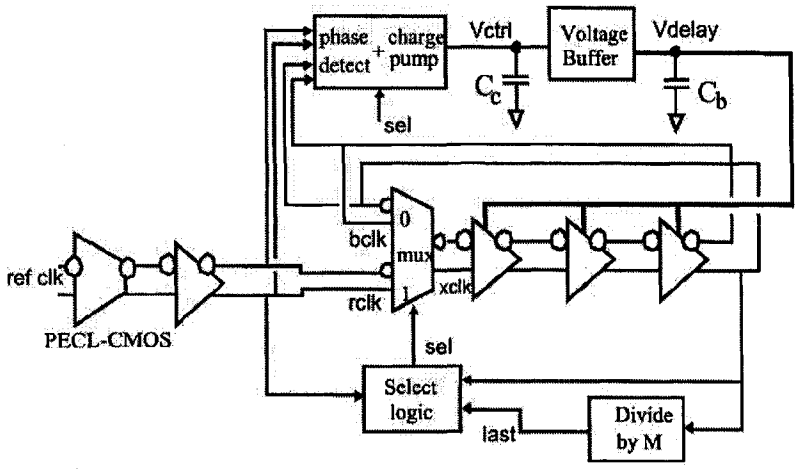


Figure 3.14 Block diagram of the programmable DLL frequency multiplier

Figure 3.15 shows the detailed timing diagram of the DLL frequency multiplier. At the circuit power on, the DLL control voltage is pulled high, and the delay line is set to its

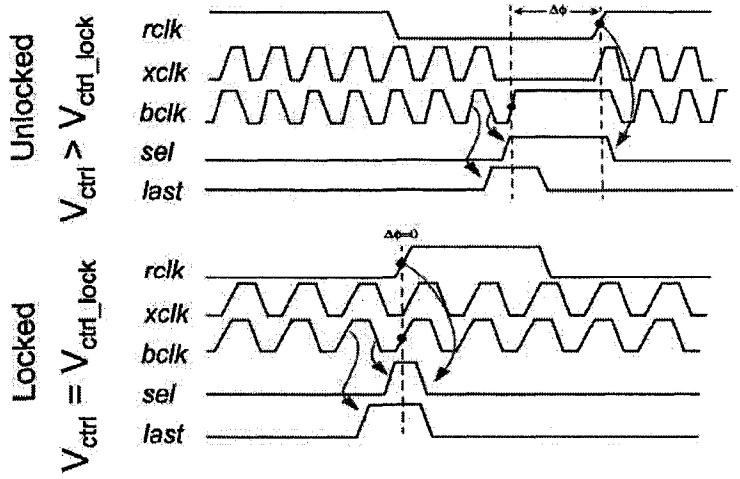


Figure 3.15 Frequency multiplier timing

minimum value while the oscillating frequency to its maximum frequency. As a result, the *bclk* signal completes M cycles before the following reference rising edge *rclk*. On the *M*th rising edge of *bclk*, the divider enables the select logic which asserts the *sel* signal at

the following falling edge of *bclk*, and switches the multiplexer to *rclk*, so a clean reference is injected. At this point, the reference signal propagates in the VCDL, and MUX is kept connected to *rclk*. The PD compares the corresponding rising edge of *bclk* with that of the *rclk* which is newly injected and a phase error $\Delta\phi$ is generated. After the reference is injected, the select logic disables the *last* signal, and the MUX is switched to *bclk*, restarting the oscillator. The phase error drives the charge pump and loop filter to decrease the control voltage and the oscillating frequency until the loop is in lock as shown in the figure when the *Vctrl* equals to the *Vctrl-lock*.

The main advantages of this architecture includes that the cyclic reference injection resets the accumulated phase noise, thus the mismatch among delay stages and edge combining circuits is avoided, and multiple multiplication ratio is achieved easily with a simple counter. The drawbacks include that the process of injecting reference edge introduces extra phase error that can hardly be detected by conventional phase detector and charge pump, and the loop needs initialization voltage at circuit start-up. Table 3.1 gives more detailed performance.

3.5 Review of reported phase error reduction technique

In the cyclic reference injected DLL frequency multipliers design, there are many phase noise and timing jitter sources, and the main sources include the in-lock error due to the mismatch in the charging and discharging current sources in the charge pump, the mismatch of the phase detector outputs, the re-alignment error caused by the reference phase injection. All of those errors can be considered as the systematic in-lock error. In the previously reviewed DLL frequency multipliers, the issue of the phase noise and timing jitter caused by the system in-lock error is mainly ignored. The other noise sources includes the VCO noise, and there are two design works reported in literature employing some techniques to mitigate the phase noise performance deteriorated by the VCO noise and noise resulted from the PD/CP mismatch in the loop.

One of the techniques is reported in [36]. The architecture is very similar to the one used in Figure 3.14 in [34]. In this reported design, the static phase error due to the imbalance of the mismatches in PFD/CP is compensated. A low-bandwidth secondary loop is added for compensating the current mismatch. The compensation loop is digital, and it comprises of a bang-bang PD and an accumulator functioning as an integrator with infinite dc gain, and the output of the integrator controls a digital to analog converter (DAC) that leaks current from either side of the charge pump. However, the detailed implementation is not available. With the compensation loop disabled, the output signal edge is split into two distinct waveforms which are 81ps apart, and with the loop enabled, the measured RMS timing jitter is 4.7ps. For comparison, more performance results are listed in Table 3.2.

The other reported phase noise reduction technique is in [37], and the circuit architecture is shown in Figure 3.16. The operation principle is similar to the one in

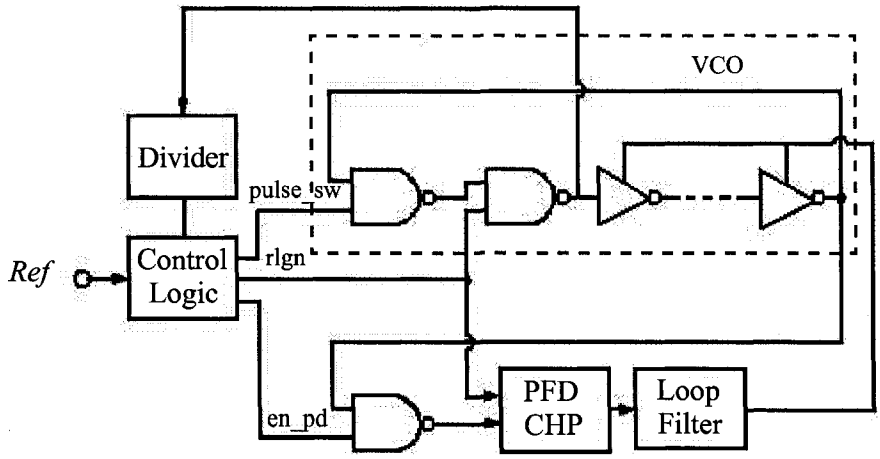


Figure 3.16 DLL frequency multiplier with phase noise suppressing

Figure 3.14. The phase noise reduction technique is used in the VCO, which contains two NAND gates functioning as an MUX and a few of delay cells. The phase noise caused by the *1/f* noise in the VCO can be reduced by periodically switching the MOSFET between '*on*' and '*off*' states. By turning off the transistor, the trapped carriers tend to be released,

resetting the noise source, and the noise correlation is interrupted.

To minimize the *1/f* noise, switched circuit is used in both the delay cells and the bias circuit as shown in Figure 3.17. The circuit switches the source and drain to turn off

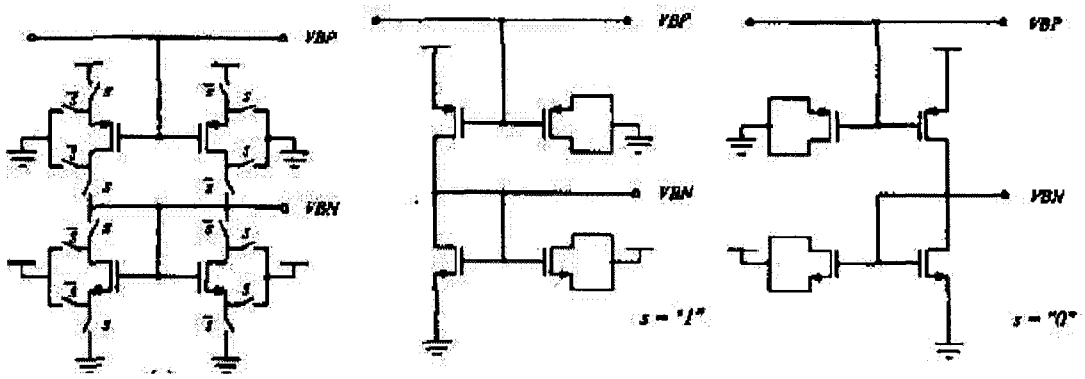


Figure 3.17 VCO switching bias circuitry

each transistor. For example, by switching both the source and the drain to VDD, an NMOS transistor is off. The figures on the right show the operation with clock at high level and low level respectively, and the clock is from the VCO itself. The measured results show that a phase noise reduction of 5dB with the noise reduction technique enabled.

Based on the reviews presented in the previous paragraph, the technique used in [36] only focuses on compensating the phase error caused by the charge pump, neglecting the phase noise caused by the cyclic reference injection, while the one reported in [37] is only targeted on reducing the phase noise in the ring oscillator by minimizing the *1/f* noise. And neither of them mentioned the harmonic locking problem due to the nature of the DLL. Consequently, the circuits have to be initialized at power on in the same way as in the circuit reported in [34].

3.6 Summary

In this section, two types of DLL based frequency multipliers and two reported techniques employed to improve the phase noise and timing jitter performance reported

most recently were reviewed.

Table 3.1 illustrates the performances of the reviewed DLL frequency multipliers. From the table, the cyclic injection technique used in [34] helps reducing the timing jitter and spurious output, and both the power and the area are small. Ref [33] used programmable edge combiner, and its timing jitter is small too, however, the power consumption is huge due to the larger multiplication ratio which requires more delay stages, and more phase noise is accumulated in the delay line. Ref [32] is also a programmable frequency multiplier, but with eight external control signals to program the multiplication ratio, and its phase noise is not as good as [29]. Besides, due to the mismatch among the edge combining paths, the spur performance is poor. Ref [29] [30] [31] are all edge combiner based ones with single frequency output, among which [29] consumes the largest area and power dissipation due to the LC load, and [30] [31] are comparable to each other regarding to the jitter performance. However, the large power consumption, single multiplication ratio and relative low output frequency limits its application.

TABLE 3.1: Performance comparison of selected DLL frequency multipliers

	[29]	[30]	[31]	[32]	[33]	[34]
Technology	0.35 μm	0.5 μm	0.35 μm		0.35 μm	0.18 μm
V_{DD} (V)	3.3	1.8-3.3	3.3	2.5	3.3	1.8
Output (Hz)	900M	0.6-1G @2 V	120M-1.2G	1.5G(max)	0.12-1.8G	0.2-2G
Multiplication ratio	9	9	4	7-10	0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4	4, 5, 8, 10
Phase noise/ jitter(dBc/ps)	-123 @ 60 k	@1 G: 3.2 (rms), 20 (p-p)	2 (rms), 14.5 (p-p)	-103.4@50 k, -93.5 @10 k	1.8 (rms) 13.2 (p-p)	@2 G: 1.73 (rms),12 (p-p)
Spur	-37 @ 200 M	N/A	N/A	-20 @120 M	N/A	-37 @ 2 G
Power(mW)	130	54 @2 V	42.9	52.2	86.6 @1.6 G	12 @ 2 G
Active area	$1.2\times1.0mm^2$	0.6 mm^2	0.07 mm^2	0.132 mm^2	0.07 mm^2	0.05 mm^2

In addition, all of the previously reviewed circuits need to be initialized at the circuit power on, and once the circuit tends to lock in the wrong direction, the circuit has to be reset.

Although the DLL based frequency multiplier with the cyclic reference injection possesses the best phase noise performance and the easiest way to achieve multiple multiplication ratio as compared in Table 3.1, its spur performance needs further improvement compared with its counterpart, the PLL based frequency multipliers. As reviewed in the previous paragraph, two reported work focus on noise reduction. The performances are listed in Table 3.2.

TABLE 3.2: Performance comparison of reported DLL frequency multipliers with phase noise reduction technique

	Technology	V_{DD} (V)	output (Hz)	ratio	phase noise / jitter (dBc/ps)	Spur (dB)	Power (mW)	Area (mm^2)	Noise source compensated
[36]	0.25 μm	2.5	500 M	4	4.7 (rms) 36.7(p-p)	N/A	28	0.21	The mismatch between the CP currents
[37]	0.18 μm	1.8	1M-160M	N/A	noise reduction of 5dB	N/A	8.6 (core)	4.84	Flicker noise in the VCO

From the table, [36] only compensates the phase noise caused by the mismatch between the current sources in the charge pump, and the phase noise due to the cyclic injection is not considered. Even with a divider, its output is a single frequency, and adding more ratio will result in more power and possibly more phase noise to the output. [37] only considered suppressing the flicker noise in the VCO, and noise reduction of 5 dB is observed in the DLL based multiplier with cyclic reference injection, however the phase noise performance is not given in the report.

CHAPTER 4

A Proposed CDR with A Digital Threshold Decision Technique

4.1 Introduction

Literature review of the reported CDR circuits given in Chapter 2 shows that the all-digital CDRs are of interest because of their portability, low power consumption and simple oscillator-less architecture. In this chapter, a digital-threshold decision oversampling CDR architecture with fast-acquisition and high jitter tolerance performance is first proposed. The principle of the proposed CDR is first discussed, followed by more detailed analysis of the proposed CDR with a 5X oversampling ratio. Second, the operation principle of the proposed CDR is verified in Simulink, and the high jitter tolerance performance is verified with an event-driven simulation in Matlab. Third, the power spectrum of the recovered clock is analyzed theoretically. Finally, a summary of this chapter is given.

4.2 The proposed digital-threshold decision CDR architecture

In the proposed CDR architecture illustrated in Figure 4.1, the incoming data is first sampled by the sampling circuits with the sampling clocks generated from a multiphase clock generator to obtain N data samples in each data symbol period and the recovered data or clock is obtained by dynamically selecting one out of the N data samples or multiple phases. To achieve such a dynamic selection or to determinate the data sampling phase position (DSPP), those samples are applied to the digital process block, in which a

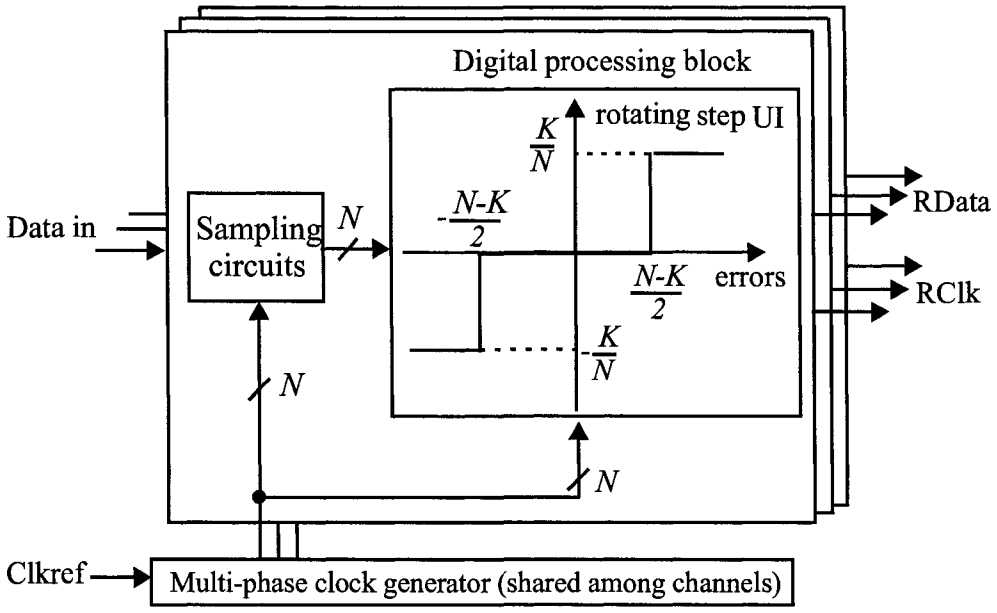


Figure 4.1 The architecture of the proposed CDR

transition detection logic first generates N data transition signals based on those samples and a phase error is evaluated based on the current DSPP and the data transition information. Then, the digital process block performs a decision technique to update the DSPP based on the phase error.

For an oversampling ratio of N, the ideal data sampling position is at the $\left\lceil \frac{N+1}{2} \right\rceil th$ phase after the data transition for any odd value of N. The phase error is assumed to be zero if no data transition is detected. If the data sampling phase is exactly the $[(N+1)/2]th$ phase after the data transition detected, the phase error of the data sampling position is defined as zero, otherwise, the phase difference between the data sampling phase and the middle phase is defined to be the phase error value, which could be 0, ±1, ±2... To ensure the correct data recovery, the maximum phase error should not be beyond $[(N-1)/2]$.

The current DSPP is updated upon an error of larger than *(N-K)/2* or smaller than *-(N-K)/2* detected within a given examining window (EW) by advancing or retarding the

DSPP by *K/N* UI as shown in the vertical axis, where *K* is the number of phase steps. All the errors with other values are ignored in order to increase the high-frequency JT. Consequently, the value of $|N-K|/2$ is considered to be the decision threshold to determine whether the current DSPP needs to be updated to keep it in the data eye opening region.

By updating the decision within each baud period, the fast acquisition can be achieved. However, the JT performance at high jitter frequencies is degraded because the tracking behavior of the CDR is always limited by a certain bandwidth. Normally, the high-frequency jitter tolerance of a CDR can be improved by good jitter averaging. To achieve such averaging, the updating decision is based not only on the phase error within each baud period, but also on the errors in the previous data bits within the examining window, where the phase errors are detected and memorized. The window length is required to be large enough so that at least 2 data transitions can be detected within any EW, while the low-frequency JT is degraded if the EW length is too large. Thus by both examining the errors in each baud period and previous baud periods in the EW, fast acquisition and high-frequency jitter tolerance are both maintained.

The followed sections describe mathematical analysis of the proposed technique mainly focused on the jitter tolerance performance.

4.3 Mathematical analysis of the proposed CDR

As the principle of the proposed CDR described in the previous section, the data sampling phase may be updated (advanced or retarded by $\frac{k}{N}$ UI) only if the phase error reaches $\frac{N-K}{2}$ or $-\frac{N-K}{2}$, while all phase errors below the threshold are ignored. In other words, theoretically the jitter amplitude at high jitter frequencies in the data can be tolerated up to $\frac{N-K}{N}$ UI, therefore the ideal peak-to-peak jitter tolerance at the high jitter frequencies can be easily obtained as

$$JT_H = 1 - \frac{K}{N} \tag{4.1}$$

To maximize the high-frequency jitter tolerance, K can be set to be its minimum value of 1 while N can be as large as possible. However, a large oversampling ratio N degrades the jitter tolerance at low frequencies where the CDR tracks the input jitter by updating the data sampling phase. By defining the normalized jitter frequency (F_j) as the ratio of the actual jitter frequency (Hz) to the baud rate (bits/s), the edge jitter at frequency F_j can be expressed as

$$Jitter(t) = A_j \cdot \sin(2\pi F_j t) \tag{4.2}$$

where A_j is the jitter amplitude function in the data, and the fastest edge changing speed can be obtained by differentiate the jitter function,

$$\frac{d(Jitter(t))}{dt} = A_j \cdot 2\pi \cdot F_j \cdot \cos(2\pi F_j t) \tag{4.3}$$

So the fastest edge changing speed is $A_j \cdot 2\pi \cdot F_j$. In the case when the input data has the maximum transition density (100%), the fastest updating speed of the CDR is $\frac{K}{N}$ UI per symbol period, which can be expresses as

$$A_j \cdot 2\pi \cdot F_j = \frac{K}{N} \tag{4.4}$$

Therefore the maximum jitter amplitude that the CDR can track in this case or the jitter tolerance at the low jitter frequencies can be derived is

$$JT_{p-p}(UI) = \frac{K}{N \cdot \pi \cdot F_j} \tag{4.5}$$

For the case of 2^7-1 PRBS data, the jitter tolerance obtained from above formula should be divided by 7 because the lowest transition density is 1/7 (one transition in every seven symbol periods). Generally speaking, the low frequency jitter is expressed as,

$$JT_{p-p}(UI) = \frac{K \cdot D_t}{N \cdot \pi \cdot F_j} \tag{4.6}$$

where D_t is the lowest transition density of the incoming data. With the high-frequency jitter tolerance and the low-frequency jitter tolerance, the normalized corner fre-

quency of the CDR can be calculated as

$$F_c = \frac{D_t}{N \cdot \pi \cdot (1 - K/N)} \tag{4.7}$$

From the equation, in the case of $N = 5$ and $K = 1$, a 5X oversampling CDR applied with 2^7-1 PRBS data, the low-frequency jitter tolerance is calculated as 91(UI p-p) at F_j=0.001and the normalized corner frequency is 0.011.

4.4 Choosing design parameters

Since the jitter tolerance at high jitter frequencies is $JT = 1 - \frac{K}{N}$ as derived previously, the JT at the high jitter frequencies is large with large N and small K at the cost of the jitter tolerance at the low jitter frequencies. It is because at low jitter frequencies, the jitter tolerance depends on how fast the CDR is capable of tracking the data jitter. For a given data jitter, a CDR with large N tracks the jitter more slowly than a CDR with a small oversampling ratio does.

However, with increasing N and smaller K, the loop is prone to capture lock in the wrong direction. Taking $N = 20$, $K = 1$, a 7-bit PRBS data and a sampling clock period of 600ps as an example, according to the threshold decision technique, the current data sampling phase will not be updated until a phase error larger than 30ps is detected. The maximum CID pattern is 6 data bits. If the initial data jitter is -5ps, and suppose the data transition is at a little bit more than 30ps later than the sampling clock as illustrated in Figure 4.2. According to the digital threshold decision technique, the data sampling phase is advanced and rotated to the left by one phase step. In case of a maximum CID data stream, the phase error will be accumulated to -35ps after 7 data bits when there is a transition coming. At this point, the data transition happens before the same sampling point clock phase, therefore the current data sampling phase will be retarded or rotated to the

right according to the technique. For the case when the data jitter is +5ps, after enough long data bit periods, the transition also happens ahead of the same sampling clock phase. Consequently, the CDR misses tracking the data jitter, and its BER performance degrades.

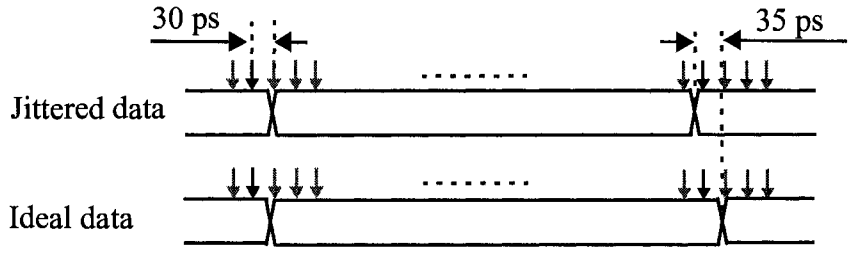


Figure 4.2 Illustration of the false phase tracking

Therefore, the trade-off between the high frequency jitter tolerance and the low frequency jitter tolerance needs to be considered for the determination of N and K. In the case where N=20 and K=4, a theoretical jitter tolerance at high jitter frequencies can be the same as the case of K=5 and K=1, but with finer phase rotating step and smaller clock jitter. However, large oversampling ratio results in more complex hardware and thus more power consumption.

The smallest values of the sampling ratio N and the decision threshold K are 3 and 1 respectively. In this case, the phase step is large, which results in large clock jitter in the re-aligned clock. In addition, the maximum theoretical jitter tolerance at the high jitter frequencies is 0.66UI, which can hardly meet the jitter tolerance specifications of the SFI-5 interface with some degradation due to the PVT conditions.

The length of the EW is also an important parameter in the updating decision technique as it affects the high-frequency jitter tolerance. If the phase position error within each baud period is examined to make the updating decision, the acquisition time can be dramatically reduced to be one baud period, and the low-frequency jitter tolerance is improved. However, by updating decision every baud period, any jitter with periods less

than one baud period is considered to be high-frequency jitter, and the jitter tolerance at those frequencies degrades.

Therefore, not only the data sampling phase error in the current baud period is considered, but those in the previous baud periods within one examining window are also examined. The examining window defines the number of data bits within which the error is detected and memorized to determine the updating decision. For the data from a 2^M-1 PRBS data, the examining window is required to be larger than M to ensure at least two data transitions is detected, so at least one data sampling phase error is defined within the examining window. The jitter with the period smaller than M baud periods is considered as high-frequency jitter which is not tracked by the loop.

Consequently, the acquisition time and the updating time are sped up by examining the data sampling phase position error in each baud period, while high jitter tolerance is maintained by examining the errors in baud periods within the examining window in the mean time. In addition, by ignoring errors that is within the threshold values can also contribute to high jitter tolerance performance. Obviously, the decision updating is not periodic as no new decision is made until the error reaches the threshold values.

Consequently, in this all-digital CDR design, by taking the jitter tolerance specifications, the acquisition time, and the power consumption into considerations, the oversampling ration N and the decision threshold K are taken as 5 and 1 respectively with the input data from an on-chip 2^7-1 PRBS data generator with an EW length of eight baud periods. The simulated jitter tolerance performance will be given after the proposed 5X oversampling CDR with the digital threshold decision technique is described in details.

4.5 The proposed CDR with 5X oversampling5

From the previous sections, design parameters of $N = 5$ and $K = 1$ with the incoming data from an on-chip 2^7-1 PRBS data are chosen. Accordingly, the decision threshold

values are -2 and +2. In this section, the proposed technique applied to this specific design with the chosen parameters is explained and analyzed in details. The system architecture of the 5X oversampling CDR technique is similar to the proposed technique in the general cases. The incoming data is first sampled by the 5 sampling clock phases in the sampling circuit, and the five sampled data is processed according to the decision technique to extract the data and clock information.

The basic operation of the 5X oversampling is illustrated in Figure 4.3. Suppose the first data samples are '00000' followed by two '11111', so a data transition is detected

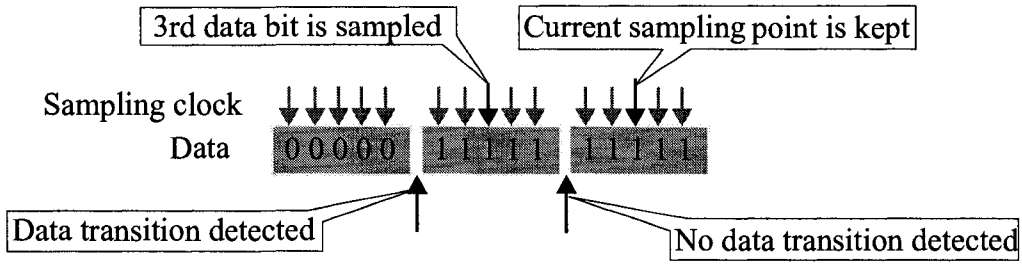


Figure 4.3 5X oversampling in the CDR

between the first and the second data bit. According to the decision technique in the general case, the third sampling clock phase after the data transition corresponds to the recovered data for the best BER performance. When no data transition is detected, the same data sampling phase is kept until next data transition is detected for decision updating.

Similar to the general case, the data sampling phase errors are first determined for each data transition, which is repeated in Figure 4.4 for the case of N=5. The data sampling phase position is assumed to be fixed as highlighted in the dark arrow. If the transition is found to be at the 3rd phase that advances the assumed data sampling phase position as indicated by the transition arrow in Figure 4.4 (c), the data sampling phase position error is defined to be '0', which means there is no phase error between the data sampling clock and the data eye center. There are also other possible data transitions as

shown in (a), (b), (d), (e). If the data transition is advanced (or retarded) by one phase step

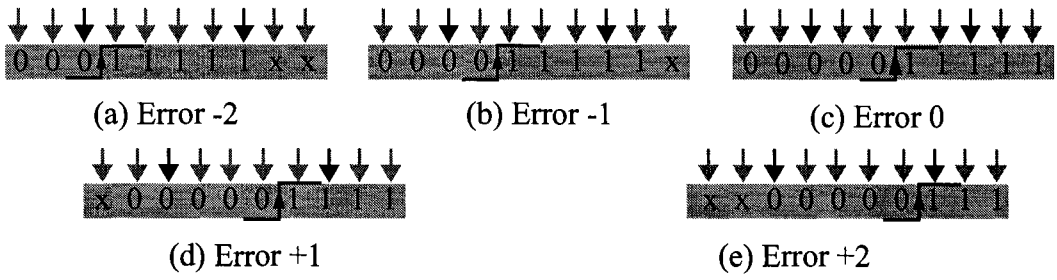


Figure 4.4 Definition of the sampling phase position errors

with respect to the one in (c) and the data sampling phase keeps at the assumed fixed position, the error is defined to be '-1' (or '+1') as shown in (b) (or (d)). Similarly, if the data transition is advanced (or retarded) by two phase steps with the same data sampling phase position, the error is defined as '-2' (or '+2') as shown in (a) (or (e)).

The defined errors determines the updating decisions according to the 5X oversampling CDR technique shown in Figure 4.5. The updating technique is as follows,

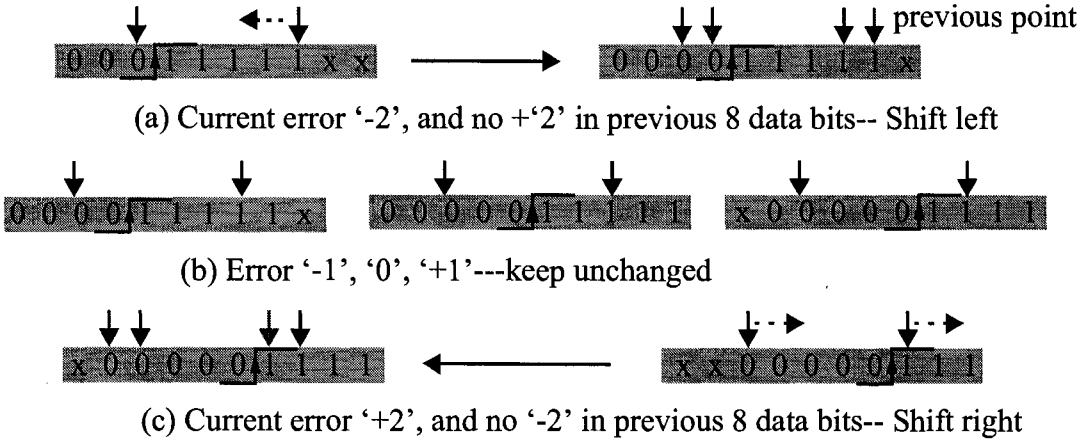


Figure 4.5 Data sampling phase position updating

- In each baud period, if an error -2 (or +2) is detected and no +2 (or -2) ever appeared within the examining window, the current DSPP is rotated to the left (or right) by one phase step as shown in Figure 4.5 (a) (or Figure 4.5 (c)).
- For all the other possible error combinations, the current data sampling phase posi-

tion is kept unchanged (Figure 4.5 (b)).

In the figure, the solid arrow indicates the updated data sampling phase position, and the dotted arrow indicates the previous one. This updating technique is capable of updating all the possible data transition positions at a given data sampling phase position as illustrated in Figure 4.6. The rectangular shows the data eye opening region with the minimum eye width of slightly larger than 0.2UI as required in this technique, and the shaded area is the jittered data transition area. The solid arrow indicates the assumed data sampling phase position. If the two adjacent sampling phase positions are aligned at the two ends of the horizontal eye opening as shown in Figure 4.6 (A) and (B), the possible transitions are detected between the sampling phase position signals *a* and *b*, *c* and *d*, *e* and *a* (in case of A) or *a* and *b*, *b* and *c*, *d* and *e*, *e* and *a* (in case of B). So according to the

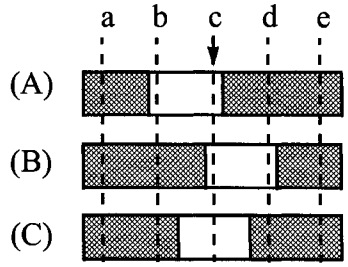


Figure 4.6 Data sampling phase position updating

definition of the error in Figure 4.4, the possible errors are 0, -1, +1, and -2 (A) or 0, -1, +1, and +2 (B). Consequently, in case of errors 0, -1 and +1, this sampling point is kept unchanged, and in case of error -2 and 2, it is rotated to the left or the right by one clock phase step. In other words, in case of (A), the error is -2 according to the data transition position, and the sampling phase selection signal will be rotated so that case A becomes case B, while in case of (B) the phase position error is +2, and the data sampling phase position will be rotated to the case (A). In case of (C), only the data sampling point is within the horizontal eye opening, so both errors of -2 and +2 appear if the updating time is long enough, and the sampled data is right in the eye opening area, therefore the data sampling point has not to be updated.

Similar in the general case, by examining each current baud period and its previous eight ones to check if there is ever one -2 error or +2 errors present, the circuit can acquire lock within one baud period. Unlike the reported CDR works where there is always a updating period in the decision making, the updating time is not periodic since according to the technique, the CDR doesn't update its decision until an error -2 or +2 is detected. In addition, by examining the phase position errors in the previous 8 data symbols and ignoring the phase errors of -1 and +1 equals to adding the averaging mechanism in the CDR, so the jitter tolerance of the CDR at high jitter frequency is increased.

4.6 An implementation of the proposed 5X oversampling CDR

An implementation of the proposed digital threshold decision CDR is shown in Figure 4.7. It consists of a phase detector with three functions, data sampling, transition

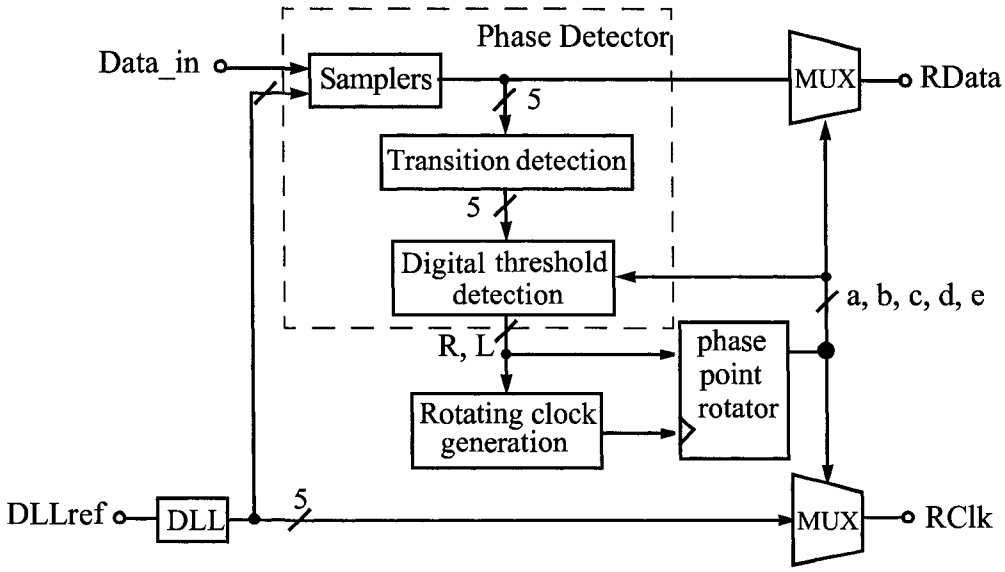


Figure 4.7 System architecture of the proposed CDR

detection and threshold detection, a rotating clock generator, a phase rotator and two multiplexers. The incoming data is first sampled by five clock phases from a DLL based multiple-phase clock generator, and the data transition is detected. Based on the transition information and the phase rotator which outputs a five-bit (*a, b, c, d, e*) one-hot coded sig-

nal, the phase position error signal is determined and the digital threshold value is detected in the threshold detection function block. If a negative phase error of a threshold value is detected, an *L* signal indicating the current data sampling phase position should be advanced is generated. In case that an error of +2 is detected, an *R* signal is generated to retard the current data sampling point. However, the CDR doesn't have to update the one-hot coded phase selection signal for all the errors with a threshold value, so a rotating clock generation block is employed to generate a rotating clock to trigger a data sampling phase updating by rotating the one-hot signal in the phase rotating block. The phase rotator updates the one-hot signal which is again fedback to the PD to generate new errors, and the phase and frequency difference between the data and the sampling clock is thus tracked.

4.7 Matlab simulation model of the proposed CDR

Based on the system implementation proposed in the previous section, an Matlab model of the whole CDR system is built to verify the functionality of the proposed CDR as shown in Figure 4.8. It consists of three functional blocks, one phase detector, one rotating clock generator, and one sampling phase rotator. The recovered data and re-

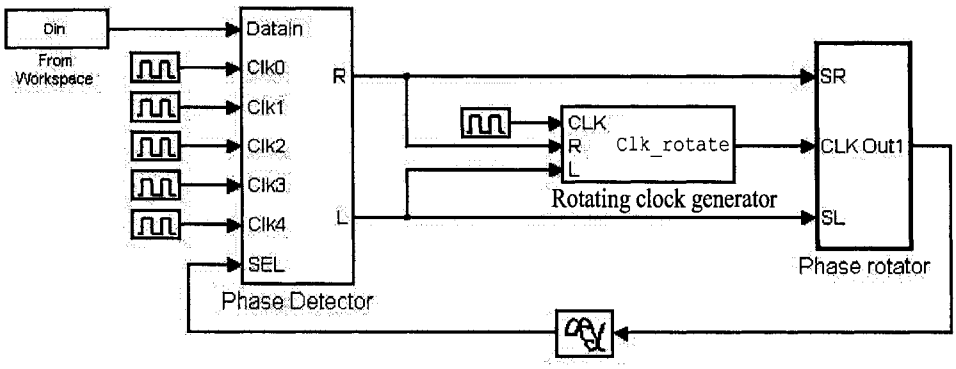


Figure 4.8 Block diagram of the CDR system model

aligned clock can be obtained with two multiplexers which are not included because the MUXes are outside the loop, and this model is mainly used to confirm the CDR loop oper-

ation based on the proposed technique.

Five clock phases are applied to the phase detector to detect the data transitions at full data rate, and the data source *Din* is from a PRBS7 generator. The PD generates signals *R* and *L* according to the proposed decision technique, and a phase updating is triggered by signal *Clk_rotate* from the rotating clock generator to determine when the data sampling phase is rotated for updating, while signals *R* and *L* determine which direction to rotate to. The phase rotator rotates the one-hot signal *a, b, c, d, e* to update the data sampling phase position and the updated one-hot coded signal is fed back to the PD to examine the new phase error until the error is 0.

