A 2-5 Gbps Fully Differential 3X Oversampling CDR for High-Speed Serial Data Link

By

Kulanathan Kiddinapillai, B.Eng.
University of Peradeniya, Sri Lanka

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements of the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Electronics
Carleton University
Ottawa, Ontario
Canada

December 2009

© Copyright 2009, Kulanathan Kiddinapillai

1*1

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliotheque et Archives Canada

Direction du Patrimoine de l'edition

395, rue Wellington
OttawaONK1A0N4
Canada

Your file Votre reference
ISBN: 978-0-494-63461-5
Our file Notre reference
ISBN: 978-0-494-63461-5

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:

L'auteur a accorde une licence non exclusive permettant a la Bibliotheque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par telecommunication ou par Nnternet, prefer, distribuer et vendre des theses partout dans le monde, a des fins commerciales ou autres, sur support microforme, papier, electronique et/ou autres formats.

L'auteur conserve la propriete du droit d'auteur et des droits moraux qui protege cette these. Ni la these ni des extraits substantiels de celle-ci ne doivent etre imprimes ou autrement reproduits sans son autorisation.

Conformement a la loi canadienne sur la protection de la vie privee, quelques formulaires secondaires ont ete enleves de cette these.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

1+1

Canada

The undersigned recommend to the Faculty of Graduate Studies and Research acceptance of the thesis

"A 2-5 Gbps Fully Differential 3X Oversampling CDR for High-Speed Serial Data Link"

Submitted by Kulanathan Kiddinapillai, B.Eng.
in partial fulfilment of the requirements for the degree of
Master of Applied Science

Prof. T.A. Kwasniewski
Thesis Supervisor

Chair
Department of Electronics

Carleton University
December 2009

Abstract

This thesis reports a fully differential 3X oversampling clock and data recovery (CDR) circuit for burst-mode high-speed serial data link. The CDR operates at a multiple data rate from 2 to 5 Gbps. The architecture of the CDR replaces the analog VCO and loop filter, used in an analog PLL based CDR, with digital circuits. The CDR uses a digital threshold decision technique to improve the jitter tolerance performance.

First the system level CDR analysis is reported, which includes the CDR architecture and operating principle, and derivation of jitter tolerance and acquisition time. It is critical to know the amount of jitter that can be tolerated by the CDR in order to recover the data with satisfied bit error ratio (BER) performance. The jitter tolerance of the CDR is estimated by an event-driven simulation model developed in Matlab. The simulated results show a very close match to the theoretical values. The CDR has a high frequency jitter tolerance of 0.67 UI and an acquisition time of 8 Baud periods.

The complete design flow is executed for the CDR circuit in 65 nm CMOS process technology. The whole CDR is designed based on current mode logic (CML) circuits. The functionality of the CDR is verified by post-layout simulation with pseudo random bit sequence (PRBS) of 2^7 - 1 . The waveforms of incoming data, recovered data and clock are presented. The power consumption and chip area also obtained from post-layout simulation. The CDR consumes 39 mW of power from 1.1 V supply at 5 Gbps. The core CDR circuit occupies an area of 0.013 mm^2. The performance parameters of the CDR are compared with recently reported digital CDRs.

Important Information

The information used in this thesis comes in part from the research program of Dr. Tad A. 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 research program and can be used by Dr. Kwasniewski and his associates for education and research purposes, including publication in open literature with the appropriate credits. Matters of intellectual property may be pursued cooperatively with Carleton University and Dr. Kwasniewski where and as applicable.

Acknowledgements

First I would like to thank my thesis supervisor Prof. Tad Kwasniewski for providing me the guidance and mentorship with patient and kindness during my graduate studies. His advice and help have been invaluable in guiding me to complete this work. I am always grateful for the time worked with him.

I also would like thank my colleagues who have helped me in my thesis and course work, special thanks to Sivakumar Kanesapillai and Guohui Situ. I also would like to express my appreciation to Dr. Dianyong Chen for his useful discussions. I am also grateful to faculty members, staffs of Department of Electronics who have helped me during my stay at Carleton University. All source of financial support for this study are greatly appreciated.

Last but not least I would like express my gratitude to my wife, Vasuki for her love, moral support and patient during my studies. This would not be possible without her understanding and help in managing