The functional block diagram of the PD is shown in Figure 4.9. The data is first sampled by multi-phase clocks in the sample/hold block, and the transition information is

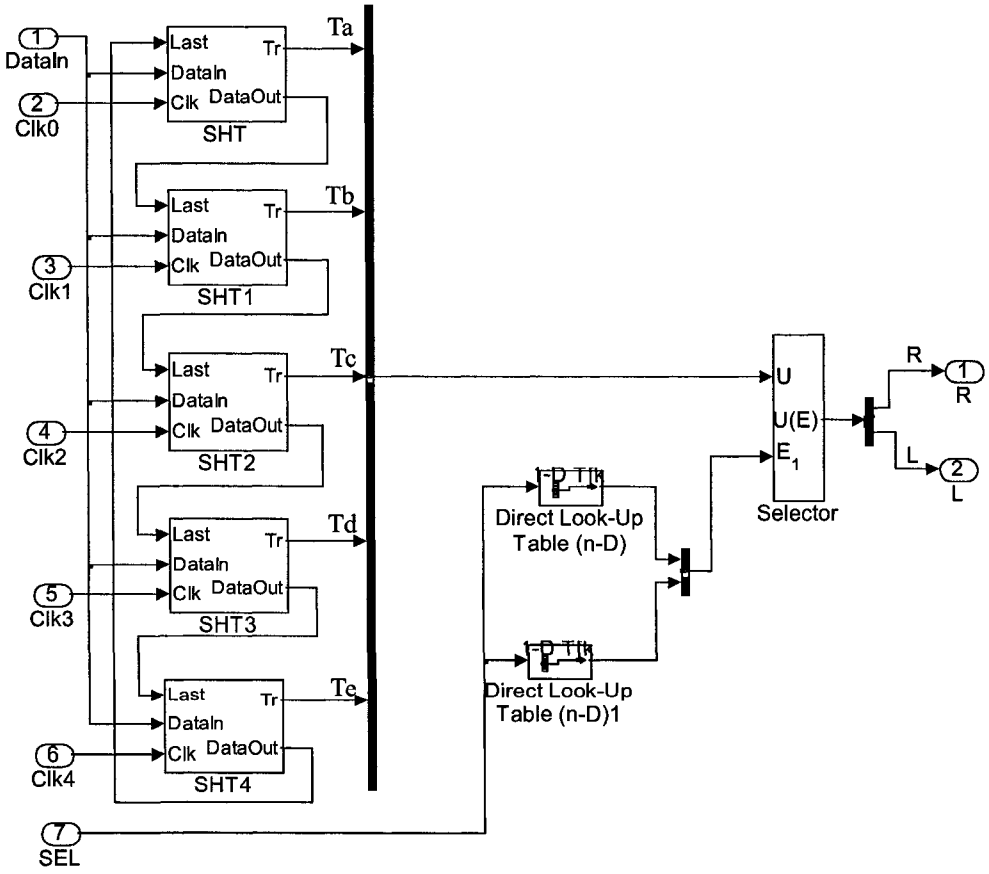


Figure 4.9 Block model of the phase detector

detected in the transition detection block shown in Figure 4.10, where the data samples are applied to the XOR gates with the data samples sampled by the clock phase which is earlier by one phase step than the current one, then an AND gate eliminates the unwanted pulses from the XOR gate, and only outputs the transition signals *Ta*, *Tb*, *Tc*, *Td*, *Te*.

Among the five transition signals, two transition signals detected by two clock phases which are adjacent to the current data sampling phase are selected as *R* and *L* signals. The selection signal *SEL* is from the clock phase rotator, and it is split into two selection signals, one of which picks up the *R* signal, and the other one picks up the *L* signal. If the data sampling phase is on the right of the adjacent clock phase which detects the transition, *L* is high and *R* is low, otherwise *L* is low and *R* is high.

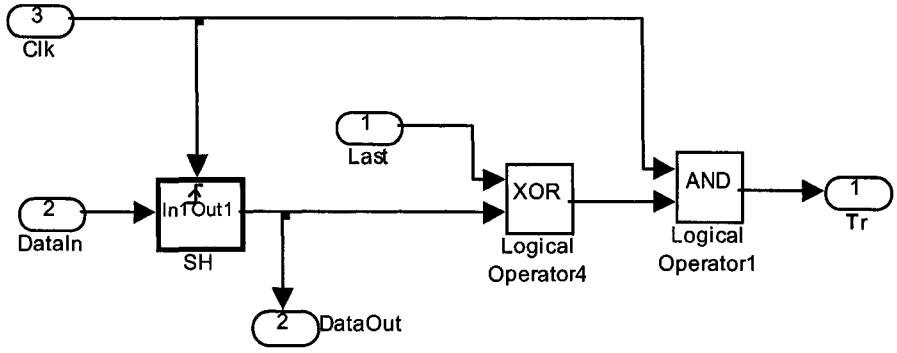


Figure 4.10 Model of the transition detector in the phase detector

Figure 4.11 shows the block diagram of the proposed rotating clock generator. Two subsystems shown in Figure 4.12 are used to detect whether there is *R* signal or *L* signal in the EW, the previous eight data symbols. If either signal *R* or *L* ever appears in the EW, a low output is generated from the subsystems (otherwise 1 is generated), and the outputs are applied to the two AND gates whose outputs are then combined to generate the rotating clock in the OR gate to rotate the one-hot signal. The output *Clk_rotate* only indicates when the phase rotator is triggered and it does not indicate which direction to rotate. For example, if node A (shown in Figure 4.11) is 0, and *L* is 1 at a time instant, which means within the 8 data symbol periods, there are both *R* and *L* detected, so according to the tech-

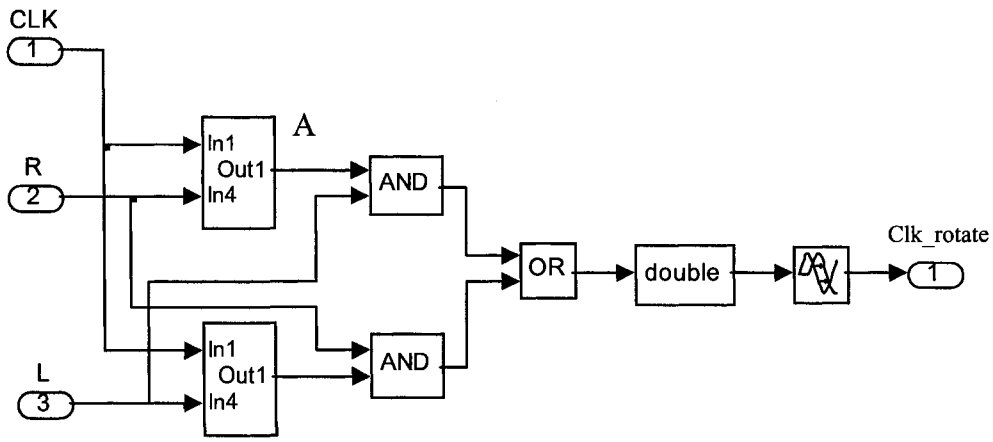


Figure 4.11 Block model of the rotating clock generator

nique, neither *R* nor *L* is activated, and this is obtained by the AND gate followed which generates a low output. If node A is 1, and *L* is 1, it means at this time instant, there is high *L* and low *R*, so the output the followed AND gate is 1, indicating the data sampling phase should be rotated.

Figure 4.12 shows the detailed mathematical block diagram of the subsystem used in the rotating clock generator block. The input *In1* is the sampling clock from the DLL

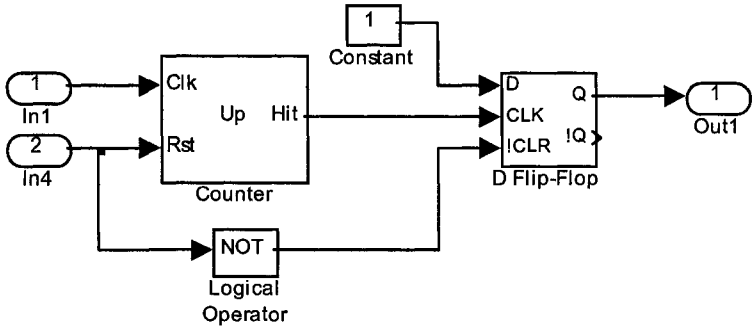


Figure 4.12 Subsystem in rotating clock generator block model

multiple-phase clock generator, and signal *In4* is the PD output *R* or *L*. With a counting-up counter, the clock cycles are counted and whenever 8 cycles are reached, one hit is generated, and the hit output is reset to 0 once there is a high *R* or a high *L* coming. Therefore if the hit output is 1, there is no *R* nor *L* within the 8 baud periods. If the hit output is 0, there are *R* or *L* signals within the EW depending on what the input In4 is. It also implies that

more than one *R* or *L* has the same function with just one *R* or *L* in the EW. The high hit value triggers the DFF followed, and a high output is produced by combining both the *R* and the *L* signals to indicate the incoming rotating action in the phase rotator. The simulation results are shown in Figure 4.13 and Figure 4.14 with different data rates and the same clock rate. When the clock period is 300ps and the data symbol period is 309ps, the simulation is shown in Figure 4.13, and Figure 4.15 shows the expanded figure for a

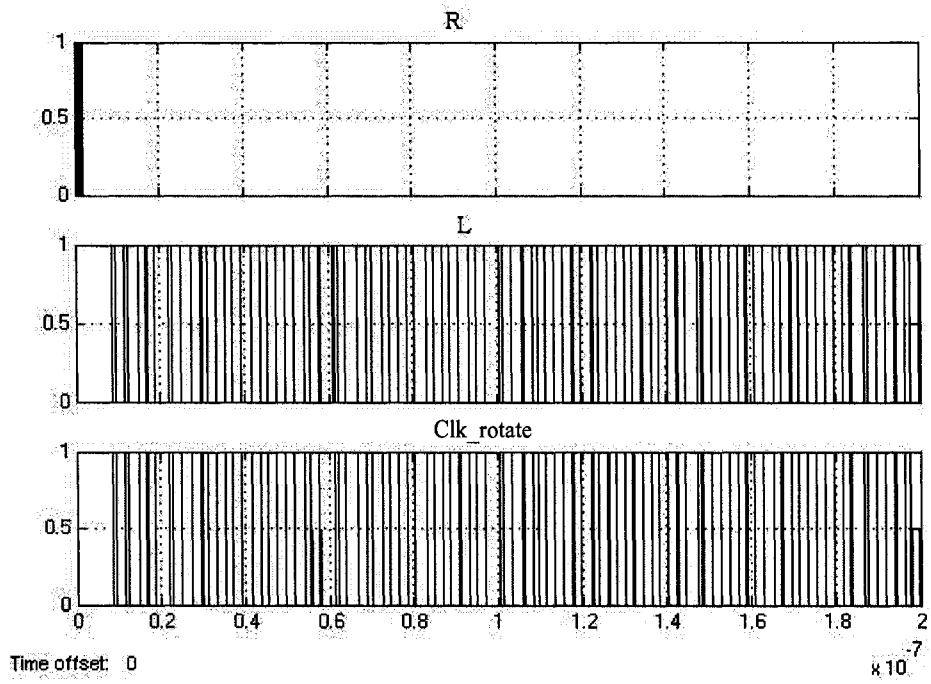


Figure 4.13 Simulation results of the rotating clock generator

clearer picture. From the figures, each pulse of *L* corresponds to a *Clk_rotate* pulse to trigger a phase rotating action.

Similarly, when the data period is 291ps with the same clock period of 300ps, the data sampling phase should be kept rotating to the right to be retarded. Figure 4.14 shows the inputs and the output of the rotating clock generator. Obviously, each *R* or *L* pulse corresponds to one *Clk_rotate* pulse too, both of which verify its functionality.

Figure 4.16 gives the model of the phase rotator. Signal *Clk_rotate* triggers the subsystem when a phase rotating action is needed, and the *R* and *L* are from the PD indicating the phase rotation direction. The updated output *PR_Out* is then fedback to the PD

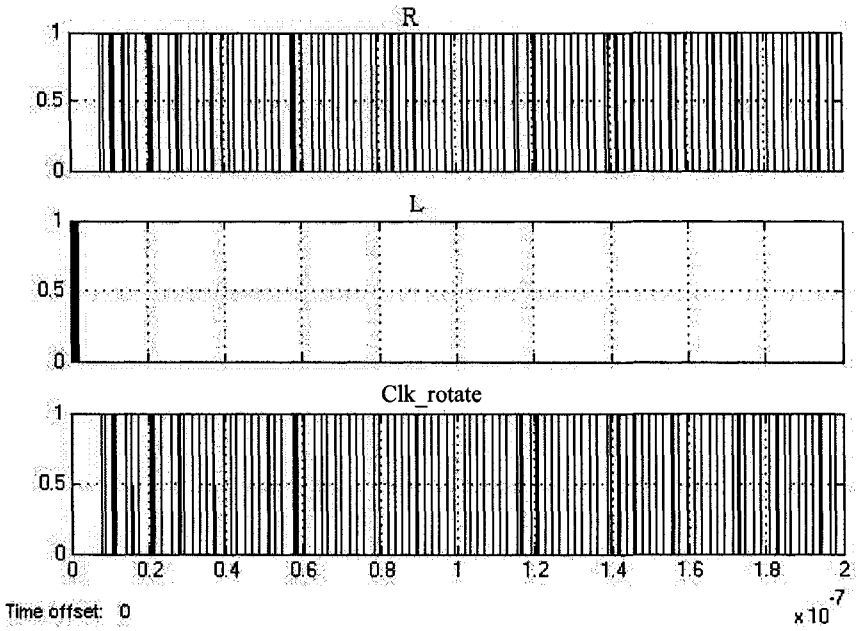


Figure 4.14 Simulation results of the rotating clock generator

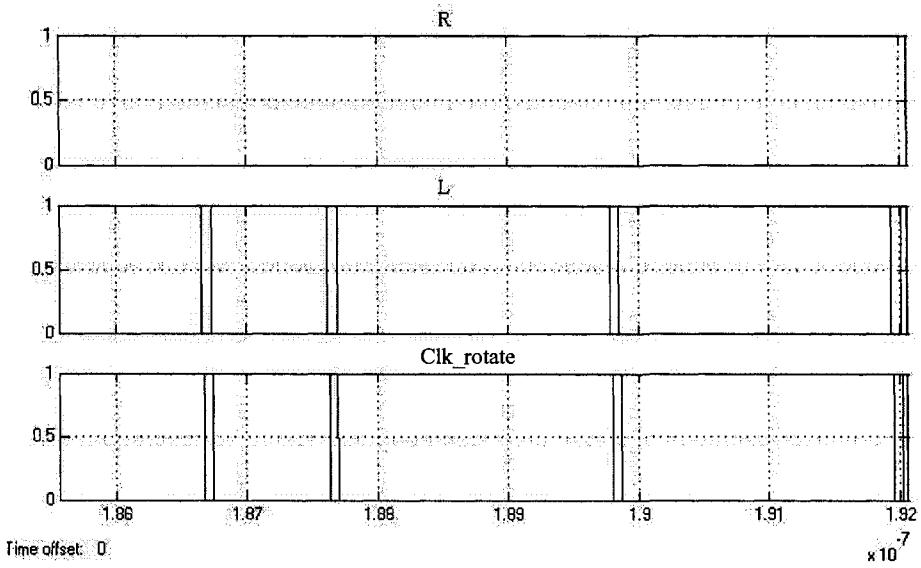


Figure 4.15 Expanded simulation results of the rotating clock generator

to recover the data by picking up the data sample that is in the data eye opening region.

The mathematical model of the phase rotator is shown in Figure 4.17. The *L* are subtracted by *R* and the resulting value is integrated by a discrete integrator, which generates a linear line with positive slope if the subtracted value is positive, otherwise the slope

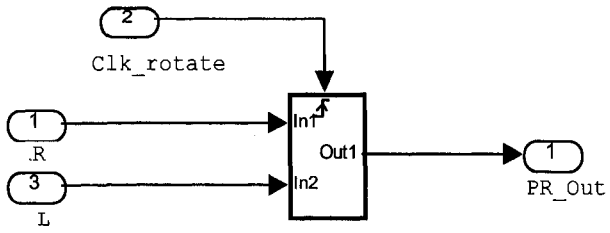


Figure 4.16 Symbol of the data sampling clock phase rotator

is negative. The integrated value is then applied to a function embedded with a floor function to generate the one-hot signal to rotate the five clock phases. The simulated phase

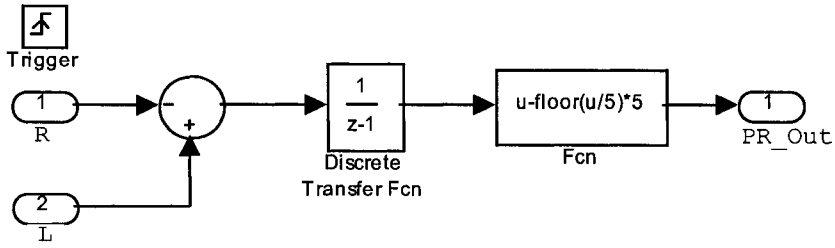


Figure 4.17 Block diagram of the clock phase rotator model

rotating is shown in Figure 4.18 with a data period of 309ps and a clock period of 300ps (3% difference, which is 3000 ppm). The CDR tracks the data by rotating the data sampling phase from the first phase to the 5th phase then rotates back to the first one. The expanded figure shown in Figure 4.19 gives a clearer illustration. The phase rotator rotates the one-hot signal on each L pulse. Similarly, when the data symbol period is 291ps (3000ppm), the phase rotator rotating is shown in Figure 4.20. The CDR rotates the one-hot signal in an opposite direction from the fifth one to the first one then rotates back to the fifth one. An expanded picture is also given as shown in Figure 4.21. The results obtained in the two cases with the clock leading and lagging the data respectively verify the functionality of the proposed CDR.

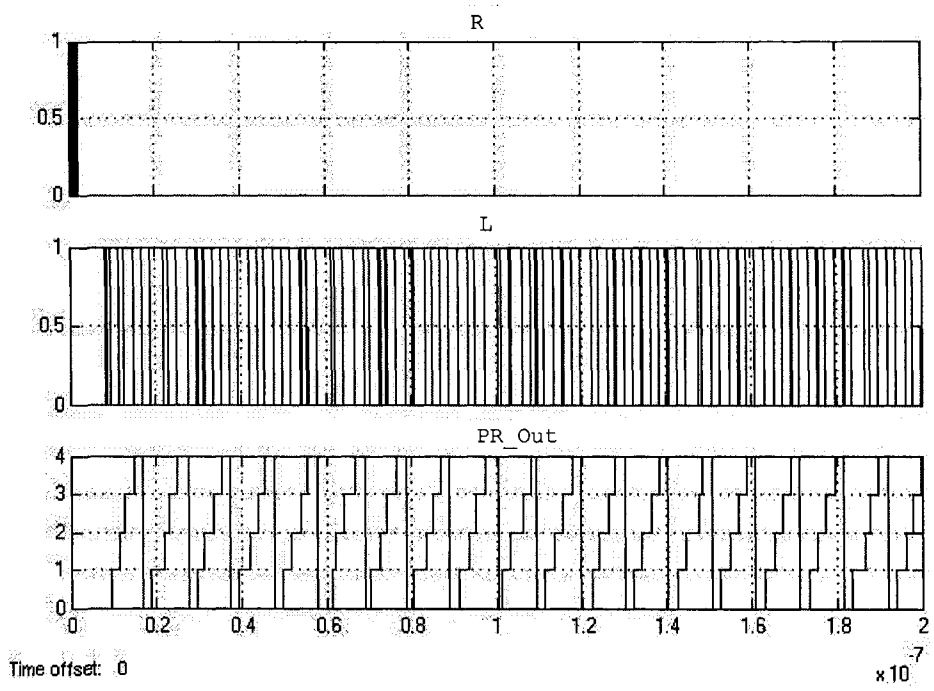


Figure 4.18 phase rotation when the clock phase leads

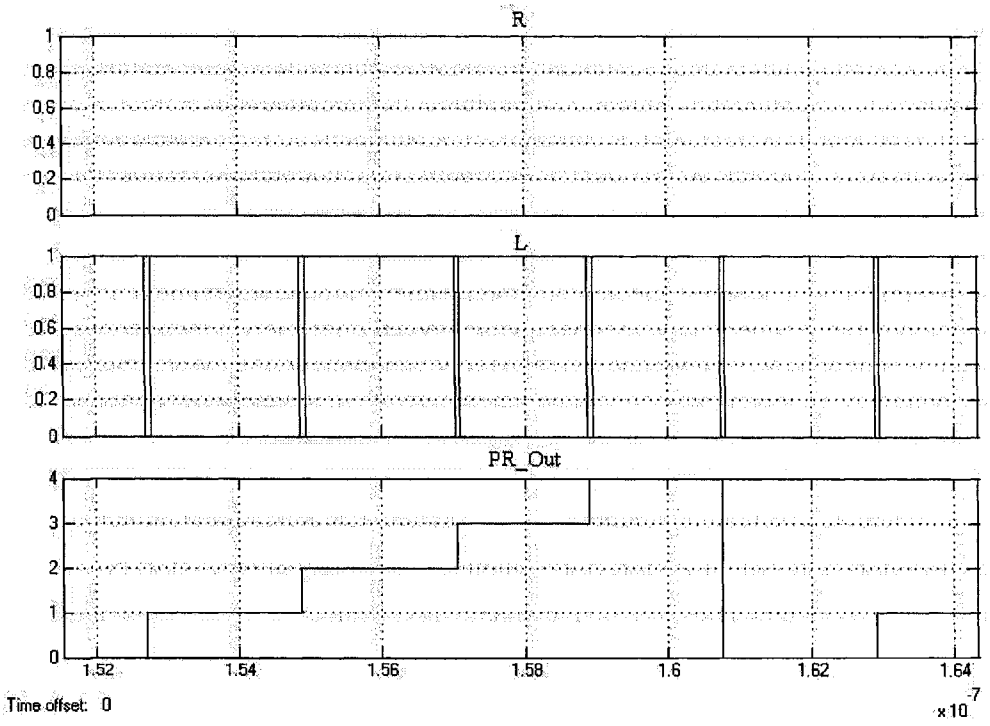


Figure 4.19 Expanded phase rotation when clock phase leads

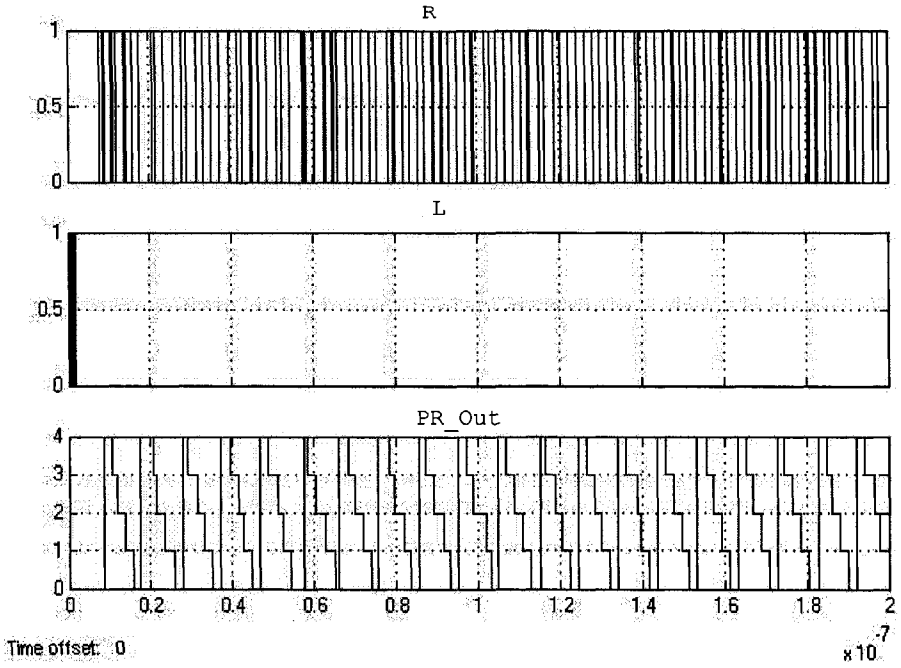


Figure 4.20 phase rotation when the data phase leads

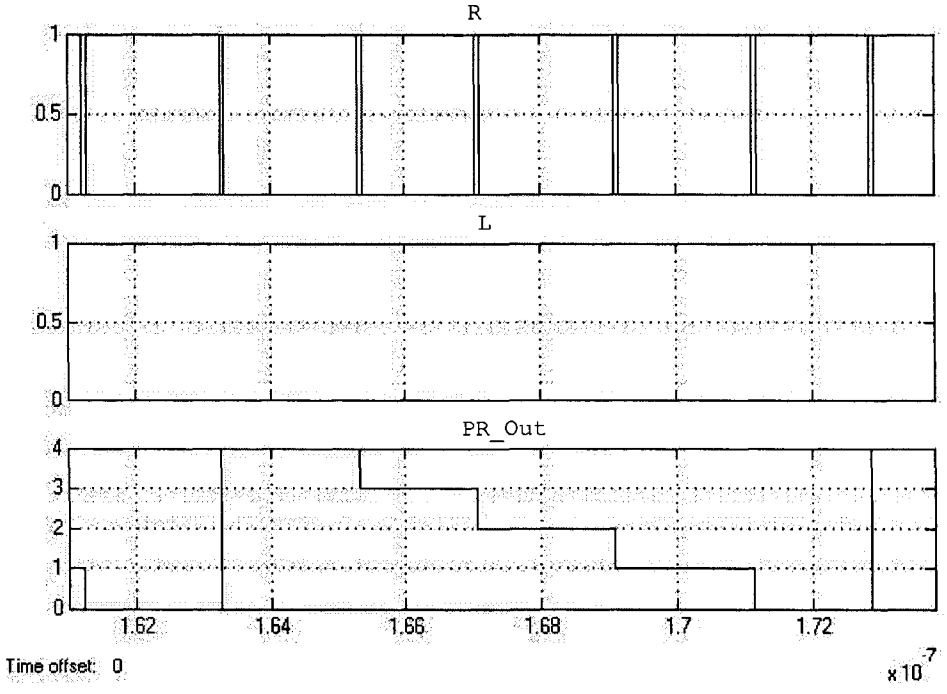


Figure 4.21 Expanded phase rotation when the data phase leads

4.8 Event-driven jitter tolerance simulation

The jitter tolerance performance is of great importance for a CDR. For the low-frequency jitter, the CDR updates its decision by tracking the data edge jitter, and high jitter tolerance performance at low jitter frequency means a fast loop response to the data jitter. However, for the jitter tolerance at high jitter frequencies, the CDR cannot track it correctly, and the performance can only be enhanced by averaging the jittered data samples so that the data sampling phase stays at the center of the data eye opening region.

The traditional way to obtain the JT value in Matlab is from Simulink simulation. By setting a BER limit, the input data and the recovered data are compared. At a certain jitter frequency, the jitter amplitude is increased during data comparison until the BER limit is reached, then changing to another jitter frequency point, and repeat the process to obtain the maximum jitter amplitude over the jitter frequencies. Therefore only limited jitter frequencies can be tested and the data length cannot be very long due to the dramatically increased simulation time.

In this section, an event-driven simulation from Matlab script is proposed to obtain the jitter tolerance of the proposed CDR technique. Its top-level description is given in Figure 4.22. First a 7-bit PRBS data is generated from the script, and then at a given jitter frequency, an initial jitter amplitude and its initial updating step size are set. The data samples are obtained by applying the jittered data which is modulated by the sinusoid jitter to the samplers based on the specified jitter parameters, and based on the proposed technique, the data is recovered from the data samples. Then the recovered data and the original data are compared to find whether there is any bit errors, and if any error is detected, the jitter amplitude will be reduced, otherwise the jitter amplitude is increased as shown in the figure. Bisection method instead of linear method is used to update the jitter amplitude for reducing the simulation time. This process is repeated until the updating step size *DeltaA* is smaller than the pre-defined value *R*, which is the defined as the resolution

value. The final jitter amplitude value is the jitter tolerance value at the current jitter frequency, then this process is repeated at the next jitter frequency point until the jitter tolerance at all desired jitter frequencies are calculated.

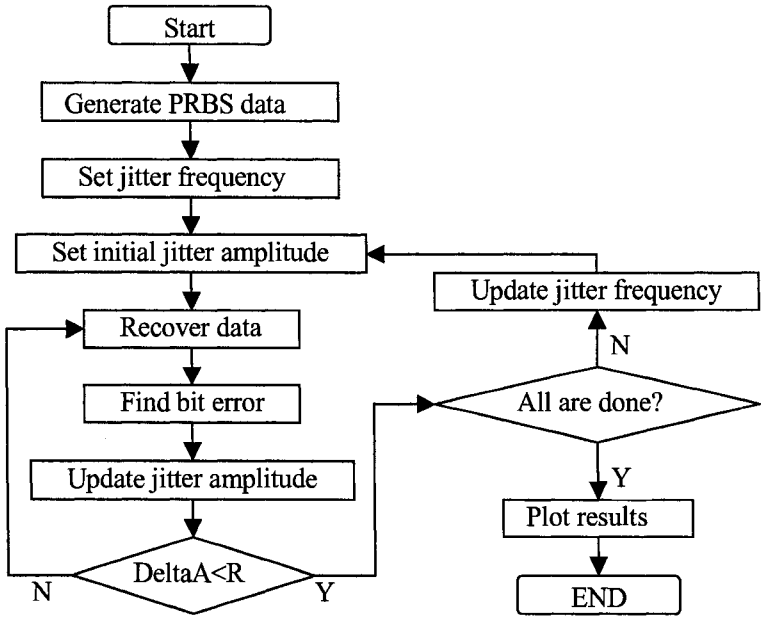


Figure 4.22 Top-level description of the event-driven jitter tolerance simulation

Based on the methodology, to confirm the proposed technique, a jitter tolerance simulation is done with 20,000 bits of 2^7-1 PRBS data including approximately 100,000 data samples, and 50 jitter frequency points are taken for the simulation. The jitter tolerance curve is shown in Figure 4.23. At starting point of 0.0001, which is the ratio of the jitter frequency to the baud rate, the peak-to-peak jitter tolerance is 93.75UI. The corner frequency point is at approximately 0.012. The peak-to-peak jitter tolerance at the high jitter frequencies is about 0.8UI as shown in the figure. The simulated jitter tolerance at the starting point, high jitter frequencies and the corner frequency are very close to the calculated values given in section 4.3, and the difference is mainly due to the calculation where the infinite frequency points, infinite data length, and so on, are assumed.

Figure 4.24 shows the jitter tolerance with different examining windows. From the figure, when EW = 8, 16, and 32, the jitter tolerance curves coincide with each other,

which means the jitter tolerance is almost the same since the jitter frequency is no larger than the corner frequency. By increasing the length of EW, the low-frequency jitter tolerance as well as the corner frequency is decreased as theoretically analyzed in Section 4.4.

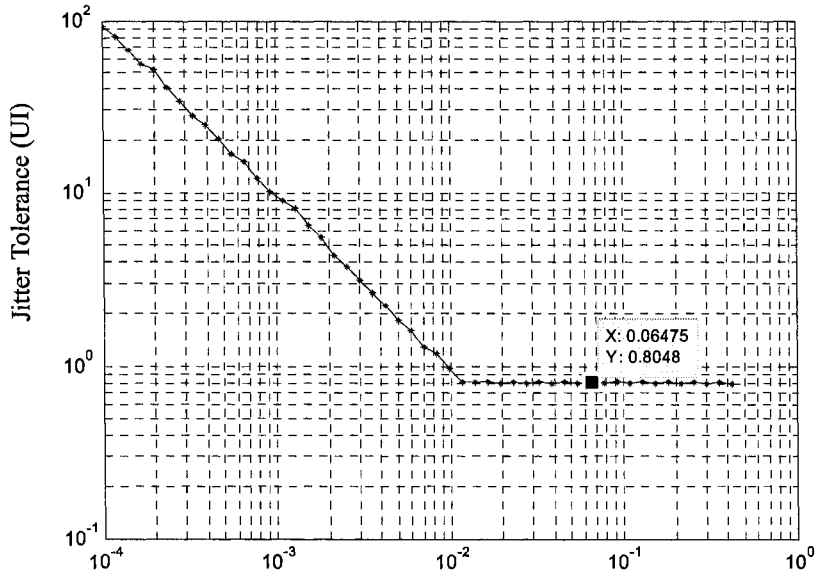


Figure 4.23 Jitter tolerance of the proposed CDR

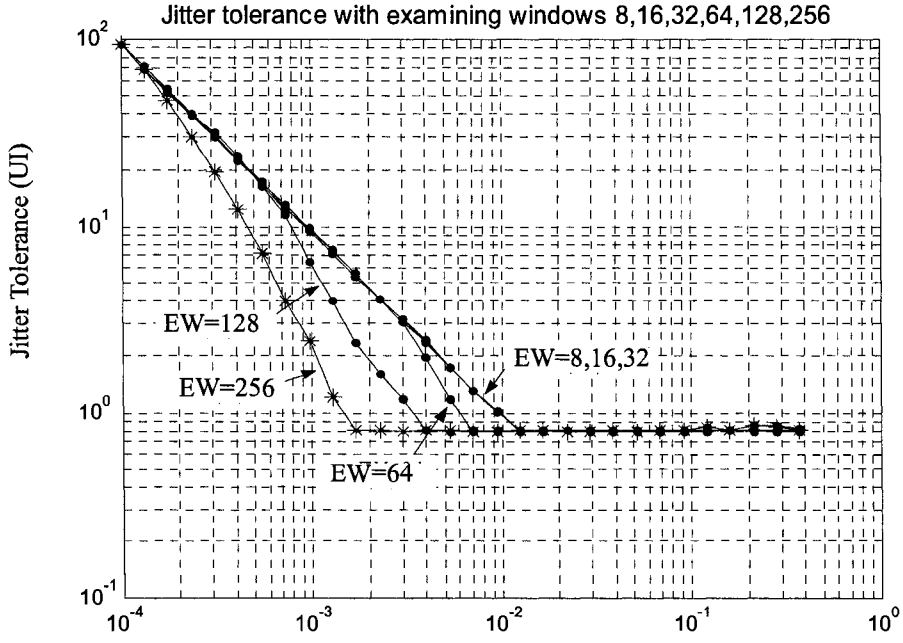


Figure 4.24 Jitter tolerance of the proposed CDR

4.9 Theoretical analysis of the re-aligned clock power spectrum

The re-aligned clock of the 5X oversampling CDR is obtained by dynamically selecting one out of the 5 phases of the multi-phase clocks. It is clear that the re-aligned clock may exhibit a large edge jitter corresponding to the phase difference between two consecutive phases. In this section, the power spectrum of the re-aligned clock is analyzed theoretically.

When the CDR clock frequency equals to the incoming data rate, the CDR is essentially an open loop system once the phase acquisition is done. So the re-aligned clock is always the same as one of the multiple-phase clocks and exhibits no extra edge jitter besides the phase noise related jitter. The same thing happens if the peak-to-peak data jitter is within 4/5UI at high jitter frequencies or within 3/5UI at low jitter frequencies because the CDR doesn't change the clock phase.

However, if the data rate *Fdata* and the CDR sampling clock frequency *Fcdr* are both constant but are not exactly equal, the CDR phase rotator continuously rotates the phase to track the incoming data. In this case, the re-aligned clock exhibits a periodic edge jitter, having a fundamental frequency f_{saw} equal to $N \times |Fcdr - Fdata|$, and has a peak-to-peak jitter of UI/5. To obtain the power spectrum of the re-aligned clock, the re-aligned clock is considered as an ideal clock at *Fdata*, and its phase is modulated by a sawtooth signal with a peak-to-peak amplitude of $\frac{2\pi}{5}$.

First, the sawtooth wave can be expressed as,

$$\phi_{saw}(t) = \frac{2\pi}{N} \cdot (f_{saw}t - floor(f_{saw}t + 0.5))$$

and it can be decomposed by using Fourier transform as,

$$\phi_{saw}(t) = \frac{2}{N} \cdot \sum_{n=1}^{\infty} \frac{\sin(n \cdot 2\pi f_{saw}t)}{n}$$

So the sawtooth signal has a fundamental component of $2/(N \cdot \sin(2\pi f_{saw}t))$, which modulates the phase of the ideal data clock and shows up as the first side spur in the power spectrum of the recovered clock. To simplify the analysis, assume the ideal clock is only modulated by the fundamental component of the sawtooth wave and the energy at the first side spur is thus calculated.

The modulated ideal data clock is

$$v(t) = A\cos\left(2\pi f_{data}t + \frac{2}{5}\cos(2\pi f_{saw}t)\right)$$

$$= A \cdot J_0\left(\frac{2}{N}\right)\cos(2\pi f_{data}t)$$

$$+A \cdot J_1\left(\frac{2}{N}\right)\cos(2\pi(f_{data}+f_{saw})t) + A \cdot J_1\left(\frac{2}{N}\right)\cos(2\pi(f_{data}-f_{saw})t)$$

$$+A \cdot J_1\left(\frac{2}{N}\right)\cos(2\pi(f_{data}+2f_{saw})t) + A \cdot J_1\left(\frac{2}{N}\right)\cos(2\pi(f_{data}-2f_{saw})t)$$

...

where $J_n(x)$ denotes the bessel function. The constant phase is eliminated from the above equation for simplification. Based on the above analysis, the power spectrum of the recovered clock has the first side spur at the frequency offset of f_{saw} and the relative power is $Rspur = 20\log(J_1(2/N)/J_0(2/N))$ dBc. For N=5, *Rspur* is calculated to be -14dBc.

All analysis so far is based on the assumption that the N phases of the multiple phase clocks are evenly spaced. However, there are always some mismatches among the multiple phases in the real circuit and some additional spurs appear at the power spectrum of the re-aligned clock. In this case, the spurs may show up at the frequency offsets of any integer multiple of the frequency difference between *Fdata* and *Fcdr*. Figure 4.25 illustrates the power spectrum of the recovered clock, in which Δf denotes the frequency difference between the data rate and the sampling clock.

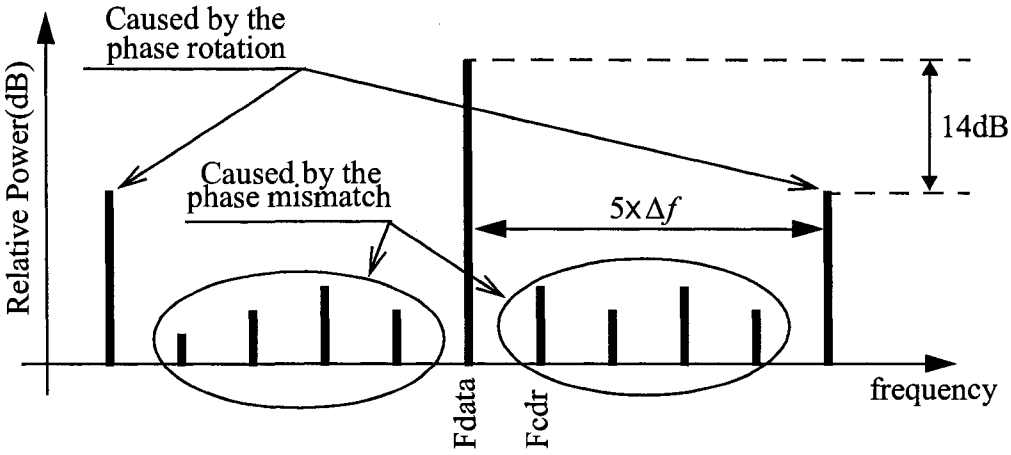


Figure 4.25 The power spectrum of the re-aligned clock

4.10 Summary

In this chapter, the proposed digital threshold decision CDR technique and architecture were presented, and the functionality of the proposed CDR as well as the theoretical jitter tolerance were verified. The theoretical analysis of the CDR with the digital threshold decision technique in the general parameters was first given, and design parameters were chosen based on that. The functionality was verified by the mathematical model simulation using Matlab, and its jitter tolerance curve was obtained by using an efficient jitter tolerance simulation methodology, the event-driven simulation. The power spectrum of the re-aligned clock was also given. Both of the Matlab model simulation and the event-driven simulation verified the functionality of the proposed technique, and it confirmed that the proposed CDR architecture can be used in some applications such as the SFI-5 interface.

CHAPTER 5

A Proposed DLL Based Frequency Multiplier

5.1 Introduction

The PLL has been widely used in the frequency synthesizer design for many years, and recently the DLL based frequency multiplier has been explored in this field because it has no phase noise accumulation and it is easy to integrate, unconditionally stable, and so on. In Chapter 3, several reported DLL based frequency multipliers were reviewed, and the main advantages and disadvantages were discussed. It was found that the output spurious power reduction and eliminating harmonic locking with low circuit complexity remain the main challenge in the DLL frequency multiplier design. In this section, the noise transfer functions of the PLL and the DLL are first compared. To reduce the resulting spurious power level, a DLL with a period error compensation loop was proposed to reduce the systematic error. Matlab models were built to verify the functionality and efficiency of the proposed DLL.

5.2 PLLs and DLLs phase noise comparison

The reference signal of the PLL-based frequency synthesizers usually comes from a low-jitter signal source such as a crystal oscillator, so the jitter in the output frequency mainly originates in the VCO, so a large bandwidth should then be employed to attenuated the VCO phase noise. Since PDs work on a discrete time basis, it is more convenient to

analyze the noise performance in discrete time domain than the continuos time domain.

A simplified discrete-time domain model for a PLL-based synthesizer incorporating a ring-oscillator is developed as shown in Figure 5.1 [40]. Mathematically, the internal noise of the ring oscillator is modelled as an equivalent voltage noise source ϕ_n on the control signal of the VCO. In the figure, ϕ_{on} denotes the jitter at the VCO output. In this section, the input signal is supposed to be an ideal signal and only the output jitter caused by internal equivalent noise ϕ_n is analyzed. The internal jitter of the VCO, $\phi_n(n)$ is accu-

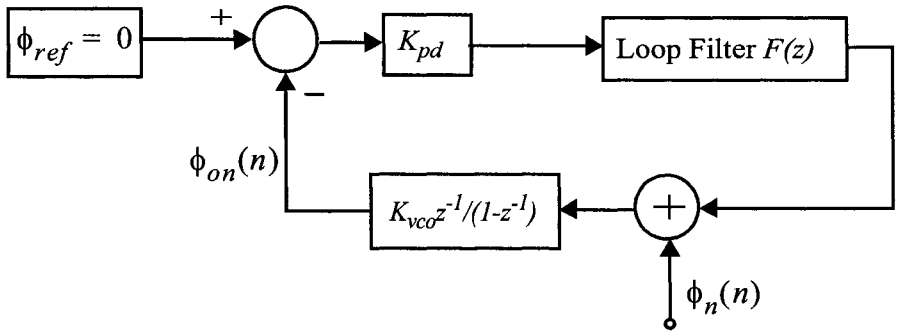


Figure 5.1 Simplified PLL discrete-time model for phase noise analysis

mulated over cycles. This would carry on infinitely given that there is no loop compensation. The jitter transfer function for PLL can be written in z-domain as

$$H(z) = \frac{\Phi_{on}(z)}{\Phi_n(z)} = \frac{z^{-1}K_{vco}}{1 - z^{-1} + z^{-1}K_{vco} \cdot K_{pd} \cdot F(z)} \tag{5.1}$$

where K_{pd} is the gain of the phase detector, $F(z)$ is the transfer function of the loop filter in z-domain and K_{VCO} is the gain of VCO in the discrete time domain.

From the equation above, the presence of K_{VCO} in the numerator suggests that an increase in the output jitter is at least possible when this parameter increases, but further examination reveals that this will not happen. The internal jitter of the VCO is modelled as an equivalent noise presented to the control signal of the VCO, and the relation between

the actual internal jitter and the equivalent jitter is,

$$\phi_{novco}(n) = K_{VCO} \cdot \phi_n(n) \tag{5.2}$$

where $\phi_{novco}(n)$ is the actual jitter of the VCO, which normally has constant power spectrum for the same VCO and under the same working environment. Thus the relation between the $\phi_{on}(n)$ and $\phi_{novco}(n)$ can be written as,

$$\frac{\Phi_{on}(z)}{\Phi_{novco}(z)} = \frac{z^{-1}}{1 - z^{-1} + z^{-1} K_{vco} \cdot K_{pd} \cdot F(z)} \tag{5.3}$$

So in fact, the output jitter will decrease while the K_{VCO} increases since it is at the position of denominator in the (5.3).

The main difference between the DLL and the PLL with a ring oscillator is that the noise at the output of the VCDL is not fed back to its input and the phase noise does not accumulate in the VCDL, so only the jitter produced in the present cycle is ever observed. Using the superposition principle, the discrete model for the noise analysis is developed as shown in Figure 5.2. The output of the VCDL, ϕ_o, is the delayed version of the reference

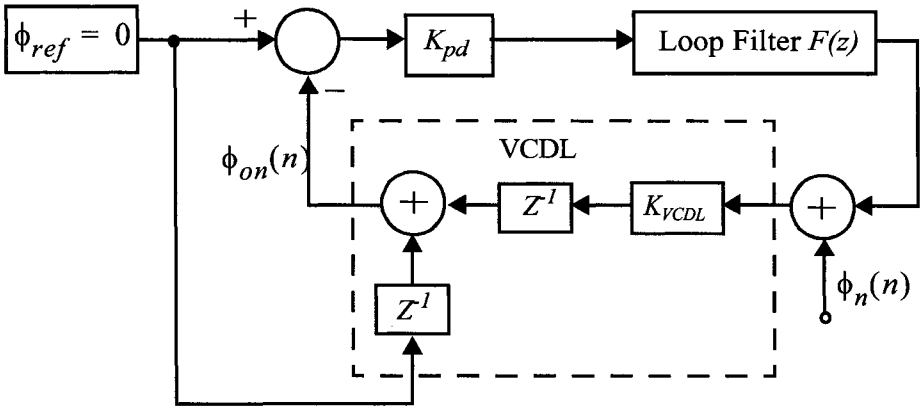


Figure 5.2 Simplified DLL discrete-time model for phase noise analysis

signal ϕ_{ref} itself after being delayed with the delay value tuned by the control signal. Similar in the previous PLL analysis, the internal jitter of the VCDL is modelled by an

equivalent noise ϕ_n applied to the control input.

The output of the VCDL is the sum of the previous phase (which is zero for noise analysis) and the delay which is the product of K_{VCDL} and the control voltage. For the sake of the noise analysis, the reference is always 0, the transfer function of the VCDL can be written as $H_{VCDL}(z) = z^{-1}K_{VCDL}$. Consequently the relation between the output jitter $\phi_{on}(n)$ and the internal equivalent noise $\phi_n(n)$ in z-domain is expressed as,

$$H(z) = \frac{\Phi_{on}(z)}{\Phi_n(z)} = \frac{z^{-1}K_{VCDL}}{1 + z^{-1}K_{VCDL} \cdot K_{pd} \cdot F(z)} \tag{5.4}$$

where K_{pd} is the gain of phase detector, $F(z)$ is the z-domain transfer function of the loop filter and K_{VCDL} is the gain of the VCDL in the discrete time domain.

On comparison with (5.1), the output jitter of a DLL is smaller than the output jitter of a PLL using the same loop filter and phase detector when $K_{VCDL} = K_{VCO}$. Although when K_{VCO} or K_{pd} is sufficiently large, the PLL can compensate the accumulated jitter very quickly, and the output jitter of a PLL is close to that of a 'similar' DLL, a PLL with a large loop gain normally results in a stability problem. A first-order DLL, on the other hand, is always stable regardless the loop gain. However, the output of DLL using an edge combiner normally exhibits higher spurs than that of a PLL due to the stage mismatch and the in-lock error, which are briefly analyzed in the following section.

5.3 Spur generation of the DLL frequency synthesizer

In DLL frequency multipliers, the spurious output power level is higher than that of the PLL based ones. The large spur is mainly resulted from the delay stage mismatch in the edge combiner based DLL multiplier and from the in-lock error in both types of DLL multipliers including the reference cyclic injected ones. Mathematical model simulation is done to analyze the spur generation in the DLL frequency multiplier in this section.

5.3.1 The principle of the edge combining DLL

To investigate the spur generation of the DLL, the principle the edge combiner based DLL is first analyzed mathematically in this section [41]. In an edge combiner based DLL frequency synthesizer, the output can be considered as the summation of N pulse signals having a small duty-cycle (approximately $1/(2N)$) and a frequency equal to the reference frequency. The fundamental frequency of each pulse is the reference frequency, and the Fourier transform of the nth pulse signal $X_n(kf_{ref})$ can be expressed as,

$$X_n(kf_{ref}) = X_0(kf_{ref}) \cdot e^{j2\pi f_{ref} t_d(n) k} \tag{5.5}$$

where k is an integer and $t_d(n)$ is the delay of the nth pulse signal relative to the first pulse signal ($t_d(0) = 0$). Ideally, the final output from the VCDL functions as an adder, and the Fourier transform of f_{out} is expressed,

$$X_{fout}(kf_{ref}) = X_0(kf_{ref}) \sum_{n=0}^{N-1} e^{j2\pi f_{ref} t_d(n) k} \tag{5.6}$$

where $X_0(kf_{ref})$ usually has non-zero values for all values of k. Ideally, all pulse signals are evenly spaced within one period of the reference T, so $t_d(n) = \frac{T}{N} \cdot n$, and the Fourier transform of the final output $X_{out}(k)$ is,

$$X_{out}(k) = X_0(k) \cdot \sum_{n=0}^{N-1} e^{j2\pi \frac{nk}{N}} = \begin{cases} N \cdot X_0(k) & k = N \cdot m \\ 0 & k \neq N \cdot m \end{cases} \tag{5.7}$$

where m is an integer, and $X_{out}(k)$ is constant for k equal to N or its integer multiple, and zero for all the other values. So the fundamental frequency of f_{out} is N times the reference frequency. However in the real circuits, the output spectrum has non-zero values for

$k \neq N \cdot m$. The power ratio of the component at $(N-1)f_{ref}$ (or at $(N+1)f_{ref}$ whichever is larger) to the centre component at Nf_{ref}, $R'_{spur} = \left|\frac{X(N-1)}{X(N)}\right|^2$, is defined as the reference spur level and usually expressed in dBc. To simplify the analysis, the term $X_0(kf_{ref})$ in (5.5) is assumed to be constant for $k = N\text{-}1, N, N\text{+}1$, and in most cases, the two sideband spurs are almost at the same power level.

5.3.2 *Spurs caused by the stage mismatch*

With the above analysis, spur generation due to the mismatches among delay stages and the in-lock error can be analyzed. The DLL based frequency synthesizer with edge combiner in [30] [31] [32] [33] is susceptible to the mismatch among the delay stages in the VCDL, and the delay variation shows up in the final output as timing jitter in the time domain and spurious output in the frequency domain.

This section analyzes the spur generation due to these mismatch based on the assumption that the reference signal is ideal and the in-lock error is zero. The delay variation of n delay stages is denoted as $\Delta D(n)$. To simplify the analysis, $\Delta D(n)$ is supposed to be linear and can be expressed as,

$$\Delta D(n) = nTk_r + \Delta D_0 \tag{5.8}$$

where k_r is the relative slope of the delay variation, or the ratio of the slope to the reference period, and since the average value of $\Delta D(n)$ $(0 \le n \le N-1)$ is zero, the term ΔD_0 in the equation is $-(N-1)Tk_r/2$.

Thus the total delay value at each stage $t_d(n)$ can be expressed as,

$$t_d(n) = \frac{T}{N} \cdot n + \sum_{i=0}^{n} \Delta D(i) \tag{5.9}$$

where n is the number of delay stages. By substituting (5.8) into (5.9),

$$t_d(n) = \frac{T}{N} \cdot n + \frac{n^2}{2} \cdot Tk_r - \frac{N}{2} \cdot nTk_r \qquad (5.10)$$

The phase alignment error of those multiple-phase pulses is shown in Figure 5.3. Although the stage delay variation is linear, the largest phase alignment error occurs at the

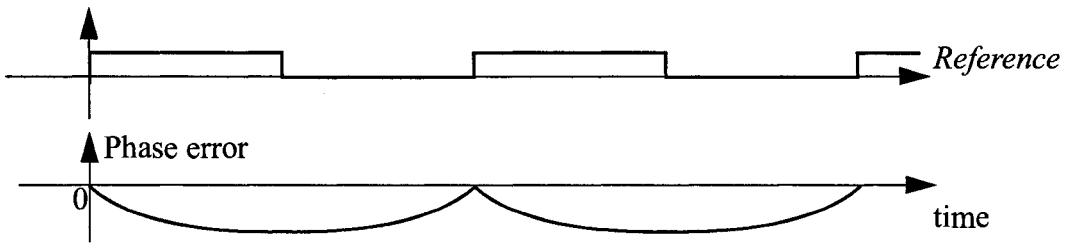


Figure 5.3 Phase alignment error

middle of the delay line if the in-lock error is zero. The relative power spectrum of the output can be calculated approximately. Figure 5.4 shows a relative power spectrum with N = 11 and k_r = 0.001. The desired frequency component ($f_{out} = 11 \cdot f_{ref}$) is at k = 11. Two sidebands appear at k = 10 and k = 12, so due to the mismatch in the delay stages, spurious output is resulted.

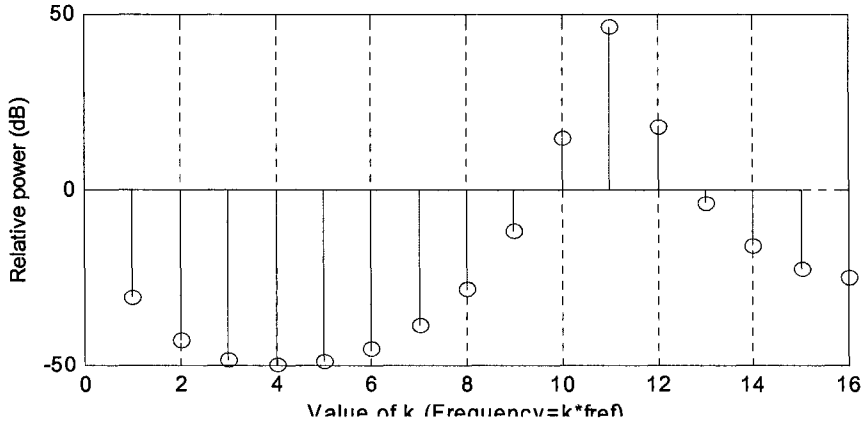


Figure 5.4 Illustration of the spurious output

5.3.3 Spurs caused by the in-lock error

Besides the stage mismatches, the in-lock error may also cause some spurs at the output. The stage mismatches are assumed to be zero and the spurs caused solely by the in-lock error is analyzed based on [42].

Because the edge jitter is reset at each reference edge, the accumulated edge jitter within each reference period is always zero if the reference signal has no jitter. Consequently, the in-lock error appears at the nth pulse signal, also having its negative error evenly distributed over all the other pulse signals. So the delay of each pulse signal output can be expressed as,

$$t_d(n) = \left(\frac{T}{N} + \Delta p_s\right) \cdot n \tag{5.11}$$

where $t_d(n)$ is the delay of nth pulse signal with respect to the first pulse signal; T is reference period; N is the multiplication ratio and Δp_s is a single period variation. Consequently, according to (5.6), the spurious power when $n = k$ is,

$$S(k) = \sum_{n=0}^{N-1} e^{j2\pi f_{ref}\left(\frac{T}{N} + \Delta p_s\right) \cdot nk} = \frac{1 - e^{j2\pi k} e^{j2\pi f_{ref} \Delta p_s \cdot Nk}}{1 - e^{j2\pi f_{ref}\left(\frac{T}{N} + \Delta p\right)k}} \tag{5.12}$$

A sketch of the relative output power spectrum may be produced by calculating the value of $S^2(k)$ for different k. As an example, Figure 5.5 shows a calculated power spectrum for $N = 10$ and $\Delta p / T_{ref} = 0.015$.

Using Δp_r as a symbol for the ratio $(\Delta p)/T_{ref}$, where Δp is the output period variation, the output spurious power level is plotted as a function of division ratio N in Figure 5.6. Obviously the spurious power level increases along with the relative variation and for a given relative variation, and large number of division ratio also results in larger

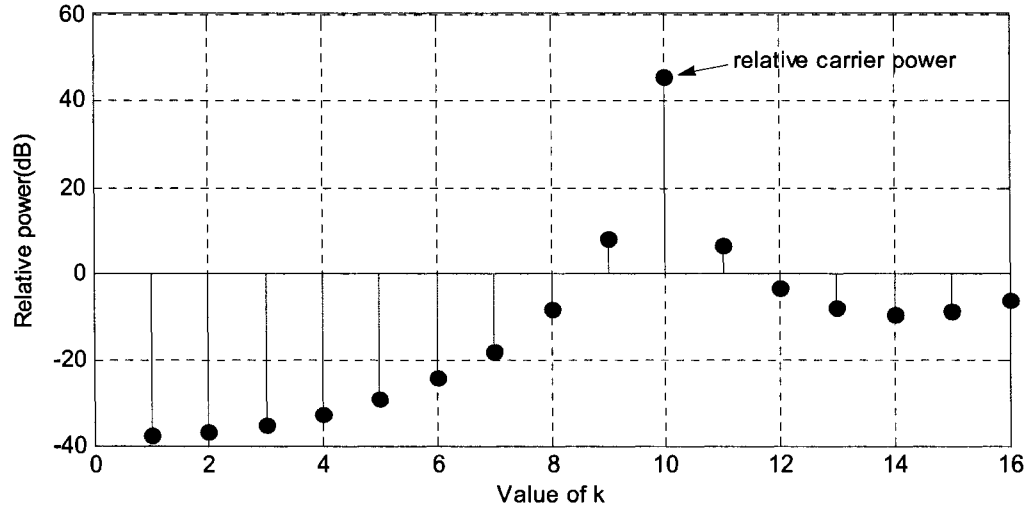


Figure 5.5 Relative power spectrum

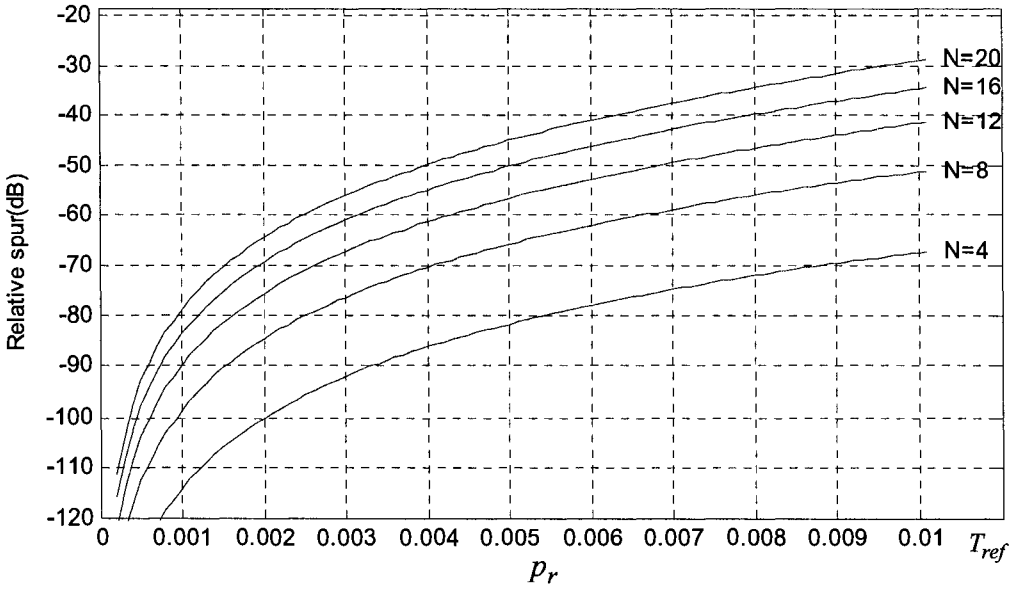


Figure 5.6 Relation of spurious power level to the in-lock error

spurious output level.

5.4 The proposed DLL with period error compensation

5.4.1 The proposed DLL architecture

Above analysis unveils the two fundamental sources of the spur generation, the

stage mismatch and the in-lock error, which need to be minimized so that the spurs at the DLL output can be effectively reduced.

Besides the edge combiner based DLLs, cyclic reference injected DLLs based on a VCO-like VCDL appear in the literature [32]. The stage mismatches are minimized in cyclic reference injected DLLs because the reference signal always circulates in the same delay stage. However, the phase re-alignment errors, in the process of injecting the clean reference edges into the VCDL, might even enlarge the total equivalent in-lock error, and they cannot be effectively minimized by optimization of the charge pump (CP) and the phase and frequency detector (PFD).

This section describes the architecture of the proposed cyclic reference injected DLL frequency multiplier with a period error compensation loop (PECL) as shown in Figure 5.7, which minimizes both the stage mismatches and the in-lock errors for the spurs

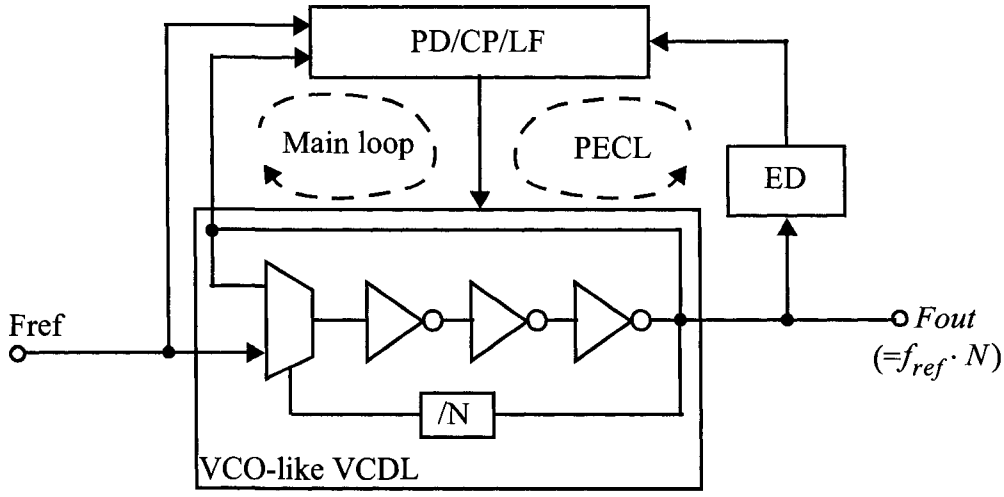


Figure 5.7 The proposed DLL frequency multiplier with a compensation loop

reduction. As shown in Figure 5.6, the proposed DLL consists of the main loop which is similar to a conventional DLL and a phase error compensation loop for reduction of the in-lock error. In the main loop, a VCO-like VCDL works as a VCO after the reference signal is injected into the delay line via the MUX, and the injected reference circulates in the inverting delay cell for N cycles. The delayed output after N output cycles is compared

with the reference signal by the phase detector (PD). The resulting phase error is filtered by the charge pump loop filter to tune the VCDL delay. In the period error compensation loop, the output signal (*Fout*) is applied to an error detector (ED) to detect the in-lock error and the output of the error detector is used to compensate the main loop so that the in-lock error is minimized. Although various sources of the in-lock error exist, all can be compensated by tuning zero offset of the PD or the zero offset of the charge pump.

To guarantee the proper operation, a switching logic circuit, which is not shown in Figure 5.7, is required to provide precise timing information for the MUX switching, the phase detection, the error detection and the divide-by-N divider. A new switching scheme is employed in the coordination of the operation among the blocks mentioned above to avoid harmonic locking without using circuit initialization and extra lock detection circuitry. The detailed operation of the switching logic circuit is given in "The divider and the switching logic" on page 126.

5.4.2 System phase error in a cyclic reference injected DLL

In this section, the system static error in a cyclic reference injected DLL is investigated. The cyclic reference injected DLL is a modified conventional DLL with the VCDL functioned like a VCO by periodically injecting the reference into the VCDL to eliminate the stage mismatches. At a given PVT condition, the system error is normally constant, and it may result in large spur as described in the Section "Spurs caused by the in-lock error" on page 83. There are a few of phase error sources in the cyclic injection based DLL frequency multiplier, mainly including the static phase error caused by the current mismatch between the *UP* and *DOWN* signal paths in the PD, and the charging and discharging current in the charge pump $\phi_{mn}(n)$ as shown in Figure 5.8, the reference phase re-alignment error $\phi_{rn}(n)$ when the reference edges are injected. All these phase errors are system static error, which can be concluded as the in-lock error $\phi_{in}(n)$ when the loop

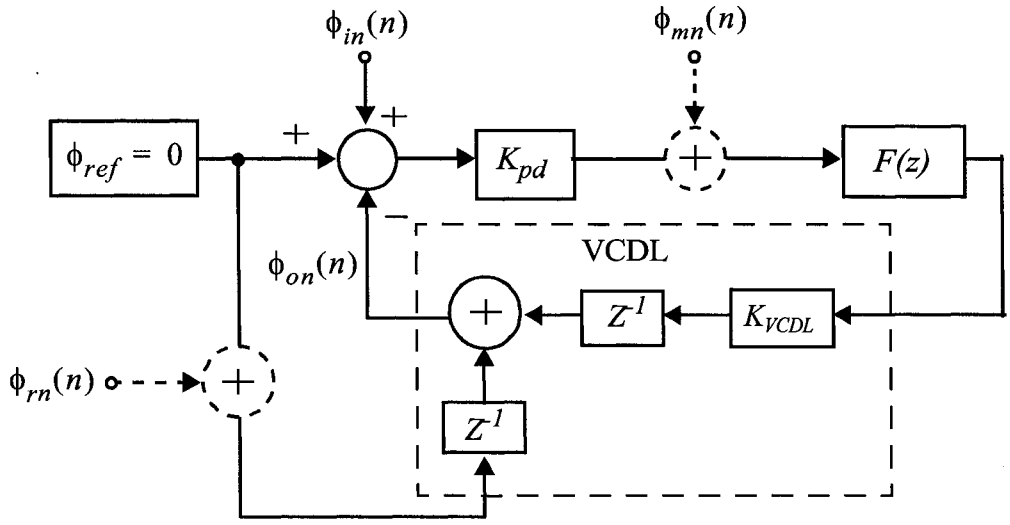


Figure 5.8 Simplified DLL discrete-time model for system static error analysis

is in lock, and this in-lock error may result in large spur in the output.

An example of the multiplication ratio of 9 is shown in Figure 5.9, within one ref-

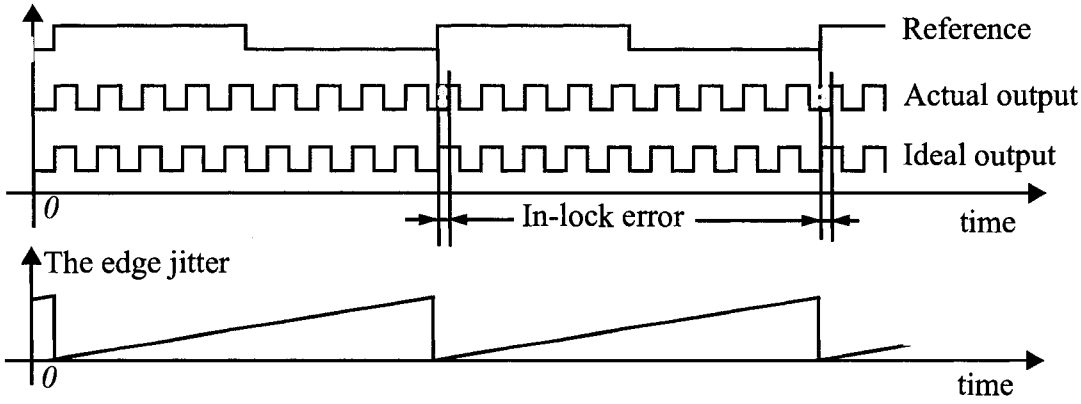


Figure 5.9 In-lock error

erence period, ideally there are 9 cycles of the high frequency signal with equal periods. However due to the in-lock error, there is one cycle with the period that might be smaller (as shown in the example) or larger than the ideal period as indicated in the shaded area in the figure. As a result of the in-lock error, the edge jitter increases linearly until a new reference edge injected. A period error compensation loop is employed in the proposed DLL

to reduce the in-lock error resulting from various sources, such as the phase injection error the static error from the mismatch in the PD and CP combination.

5.4.3 The period error compensation and model simulation

To minimize the in-lock error, an error detection circuit is employed in the proposed DLL as the key component of the period error compensation loop as illustrated in Figure 5.7. The operation principle of the error detection is shown in Figure 5.10, where $T_{out}(n)$ is the nth output period, and G_1 and G_2 is two gain stages with similar value. G_1 is used to amplify the period error between the nth output period and the average period, so the error can be large enough to be identified by the followed circuit blocks. G_2 is used to decrease the error before it is integrated by the integrator, and the nth output period is deducted by the integrated value to extract the error signals which are again amplified and then attenuated. This procedure is repeated for many times to obtain the ideal average output period, and by subtracting this error from the current output period, an instantaneous period error is detected. Since the main in-lock error is considered to appear in the last out-

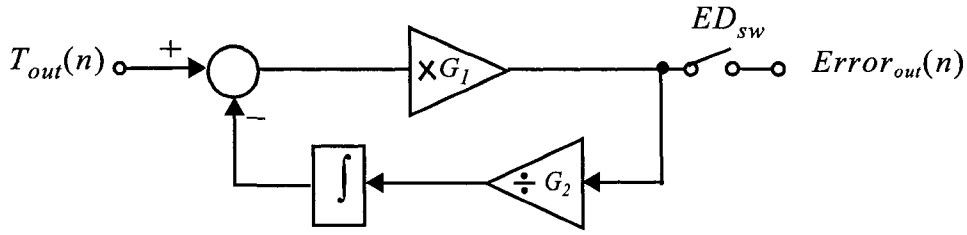


Figure 5.10 Principle of error detection

put period within one reference cycle, this error detection is controlled by a control signal *EDsw* so that the nth period, which may have a different period from all other periods due to the in-lock error, is compared with the average period for the error detection.

To verify the efficiency of the period error compensation loop, a top-level model consisting of a mathematical DLL, an error detector and an integrator is built as shown in Figure 5.11. The DLL model produces an output period signal $T_{out}(n)$ corresponding to a

certain in-lock error defined as its input (*Error*). $T_{out}(n)$ and a switching signal (ED_{sw}) are applied to an error detector to detect the in-lock error. The initial in-lock error is shown in *constant1* and it is subtracted by the integrated error signal. The subtracted value is applied to the DLL model to adjust the in-lock error.

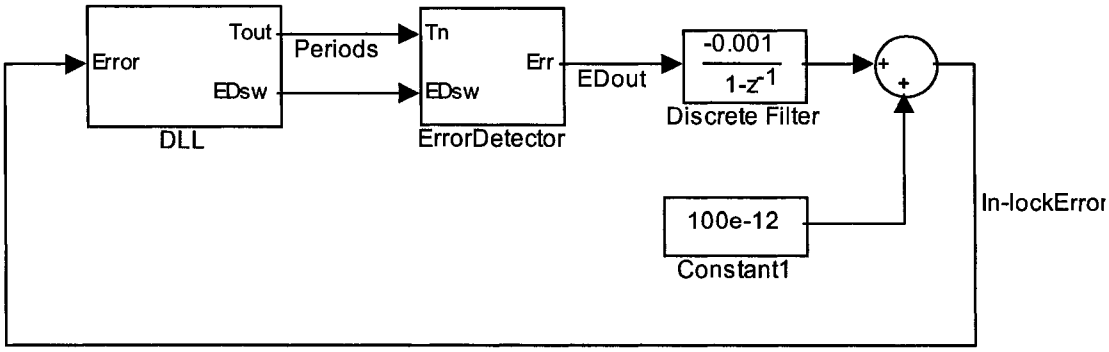


Figure 5.11 Top level model of error detection

The DLL model is used to generate the output signal periods, with an in-lock error specified its input (*Error*), and the switching signal for the error detector as shown in Figure 5.12. The constant '*1*' is integrated first until *N* times is reached, and at this point,

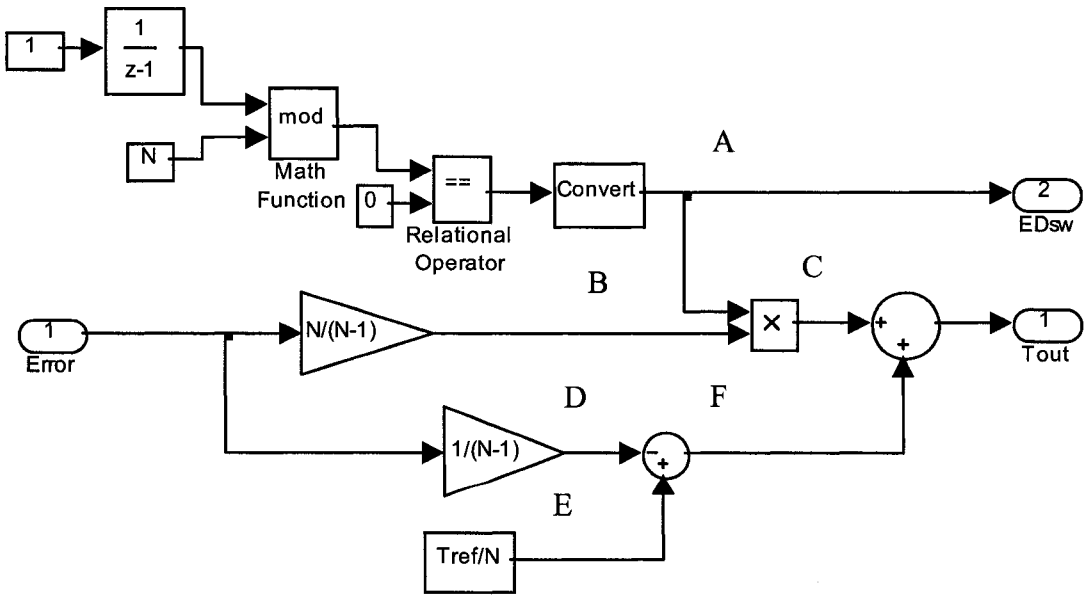


Figure 5.12 Mathematical equivalent DLL model

the *mod* function block outputs values from 0 to *N-1*. The *mod* output is compared with '0'

in the relational operator to generate either '1' or '0' at node A, indicating where the in-lock error appears. When node A or *EDsw* is '1' at the *N*th output period, the error detector is enabled, and when it is 0, the error detector is disabled during the other output periods.

The error signal at terminal *1* determines the in-lock error of the DLL, and the in-lock error is assumed to be evenly distributed over *N-1* cycles with the *n*th cycle having larger or smaller period value. The output at node E is the ideal output period, while output at D is the average period error at *N-1* output cycles. Consequently, the output at node F is the periods of the *N-1* output cycles. Node B generates the output period at the *n*th output cycle which has the different period values from the other output cycles, by multiplying 1 or 0 from node A at the reference frequency, the *n*th output period value is obtained at node C. The final desired T_{out} is generated by summing the *N-1* output periods and the *n*th output period.

The mathematical model of the error detector shown in Figure 5.13 is very similar to the theoretical one. In the simulation, G_1 and G_2 are both 50. By taking the reference period of 6ns, a division ratio of 20, and the initial in-lock error of 100ps, the simulation results from the mathematical model simulation of the whole loop is shown in Figure 5.14.

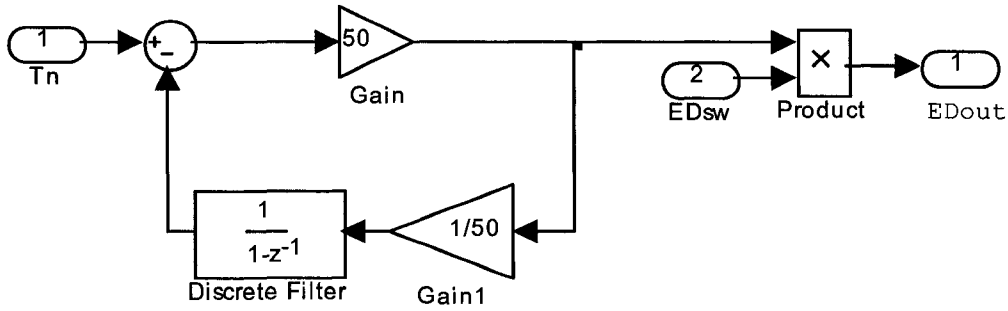


Figure 5.13 Mathematical model of the error detector

In the loop acquisition process, the output of the error detector (*EDout*) is reduced while the in-lock error and the difference of the DLL output periods are reduced gradually. When the loop is in lock, and all the periods are 300ps, and the in-lock error is zero, con-

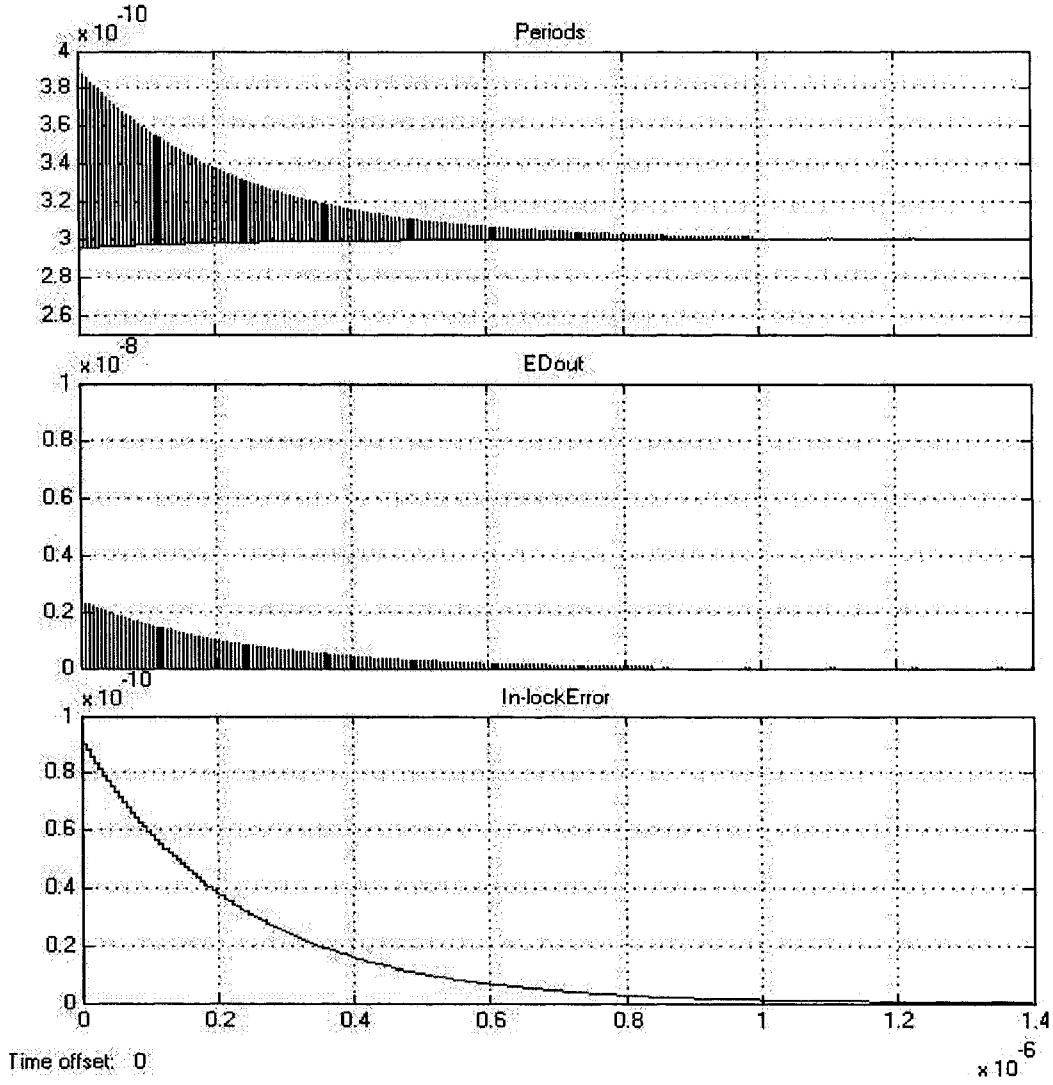


Figure 5.14 The DLL and error detector outputs

firming the efficiency of the proposed error detection and compensation technique.

5.5 Summary

In this chapter, the noise performances of the PLLs and the DLLs were first compared using their z-domain transfer functions. Although DLLs normally exhibit a better phase noise performance, they suffer the problems of stage mismatch and the in-lock error, which result in spurious output. After the analysis of the relation of the mismatch and the in-lock error to the output spur, a cyclic reference injected DLL with a period error compensation loop for spur reduction was proposed. Finally, the efficiency of the period

error compensation loop was confirmed by Matlab model simulation.

CHAPTER 6

Implementation of the CDR and the DLL Frequency Multiplier

6.1 Introduction

In this chapter, the implementations of the CDR and the DLL based frequency multiplier are described in details. In the CDR test chip, besides the CDR core circuit, an on-chip 2^7-1 PRBS for data pattern generation, a BER test circuit and an on-chip multi-phase clock generator based on a DLL were implemented. Design considerations with respect to the jitter tolerance performance, the operation speed, the power consumption and so on are also discussed in this chapter. The implemented DLL based frequency multiplier mainly consists of a phase detector, a charge pump, a loop capacitor, a VCO-like VCDL with an MUX, a divider and a switching logic. Design details of the frequency multiplier and the design considerations regarding to the phase noise and output spur performance are also given in this chapter.

6.2 CMOS Implementation of the proposed CDR

The CDR circuit was implemented in 90nm CMOS technology, and the system-level architecture is shown in Figure 6.1. The CDR consists of a phase detector, a rotating clock generator, a phase rotator, and two multiplexers to generate the recovered data and re-aligned clock. The five clock phases are generated from a DLL based multiple phase clock generator with the reference from a low-jitter external source. The data applied to

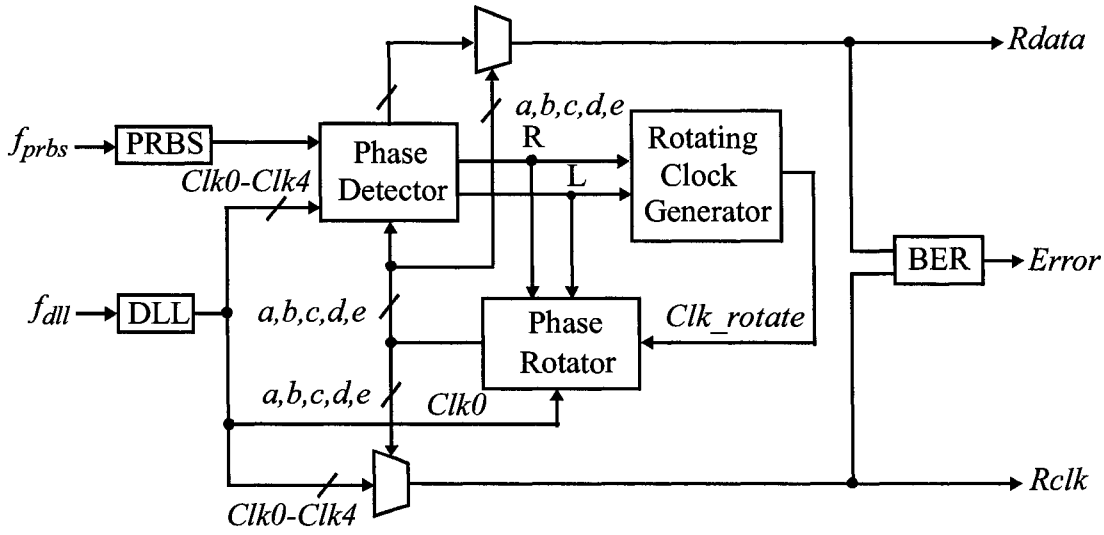


Figure 6.1 System architecture of the implemented CDR

the CDR is generated by the on-chip PRBS 7 data generator clocked by an external clock source. The recovered data *Rdata* and re-aligned clock *Rclk* are applied to an on-chip BER test circuit to achieve bit error detection, which is required for the measurement of the jitter tolerance performance.

In the previous chapter, the jitter tolerance performance of the proposed CDR is obtained by assuming the CDR works in an ideal environment, where the sampling clock is jitter-free, and the samplers and the logic gates in the models are ideal and so on. However, the jitter tolerance performance is degraded in the real circuit design, due to mismatches among multiple phase clocks, the parasitics in the CMOS transistors and the PVT conditions. Consequently, implementation considerations are taken at some critical points on enhancing the jitter tolerance, meeting the targeted operation speed and reducing the power consumption in order to obtain the desired performance. In the followed sections, detailed implementation for each blocks shown in Figure 6.1 will be given.

6.2.1 The phase detector

In the phase detector, the data is first sampled by five clock phases *Clk0*, *Clk1*,

Clk2, *Clk3*, *Clk4* and data transition information can be detected by five XOR gates which output the transition signals, *Ta*, *Tb*, *Tc*, *Td*, and *Te*. The one-hot coded phase selection signal, (*a b c d e*), from the phase rotator selects the clock phase out of the five sampling phases as the data sampling phase for data recovery. Signals, *a*, *b*, *c*, *d*, *e*, corresponds to the clocks *Clk0*, *Clk1*, *Clk2*, *Clk3*, *Clk4*, coming from the multiple phase clock generator respectively.

According to the proposed CDR decision technique, if the one-hot signal selects the data sample that is immediately before the data transition, the one-hot phase selection signal will be rotated to the left by 0.2UI. In other words, the current data sampling position should be advanced. Similarly, if the data sampling phase position is at the data sample that is right after the data transition, the phase selection signal should be rotated to the right or the data sampling position should be retarded.

For example, when the transition is *Ta*, and phase selection signal *a* is high, the phase rotator is rotated to the left by one phase step according to decision technique. While if the data transition is found to be elsewhere, no phase rotating is performed. Similarly, when *a* is high, and only when the data transition is found to be *Te*, the one-hot phase selection signal is rotated to the right, or data sampling clock phase is retarded. It is similar to other cases as illustrated in Figure 6.2.

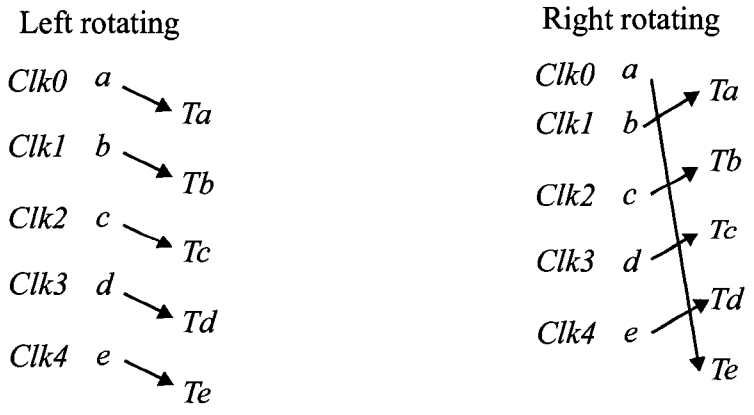


Figure 6.2 Conditions of left rotating and right rotating

The left rotating and right rotating can also be expressed as

$$L = a \cdot Ta + b \cdot Tb + c \cdot Tc + d \cdot Td + e \cdot Te$$

$$R = a \cdot Te + b \cdot Ta + c \cdot Tb + d \cdot Tc + e \cdot Ta$$

where L indicates an left rotating action in the phase rotator, while R indicates a right rotating. According to the equations, ten AND gates and two five-input OR gates are needed to perform the functions required in the phase detector. At this point, the phase detector implemented in transistor-level is as shown in Figure 6.3. However, this phase

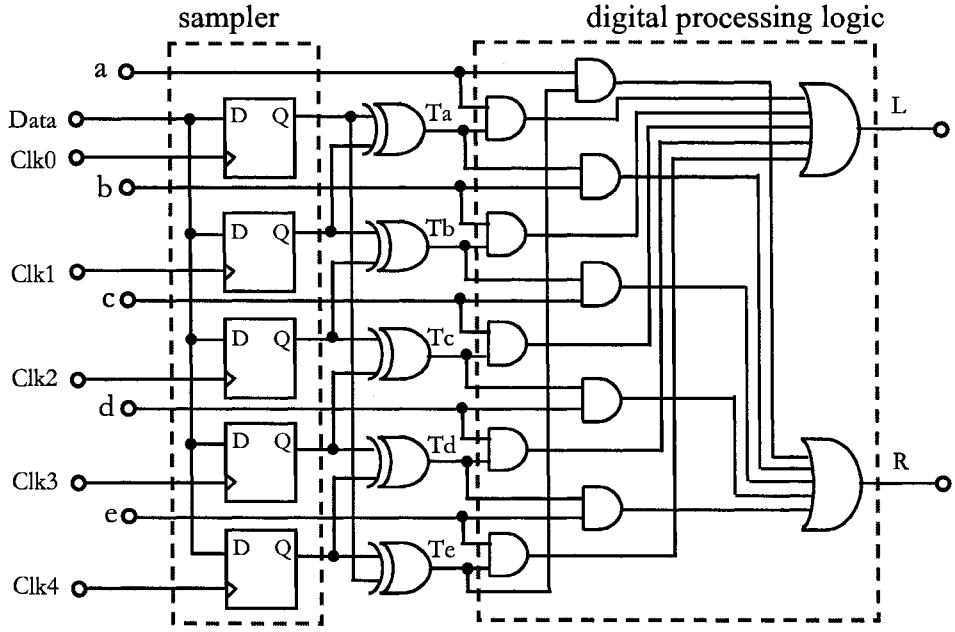


Figure 6.3 Functional block diagram of the phase detector

detector is still redundant. To reduce the complexity and the power consumption incurred, the set of digital logic gates that process the five transition signals and the phase selection signal can be replaced by an analog circuit with the AND and OR functions combined together as shown in Figure 6.4. From the figure, the five bits of the one-hot phase selection signal control five NMOS transistors functioning as switches respectively to turn on and off the current in the five differential pairs. The adjacent NMOS transistors in the five differential pairs share the same data transition signal. The output signal R or L is pulled

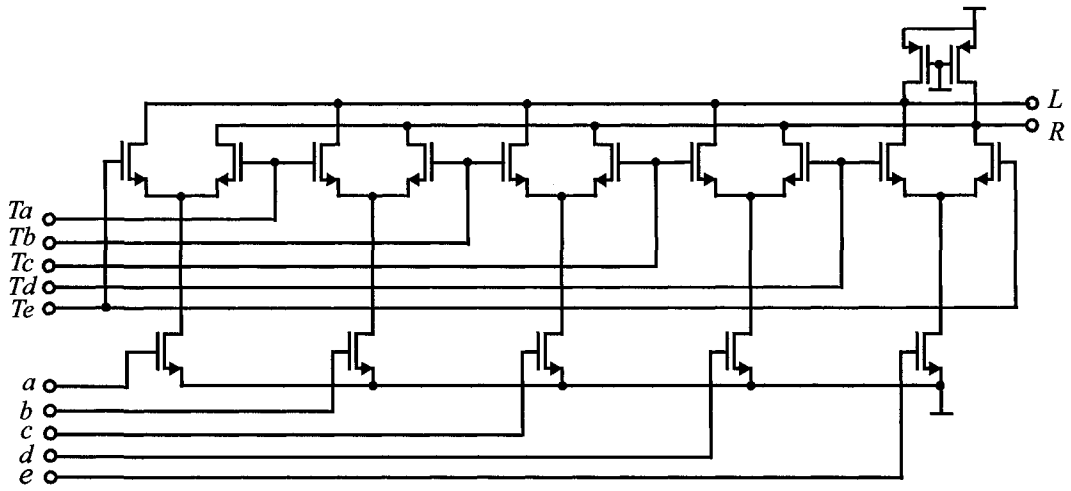


Figure 6.4 R, L-generating logic gate in the PD

low to enable a phase rotating when the one-hot coded phase selection signal turns on the differential pair at a high transition signal, and both are kept high when the data sampling position does not need to be updated. The size of both PMOS transistors are large for large pull-up impedance. Thus the 60-transistor set of digital logic gates is finally reduced to a 17-transistor logic gate.

The phase detector given in Figure 6.3 is modified as shown in Figure 6.5. The outputs of the XOR gates should be one-hot coded signal with a single pulse too, in other words, only one output among the five outputs is high within one data transition period. However, additional pulses appear at the data transition position from the XOR gates because of the clock skew among multiple phase clocks. Fortunately, these unwanted pulses can be removed with little extra complexity by adding a three-input AND gate with one input inverted as shown in the modified phase detector. Consequently, only one pulse appears at the five outputs of the five AND gates at one data transition position. The elimination of the unwanted pulses is illustrated in Figure 6.6. Signals *D0* and *D1* are sampled by *Clk0* and *Clk1* respectively, and *X1* is the XOR output with the unwanted pulses which appear even if there is no data transition. With the AND gate, the data transition signal *Ta* is obtained only at the data transition. As shown in the figure, there are two transitions in

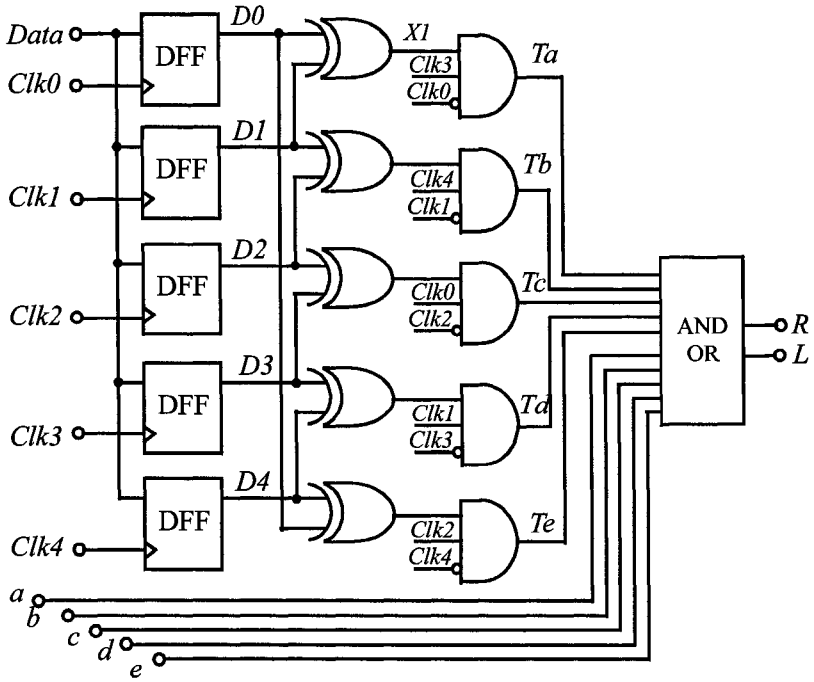


Figure 6.5 Schematic of the phase detector

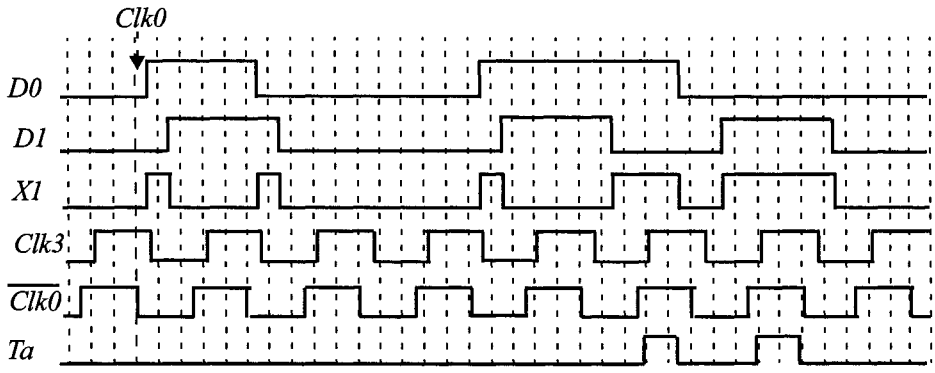


Figure 6.6 Timing of eliminating the unwanted pulse

the data, so there are two pulses in *Ta*.

By employing the low-complexity *R*, *L* generating logic gate, the delay in the PD is reduced significantly. The loop latency can be further reduced by applying rotating signals *R* and *L* to the phase rotator block directly from the PD instead of from the decision logic (rotating clock generator in this design) as reported in [28]. In addition, it can also be further reduced significantly by employing an asynchronous architecture instead of syn-

chronous one reported in [28] and by using multiplexers outside of the loop to generate the recovered data and the re-aligned clock.

6.2.2 *The samplers of the incoming data*

The performance of samplers, which sample the incoming data at the rising edges of the multiple-phase clocks, affects the jitter tolerance of the CDR. This section first reviews some parameters of a D flip-flop, followed by the impact of those parameters on the CDR performance. Finally, the selection of different types of D flip-flops are given together with the transistor level simulation result.

Two well known parameters of D flip-flops are setup time and the hold time, which are defined as the threshold values of the data to clock delays. The sampled result of a D flip-flop may be unpredictable or may not be what was expected when the edges of the data and the clock happen at the same time, which is called metastability. In most circuits, the metastability is avoided by ensuring the data is held constant for a certain time before or after the clock edge, the minimum required time period for such purpose is defined as the setup time and the hold time respectively. However, for the samplers of the CDR, there is no way to meet such a timing requirement because any data to clock delay may happens.

When the data edge appears between the two threshold values defined by the setup and hold time, the output of the sampler actually depends on its previous output as illustrated in Figure 6.7. Such characteristic is called hysteresis, which is usually needed to be

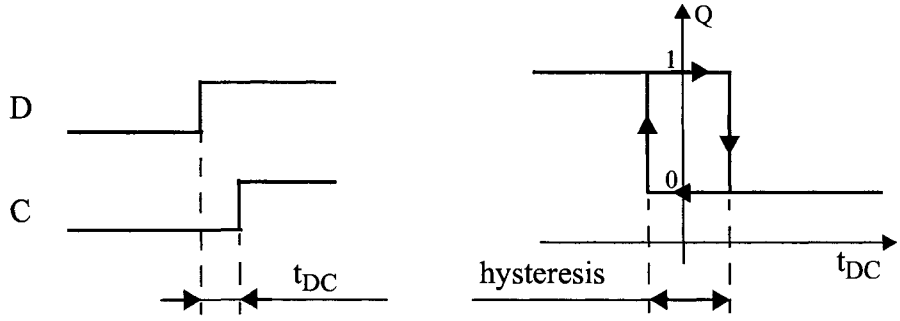


Figure 6.7 Illustration of the hysteresis of the DFF

minimized in most asynchronous applications.

Usually, such hysteresis is approximately the time period between the setup time and the hold time. Although the setup time and the hold time individually do not affect the CDR performance as long as they are constant, the difference between them does degrade the jitter tolerance performance, which is similar to an additional edge jitter of the multiple phase clocks in the same amount. Theoretically, the delay between the clock and the sampled output may affect the stability of the CDR loop because it actually reduces the phase margin. However, the proposed CDR is essentially a first order system for the stability point of view, and the stability margin is far larger than the effect of the clock to output delay.

Based on the above analysis, the main criteria of the selection of the D flip-flops for the proposed CDR is the hysteresis performance. For a 3GHz 5X oversampling CDR, the space between each two consecutive samples is 66.7ps and the hysteresis of the D flip-flops must be far smaller than such a time period so that the performance degradation caused by the hysteresis of the samplers is insignificant.

Normally, the dynamic DFFs possess smaller hysteresis than the static ones. Since the incoming data might be '1' or '0' for consecutive periods, the dynamic DFFs might lose data information due to leakage, thus only static DFFs are considered in this design. Several types of DFF were simulated to find one structure with the minimum hysteresis under the same simulation condition, and it is found that the conventional static DFF comprising of inverters and pass gates has the minimum hysteresis in the simulated static DFFs including sense amplifier based DFF. Figure 6.8 shows its simulated hysteresis. With careful transistors sizing, the pk-to-pk value is 4.7ps, compared with the sense-amplifier based DFF which has the pk-to-pk value of 11.3ps from simulation under the same simulation conditions as shown at top of the figure, and it is even well comparable to a dynamic DFF with a hysteresis of 3.0ps used in a timing generator in [44].

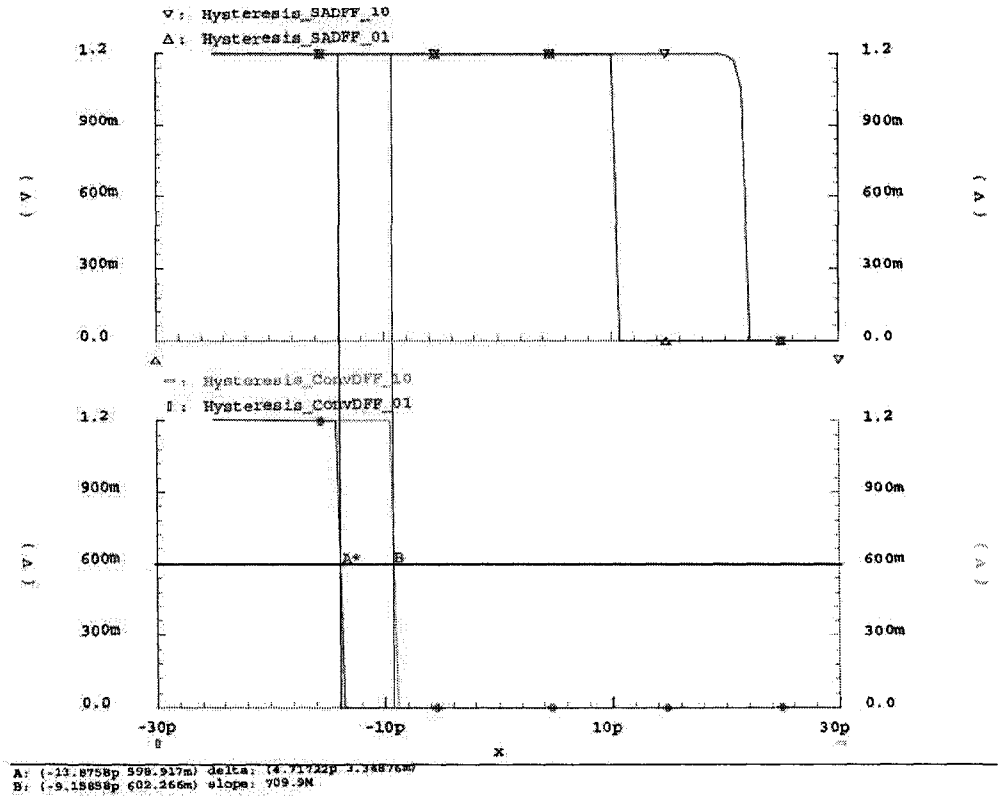


Figure 6.8 Hysteresis of the DFF in the PD

6.2.3 The rotating clock generator

The sampling position is not always updated when the *R* or *L* is generated from the PD. The decision circuit is used to output a rotating clock signal to determine when a clock phase rotating is needed to update the data sampling phase. In the circuit, two 3-bit counters are employed to memorize the location of the last *R* and *L* pulses from the PD within the EW. The outputs of the counters always indicate if there is any *R* or *L* ever appearing in the previous 8 baud periods, the examining window. According to the decision technique elaborated in Chapter 4, if there is a *R* (or *L*) pulse in the current baud period, and no *L* (or *R*) exists in the previous eight baud periods, the data sampling position should be rotated to the right (or left), or it should be retarded (or advanced). Consequently, whenever R or L pulse appears, the rotating clock generator outputs a clock pulse *Clk_rotate*, by examining the outputs of the counters, to trigger a phase rotating operation

in the phase rotator.

Because a low complexity binary ripple counter is not suitable for the high-speed operation, a parallel 3-bit counter with set or reset function is employed in this work as shown in Figure 6.9. Taking the upper branch consisting of DFF1, DFF2, and DFF3 as an example, on coming of a *R* pulse, DFF1 is set to 1, while DFF2 and DFF3 is reset to 0, resulting a low output at the three-input NOR gate and a low input at DFF1 for the new state transition. Thus the initial state of the three DFFs is '100', and on arriving of next clock signal, the state is updated to '010'. With the three feedback NAND gates, the next six states are '101', '110', '111', '011', '001' and '000' respectively. Once the state turns to be '000', counting-to-eight is reached, and the state of the three DFFs is locked in state '000' until the next *R* pulse arrives to set the state into the initial one '100' to repeat the counting process. By employing the DFF with set function as well as DFFs with reset function and applying both the positive and the negative DFF outputs to the three-input NAND gates, the circuit complexity is reduced. The lower 3-bit counter mainly consisting of DFF5, DFF6, DFF7 and three more NAND gates operates exactly the same way as the upper one does.

The output of the three-input NOR gate remains high and a low pulse is generated only when there is '1' in the state, indicating there is a *R* pulse (or an *L* pulse for the bottom branch) in the previous eight baud periods, and DFF7 is triggered to pass the NOR gate output, and if the NOR gate output is high, which means there is no *R* pulses in the previous eight baud periods, a high pulse is generated at the DFF7 positive output. The pulse width is determined by the delay of the feedback with a reset path to the DFF itself, and a data sampling phase updating is thus enabled. Otherwise, if the NOR gate output is 0, which means there is an *R* signal in the previous eight clock periods, the output of DFF 7 will be low, indicating no data sampling phase updating is needed.

Similarly, if the NOR gate output in the lower branch is high, and there is no *L* in

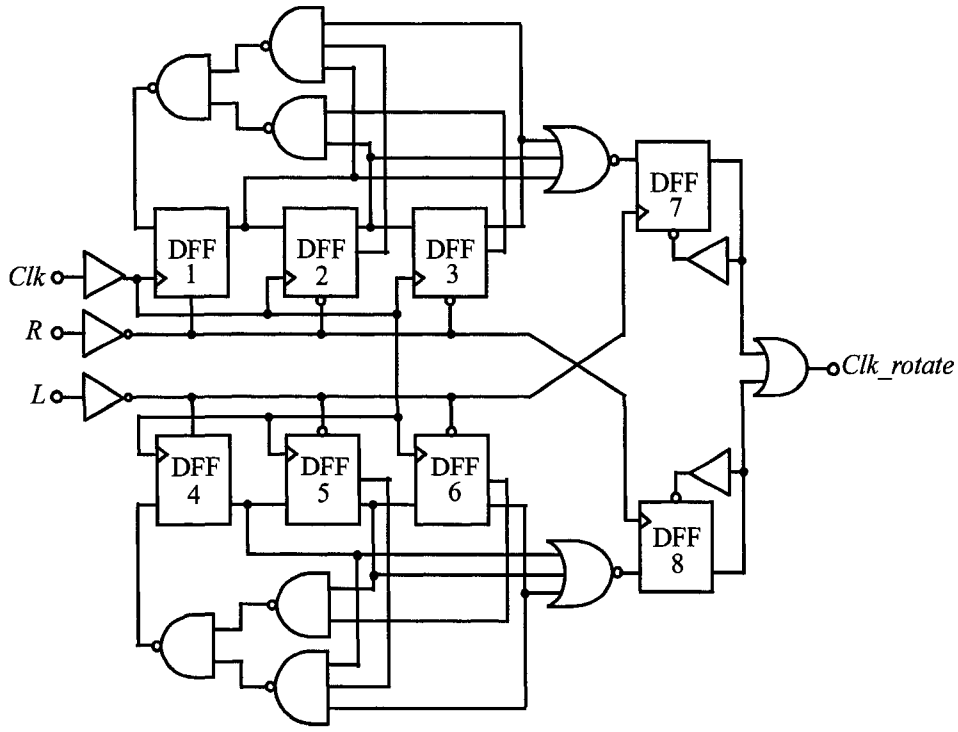


Figure 6.9 Schematic of the rotating clock generator

the previous eight baud periods, on arrival of an *R* pulse, DFF 8 is triggered to output '1' with the pulse width also determined by the delay in the reset path. The outputs of DFF 7 and DFF 8 are summed by the two-input OR gate to generate signal *Clk_rotate*.

To illustrate the inputs and output of the rotating clock generator clearer, Figure 6.10 shows the post-layout simulation with the data rate of 3.125Gbps from a PRBS 7, and the five clock phases from the DLL based multiple phase generator. When the clock phases lag the data, *R* remains constant while some pulses appear on *L*. Each pulse in the *L* corresponds to a rotating clock pulse, which enables a phase rotating operation in the phase rotator.

6.2.4 The phase rotator

The schematic of the phase rotator is shown in Figure 6.11. It consists of five phase rotating cells with the schematic shown in Figure 6.12. The one-hot coded phase

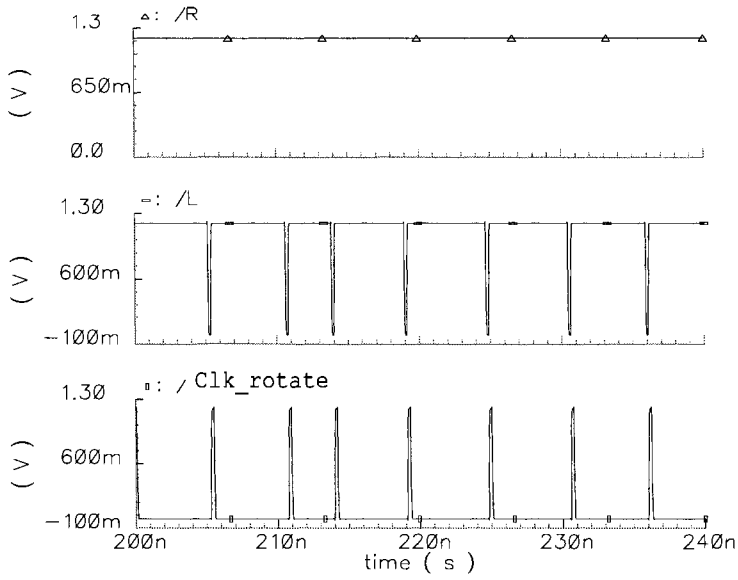


Figure 6.10 Post-layout simulation of the rotating clock generator

selection signal consisting of five clock phase position signals *a*, *b*, *c*, *d*, *e* keeps rotating in this phase rotator, and the initial state '10000' is obtained by setting the first rotating cell and resetting the left four cells. As shown in the phase rotator schematic, all the set or reset signals in the rotator cells are connected to *RST*, and on a falling edge of signal *RST*, the phase rotator is initialized to be '10000', then the following states are determined by the *R* and *L* pulses. For an instance, if there is an *R* pulse coming, the state is changed to '01000', and if there is an *L* pulse coming, the state is rotated to '00001'. In this way, the phase selection signal keeps rotating to track the phase and frequency difference among the data and the data sampling phase by feeding back the one-hot signal *a*, *b*, *c*, *d*, *e* to the phase detector to recover the data and the re-aligned clock, in the mean time, the updated *R*, and *L* signal for the next decision updating is generated.

In the schematic of the phase rotator cell shown in Figure 6.12, the inputs *R* and *L* are first inverted before applying to the NAND gates, and output signal *out* is applied to the *VR* in the followed rotator cell as well as to the *VL* in the previous rotator cell. In other words, the input *VR* is the previous state of the cell to the left, and *VL* is the previous state

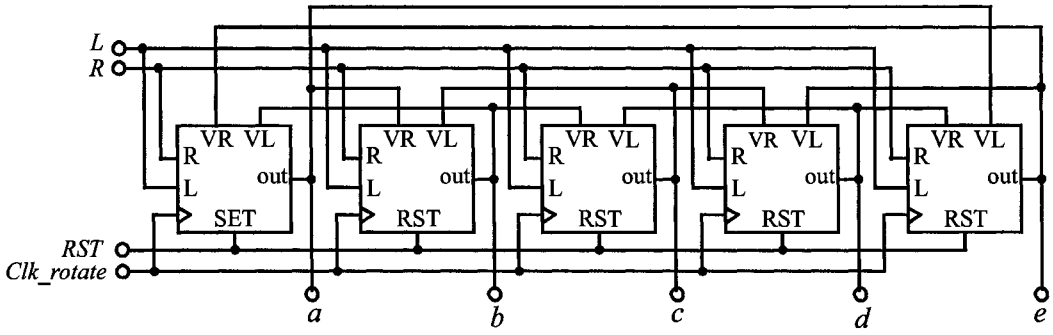


Figure 6.11 Schematic of the phase rotator

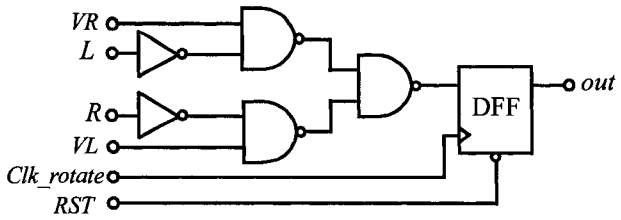


Figure 6.12 Schematic of the phase rotator cell

of the cell to the right one. According to the CDR decision technique, obviously it can be derived that when *VR* is high, and *L* is low, the current cell outputs '1' on arriving of *Clk_rotate*, indicating the one-hot signal is updated with a left rotating. Similarly when *VL* is high, and *R* is low, no matter what *L* is, the '1' state is rotated to the right. Therefore whenever a rotating is necessary, a narrow pulse appears at the input of the DFF, and this narrow pulse should be sampled by the *Clk_rotate* signal to ensure capturing any decision updating command, and correct timing must be also guaranteed against all the corner conditions.

Figure 6.13 shows the one-hot signal rotating in the phase rotator triggered by *Clk_rotate* under the same simulation conditions. The data period is 320ps and the clock period is set to be 316ps with 1.25 percent difference. From Figure 6.10, the *R* is constant, and *L* enables the rotating clock. So in this figure, the phase rotator keeps rotating to the left, and each rotating pulse corresponds to a phase rotating.

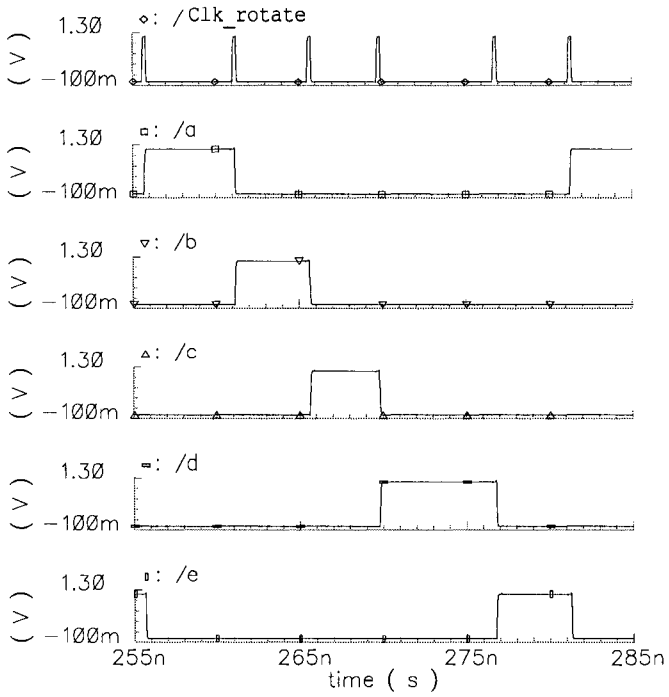


Figure 6.13 Phase rotating from the post-layout simulation

6.2.5 The MUX

The recovered data and re-aligned clock are obtained with two multiplexers shown in Figure 6.14. The phase selection signal *a*, *b*, *c*, *d*, *e* controls six NMOS transistor

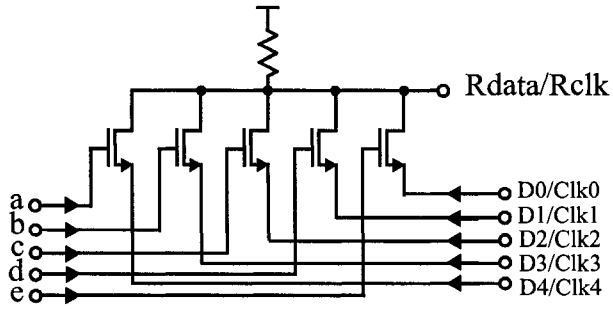


Figure 6.14 Schematic of the MUX

respectively, and the sources of the transistors are connected to *D0*, *D1*, *D2*, *D3*, *D4* respectively to obtain the data, or connected to *Clk0*, *Clk1*, *Clk2*, *Clk3*, *Clk4* to re-align the clock. By turning on and off the NMOS transistors, the signals connected to the sources

are passed to the drains which are summed together as the output. A load resistor is connected to the shared node, and the trade-off between the output swing and operation speed are balanced by sizing the resistor. A large resistance results in a large output swing, but it also results in a large time constant and a low operation speed.

6.2.6 The DLL based multiple phase clock generator

In the CDR, five clock phases are needed to sample and recover the data, so a multiple-phase clock generator based on a DLL is used due to its simple design, unconditional stability, and non-phase noise accumulation. In this CDR design, except for the hysteresis that affects the jitter tolerance, reducing the timing jitter in the sampling clock helps the circuit tolerate more jitter from the data. So timing jitter performance of the clock generator is among the most important concerns in the design.

The system-level architecture of the circuit is shown in Figure 6.15. It consists of a

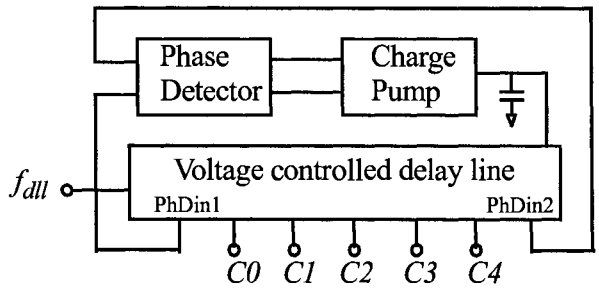


Figure 6.15 Architecture of the DLL multi-phase clock generator in the CDR

phase detector, a charge pump, a VCDL and a loop capacitor. A reference signal f_{dll} from a low-jitter source is applied to the VCDL where five phases are generated. The VCDL also outputs a signal *PDin1* from the first delay stage to the PD together with the last clock phase *PDin2*, and the phase error among the two signals is generated, which is converted to a control voltage by the charge pump and the loop capacitor. The resulting control voltage at the capacitor tunes the delay of the VCDL in a direction to reduce the phase error until the VCDL delay is exactly one period of the reference signal and the loop is in lock.

The clock phases *C0*, *C1*, *C2*, *C3*, *C4* are generated from the five delay stages in the VCDL as the output of the phase generator.

The schematic of the phase detector is shown in Figure 6.16. It is a conventional

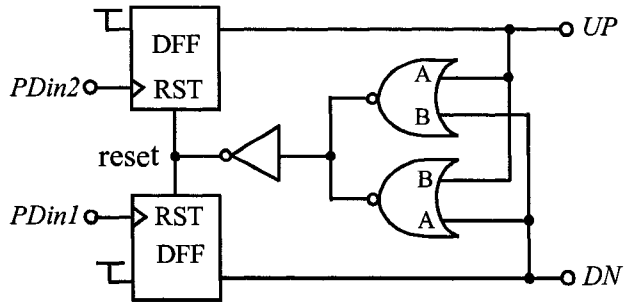


Figure 6.16 Schematic of the phase detector

dual-DFF phase detector with two extra NOR gates added. The inputs *A* and *B* of the two NOR gates are cross connected to cancel the mismatch caused by the NOR gate between the *UP* and *DN* paths. If the rising edge of the *PDin2* leads *PDin1*, signal *UP* is high, and on the arriving of the followed rising edge of *PDin1*, *DN* goes high. After a circuit delay, the *DN* pulse goes low. At this point, both *UP* and *DN* is reset by the reset signal as indicated in the figure. When the loop is in lock, *UP* and *DN* should have the same pulse width. However, any mismatch in the reset path results in the pulse width difference.

Another mismatch source is from the charge pump shown in Figure 6.17. From the

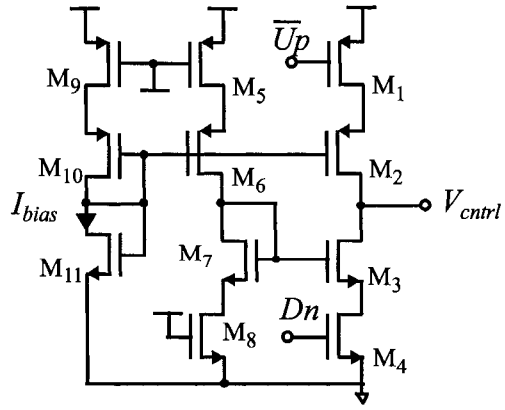


Figure 6.17 Schematic of DLL charge pump

figure, M_{11} generates the bias voltage and current I_{bias} for the whole circuit. M_4, M_8 functions as the degeneration resistors, so do M_1, M_5 and M_9 to reduce the steady state phase error caused by the charge pump. M_1 and M_4 used as current switches is controlled by signals *UP* and *DN* to source and sink current respectively.

The schematic of the VCDL is shown in Figure 6.18. It consists of a drive circuit

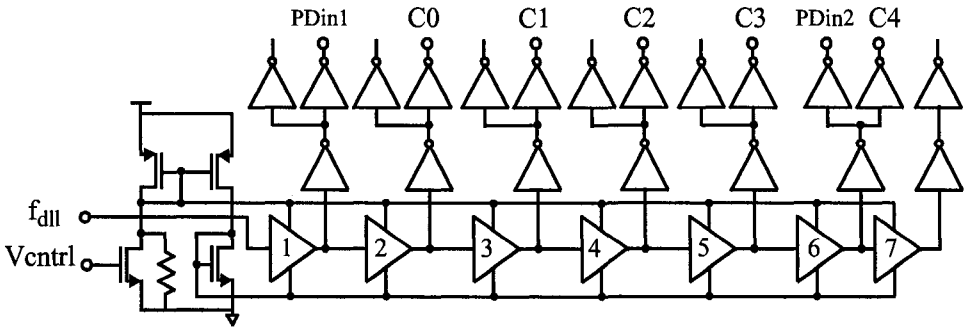


Figure 6.18 Schematic of the proposed DLL VCDL with reduced mismatches

and seven delay stages. The voltage control signal is divided to two voltage control signals to tune the VCDL delay by changing the current in the delay cells. In the drive circuit, the input control voltage is converted to a current first and the diode connected PMOS and NMOS transistors convert the current to the positive and negative control voltage respectively. A resistor in parallel with the NMOS transistor keeps the current flowing to the input NMOS transistor non-zero when the input voltage is below its threshold voltage.

The first delay stage output *PDin1* and the sixth delay stage output *PDin2* are applied to the PD as a buffered reference clock and the delayed reference signal respectively, and the phase error between the above two delay stages is reduced to ensure the delay of stage 2 to stage 6 are exactly one reference period. Each delay stage output is buffered with two inverters to mitigate the load imbalance among the six delay stages. Similarly, an extra inverter is added to the output of the load inverter, in parallel with the second inverter buffer in order to reduce the load imbalance among the stage 6 and the

other delay stages. Figure 6.19 shows delay range of the VCDL by sweeping the control

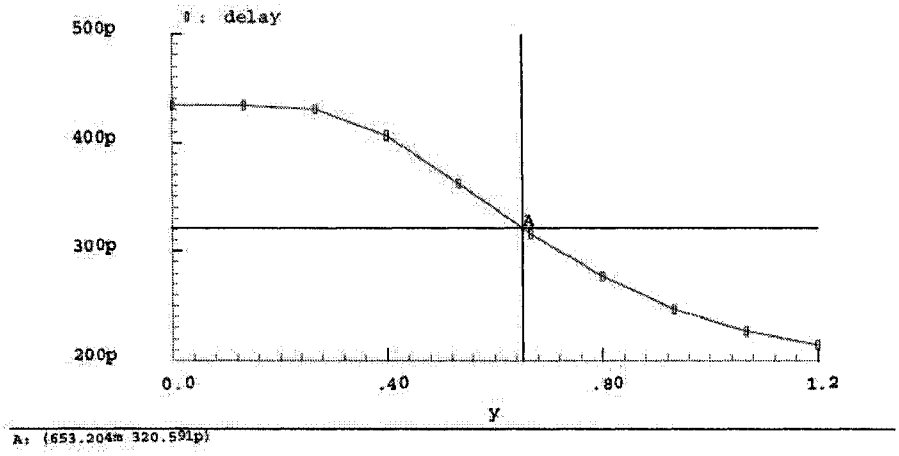


Figure 6.19 Control voltage vs VCDL delay value

voltage from post-layout simulation. The VCDL delay is from 200ps to 430ps and the targeted delay of 320ps was obtained with *Vcntrl* equal to 653mv which is an optimum output control voltage, at which the charging and discharging current sources can be better matched.

From the above description and analysis, the main concern is reducing the timing jitter by matching the *UP* and *DN* paths in the PD and by balancing the loads among different delay stages in the VCDL. From the VCDL schematic, obviously, the first delay stage and the sixth delay stage is more vulnerable to the load mismatch than the others, and the steady-state phase error between them as well as the in-lock error caused by the PD and charge pump combination appears directly in the output as clock phase mismatches. Therefore the largest phase difference after the loop is in lock is between the first clock phase *C0* and the last phase *C4*. Figure 6.20 shows the five output phases at 2.5GHz from the post-layout simulation. The delay between *C0* and *C4* is 83.78ps with a deviation of 3.78ps from the ideal value of 80ps.

In a summary, the DLL multi-phase clock generator outputs five clock phases from the VCDL with the simulated input and output range of 2.5GHz to 3.5GHz. The DLL consumes 5.5mW at 2.5GHz, and the peak-to-peak period difference among the five clock

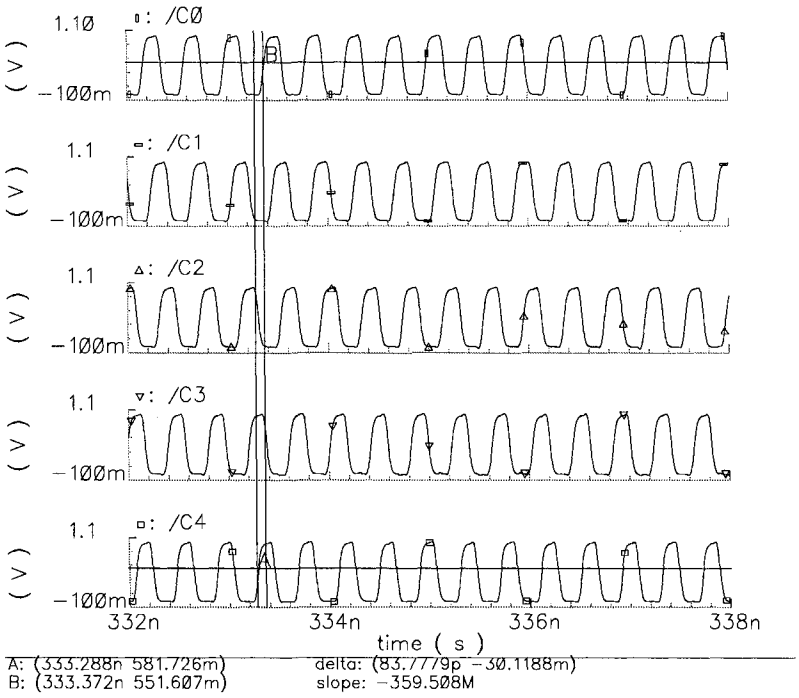


Figure 6.20 Output phases at 2.5 GHz from post-layout simulation

phases is optimized by balancing the loads of each delay stage with dummy loads.

6.2.7 The data generation and BER test circuit

In this design, the data is generated from a 7-bit PRBS data generator as shown in Figure 6.21. It consists seven DFFs and one XOR gate. By feeding back the outputs of the

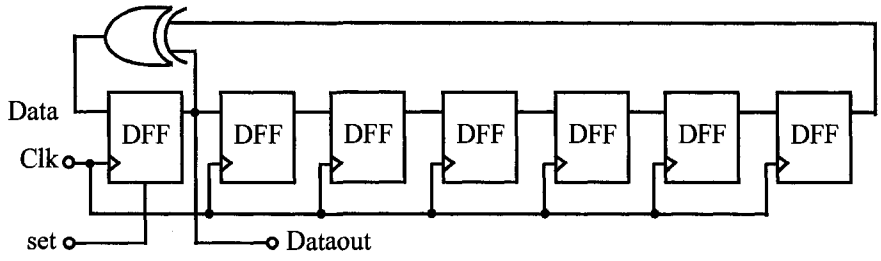


Figure 6.21 Schematic of the PRBS generator

first and the last DFF with the XOR gate to the input of the first DFF, a 127-bit sequence can be generated, and the only data sequence it cannot generate is '0000000'. The initial state is obtained by setting the first DFF, and the output data is from the first DFF output

as shown in the figure.

A BER test circuit was also implemented on the chip, and its schematic is shown in Figure 6.22. The shaded parts in the figure is the added components to the conventional

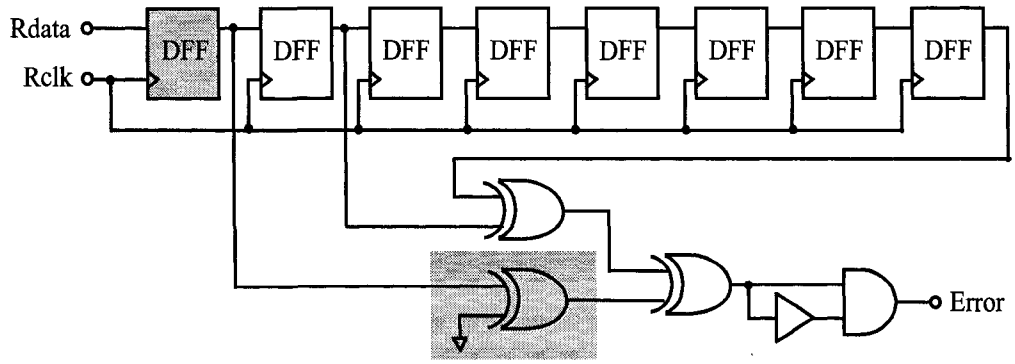


Figure 6.22 Modified BER test circuit

BER test circuit. The added DFF is used to re-time the recovered data with the re-aligned clock, and the added XOR gate is used to match the delay caused by the XOR gate in the PRBS data generator. The data from the data generator is compared with the re-timed data recovered from the CDR circuit, and an error signal *Error* is generated to indicate if there is any discrepancies between the recovered data and the PRBS data.

6.2.8 Top-level post-layout simulation

The top-level circuits including the extracted PRBS data generator, the BER test circuit, the core CDR and the DLL multi-phase clock generator were simulated to verify the functionality. The three main outputs of the chip is the recovered data *Rdata*, the re-aligned clock *Rclk*, and the BER test circuit output *Error*. The circuit was first checked at data rate of 3.125Gbps for simulation time of 320ns, and the sampling clock period was chosen to be 316ps. It was observed that the CDR keeps in lock with the one-hot signal keep rotating to compensate the phase error between the data sampling position and the data eye center to maintain the correct data and clock recovering. Figure 6.23 shows the simulation results. After the circuit acquires lock, no error was observed. For clearer

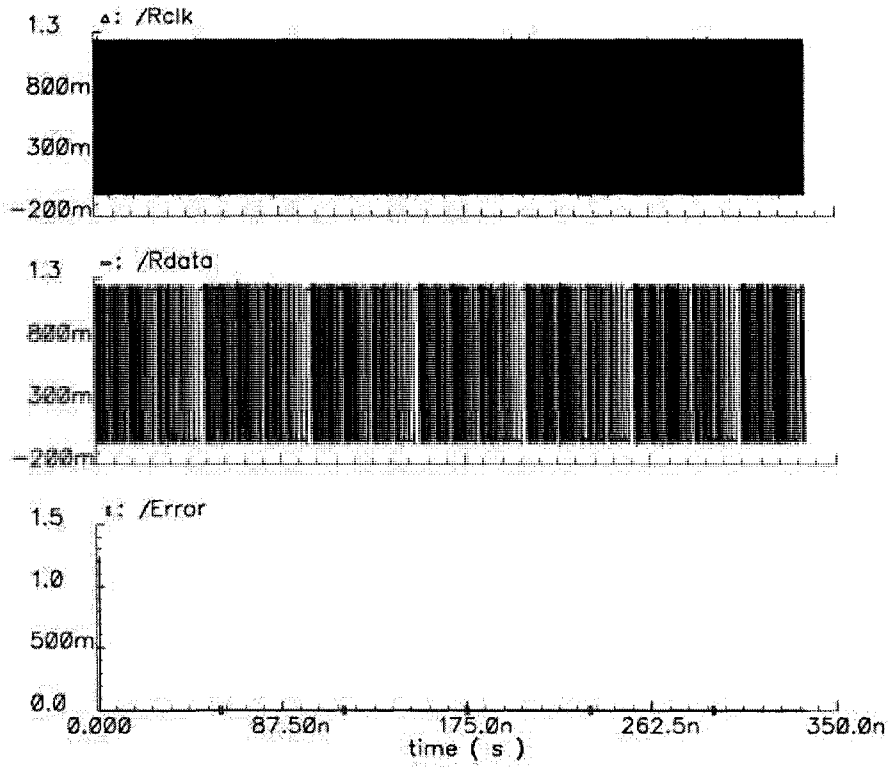


Figure 6.23 The three main chip outputs

details, an expanded figure is shown in Figure 6.24. In the figure of *Rclk*, a phase rotating action indicated with an arrow was observed, and the pulse width is different from the others due to the discrete phase step.

The top-level circuit was also checked with the post-layout simulation operating at data rate of 2.5Gbps with the baud period of 400ps and a clock period of 405ps. Similar simulation results were obtained. The phase rotator keeps rotating to track the phase error, and from the simulation time 0 to 350ps, no error was observed. One expanded figure is shown in Figure 6.25. It shows the waveform of the original data, *data*, the re-aligned clock *Rclk*, the recovered data, *Rdata*, and the error flag from the BER test circuit *Error*. In the expanded figure, the phase rotating action is indicated by the arrows in the waveform of *Rclk* where the width of the clock pulses is smaller than the others. To make *Rclk* clearer, another expanded figure is shown in Figure 6.26 with additional signals *R*, *L*, and *Clk_rotate*. In the figure, signal *L* keeps high, and the two low *R* pulses results in two cor-

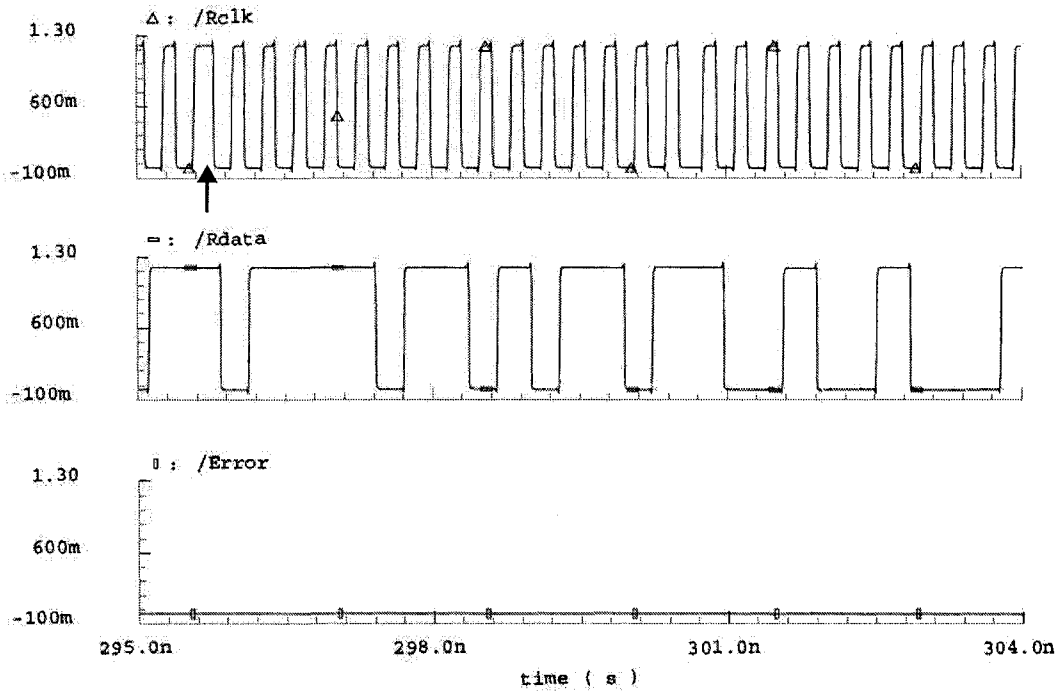


Figure 6.24 Expanded three main chip outputs at 3.125Gbps

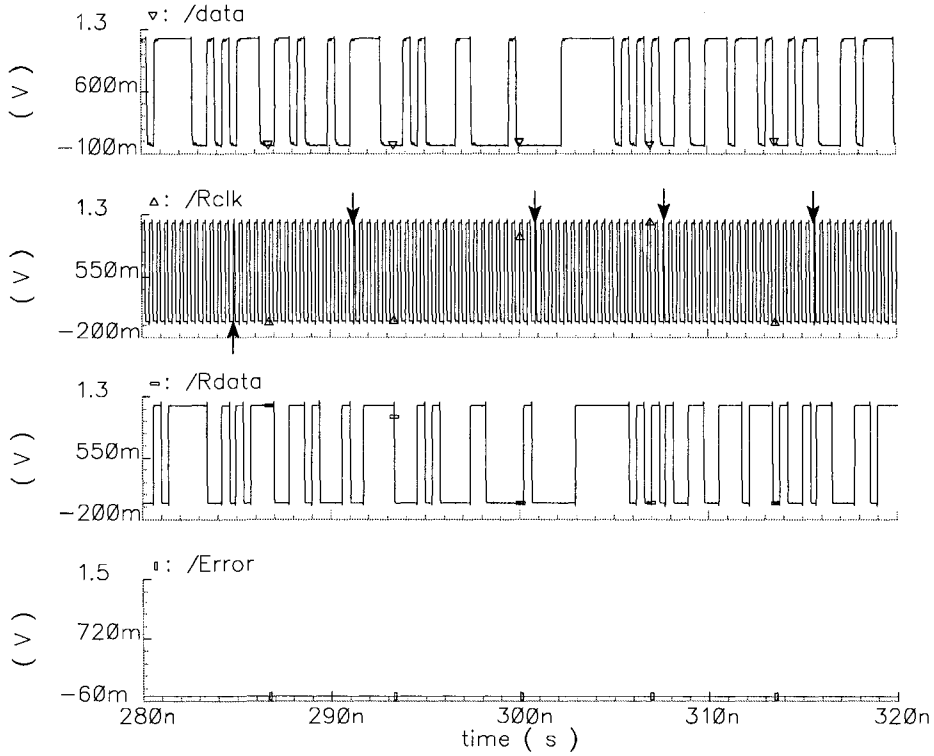


Figure 6.25 Expanded input data and chip outputs at 2.5Gbps

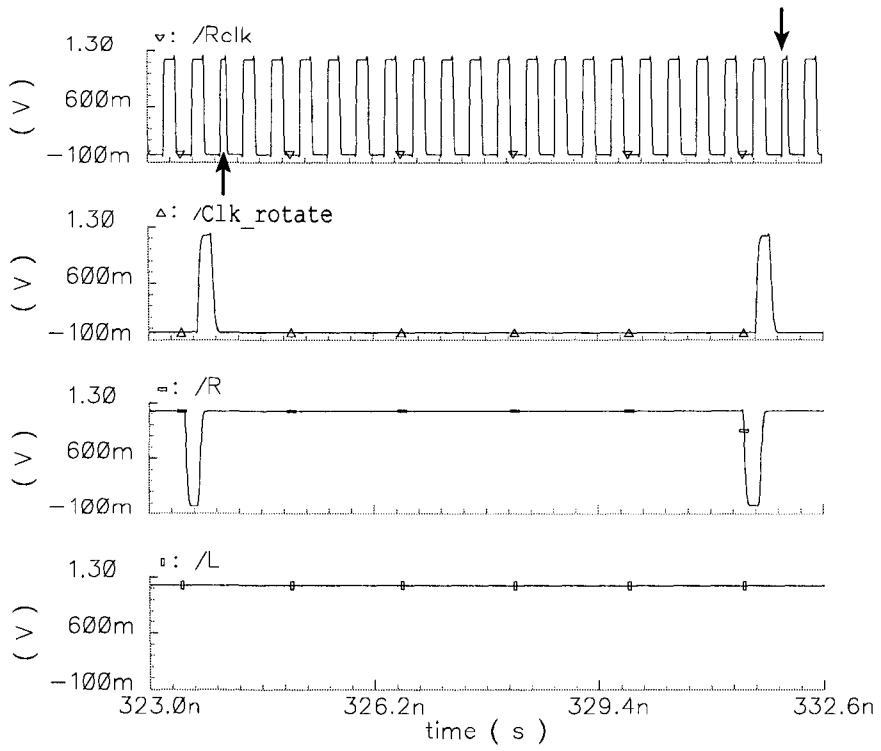


Figure 6.26 Expanded Rclk and rotating signals at 2.5Gbps

responding *Clk_rotate* pulses which trigger phase rotating twice as indicated by the arrows in the *Rclk* waveform.

The total transistors used in the core CDR circuit are approximately 900 without the output buffers. The circuits including the DLL multi-phase clock generator, the BER test circuit, and the PRBS generator were verified in the post-layout simulation in the range of 2.5GHz to 3.5GHz. At 3.125Gbps, the current consumption of the core CDR from the post-layout simulation is 3.25mA, consuming 3.9mW. The chip area including the DLL, the PRBS, the BER test circuit is 550 x 150 μm^2, and the core CDR area is 250 x 70 μm^2.

6.3 CMOS implementation of the proposed DLL frequency multiplier

In the frequency synthesis design, the PLL based architecture is dominant, how-

ever, compared with the DLL counterpart, it suffers from phase noise accumulation in the VCO induced by the power supply and the substrate noise. So the DLL based frequency multipliers are under exploration due to its superior phase noise performance, while relatively high output spurs limit its wide use. To date, two typical DLL-based frequency multiplication are reported as reviewed in Chapter 3, which is briefly repeated here. One is based on a VCDL and an edge combiner, and the other one is based on a VCDL that can be figured as a VCO. The main sources of spurious output of the first approach are from the mismatches in the VCDL and the edge combiner as well as from the loop in-lock error. The mismatches are minimized in the second approach, but the phase realignment errors, introduced in the process of injecting the clean reference edges into the VCDL, might even enlarge the in-lock error, which may not be effectively minimized by optimization of the CP and the PFD. In this design, a multiplying DLL frequency multiplier with a period error compensation loop (PECL) to achieve both low phase noise and low spurious level was implemented. The compensation loop is used to minimize the systematic error, resulting from various sources, and allows relaxing strict design criteria in the design of the PFD, CP, and multiplexer (MUX) and so on. To prevent the loop from locking to harmonic reference edges, a proposed switching control logic circuit and a resetable dynamic frequency divider is employed in the circuit.

The system level architecture of the implemented DLL frequency multiplier is shown in Figure 6.27. It comprises of a PFD, a CP, an MUX, a divide-by-N divider, switching logic (SL), an error detector (ED), a VCO-like VCDL, and one loop filter. The delay chain is configured as a VCO or a VCDL depending on which setting is chosen in the MUX, i.e. when the setting is 0, the rising edge of the reference signal f_{ref} is propagated in the delay chain, and when the setting is changed to 1, this rising edge circulates in the delay chain which can be essentially configured as a VCO. After N output cycles, where N is determined by the division ratio of the divider, this rising edge is steered out and compared with the phase of next rising edge of the reference in the phase detector,

where *UP* and *DN* signals are generated to tune the VCO frequency to the correct value. Ideally, the feedback loop is able to tune the VCO period to *1/N* of the reference period. However, due to the mismatches in the feedback path including the PFD/CP combination

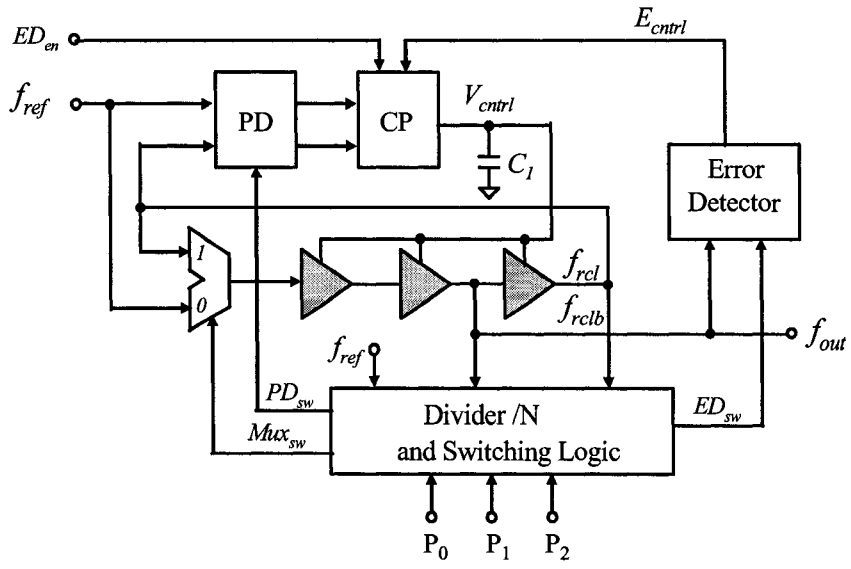


Figure 6.27 Architecture of the implemented DLL frequency multiplier

and the injection error introduced while the new reference edge is injected into the VCDL once every *N* output cycles, the *n*th cycle always has its period different from periods of the previous *N-1* cycles. Figure 6.28 shows the operation of the frequency multiplier with

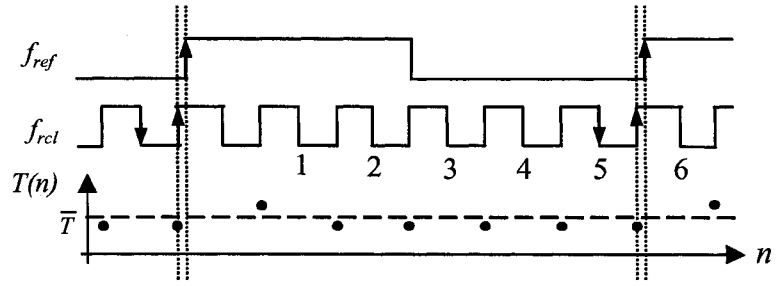


Figure 6.28 Generation of the injection error

an example of $N = 6$. When the 5th falling edge comes, the MUX is set to 0, which means the reference is connected to the VCDL, and this falling edge is inverted to a rising edge after passing the VCDL before it is compared with the reference edge to be injected. Due to the systematic errors as shown in the figure, the reference edge might be injected either

before or after the inverted sixth falling edge even if the loop is in lock, resulting a smaller or larger period than the previous 5 periods. This difference repeats at the reference frequency and results in large spurs at the VCO output.

To minimize both the alignment error analyzed above and the static phase error in the feedback path, the output clock f_{rclb}, which contains all the phase error information, is applied to an error detector in the auxiliary low-bandwidth feedback loop to detect the period difference and compensate it by tuning the charge pump current. In addition, a resetable divider and a switching logic circuit with proposed switching scheme is employed to prevent the DLL from locking harmonically and avoid the use of start up circuitry.

Since the phase error cannot be accumulated in DLL, the DLL based frequency multiplier may suffer from false locking. The PLL based ones have no such concern since the phase error can accumulate in the VCO over many cycles, and by slipping a few of cycles, the loop can acquire lock without affecting the output. While in DLLs, due to the nature of multiplication scheme, any kind of false locking can result in a false output.

When the VCDL delay is correct, the reference rising edge is exactly aligned with the inverted version of the 5th falling edge mentioned above. If the VCDL is too small as shown in Figure 6.29 (b), in other words, the reference edges are injected much earlier than the arrival of the rising reference edges, and the rising output edges are compared with the rising reference edges as indicated in the dashed arrow line, and the phase error between the two edges are generated. If the delay is too large, within one reference cycle, less than N cycles of the high-frequency output signal are counted in the divider, and the reference edges are injected much later than the reference edge as shown in Figure 6.29 (c). For this case, the DLL might lock to two adjacent reference edges. If the loop locks the output edge to the reference edge that comes early, the resulting high frequency output is correct, otherwise, if it locks to the reference edge that comes late, the resulting output

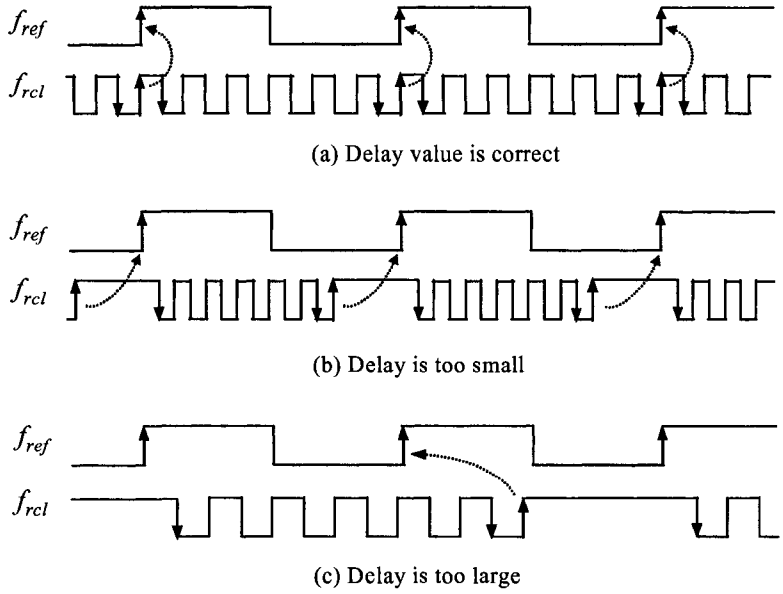


Figure 6.29 Illustration of the harmonic locking

frequency will be two times smaller than the correct value.

The harmonic locking problem is resolved in the implemented circuit, and the in-lock error due to the systematic error is reduced with the proposed period compensation loop. The followed sections elaborates on the design details of each block.

6.3.1 The phase detector and the charge pump

The schematic of the phase detector is shown in Figure 6.30. It is based on a conventional dual-DFF based PFD with a switching function controlled by signal PD_{sw} from the switching logic. In the conventional PD, the rising edge of either of f_{ref} or f_{rcl} triggers a pulse of *UP* or *DN* which are both reset to low by the following f_{rcl} or f_{ref}. In this modified structure, an additional control signal from the switching logic PD_{sw} is added. To properly operate with the proposed switching scheme, the control signal PD_{sw} does not simply disable and enable the PFD. This additional signal only functions at the falling edge by resetting the *UP* path to low, while *DN* is kept low when PD_{sw} is low, and only when PD_{sw} is high, the *DN* path is enabled. To make it clearer, the operation (For the case $N = 6$) is illustrated in Figure 6.31. The signal *UP* is enabled by the rising edge of f_{ref}, no matter PD_{sw} is

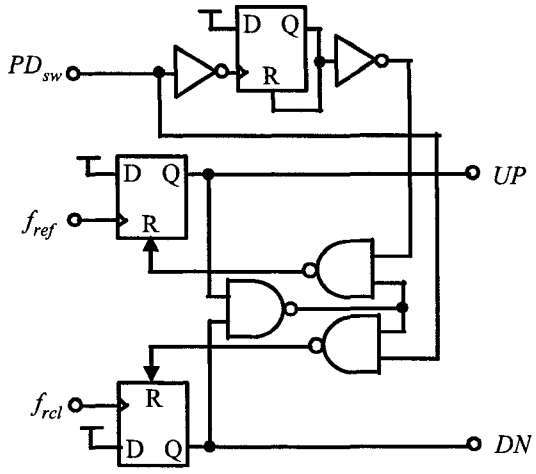
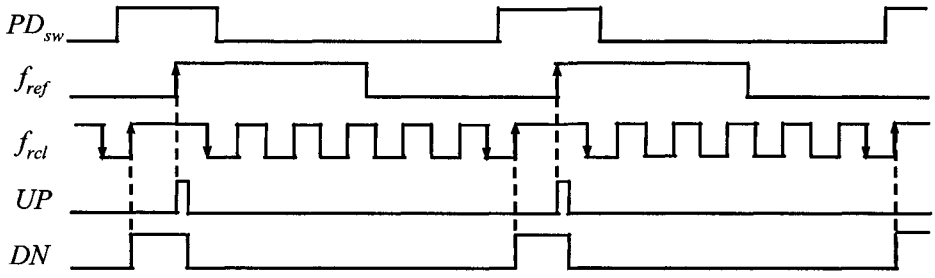
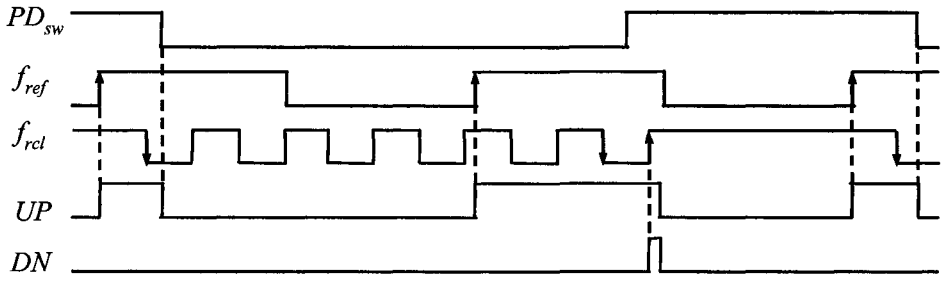


Figure 6.30 Schematic of the PD

enabled or not. However, signal *DN* is enabled at the rising edge of f_{rcl} only if the PD_{sw} is



(a) The VCDL delay is too small



(b) The VCDL delay is too large

Figure 6.31 Operation of the PD delay is too large

enabled. Both of them are reset at the same time after a certain delay in the same way as in the conventional PD. In addition, to guarantee a correct initial state of the PFD, it is also reset at each falling edge of PD_{sw}. PD_{sw} turns low right after a new reference edge is

injected and shows up at the 5th falling edge of f_{rcl} as shown in the figure. In this way, obviously, the phase detector always compares the phase error between the rising edge of f_{rcl} and the reference rising edge next to the one that was just injected, and the correct phase difference can always be obtained no matter how much the initial delay of the VCDL is. Consequently, the harmonic locking and infinite acquisition range can be achieved.

The schematic of the charge pump is shown in Figure 6.32. It is similar to the one

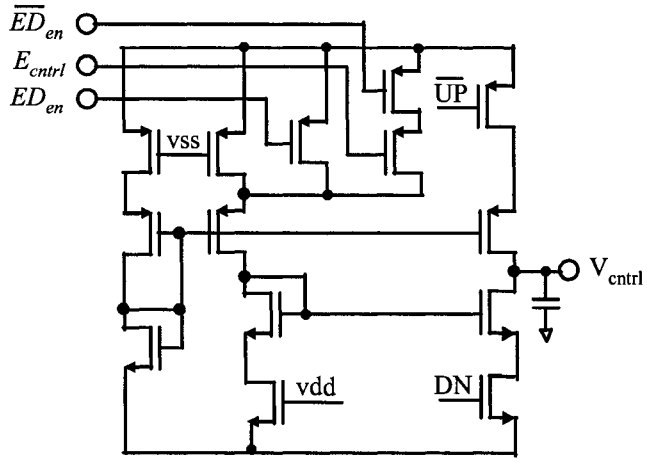


Figure 6.32 Schematic of the CP

shown in Figure 6.17 used in the DLL multiple phase generator in the CDR design. The *UP* and *DN* signals control a charge and discharge current source respectively. Normally the two currents are equally matched by the current mirrors so that the zero net charge is achieved when *UP* and *DN* have the equal width. Any current mismatch results in-lock error. In this design, to reduce the in-lock error as well as other injection errors, a current tuning capability is implemented in the CP as shown in the figure. Three additional control signals are added, two enabling signals, the positive and negative ED_{en}, and one control signal E_{cntrl} from the period error compensation loop. When ED_{en} is high, the output current I_{out} is tuned by the E_{cntrl}. Since the relative error between the charge current and the discharge current is of importance, only charging current is finely tuned by the compensation loop while the discharging current is controlled by *DN* in this design.

6.3.2 The error detector

The conventional PD compares the phase error between the reference edges and the edges of f_{rcl}, and any mismatch between the two paths would cause in-lock error. To minimize this in-lock error as well as the re-alignment error, not only the phases of f_{ref} and f_{rcl} are compared in the PFD, but also the instantaneous f_{ref} edges are compared with the edges of the averaged f_{out} in the error detector, where a period error signal is generated to help tuning the VCDL delay to minimize the errors.

Due to the mismatches in the feedback path including the PFD/CP combination and the re-alignment error introduced while the new reference edge is injected into the VCDL every reference cycles, the n-*th* cycle always has its period different from periods of the previous cycles even when the loop is in lock and large output spur is thus resulted. As illustrated in Figure 6.33, there are N output periods within one reference period, and each corresponding period is plotted at the bottom of the figure. One period out of N periods may be always larger or smaller than other periods, which corresponds to a positive or negative error as shown in Figure 6.33 (a) and (b) respectively. This static error is referred

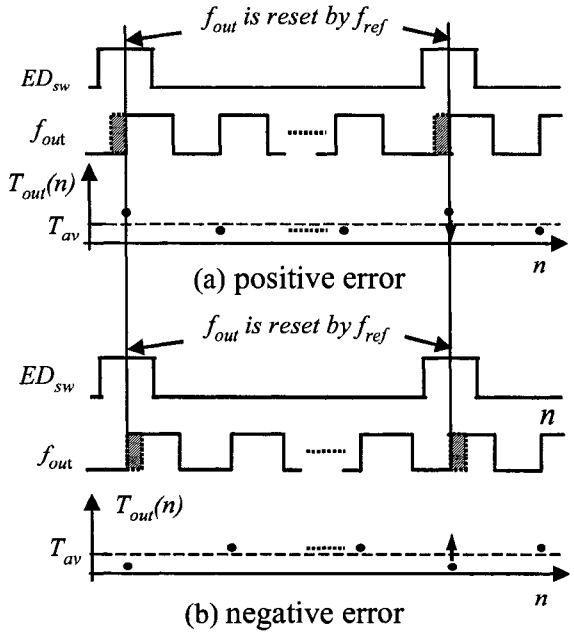


Figure 6.33 Period error when the loop is in lock

as the period error. To minimize this error, the output is applied to the ED to extract this error first and compensate it by tuning the charge pump current. The ED compares the period of the *n-th* output cycle $T_{out}(n)$ with the average period T_{av}. The difference between $T_{out}(n)$ and T_{av} is amplified and its polarity is determined. The comparison offset is self-cancelled in the ED so that it can provide more precise error information than the PFD, where the offset of the phase comparison cannot be self-cancelled. In addition, the error information provided by the ED corresponds to the total period error appearing at the output of the frequency multiplier while the error provided by the PFD only corresponds to part of this period error. In this design, besides the loop governed by the PFD, a low bandwidth PECL governed by the ED is used to compensate for the static period error to significantly reduce the output spurious level.

The mathematical model of the period error detection was given in Chapter 5, which will be repeated here. The period error is obtained by subtracting the average period from the period $T_{out}(n)$ of the ED input signal f_{out}, and this error is amplified to increase the error detection resolution. By sampling the amplified period error with signal ED_{sw}, an output $E_{out}(n)$ is generated to finely tune the source current in the CP, and then the VCDL delay value. In this way, the phase noise and timing jitter performance can be improved.

The advantages of the proposed jitter reduction technique are as follows. The output signal is not affected by the mismatch between the charging and discharging path in the PFD/CP combination. The second advantage is that the output signal f_{out} or f_{rclb} is the only ED input which contains the static phase errors from all the sources, and the jitter reduction technique targets to reduce those phase errors in one step and no additional phase noise is introduced. The third is that the phase error detection resolution can be very high by improving the error detector gain to reduce the offset phase error, and the static error resulting in a constant jitter at the output is greatly reduced by finely tuning the CP.

The error detector is the main part of the PECL. The schematic is shown in

Figure 6.34. From the figure, all VCO periods are subtracted by their average period

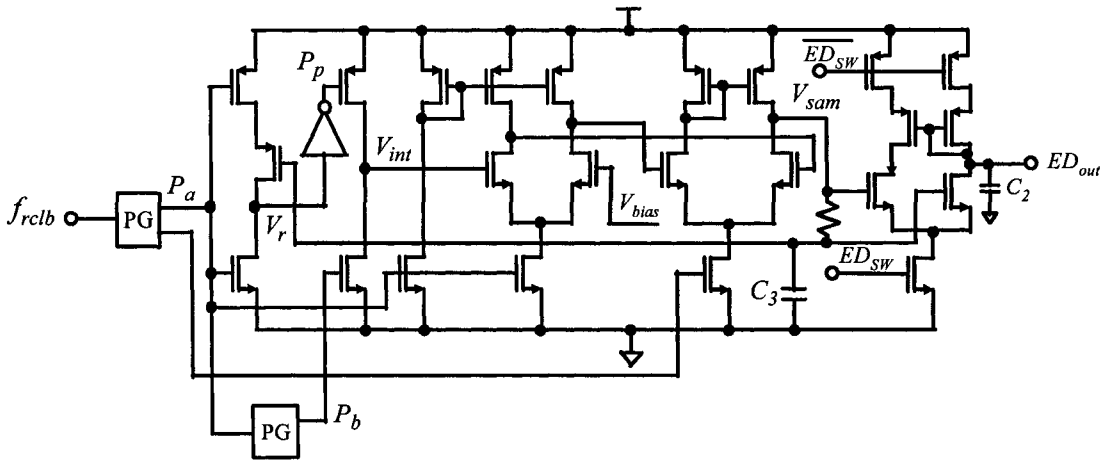


Figure 6.34 Schematic of the error detector

value, which is determined by an internal feedback loop, and the subtracted results are amplified and their polarities are examined to produce the error output whenever *EDsw* is high. The operation of the error detector is shown in Figure 6.35 for the case of positive period error. On the falling edge of signal f_{rclb}, the pulse generator (PG) generates a narrow pulse (P_a), and its width is then increased by an amount corresponding to the average

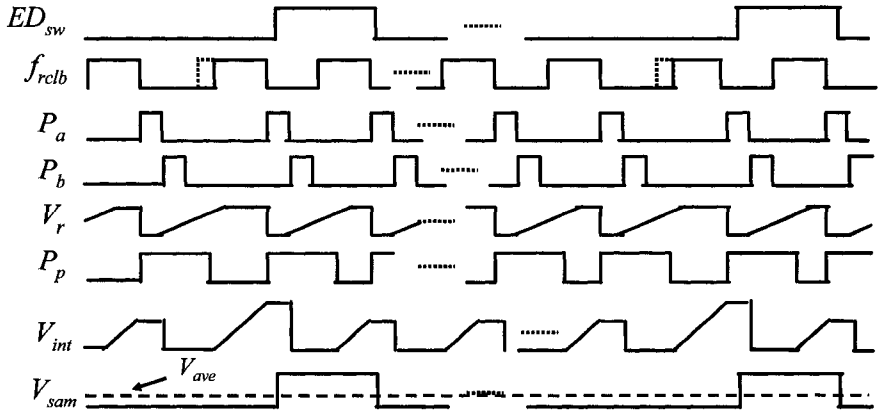


Figure 6.35 Operation of the error detector

period duration. The resulting pulse train (P_p) is inverted and integrated by a high-gain integrator whose output (V_{int}) is sampled and then reset before the next pulse coming. Those sampled results (V_{sam}), corresponding to the VCO periods, drive the internal feedback, so that the average value (V_{ave}) of the sampled results can be dynamically main-

tained by the internal feedback at a certain level determined by V_{bias}. When signal ED_{sw} is high, a comparator is switched on to compare the samples (V_{sam}) with its average (V_{ave}) to generate the error output signal. The internal feedback loop compares V_{ave} with a pre-defined value (V_{bias}) and adjusts the increment value of the pulse width of signal P_a accordingly. With the internal loop, a large integrator gain can be used to achieve high period error resolution. The bias voltage (V_{bias}), which determines V_{ave}, is set to be approximately half of the supply voltage *vdd* to ensure the integrator works in its high-gain region. The output of the ED is to finely tune the charging current in the charge pump after it is low-pass filtered to compensate the period error.

The pulse generator schematic is shown in Figure 6.36. It is basically a DFF with its output connected to the reset to generate the pulse. The equivalent circuit is shown in Figure 6.36 (b). The DFF input is the inverted *Q*, and at each rising edge of *Input*, the *Ouput* is sampled. The sampled signal keeps high until it is reset by the feedback. The DFF output *Q* and its inverted signal *Output* is connected to the NMOS and the PMOS transistors respectively which are both added to the dynamic DFF to reset the output.

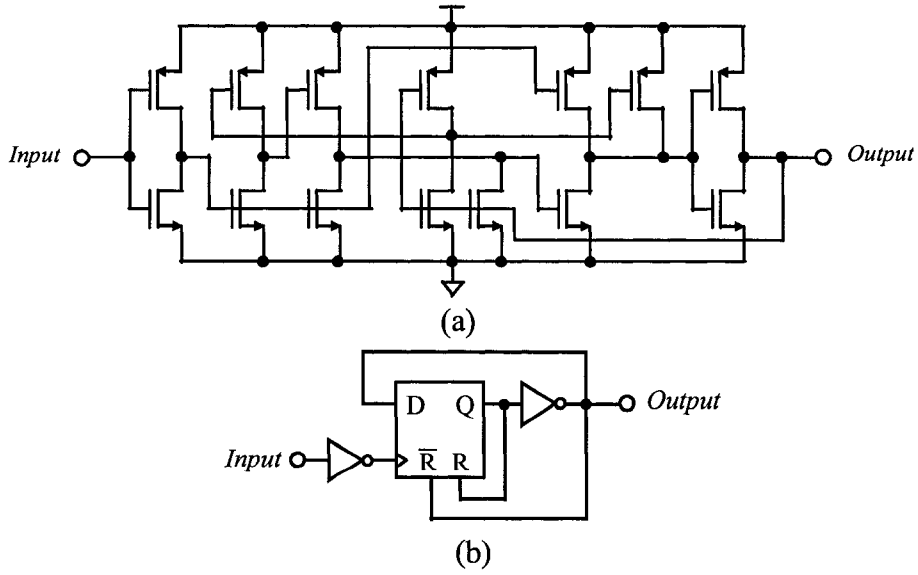


Figure 6.36 (a) Schematic of the pulse generator, (b) Equivalent circuit

6.3.3 The VCDL and MUX

The VCDL is based on 3-stage current starved inverters as shown in Figure 6.37.

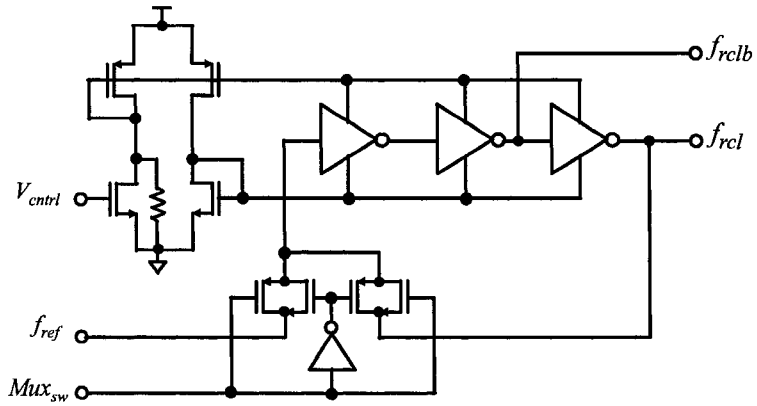


Figure 6.37 Schematic of the VCDL and MUX

Its propagation delay is controlled by two bias voltages which are derived from the control signal V_{cntrl}. Two outputs are generated from the VCDL, with one output f_{rclb} instead of f_{rcl} used as the multiplier final output because the switching activity changes the last VCDL stage load condition every reference period, while f_{rclb} is less affected. The input of the VCDL is either f_{ref} or f_{rcl} depending on the switching signal *Muxsw*. The MUX is a selecting switch of the VCDL, consisting of two pass gates, and f_{ref} or f_{rcl} is passed through the gates controlled by signal *Muxsw*. For example, when *Muxsw* is low, the rising edge of f_{ref} is injected, and when *Muxsw* turns high, and f_{rcl} is connected. The injected reference edge circulates in the VCDL for N cycles with the clock frequency of $f = 1/2T$, where T is the VCDL delay. This selection action is periodic and the phase noise accumulated over every $N-1$ output cycles are zeroed.

6.3.4 The divider and the switching logic

The frequency divider and the SL circuit shown in Figure 6.38 coordinate the frequency multiplier operation by providing four signals *Muxsw*, PD_{sw}, ED_{sw} and *reset*. The frequency divider counts the high frequency output f_{rcl} to achieve a programmable multiplication ratio. After a reference edge is injected in the VCDL, the frequency divider is

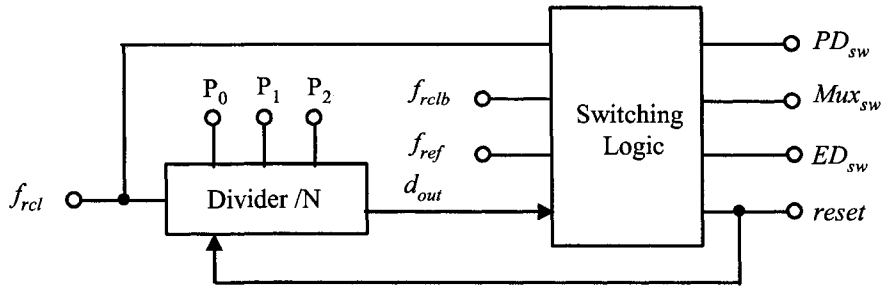


Figure 6.38 The divider and the switching logic

reset by a reset signal from the SL, and then it starts to count from zero again

To design the dynamic programmable divider, a dynamic divide-by-2 divider is first analyzed. Figure 6.39 shows the schematic of the divide-by-2 divider. It is basically a

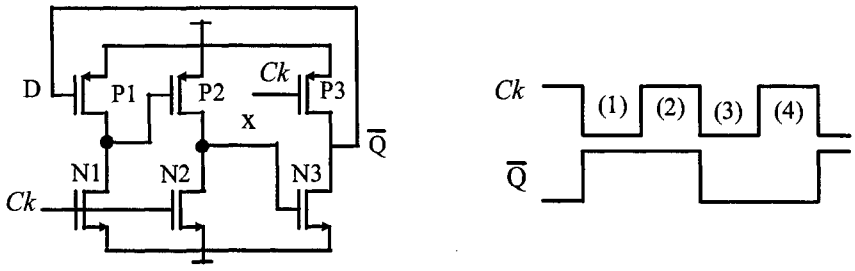


Figure 6.39 The divide-by-2 divider

dynamic DFF with the output connected to the input *D*. It consists of 6 transistors labeled as P1, P2, and P3 which are PMOS transistors and N1, N2 and N3 which are NMOS transistors. Since the current switching of the three pairs of transistors with drains connected together, depends on the driving ability, the transistor sizing is very important. For equal driving ability of each transistor pair, the ratio of the W/L of the PMOS to that of the NMOS transistor is approximately 2.5 and if the size ratio is larger than 2.5 (for better results, the size ratio can be much larger than 2.5), the PMOS transistor charges up the node harder than the NMOS transistor discharges it, and the voltage on the node approaches the supply voltage. Similarly, if the ratio is less than 2.5, the node voltage approaches the substrate voltage.

Accordingly, the circuit is connected in a toggle configuration for the sake of facil-

itating the sizing consideration, and the timing waveforms of the clock signal *Ck* and the output $\overline{Q}$ are also shown for the desired operation. In the precharge phase labeled as (1) in Figure 6.40, before the falling edge of *Ck* arrives, *Ck* is high and $\overline{Q}$ is low, the circuit configuration is the same as in the evaluation phase (4), where node *x* is discharged to low, and since the driving ability of P1 is larger than N1, *y* is high. On the falling edge, N1, N2 is cutoff. This connection changes are highlighted in the dashed line and X which means the transistor is connected and cutoff in the previous phase respectively. P1 is still connected at this point, so $\overline{Q}$ is charged up more quickly than node *x* is charged up, thereby N1 is cutoff. So in phase (1), *x* is low while *y* is high and $\overline{Q}$ is high.

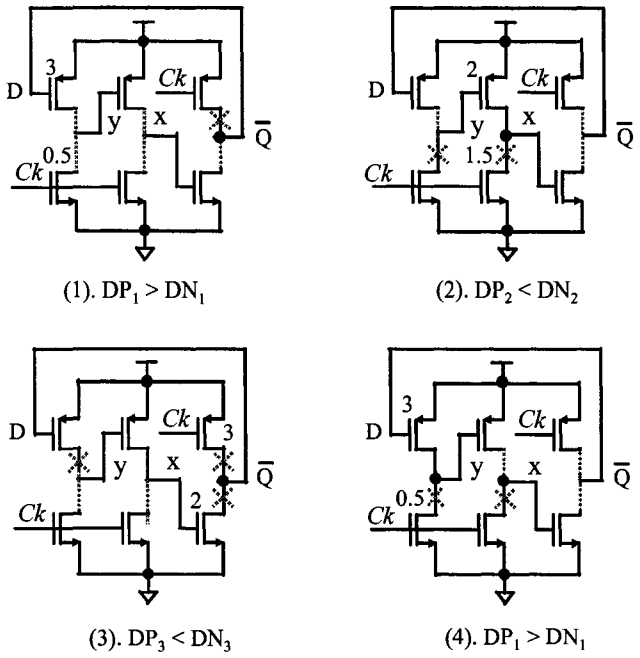


Figure 6.40 The divide-by-2 divider operation

For the evaluation phase (2), N1 and N2 are connected on the rising edge *Ck*, both P2 and N2 is connected, and node *y* is discharged immediately, which turns on P2. For the desired operation, where voltage on node x needs to be low, the driving ability of N2 should be larger than that of P2, so the transistor sizing ratio should be smaller than 2.5. In this way, N3 is still cutoff, and $\overline{Q}$ keeps high. Similarly, the driving ability of N3 should be larger than P3, and the driving ability of P1 should be larger than that of N1. The transistor

sizes are labeled in the circuit.

One of the advantages of this structure is that it uses less transistors. Compared with the more conventional structure, the source coupled logic (SCL) circuit consists of 18 transistors, and a few of extra transistor as output buffer.

Figure 6.41 shows the divide-by-2/3 divider consisting of two DFFs described

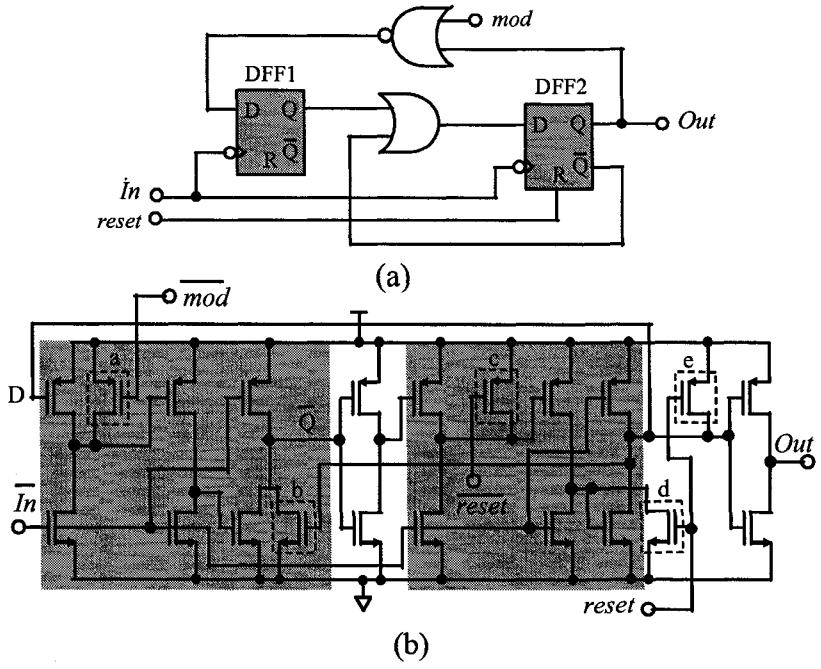


Figure 6.41 Divide-by-2/3 divider, (a) block diagram (b) detailed schematic

above, one OR gate and one NOR gate. For the normal mode operation, when *mod* = 1, the output Q of DFF1 is always 0 and DFF2 is in a toggle configuration, so the divider works as a divide-by-2 divider. When *mod* = 0, DFF2 remains high for an extra clock cycle before setting back to 0 and toggles again. At the same time, signal *mod* is set back to 1 to prevent extra pulses from being swallowed.

The transistor-level schematic of the divide-by-2/3 divider is shown in Figure 6.41 (b). The two DFFs are shown in the shaded region. With the extended true-single-phase-clock (E-TSPC) logic employed in the divider, the power consumption is reduced due to the less number of transistors, and the interconnections among the transistors are much

shorter, all of which contributes to high operating frequency. Another advantage is that this structure allows embedding some logic functions. By simply adding one additional PMOS transistor labeled '*a*' parallel to the PMOS transistor whose drain is the input of the DFF, the NOR gate is formed. Similarly, the OR gate is formed by adding one NMOS transistor *b* with the drain connected to one output of DFF1. The reset circuit is only added to DFF2 since the DFF2 generates the final divided output. To ensure the correct reset, two PMOS transistors *c*, *e*, and one NMOS transistor *d* are added. Compared with the conventional divide-by-2,3 dividers, this structure uses less number of transistors and thus resulting in a compact circuit.

By cascading three divide-by-2/3 dividers with a control qualifier which consists of three NOR gates, the divide-by-N divider is formed as shown in Figure 6.42. The

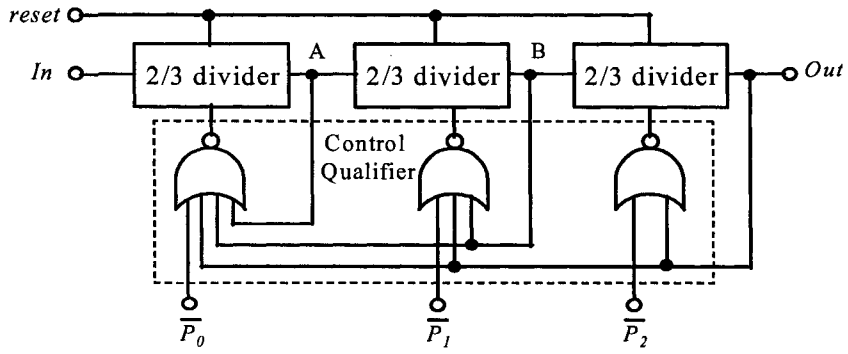


Figure 6.42 Block diagram of the divide-by-N divider

divider is reset by a reset signal from the switching logic, and starts to count from zero again. The three control signals P_0, P_1, and P_2 programs the multiplication ratio which can be calculated as

$$N = \sum_{i=0}^{2} 2^i \cdot P_i + 8$$

From the equation, the frequency division ratios are integers from 8 to 15. Due to the delay of the circuit, the actual multiplication ratio of this clock generator is from 13 to 20. To verify the functionality of the divider, simulation results are obtained from HPICE

simulation. Figure 6.43 shows the operation of the divide-by-N divider. The simulation takes $N = 11$, $P_0P_1P_2 = 110$ as an example. By feeding back the three outputs, *A*, *B* and *out* as indicated in the figure, the first and the second stage divide-by-2,3 dividers divide by 3, while the last stage divides by 2 due to $P_2 = 0$.

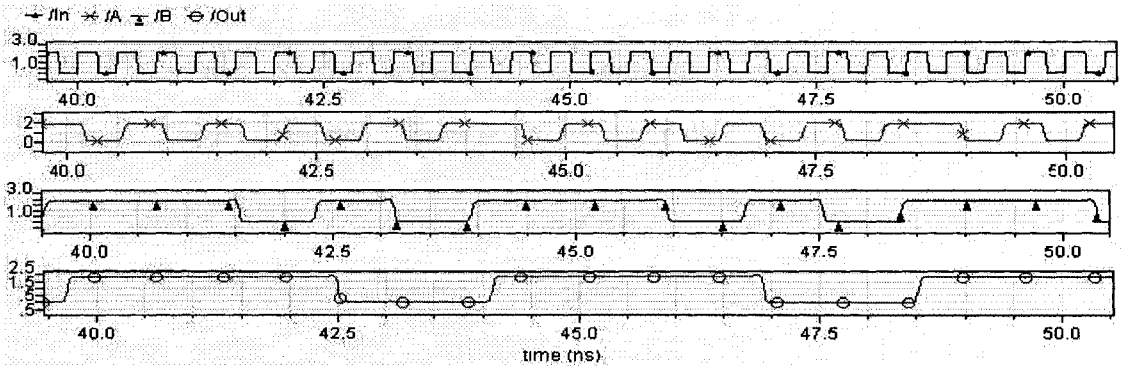


Figure 6.43 Simulated operation of the divide-by-N divider

The switching logic works as a timing control unit by synchronizing the operation of the PD, the MUX, the divider and the error detector. The schematic is shown in Figure 6.44. It consists of six DFFs with reset function and two delay elements. There are

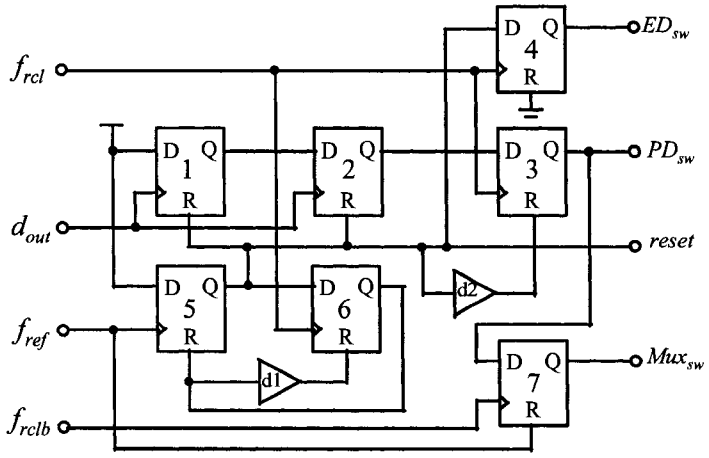


Figure 6.44 Block diagram of the switching logic

four inputs d_{out}, f_{rclb}, f_{ref}, and f_{rcl}. The reference f_{ref} is used to synchronize the loop with the reference by providing switching signals PD_{sw}, *reset*, ED_{sw}, and *Muxsw*. The signal *reset* not only resets the divider, but the other two DFFs (the count-to-2 circuit) in the switching logic as well.

One of the four inputs, d_{out}, is from the frequency divider, and its second rising edge after the divider was reset indicates that the counting-to-N is reached. The DFF1 and DFF2 form a count-to-2 circuit and the output of the DFF2 Q_2 goes high at the second rising edge of the d_{out}. The operation after Q_2 goes high is illustrated in Figure 6.45. From

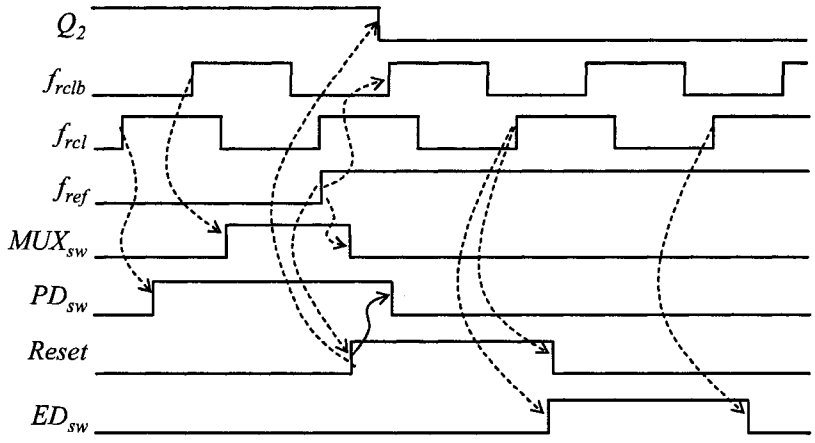


Figure 6.45 Operation of the switching logic

the figure, signal PD_{sw} is obtained by sampling $Q2$ at the rising edge of the f_{rcl}. The *MUXsw* is generated by sampling the PD_{sw} at the rising edge of f_{rclb}. The *reset* signal appears at the first rising edge of f_{rcl} after the rising edge of the f_{ref}. The ED_{sw} is obtained by sampling the *reset* signal at the rising edge of f_{rcl}, and it is a positive pulse with the width equal to one output period. This new switching scheme is employed in the synchronized operation timing for the PD, the Mux, the error detector, and the divider by generating the corresponding switching signals for the purpose of self-correcting of the loop for arbitrary initial VCDL delay values.

Figure 6.46 shows the simulated timing sequence of the inputs and outputs when the loop is in lock for better understanding the operation of the switching logic with respect to how to avoid harmonic locking. From the figure, on the rising edge of the reference signal, the *reset* signal (the output of DFF4) goes high and keeps high for a certain time depending on the delay of the delay element d_1. Then the divider is reset and its output d_{out} goes down. On the rising edge of f_{rcl}, which means a rising reference edge was

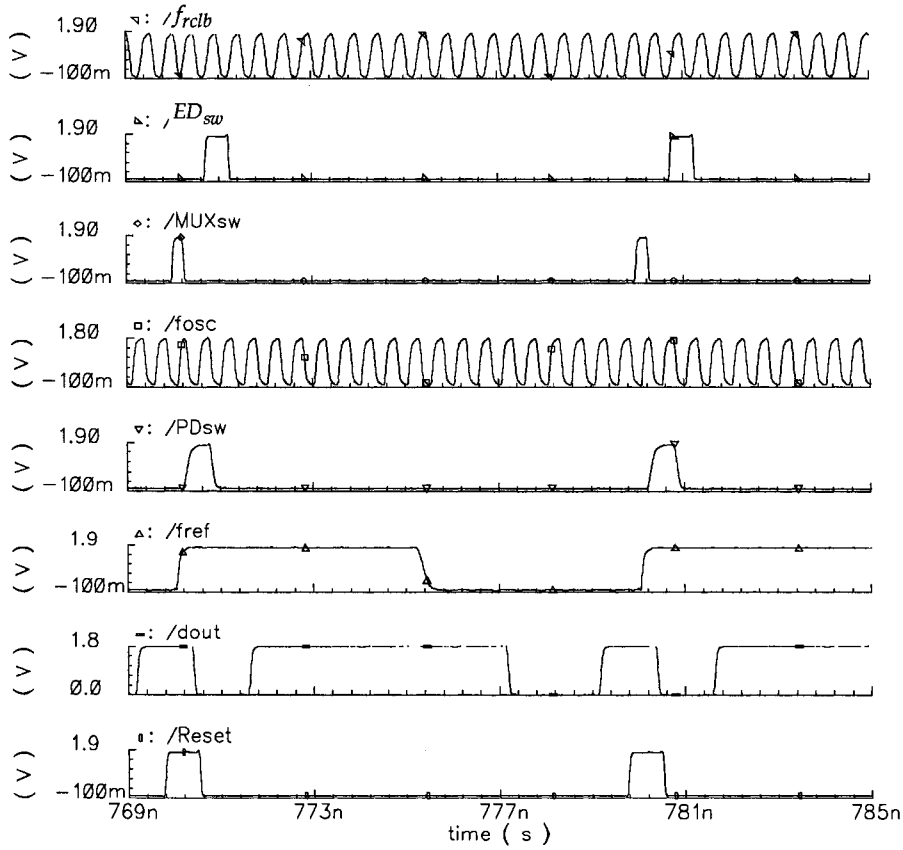


Figure 6.46 Simulated operation of the switching logic

just injected, the *reset* goes down, and the divider starts to count from zero again upon the falling edge of reset every reference cycle.

The timing of signal PD_{sw} is critical as it helps self-locking for the case when the VCDL delay is too large shown in Figure 6.31 (b). From Figure 6.45, the d_{out} has two rising edge at every reference cycle. At the first rising edge, the output of DFF1 is high, and at the second rising edge when the counting-to-N is reached, the DFF2 goes high. Once the rising edge of f_{rcl} arrives, the reference edge is injected, and PD_{sw} goes high and keeps high for a certain time determined by the delay value of d_2 and the reset signal that is enabled once a reference edge is injected. Consequently, signal PD_{sw} is enabled right before the rising edge of the n-th cycle and disabled right after the injected reference rising edge. This characteristic ensures that the PD always compares the n-th output rising edge

with the correct reference rising edge, hence the harmonic locking problem is avoided. The Mux_{sw} is triggered to be high in DFF6 at the rising edge of f_{rclb} when the PD_{sw} is high. In other words, the MUX input is connected to the reference after the *n-th* falling edge of f_{rcl}, and connected to f_{rcl} after the rising reference edge.

Due to the circuit delay, the divider is not reset immediately after the counting-to-N is finished, and it does not start the count-to-N right after the falling edge of the reset either. So five more output cycles re-circulate in the VCO, resulting in a multiplication ratio larger than the calculated division ratios of the frequency divider.

6.3.5 System-level post-layout simulation

The circuit was implemented in CMOS 0.18um technology at a 1.8v power supply. The functionality of the ED can be observed from signal V_{cntrl} and E_{cntrl} in the locking process from the post-layout simulation result. Figure 6.47 shows signal V_{cntrl} and E_{cntrl} from post-layout simulation for the locking process. From Figure 6.47(a), when V_{cntrl} is stable, the main loop is in lock, and the period error compensation loop slowly tunes the charge pump current to increase the E_{cntrl} until the auxiliary loop captures lock. To show the details how E_{cntrl} changes V_{cntrl} after the loop is in lock, the figure between the two lines is expanded in Figure 6.47(b) on the right as indicated by an arrow. When the main loop is in lock, V_{cntrl} is stable, and the error detector detects in-lock errors based on the output f_{out}, so the period compensation loop tunes E_{cntrl} to increase or decrease the charging current of the charge pump, which results in a small change in V_{cntrl} as shown in the zoomed-in figure. V_{cntrl} is decreased by about 2mV, and the large current spikes are also reduced. All these significantly contribute to period error reduction. To verify the functionality of the compensation loop, the edge jitter from post-layout simulation is shown in Figure 6.48. Simulation results show when the compensation loop is enabled, the peak-to-peak edge jitter is 3.2ps, and when disabled, it is larger than 20ps.

Simulation was also done with other reference frequencies and other division

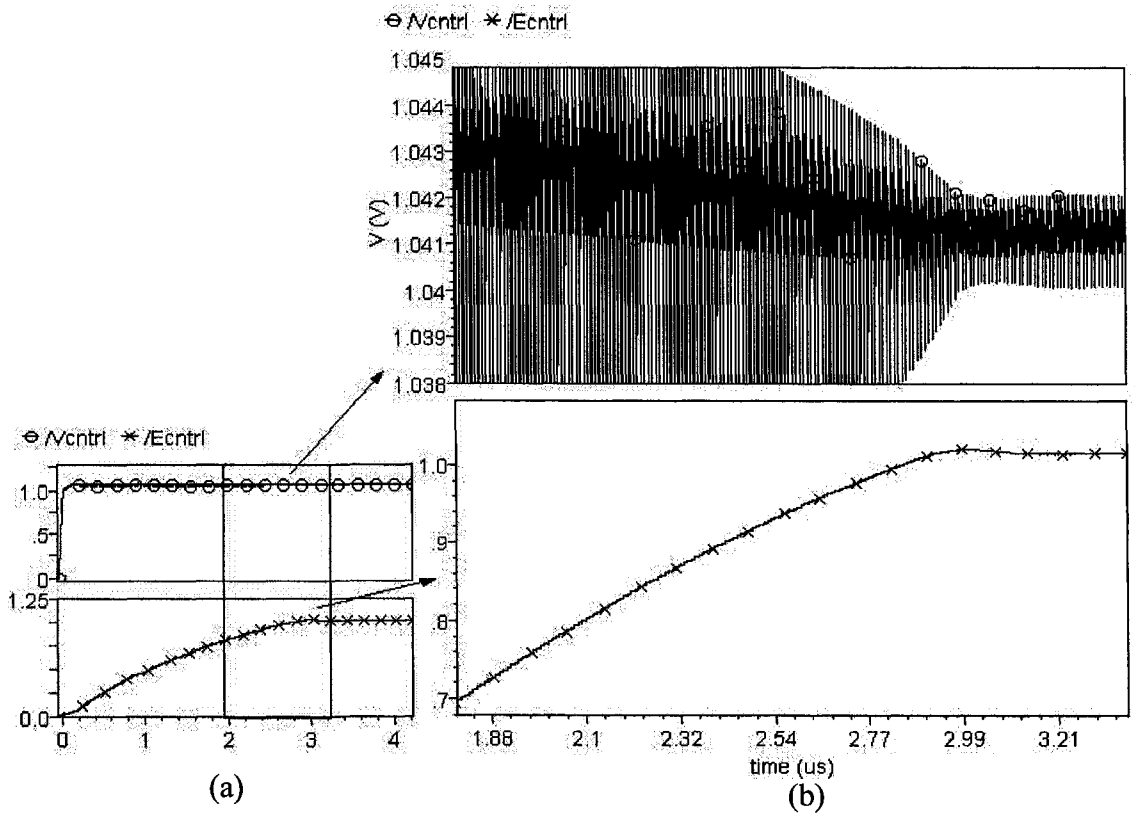


Figure 6.47 The locking process--waveform of V_{cntrl} and E_{cntrl}

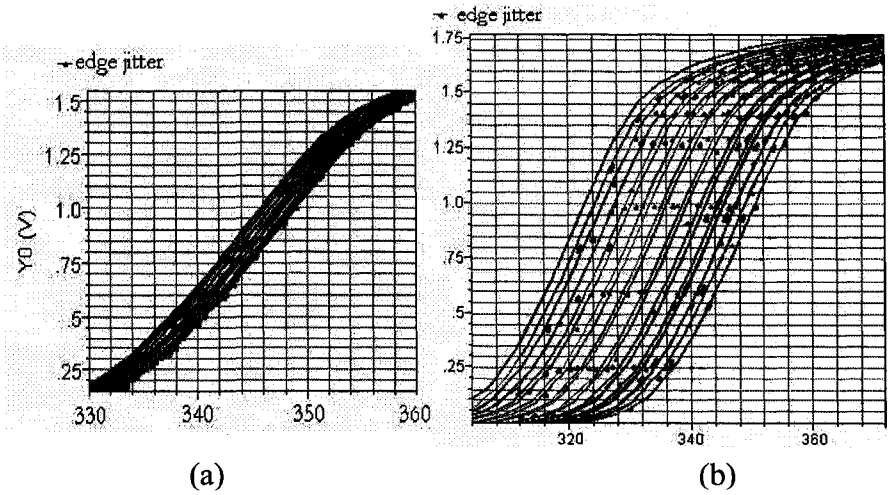


Figure 6.48 Output edge jitter at 3GHz (a) PECL enabled (b) PECL disabled

ratios. Figure 6.49 shows the simulation results with the output frequency of 2.22GHz with a division ratio of 15 from the top-level post-layout simulation. The pk-to-pk timing jitter of the output is 2.27ps, and after the loop control voltage *Vcntrl* is stable, the com-

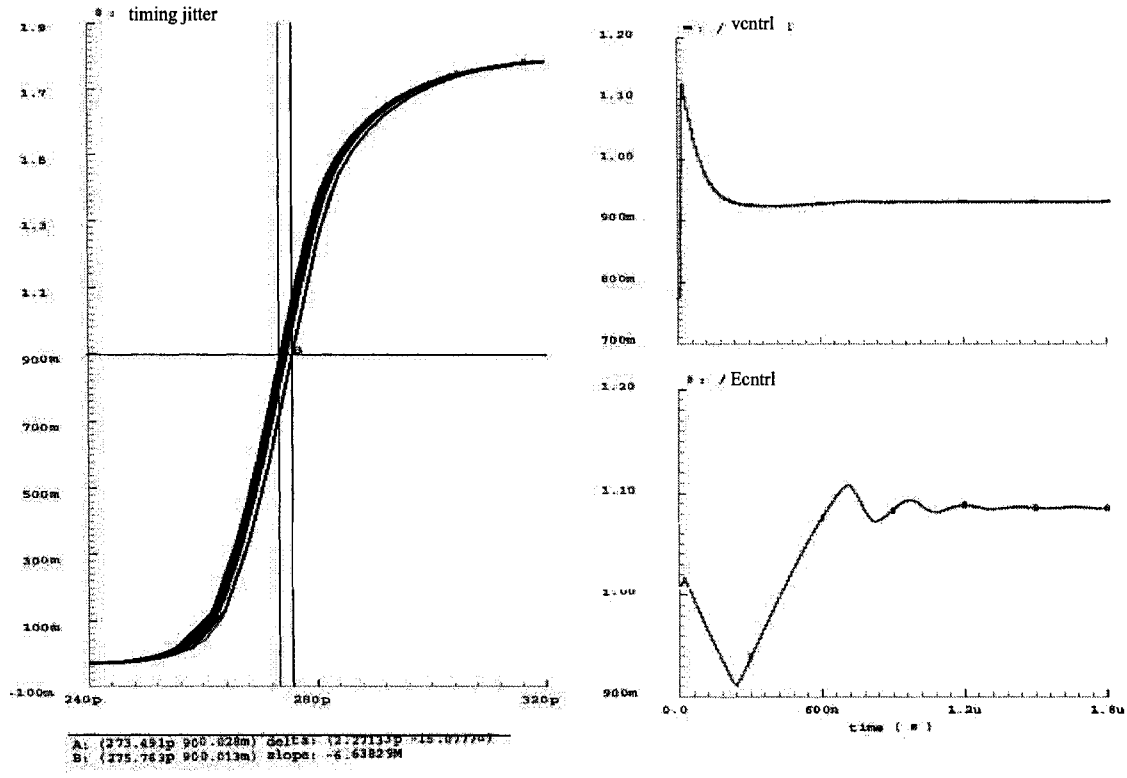


Figure 6.49 Timing jitter and the control signals at 2.22GHz output

pensation loop control voltage E_{cntrl} continues to tune the charging current finely in the CP to compensate the period error caused by the in-lock error until E_{cntrl} is stable.

6.4 Summary

In this section, the detailed CMOS implementations of the digital CDR and the DLL based programmable frequency multiplier were given. The main design considerations in circuit complexity reduction, increasing operation frequency and jitter tolerance were elaborated in the CDR design. In the DLL frequency multiplier design, the details on how to compensate the in-lock error caused by the systematic error and how to avoid harmonic locking without using extra lock detection circuitry were described in the circuit operation and implementation.

CHAPTER 7

Test Setups and Measurements

7.1 Introduction

The proposed digital CDR and the multiplying DLL frequency synthesizer were implemented and fabricated in CMOS 90nm and CMOS 0.18 μm technology respectively. This chapter covers the layout considerations, test setups and the measurement results of the two implemented circuits.

7.2 Layout considerations

Two main parts, the DLL multiple phase clock generator and the digital CDR, were implemented in the CDR test chip. The clock generator is to provide five clocks with the same frequency and five different phases, for the data sampler in the CDR, and the desired phase difference between every consecutive clocks is one fifth of one cycle to achieve maximum jitter tolerance. Consequently, the generation and transmission paths of the five clocks need to be as symmetrical as possible in the chip layout. Because the clock generator is based on a DLL, many layout considerations of the DLL design are also applied in the clock generator. However, once the incoming data is sampled by the data sampler, all remaining operations of the CDR are done in digital domain and no special layout considerations are required due to the nature of digital circuits.

In the DLL frequency multiplier design, careful considerations were taken to

improve phase noise and spur performance. In the phase detector, the two paths from f_{ref} to *UP* and from f_{rcl} to *DN* should be as identical as possible to minimize the in-lock error. The logic gates were placed symmetrically to ensure that the signal paths are matched. In the charge pump, in addition to constructing the charging and discharging paths to make them symmetrical, the circuit is surrounded by guard rings to mitigate the effects of the voltage fluctuations caused by the electromagnetic coupling in the chip.

To mitigate the effect caused by the supply noise, decoupling capacitors connected to the supply were added on the chip with the capacitance as large as possible. Taking the CDR as an example, seven metal layers were used to form the capacitor and to maximize the capacitance. The capacitor structure is shown in Figure 7.1, and the estimated total capacitance added to the chip is about 15pF.

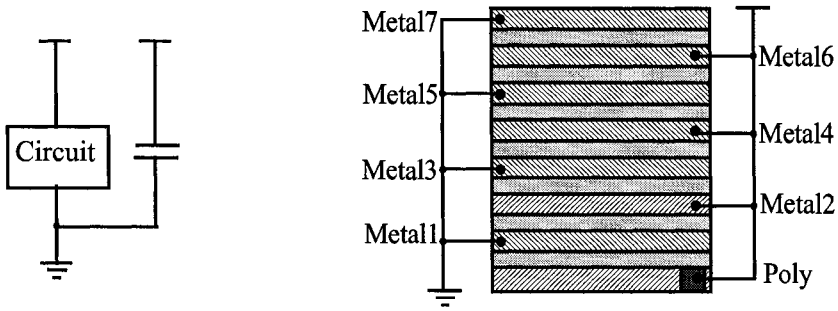


Figure 7.1 Structure of the capacitor layout

7.3 CDR floor plan and pad arrangements

The CDR circuitry presented in the previous chapters was fabricated with an die area of $mm \times 1mn$, in which, the CDR core circuit occupies $250\mu m \times 70\mu m$. Figure 7.2 shows the chip micro photo. Besides the CDR core circuit, a PRBS7 data generator, a bit error test circuit and a DLL-based multi-phase clock generator are included in the test chip to provide the test signals and to monitor the outputs of the CDR, resulting in a total circuit area of approximately $250\mu m \times 120\mu m$. In addition, a few of output buffers are used to

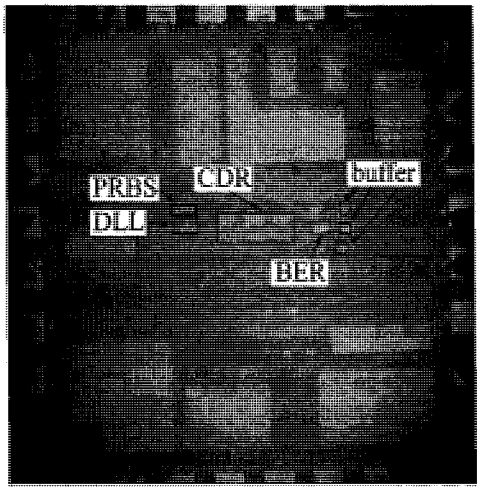


Figure 7.2 Die Microphoto

enhance the driving capability of the circuit. The 'floor plan' of the design is shown in Figure 7.3. The CDR core circuit consists of the PD, the rotating clock generation block

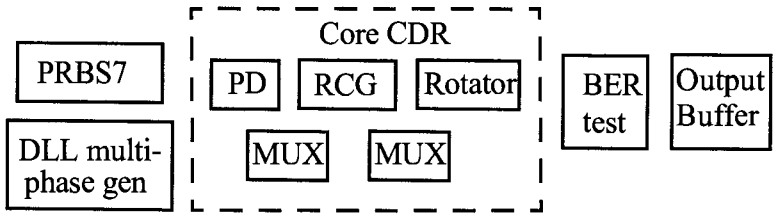


Figure 7.3 Floorplan of the CDR

(RCG), the phase rotator, and two MUXes as described in the previous chapter.

The pads arrangement of the CDR test chip is shown in Figure 7.4. There are 8 pads aligned on each side of the chip with a 150μm pitch to match the probes of two available probing cards. The top-level connections to the CDR circuitry are also shown in Figure 7.4. The upper and bottom arrays of pads are connected to the extra DLL based multiple-phase clock generator supplied by *vddDPG*. This extra DLL was not used in the CDR testing since the DLL included in the CDR circuitry was proven to function correctly. The CDR shares the same power supply *vddCDR* with the PRBS, the buffer, and the BER test circuity, and the DLL based multiple-phase clock generator draws current

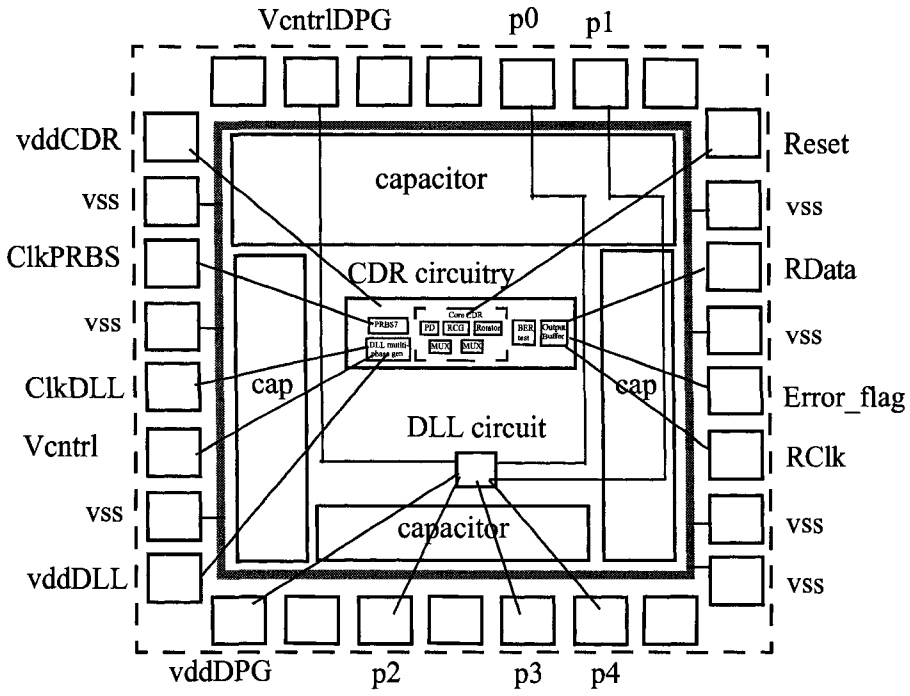


Figure 7.4 Pad arrangement

from *vddDLL*. The PRBS is clocked by *ClkPRBS*, and the reference clock of the multiple phase generator is *ClkDLL*. The control voltage of the VCDL in the DLL multi-phase generator is connected to pad *Vcntrl* where an off-chip capacitor can be connected in the case that the loop capacitor of 3pF is not large enough. During the chip testing, this pad was not connected as the loop was confirmed to operate well with the 3pF capacitor. The *Reset* signal initializes both the PRBS data generator and the phase rotator in the core CDR. In a summary, the main signals are listed below:

- *vddCDR*-- the DC power supply(1.2V) of the CDR circuitry
- *vddDLL*-- the DC power supply (1.2V) of the DLL phase generator
- *VddDPG*-- the DC power supply(1.2V) of the extra DLL phase generator
- *ClkPRBS* -- the input clock of the PRBS data generator
- *ClkDLL* -- the input clock of the multiple phase clock generator

- *Error*-- the output of the BER output error flag
- *RData*-- the output of the recovered data
- *RClk*-- the output of the re-aligned clock
- *Reset*-- the reset signal for PRBS and the phase rotator

There are no ESD protection circuits on any pad in both designs, so extreme care is necessary when handling and testing the devices. This is especially true for the pads connected to the input signal *ClkDLL* and *ClkPRBS* since they are directly connected to the gates of a CMOS inverter without any discharging path.

7.4 CDR Test setups

The CDR test setup is shown in Figure 7.5. One loose die was mounted in the

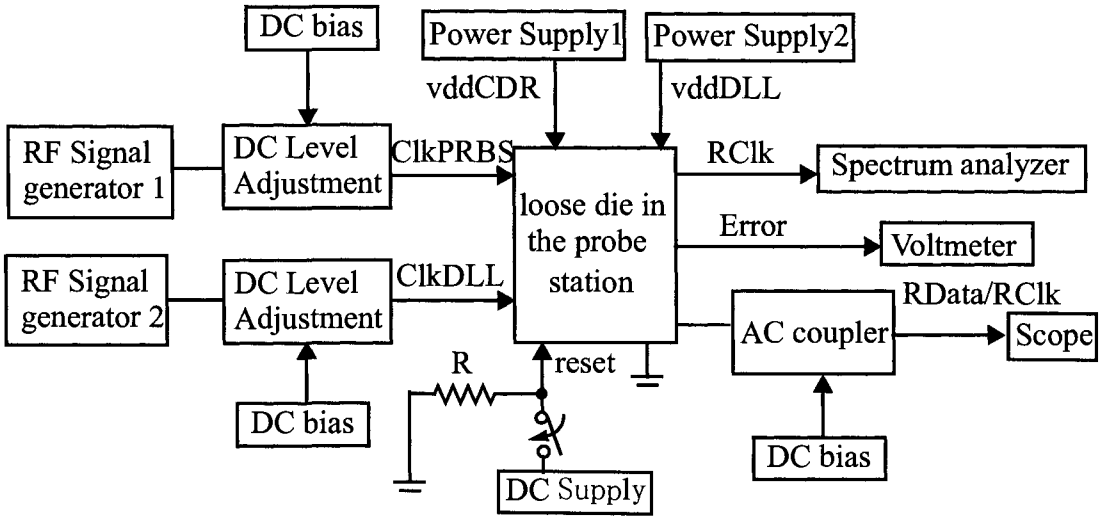


Figure 7.5 Top level test setup for the CDR circuit

probe station, and supplied by two power supplies, one for the DLL multi-phase clock generator, and one for the rest of the circuitry. Two RF signal generators were connected to the station to provide the reference clocks for the on-chip PRBS7 data generator and the DLL clock generator. DC bias circuits were added at the output of the RF generators to match the chip DC level because the DC level of the signal generator is zero while the

required DC level for the tested chip is approximately half of the supply voltage (0.6V). Similarly, at the output of the chip, another DC bias circuit working as an AC coupler was placed at the input of the oscilloscope to filter out the DC signal.

The output signal of the bit error detector, *Error,* was applied to a Voltmeter to monitor bit errors. The measured voltage keeps low if no bit error appears, and it increases with the increase of the bit error rate. The recovered data and the re-aligned clock were observed in the oscilloscope triggered with *ClkPRBS*, the clock of the PRBS data generator, and the power spectrum of the re-aligned clock was observed in the spectrum analyzer. The RF signal generator can generate phase or frequency modulated clock signal to trigger the PRBS7 to generate jittered input data. Unfortunately the frequency or phase modulation range of the RF signal generator is not wide enough to test the jitter tolerance of the proposed CDR.

In a summary, the test equipment used is listed in the table.

TABLE 7.1: Test Equipment for the CDR chip

No.	Name and function	Quantity
1	Wentworth Probe Station	1
2	Rohde&schwarz Signal Generator with output 5K~6GHz	2
3	HP 3630A DC Voltage Source, 0--6V	4
4	HP8564E Spectrum Analyzer	1
5	ZFBT-6GW DC level adjustment circuit	3
6	Oscilloscope with maximum input frequency of 40GHz	1
7	Multimeter for current measurement and observing error flag	1

7.5 Measured results of the CDR

In the testing process, the functionality of the BER test circuit was first verified by applying all ones or all zeros to the BER test circuit. This can be achieved by controlling two signals, *Reset* and *ClkPRBS*. By keeping this reset signal high, a data pattern with all ones was generated. This all-one data was recovered by the CDR and was applied to the

BER test circuit. Theoretically, continuous error pulses would be generated by the BER test circuit in this case. In the testing, a high voltage of 1.025V was measured at the error output *Verror*, which confirms a correct operation of the BER test circuit. An all-zero data pattern was also generated by resetting the PRBS data generator and manually providing clock cycles at *ClkPRBS*. In this case, no error pulse was observed because the error voltage *Verror* kept low (0.04V) as expected, which again verifies the correct function of the BER circuit. To double check the function of the chip, both *ClkPRBS* and *ClkDLL* were tuned to be 2.5GHz, so the data and sampling clock are at the same frequency, the error voltage was observed to be 0.04V showing no bit error was detected.

To observe the data tracking behavior, the *ClkPRBS* is tuned to be 2.5GHz to generate the data at 2.5Gbps, and *ClkDLL* is 2.52GHz, with the DLL multi-phase clock generator operated at a 1.0V supply due to the limited delay in the VCDL. The recovered data *RData* and re-aligned clock *RClk* were monitored with the oscilloscope triggered by *ClkPRBS,* which is used to generate the input data. An eye diagram together with *RClk* is shown in Figure 7.6. With the constant frequency offset between the *ClkPRBS* and

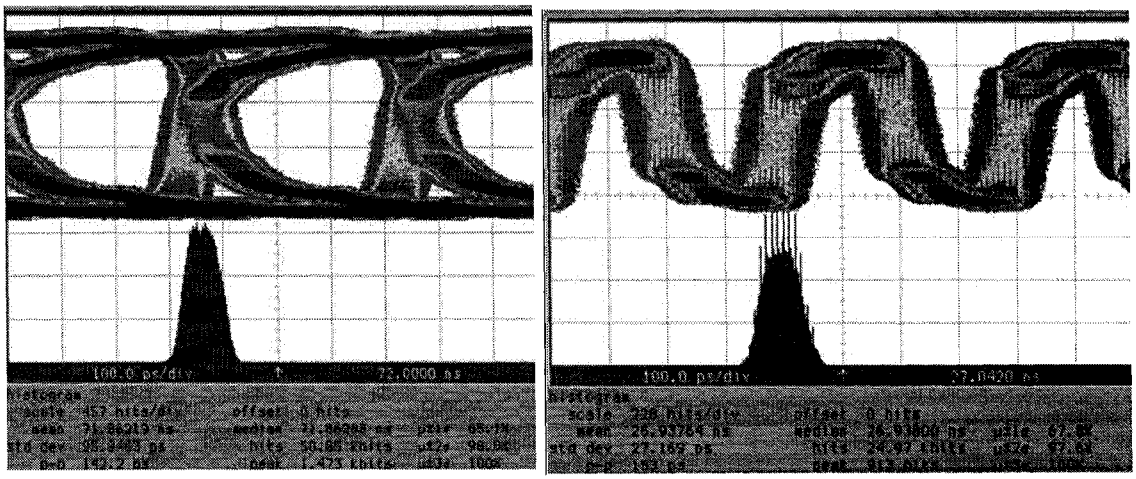


Figure 7.6 Rdata and RClk at 2.5Gbps

ClkDLL, the CDR tracks the incoming data by keeping updating its phase rotator, resulting in 0.2UI phase jumps. The measured result shows that an *rms* edge jitter σ of the re-aligned clock is 27ps, and the value of 3σ is 81ps, which is the major jitter at the clock

edges and it agrees with the theoretical value of 0.2UI. The similar edge jitter of the recovered data was also observed from the recovered data eye diagram as expected.

To further illustrate the loop phase tracking, the power spectrums of the re-aligned clock, when the CDR is tracking (a) or is not tracking (b), are given in Figure 7.7. When the CDR is tracking the incoming data, the center frequency of the re-aligned clock is

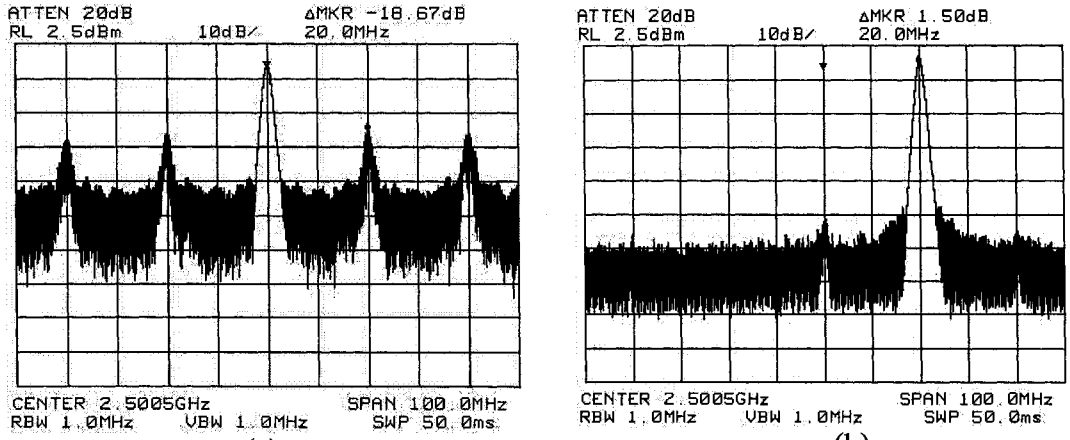


Figure 7.7 Spectrum of RClk with loop tracking and no tracking

measured as 2.5GHz, which is the same as the *ClkPRBS* and confirms that the loop tracks the frequency difference correctly. Ideally, the first side spur is expected at 100MHz as analyzed in Chapter 5. The measured power spectrum shows some spurs at 20MHz frequency offset from the center frequency because of the non-ideal sampling phase space. When the reset signal is kept high, the one-hot signal is locked in one state and the loop does not track any frequency. In this case, the spectrum of the re-aligned clock shows a center frequency of 2.52GHz as shown in the figure. No harmonic components in the spectrum due to the absence of frequency tracking, and only the DLL clock frequency of 2.52GHz was observed.

The CDR was also confirmed to track the phase difference from data rate of 2.0Gbps to 3.5Gbps. Because of the limited delay range of the delay cells in the DLL clock generator, the supply voltage of the multi-phase DLL clock generator was changed to extend the operation frequency range. Figure 7.8 shows the eye diagram of the recov-

ered data and the re-aligned clock both triggered by *ClkPRBS* at 2.2Gbps and 2.7Gbps

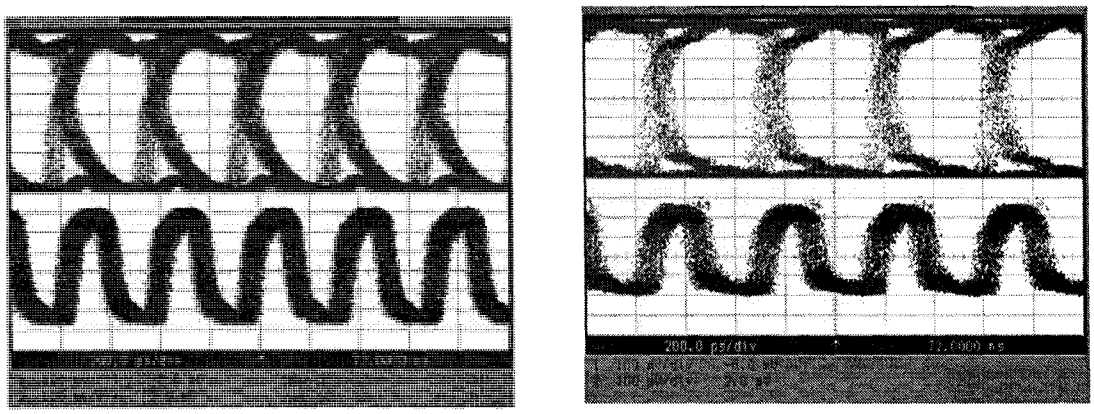


Figure 7.8 RData and RClk at 2.2Gbps and 2.7Gbps

with a frequency difference of 4.5Kppm and 3.7Kppm respectively. Figure 7.9 shows time domain waveforms of *RData* and *RClk* at 3.0Gbps with a frequency difference of 6.66Kppm.

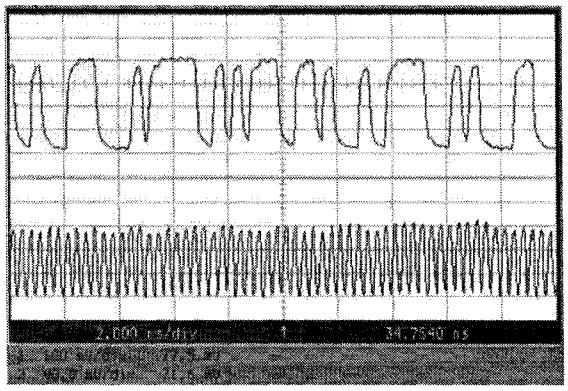


Figure 7.9 Rdata and RClk at 3.0Gbps

Figure 7.10 gives the re-aligned clock *RClk* to illustrate the phase shift when the loop tracks the phase difference captured from the scope. The edge indicated in an arrow was observed to shift gradually to the next edge on the right and then jump back to its original place, which means a threshold error was encountered and the loop adjusted its data sampling phase by a discrete phase step.

To test the jitter tolerance of the CDR, the clock used to generate the PRBS data is modulated by a sine wave so that a jittered incoming data can be generated. For a certain

Figure 7.10 Phase tracking behavior observed in *RClk*

jitter frequency (the modulation frequency of the *ClkPRBS*), the maximum jitter amplitude that the CDR can tolerate, which is defined as the jitter tolerance, can be measured by examining the output of bit error detection circuit. Unfortunately, the phase or frequency modulation range of available RF signal generators in the department are not wide enough for the CDR to result in bit errors. For example, the maximum phase modulation of the HP 83640B RF generator is 12.566rad at 50kHz. By modulating *ClkPRBS* at 2.0GHz with such maximum modulation, a jittered data was generated and recovered by the CDR without any bit error.

Fortunately, the low-frequency JT was measured with the aid of Equation (7.1) by measuring the maximum tolerable frequency offset between *ClkPRBS* and *ClkDLL*. One set of measured figures was obtained as follows. The *ClkPRBS* of 2.4GHz is applied, and *ClkDLL*=2.4GHz, the error output *Verror* is observed to be 0.072V. By increasing *ClkDLL* gradually, and the *Verror* was read as follows,

ClkDLL=2.41GHz, *Verror*=0.073

ClkDLL=2.42GHz, *Verror*=0.073V

ClkDLL=2.430GHz, *Verror*=0.073V

ClkDLL=2.440GHz, *Verror*=0.072V

ClkDLL=2.450GHz, *Verror*=0.073V

ClkDLL=2.458GHz, *Verror*=0.073V

ClkDLL=2.460GHz, *Verror*=0.077V

The error output, *Verror*, starts to increase when the *ClkDLL* increases to 2.46GHz, therefore the maximum frequency difference the loop can track is approximately 58MHz. With this measured maximum tolerable frequency difference, the low frequency jitter tolerance can be calculated by examining the maximum jitter tracking speed of the CDR. For a sinusoid jitter with a jitter frequency of f_j (Hz), and an amplitude of A_j (UI), the maximum jitter change speed is, $A_j \cdot 2\pi \cdot f_j$ UI/s, while the measured maximum tolerable jitter change speed is, 58M UI/s. The low-frequency jitter tolerance can thus be derived as below,

$$JT_{p-p}(UI) = \frac{58 \times 10^6}{\pi \cdot f_j} = \frac{18.5 \times 10^6}{f_j} \qquad (7.1)$$

If the normalized jitter frequency (F_j=f_j/*Fdata*) is used, the above equation can be rewritten as below,

$$JT_{p-p}(UI) = \frac{0.0077}{F_j} \qquad (7.2)$$

The jitter tolerance can thus be plotted as illustrated in Figure 7.11. At the normalized jitter frequency $F_j = 0.001$, the experimental jitter tolerance is 7.7UI, which is very closed to the simulated value of 9.7UI. Since the jitter tolerance of the CDR is far above the maximum modulation phase and frequency of the only available RF signal generator, the high-frequency jitter tolerance cannot be measured directly. It was estimated by assuming the same degrading as the low frequency jitter tolerance with respect to the simulated results. The estimated high frequency jitter tolerance value is 0.68UI. Figure 7.11

shows the jitter tolerance performance from the simulated value as well as the experimen-

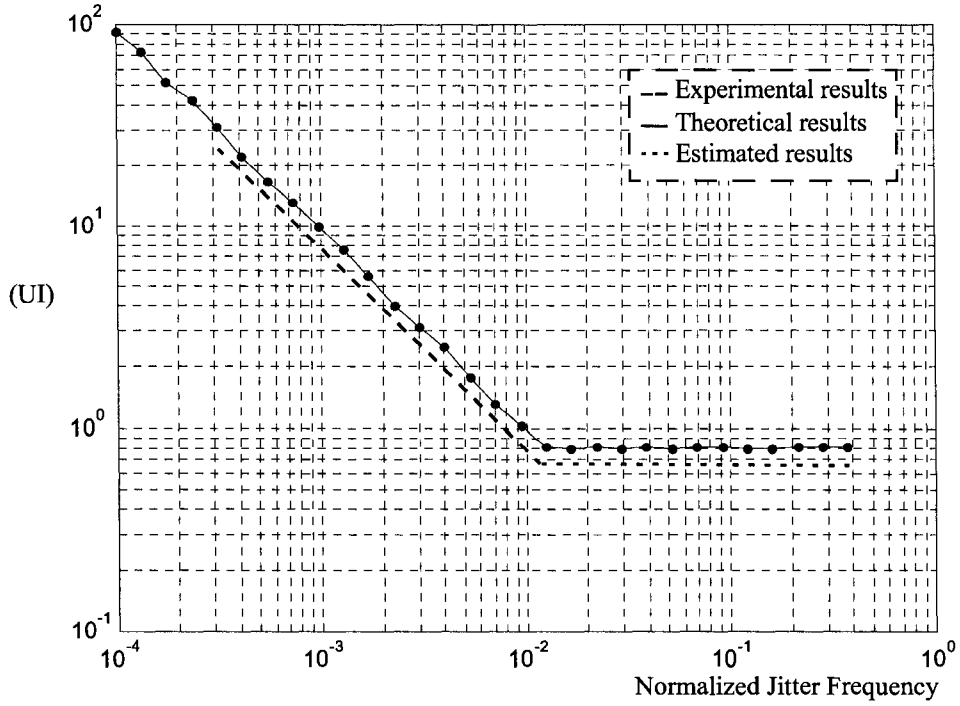


Figure 7.11 Theoretical jitter tolerance of the proposed decision technique

tal jitter tolerance which is from measurement in the low jitter frequency and estimation in the high-frequency jitter.

7.6 Power consumption and performance summary

Since the CDR core circuit shares the same power supply with the rest circuitry in the chip except for the DLL based multi-phase clock generator, a large portion of the measured current is from the five inverter-based output buffers.

The post-layout simulation shows that at 3.125Gbps, the core CDR consumes 3.227mA, and the remaining circuits excluding the DLL consumes 4.73mA, while the DLL consumes 3.434mA. At 2.5Gbps, the core CDR consumes 2.57mA, and the DLL consumes 2.63mA, while the remaining circuits including the output buffers and the PRBS data generator consumes 3.637mA. According to these currents values, the currents of core CDR at both baud rates are approximately 40.5% of the current derived from

vddCDR.

In the measurement, the current consumption of the circuits excluding the DLL was measured by only connecting the error signal to the Voltmeter, and the left two output pads have no load. According to the post-layout simulation results, the current consumption of the core CDR is about 40.5 percent of the current that the whole circuits excluding the DLL consumes, thus the core CDR current consumption can be estimated to be 40.5% of the measured current on *vddCDR*. Figure 7.12 shows the measured chip current excluding the DLL current, and the estimated core CDR current. The core CDR consumes less than 5.3mA at 3Gbps. This current estimation result tends to be larger than the actual current consumption because the current to drive the output pads was not included in the post-layout simulation.

The performance is summarized in Table 7.2 with the reported digital CDR performances for comparison. With the low-complexity CDR, the power and area are reduced, and the jitter performance is well comparable to the other reported CDRs.

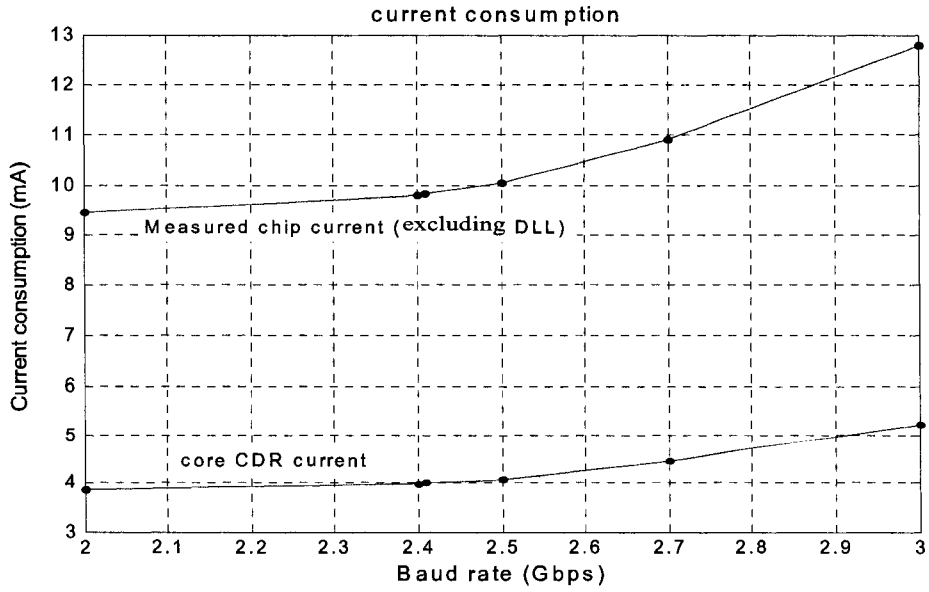


Figure 7.12 Current consumption vs. baud rate

TABLE 7.2: **Performance Comparison**

	[3]	[5]	[6]	This work
Technology	0.13μm CMOS	0.18μm CMOS	0.5μm CMOS	90nm CMOS
Supply	1.2	1.8	2.7-4.0	1.2
Data rate	3.125 G	2.5 G	4 G	2.0-3.5
Jitter tolerance	0.67UI	0.7UI	N/A	0.68UI
Acquisition time	40,80 data bits	16 data bits	24 data bits	1 data bit
Power (mW)	50 per channel	50	about 175	4@2.5Gpbs
Area (mm^2)	1.5 x 1.3	0.02	3 x 3	0.0175

7.7 DLL frequency synthesizer floor plan and pad arrangements

The floor plan of the DLL based frequency synthesizer is shown in Figure 7.13.

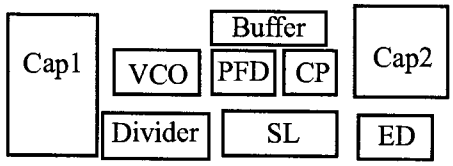


Figure 7.13 Floorplan of the DLL frequency synthesizer

The VCO includes the VCDL and the MUX, and the PFD is aligned with the CP for better symmetrical paths in *UP* and *DN* signals. Since the signal paths are not timing critical, the divider, the switching logic (SL) and the error detector (ED) were placed below. A capacitor Cap2 with a capacitance of 2pF is connected to the ED to provide the integration path for the finely tuning control signal. Cap1 is the on-chip loop capacitor filter of the main loop with a capacitance of 5pF.

The design was packaged in a 24-leaded CFP (ceramic flat pack) which has a small cavity (3.5mm x 5.5mm) and is suitable for high frequency (up to 2GHz) designs. The pads arrangement is shown in Figure 7.14. There are two designs in the chip, design version A and design version B. The main difference is that in version B, the loop capacitor *Cap1* and the capacitor connected to the ED, *Cap2,* are off-chip, which are connected

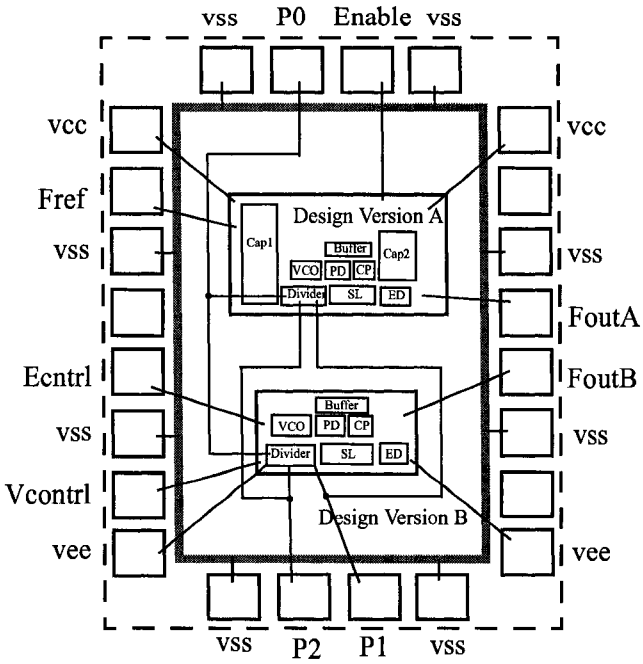


Figure 7.14 Pad arrangement

to port *Vcntrl* and *Econtrl* respectively. Both design versions were proven to function correctly, and the measurements are mainly focused on version A. To program the division ratio, three index are connected to port *P0*, *P1*, and *P2*. Signal *Enable* was set to high or low to enable or to disable the compensation loop for comparison. The low-frequency reference signal is from the input port *Fref*, and the synthesized high-speed output signal was obtained from the output port *FoutA*.

In the test process, a DC power supply was used to provide the 1.8V DC supply voltage. The reference signal was generated from an RF signal generator, and an oscilloscope was used to observe the input and output waveforms as well as the jitter histogram. The phase noise performance and the spurious output power level were measured with a spectrum analyzer. To measure the current consumption, a multi-meter was used in the test process. Equipment designations are listed in Table 7.3. The top level test setup is shown in Figure 7.15. There is DC adjustment at the output of signal generator to match the DLL DC level because the DC level of the signal generator is zero while the required DC level

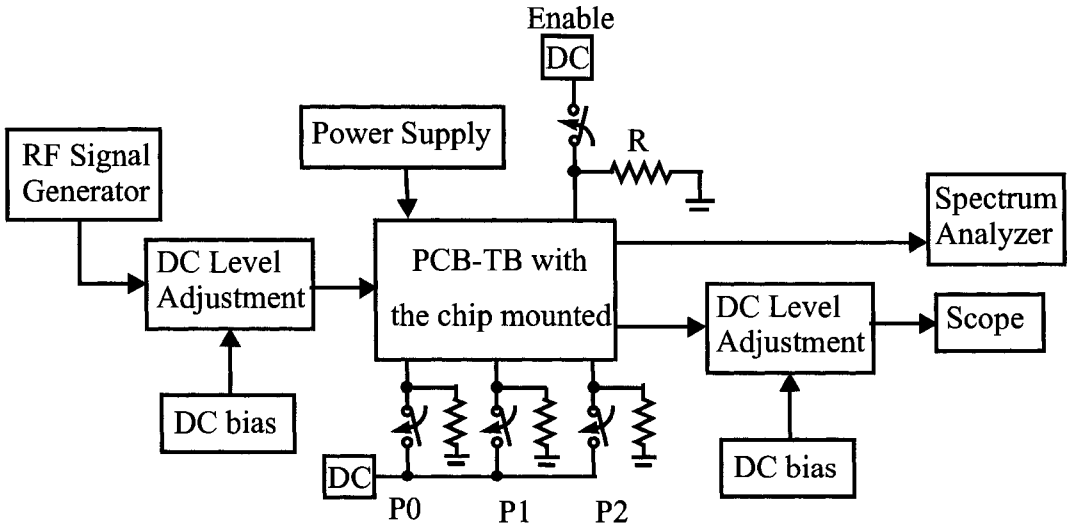


Figure 7.15 Top level test setup

for the tested chip is about VDD/2 (0.9V), which is the same reason as it was used in the CDR chip testing. Changing this bias DC voltage can actually change the duty cycle of the signal inside the VCDL since the output signal of the signal generator is sinusoid wave instead of an ideal square wave. The programmable index are connected to three DC voltage sources with switches, and programmable division ratios can be obtained by turning on or off these switches. The compensation loop is enabled by signal *Enable* which is connected to a similar switch as used in the programmable index.

TABLE 7.3: Test Equipment for the DLL frequency multiplier chip

No.	Name and function	Quantity
1	Rohde & schwarz Signal Generator with output 5K~6GHz	1
2	HP 3630A DC Voltage Source, 0--6V	4
4	HP8564E Spectrum Analyzer	1
5	ZFBT-6GW DC level adjustment circuit	2
6	Oscilloscope with maximum input frequency of 40G/s	1
7	Multi-meter for current measurement	1

7.8 Measurement results

The timing jitter and phase noise performance are of importance in the frequency

synthesizer measurement. By enabling the compensation loop, the jitter histogram was obtained at 2.162GHz as shown in Figure 7.16. The cycle-to-cycle RMS edge jitter is 1.6ps and the peak-to-peak value is 12.9ps at the 2.16GHz output frequency with the divi-

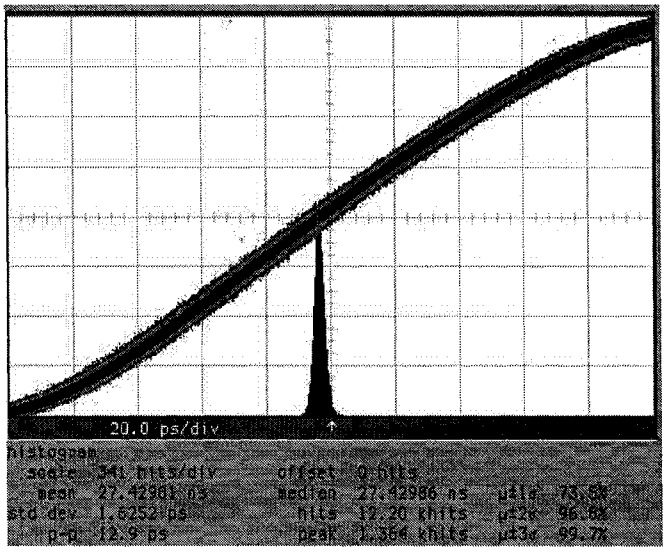


Figure 7.16 Jitter histogram at 2.162GHz

sion ratio of N = 20. The phase noise and the waveform of the same output are shown in Figure 7.17. The measured phase noise is -110dBc/Hz at 100kHz offset, and -106dBc/Hz

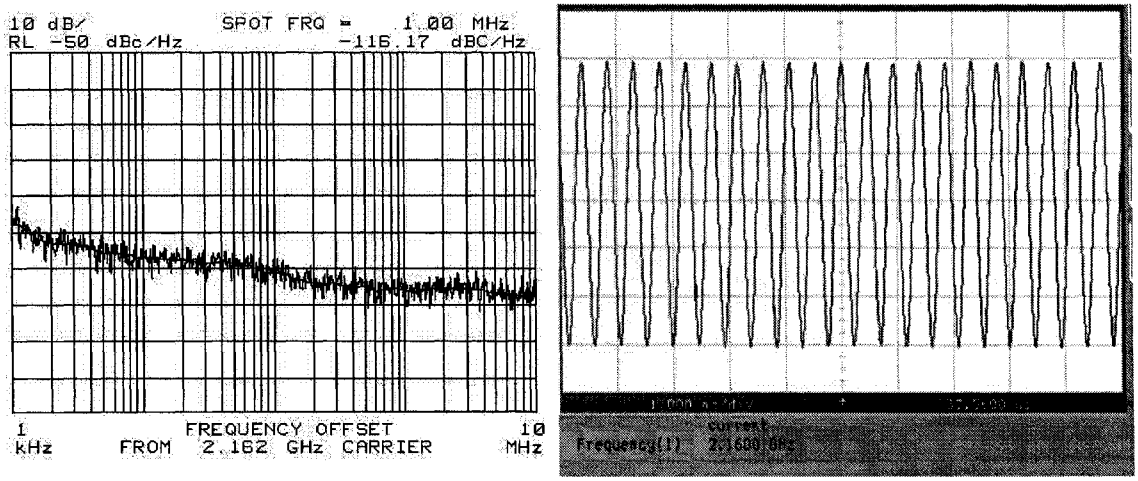


Figure 7.17 Phase noise and waveform at 2.162GHz

at 10kHz offset. The phase noise does not change much when the compensation loop is

enabled.

Figure 7.18 presents the measured jitter histogram with the spur reduction disabled and enabled. At 2.9GHz, with the multiplication ratio N = 14, when the spur reduction

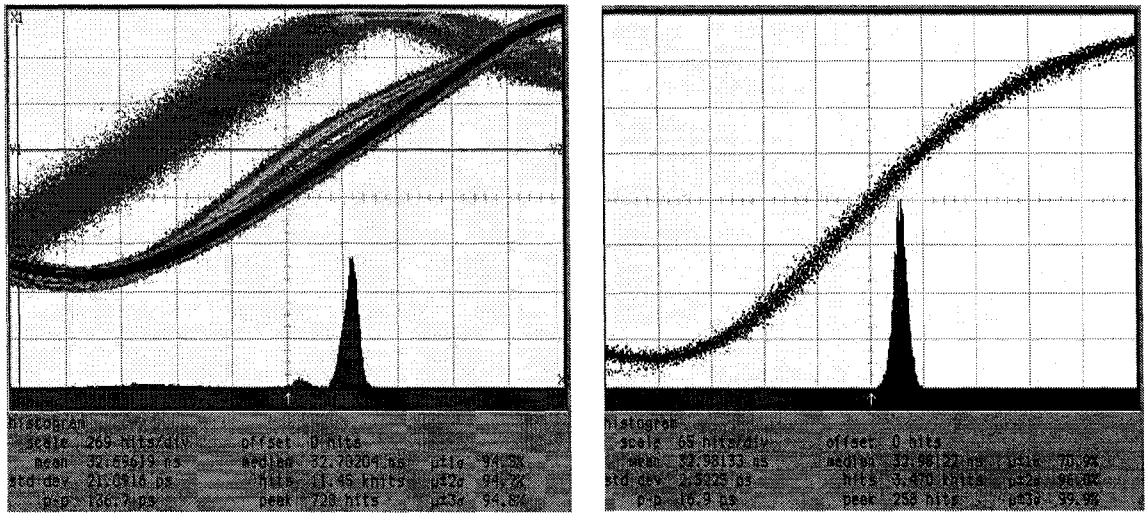


Figure 7.18 The jitter histogram at 2.9GHz

technique is disabled, the measured cycle-to-cycle RMS edge jitter is 21ps and the pk-to-pk jitter is 136ps. When the technique is enabled, the measured jitter is 2.5ps (RMS) and the pk-pk timing jitter is 0.049UI which is suitable for the data transmission. It confirms that the compensation loop is able to dramatically reduce the in-lock error which passes directly to the output as timing jitter.

To confirm the phase noise performance of the circuit, the circuit was tested at more frequencies with more division ratios. Figure 7.19 shows the measured phase noise at the output of 1.786GHz and 1.88GHz. With the division ratio of 17 and 18, the phase noise are -117dBc/Hz and -116dBc/Hz respectively, both at the offset of 1MHz. Figure 7.20 shows the measured phase noise at the output of 2.756GHz with the division ratio of 13 and the phase noise at 1.996GHz with a division ratio of 15. The measured phase noise at 2.756GHz is -119.8dBc/Hz at 1MHz offset, and approximately -110dBc/Hz at 100kHz offset. At 1.996GHz, the measured phase noise is -116dBc/Hz at 1MHz offset. Similar phase noise performance was also observed at other frequencies test, all of which

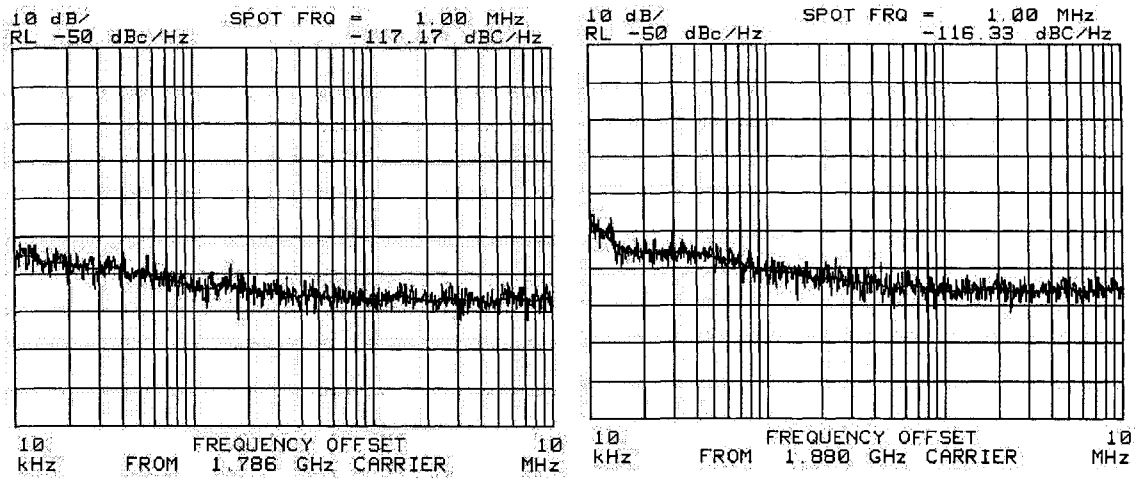


Figure 7.19 Phase noise at 1.786GHz, 1.88GHz

confirms the noise performance of the circuit.

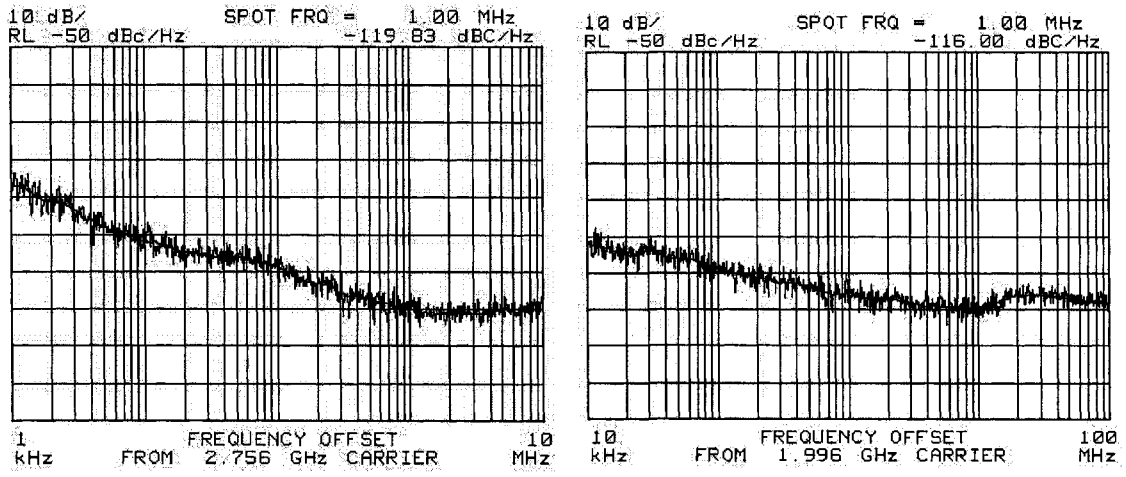


Figure 7.20 Phase noise at 2.756GHz, ratio of 13 and 1.996GHz, ratio of 15

Figure 7.21 shows the output clock at 800MHz with the division ratio of 13 and the reference is less than the minimum reference frequency. Because the reference period is beyond the maximum delay of the VCDL, the DLL is not able to lock.

Figure 7.22 presents the measured output spurious power level with a carrier frequency of 1.2GHz with the input frequency of 64MHz and ratio of 19. With the compen-

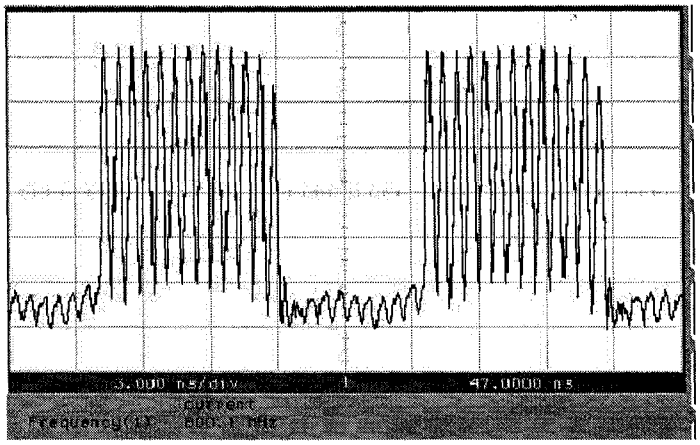


Figure 7.21 Waveform at frequency of 800MHz

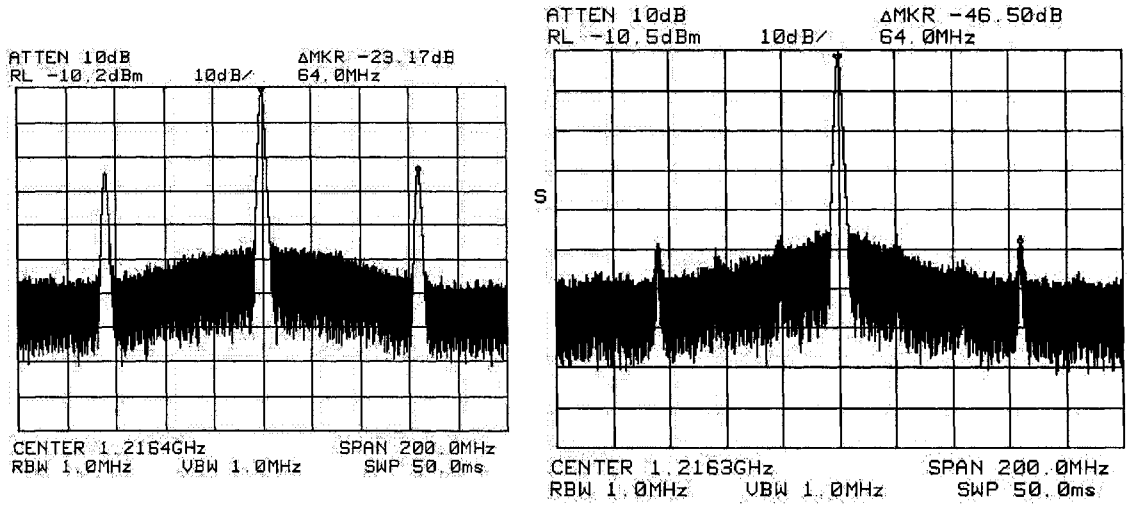


Figure 7.22 Spurious output at 1.216GHz

sation loop disabled and enabled, the spurious tones at the reference frequency are -23.17 dBc and -46.5dBc respectively, and a significant spur reduction of 23dB was observed. However, it was found that the power supply of the output driver circuit is not clean and the substrate and the supply coupling likely prevents the spurious level from further reducing because all circuit components share the same substrate and power supply in the test chip. Consequently, the actual spurious performance without such coupling is likely better than the measured -46.5dBc and the spur reduction is likely more than 23dB. To prove this statement, several designs were reviewed and the spur performances were compared. In

the reduced in-locked DLL frequency synthesizer [30], a -46.17dB reference spur was measured. The digital CDR test chip was tested by supplying two clock signals with different frequencies (3GHz and 150MHz respectively). In the CDR test chip, two clock signals go through the test chip by two separate paths and two separate output buffers without any deliberate coupling between them. The power spectrum of the signal with frequency of 3GHz was measured as shown in Figure 7.1, in which a spur level of -45dB at 150MHz frequency offset was observed.

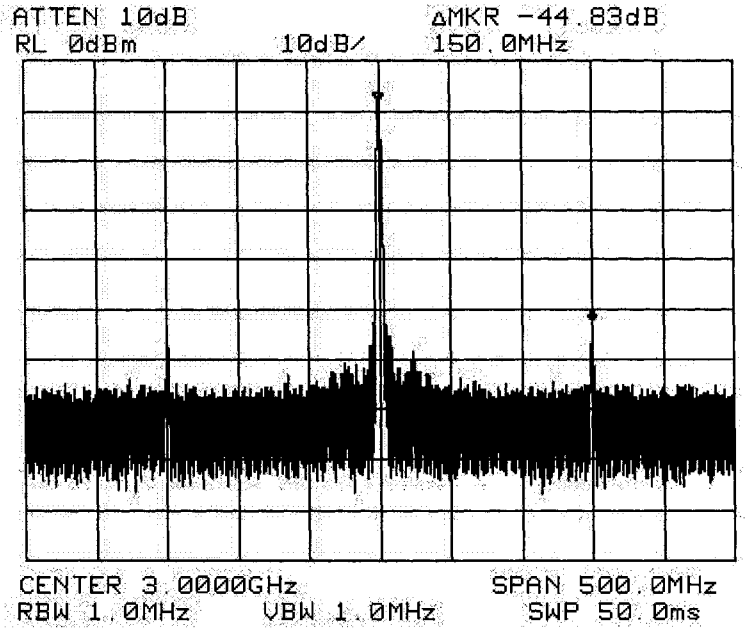


Figure 7.1 The measured spur from the digital CDR test chip

Approximately the same spur levels ($-44 \pm 2dB$) were observed in two different designs listed above as well as the DLL in this work, and it is reasonable to believe that those spurs are dominated by a common factor among those designs, and it is that different parts of the circuit in each chip share the same power supply. Consequently, those spurs are dominated by the strong AC coupling through the power supply and the ground in those designs. Actually, in [39], the measured spurious performance of the actual output clock was not given because the author claims that it was discovered that the power supply of the clock output drive was unclean, and hence the clock output had the reference fre-

quency coupling into it through the power supply.

The chip microphoto is shown in Figure 7.23, and the floor plan of the chip

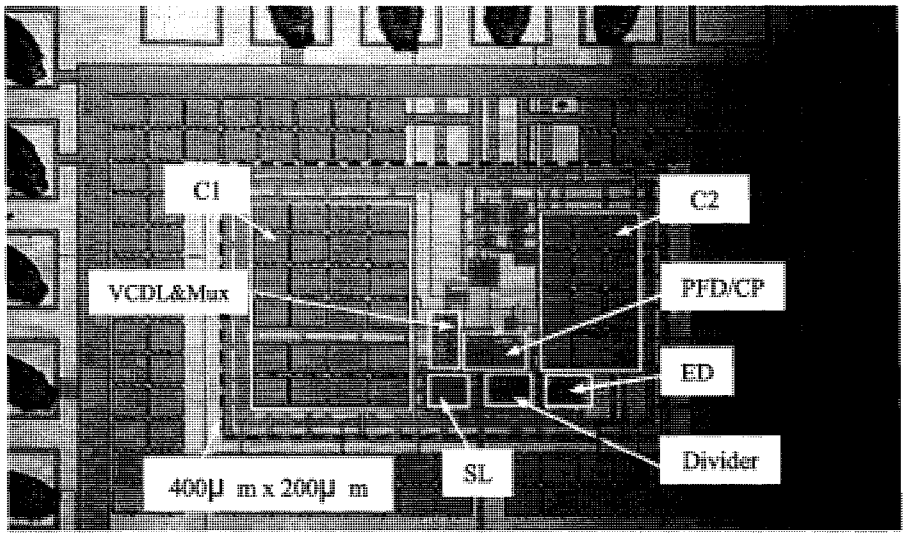


Figure 7.23 Chip microphoto

including the dimension can be observed from the figure. Supplied by a 1.8V DC source, the current consumption including the output buffers is increased with the increase of the operating frequency, and the maximum current is less than 12mA, and the circuit consumes less than 22mW. The current consumption can be further reduced to almost half of the measured value if static DFFs consisting of inverters and pass gates were used in the switching logic. The output frequency range is from 900MHz to 2.9GHz, and division ratios of from 13 to 20 were observed from the measurement. Compared with the measured results reported in [33] and [34], the measured performances of the circuit are summarized in Table 7.4. Obviously, with the PECL enabled, the performance is improved.

TABLE 7.4: **Summary of measured performance**

Items	[33]	[34]	This design
Supply voltage	3.3 v	1.8 v	1.8 v
Division ratios	1-8	4, 6, 8, 10	13,14,15,16,17,18,19,20
Output frequencies	120MHz-1.8GHz	200MHz-2GHz	900MHz-2.9GHz
Phase noise	N/A	N/A	-110dBc/Hz at 100kHz offset
Timing jitter	1.8ps at 1.8GHz (rms) 13.2ps at 1.8GHz (pk-pk)	1.62ps at 2GHz (rms) 13.1ps at 2GHz (pk-pk)	1.6ps at 2.16GHz (rms) 12.9ps at 2.16GHz(pk-pk)
Spurious tones	N/A	-37dB at 2GHz	-46.5 dB at 1.2 GHz (PECL enabled) -23.17 dB at 1.2GHz (PECL disabled)
Power	86 at 1.6GHz	12 at 2GHz	19.8 at 2.16 GHz output
Loop capacitor	N/A	N/A	C1: 5pF (on-chip), C2:2pF (on-chip)
Active area	$0.07mm^2$	$0.05mm^2$	$0.07mm^2$
Technology	$0.35\mu m$ CMOS	$0.18\mu m$ CMOS	$0.18\mu m$ CMOS

Two newer contributions of the DLL-based frequency multipliers [38] [39] have been published. The design in [38] actually refers to this work (reported in [43]) and employs the similar idea of the proposed period compensation method to compare the period ending with the injected clock edge with other periods. A -59.6 dBc reference spur was achieved with further digital averaging in the period error feedback path, and less reference signal coupling level. The achievement of [38] further confirms the feasibility of the proposed period error compensation method. Another design was reported in [39], and a -70dBc reference was achieved for an output frequency of 176MHz, which is equivalent (for the same amount of in-lock timing error) to a -50dBc (-70+20*log(10)) reference spur for an output frequency of 1.76GHz. Although this result is slightly better than this work, the measured spurious performance doesn't count for any coupling of the power supply or substrate. The spur of -70dBc was measured indirectly and the spurious performance measured from the actual output clock was not given in the paper because the author claims that strong coupling was discovered in the design.

7.9 Summary

This chapter presents the layout considerations, test setups and measurement results of the two designed circuits, the CDR with a digital threshold decision technique and the anti-harmonic, programmable DLL based frequency multiplier with the period error compensation loop. The measured results verified the theoretical analysis and the transistor-level implementation in both designs. The digital threshold technique and implementation were confirmed in the CDR design, and the timing jitter and spur reduction with the period error compensation loop were verified in the DLL frequency multiplier design. In addition, explanation and analysis were also given in some measured results.

CHAPTER 8

Conclusion

8.1 Conclusion

The rapid development of the CMOS technology makes the trend of digitizing the RF transceiver possible. The all-digital CDR circuits reported recently proves that it is better choice especially with respect to the portability than the analog CDR circuits. In this thesis, two critical blocks, CDRs and clock generators in the transceiver were first reviewed. Most existed CDR circuits consume significantly large power and area, and require long time to acquire locking. The two proposed designs, a low-complexity, all-digital CDR with high jitter tolerance performance, and an anti-harmonic DLL based frequency multiplier with reduced spurious output power are analyzed and verified with model simulation, post-layout simulation and finally chip measurements respectively.

This proposed CDR, which enables fast acquisition, high jitter tolerance performance was analyzed, and its functionality as well as its jitter tolerance performance was confirmed by the model simulation from Simulink and script respectively. A CMOS implementation in 90nm technology was done to verify the theoretical performance with some new circuit ideas to further reduce the circuit complexity, especially in the PD design and the rotating clock generation design. Measured results were given to verify the proposed CDR architecture and its CMOS implementation. Compared with the most recently reported all digital CDR circuits, this design consumes less power, and exhibits

higher low-frequency jitter tolerance. Its estimated high-frequency jitter tolerance is well comparable to the best reported counterparts, the all-digital CDR circuits.

In addition to the CDR circuit in the receiver, the frequency multiplier that provides clock to both the transmitter and the receiver is also an important building block. In this work, DLL-based frequency multipliers were reviewed due to their advantages over the PLL based ones. There are two types of DLL frequency multiplying technique, both of which exhibit high spurious output power mainly due to the stage mismatch and the in-lock error. The harmonic locking problem in the DLL was normally resolved either by circuit initialization at power on or by complex additional locking detection circuitry. In the proposed DLL frequency multiplier, a period error compensation loop was proposed to reduce the in-lock error, and model simulation verified the efficiency of the compensation loop. A CMOS implementation was done with measured results confirming its functionality and efficiency. With a new switching scheme and the compensation loop, the circuit was verified to reduce the period error significantly and no initialization at power on is needed without any locking detection circuitry.

References

[1] J. Kim, D.-K. Jeong, "Multi-Gigabit-Rate Clock and Data Recovery Based on Blind Oversamping", IEEE Communications Magazine, Pages: 68-74, December 2003.

[2] M. H. Perrott, "Clock and Data Recovery (CDR) Design Using The PLL Design Assistant and CppSim Programs", Tutorial from http://www-mtl.mit.edu/~perrott.

[3] Y. Choi, D.-K. Jeong, W. Kim, "Jitter Transfer Analysis of Tracked Oversampling Technique for Multigigabit Clock and Data Recovery", IEEE Transaction on Circuits and Systems II, Vol. 50, No.11, Pages: 775-783, November 2003 .

[4] H.-T. Ng, R. Farjad-Rad, M.-J. E. Lee, W. J. Dally, T. Greer, J. Poulton, J. H. Edmondson, R. Rathi, R. Senthinathan, "A Second-Order Semidigital Clock Recovery Circuit Based on Injection Locking", IEEE Journal of Solid-State Circuits, Vol. 38, No.12, Pages: 2101-2108, December 2003.

[5] K.Kishine and H.Onodera, "Acquisition-time estimation for over 10-Gbits/s clock and data recovery ICs", Electronics Letters, Vol. 41, No. 23, Pages: 1273-1275, November 2005.

[6] H. Djahanshahi, C.A.Salama, "Differential CMOS Circuits for 622-MHz/933-MHz Clock and Data Recovery Applications", IEEE Journal of Solid-State Circuits, Vol. 35, No. 6, Pages: 847-855, June 2000.

[7] A. Pottbacker, U. Langmann, and H. Schreiber, "A Si Bipolar Phase and Frequency Detector IC for Clock Extraction up to 8 Gb/s", IEEE Journal of Solid-State Circuits, Vol. 27, Pages: 1747-1751, Decmeber 1992.

[8] A. Pottbaker, U. Langmann, "An 8 GHz Silicon Bipolar Clock-Recovery and Data-Regenerator IC", IEEE Journal of Solid-State Circuits, Vol. 29, No. 12, Pages: 1572-1576, December 1994.

[9] Y. M. Greshishchev, P.r Schvan, J. L. Showell, et al, "A Fully Integrated SiGe Receiver IC for 10-Gb/s Data Rate", IEEE Journal of Solid-State Circuits, Vol. 35, No.12, Pages: 1949-1957, December 2000.

[10] H. Nosaka, K. Ishii, T. Enoki, T. Shibata, " A 10-Gb/s Data-Pattern Independent Clock and Data Recovery Circuit with a Two-Mode Phase Comparator", IEEE Journal of Solid-State Circuits, Vol. 38, No. 2, Pages: 192-197, February 2003.

[11] J. Cao, M. Green, A. Momtaz, K. Vakilian, D. Chung, K.-C. Jen, et al, "OC-192 Transmitter and Receirver in Standard 0.18-um CMOS", IEEE Journal of Solid-State Circuits, Vol. 37, No. 12, Pages: 1768-1780, December 2002.

[12] N. Krishnapura, M. B.-Pour, et al, "A 5Gb/s NRZ Transceiver with Adaptive Equalization for Backplane Transmission", IEEE International Solid-State Circuits Conference, Pages: 60-62, Febuary 2005.

[13] R.-J. Yang, S.-P. Cheng, S.-I. Liu, "A 3.125-Gb/s Clock and Data Recovery Circuit for the 10-Gbase-LX4 Ethernet", IEEE Journal of Solid-State Circuits, Vol. 39, No. 8, Pages: 1356-1360, August 2004.

[14] J. Savoj, B. Razavi, "A 10-Gb/s Clock and Data Recovery Circuit with a Half-Rate Binary Phase/Frequency Detector", IEEE Journal of Solid-State Circuits, Vol. 38, No.1, Pages: 13-21, January 2003.

[15] J. Savoj, B. Razavi, "A 10-Gb/s Clock and Data Recovery Circuit with a Half-Rate Linear Phase Detector", IEEE Journal of Solid-State Circuits, Vol.36, No.5, Pages: 761-768, May 2001.

[16] T.-S. Chen, "A 10Gb/s CMOS Half-Rate Clock and Data Recovery Circuit with Direct Bang-Bang Tuning", IEEE International Workshop on Radio-Frequency Integration Technology, Singapore, Pages: 57-60, November 2005.

[17] S.-J. Song, S. M. Park, H.-J. Yoo, "A 4-Gb/s CMOS Clock and Data Recovery Circuit Using 1/8-Rate Clock Technique", IEEE Journal of Solid-State Circuits, Vol. 38, No. 7, Pages: 1213-1219, July 2003.

[18] P. S. and S. Mirabbasi, "A 1/8-Rate Clock and Data Recovery Architecture for High-speed Communication Systems", IEEE International Symposium on Circuits and Systems, Pages: IV-305-8, 2004.

[19] J. K. Kwon, T. K. Heo, S.-B. Cho, and S. M. Park, "A 5-Gb/s 1/8-Rate CMOS Clock and Data Recovery Circuit", IEEE International Symposium on Circuits and Systems, Pages: IV-293-6, 2004.

[20] M. heijningen, M. Badaroglu, S. Donnay, G. G. E. Gielen, and H. J. De Man, "Substrate Noise Generation in Complex Digital Systems: Efficient Modeling and Simulation Methodoloy and Experimental Verification", IEEE Journal of Solid-State Circuits, Vol. 37, Pages: 1065-1072, August 2002.

[21] S.-H. Lee, M.-S. Hwang, Y. Chor, S. Kim, Y. Moon, B.-J. Lee, D.-K. Jeong, W. Kim, Y.-J. Park and G. Ahn, "A 5-Gb/s 0.25-um CMOS Jitter-Tolerant Variable-Interval Oversampling Clock/Data Recovery Circuit", IEEE Journal of Solid-State Circuits, Vol.37, No.12, Pages: 1822-1830, December 2002.

[22] A. L. Coban, M. H. Koroglu, K. A. Ahmed, "A 2.5-3.125-Gb/s Quad Transceiver with Second-Order Analog DLL-Based CDRs", IEEE Journal of Solid-State Circuits, Vol. 40, No. 9, Pages: 1942-1947, September 2005.

[23] R. Zhang, G. S. La Rue, "Fast Acquisition Clock and Data Recovery Circuit with Low Jitter", IEEE Journal of Solid-State Circuits, Vol. 41, No. 5, Pages: 1016-1024, May 2006.

[24] P. Larsson, "A 2-1600-MHz CMOS Clock Recovery PLL with Low-Vdd Capability", IEEE Journal of Solid-State Circuits, Vol. 34, No. 12, Pages: 1951-1960, December 1999.

[25] Ch.-K. Ken Yang, R. Farjad-Rad, M. A. Horowitz, "A 0.5um CMOS 4.0-Gbits/s Serial Link Transceiver with Data Recovery Using Oversampling", IEEE Journal of Solid-State Circuits, Vol. 33, No.5, Pages: 713-722, May 1998.

[26] B.-J. Lee, M.-S. Hwang, J. Kim, D.-K. Jeong, W. Kim, "A Quad 3.125 Gbps Transceiver Cell with All-Digital Data Recovery Circuits", IEEE Symposium on VLSI Circuits Digest of Technical Papers, Pages: 384-387, 2005.

[27] J. Sonntag, J. Stonick, "A Digital Clock and Data Recovery Architecture for Multi-Gigabit/s Binary Links", IEEE Custom Integrated Circuits Conference, Pages: 537-544, 2005.

[28] Y. Miki, T. Saito, H. Yamashita, F. Yuki, T. Baba, A. Koyama, M. Sonehara, "A 50-mW/ch 2.5-Gb/s/ch Data Recovery Circuit for the SFI-5 Interface With Digital Eye-Tracking", IEEE Journal of Solid-State Circuits, Vol. 39, No.4 , Pages: 613-621, April 2004.

[29] G. Chien, P. R. Gray, "A 900-MHz Local Oscillator Using a DLL-Based Frequemcy Multiplier Technique for PCS Applications", IEEE Journal of Solid-State Circuits, Vol. 35, No. 12, Pages: 1996-1999, December 2000.

[30] D. J. Foley and M. P. Flynn, "CMOS DLL-Based 2-V 3.2-ps Jitter 1-GHz Clock Synthesizer and Temperature-Compensated Tunable Oscillator", IEEE Journal of Solid-State Circuits, Vol. 36, No. 3, Pages: 417-423, March 2001.

[31] C. Kim, I.-C. Hwang, and S.-M. Kang, "A Low-Power Small-Area +-7.28-ps-Jitter 1-GHz DLL-Based Clock Generator", IEEE Journal of Solid-State Circuits, Vol. 37, No. 11, Pages: 1414-1420, November 2002.

[32] C.-C. Wang, H.-C. She and R. Hu, "A 1.2 GHz Programmable DLL-Based Frequency Multiplier for Wireless Application", Transaction on VLSI, Pages: 1377-1381, 2002.

[33] J.-H. Kim, Y.-H. Kwak, S.-R. Yoon, M.-Y. Kim, S.-W. Kim, C. Kim, "A CMOS DLL-Based 120MHz to 1.8GHz Clock Generator for Dynamic Frequency Scaling", IEEE International Solid-State Circuits Conference, Pages: 516-517, 2005.

[34] R. Farjad-rad, W. Dally, H.-T. Ng, J. Poulton, T. Stone, R. Rathi, E. Lee, D. Huang, J. Edmondson, R. Senthinathan, "A Low-Power Multiplying DLL for Low-Jitter Clock Generation in Highly-Integrated Digital Chips", IEEE Journal of Solid-State Circuits, Vol. 37, No. 12, Pages: 1804-1812, December 2002.

[35] S. Ye, L. Jansson, I. Galton, "A Multiple-Crystal Interface PLL with VCO Realignment to Reduce Phase Noise", IEEE Journal of Solid-State Circuits, Vol. 37, No. 12, Pages: 1795-1803, December 2002.

[36] G.-Y. Wei, J. T. Stonick, D. Weinlader, J. Sonntag, S. Searles, "A 500MHz MP/DLL Clock Generator for a 5Gb/s Backplane Transceiver in 0.25 um CMOS", IEEE International Solid-State Circuits Conference, Pages: 464-465, February 2003.

[37] S. Ye, L. Jansson and I. Galton, "Techniques for In-band Phase Noise Suppression in Re-Circulating DLLs", IEEE Custom Integrated Circuits Conference, Pages: 297-300, 2003.

[38] B. M. Helal, M. Z. Straayer, G. Y. Wei and M. H. Perrott, "A Low Jitter 1.6GHz Multiplying DLL Utilizing a Scrambling Time-to-Digital Converter and Digital Correlation", 2007 Symposium on VLSI Circuits, Digitest of Technical Papers. Pages: 166-167.

[39] P. C. Maulik and D. A. Mercer, "A DLL-Based Programmable Clock Multiplier in 0.18um CMOS With -70 dBc Reference Spur", IEEE Journal of Solid-State Circuits, Vol 42, No. 8. August 2007. Pages: 1642-1648.

[40] Q.J. Du, "A DLL Based Frequency Synthesizer", Master thesis, Carleton University, 2003.

[41] J.C. Zhuang, "DLL Frequency Synthesizer Designs", Master thesis, Carleton University, 2004.

[42] J.C. Zhuang, Q. J. Du, T. Kwasniewski, "Noise, Spur Characteristics and In-lock Error Reduction of DLL-based Frequency Synthesizers", IEEE International Conference on Communication and Systems, Pages: 1443-1446, July 2004.

[43] Q. Du, J. Zhuang, and T. Kwasniewski, "A Low-Phase Noise, Anti-Harmonic Programmable DLL Frequency multiplier With Period Error Compensation for Spur Reduction," Transaction on Circuits and Systems. II, Exp. Briefs, vol. 53, Pages. 1205–1209, Nov. 2006.

[44] T. Okayasu, M. Suda, K. Yamamoto, S. Kantake, S. Sudou, D. Watanabe, "1.83ps-Resolution CMOS Dynamic Arbitary Timing Generator for > 4 GHz ATE Applications", IEEE International Conference on Solid-State Circuits, Pages: 2122-2131, 2006.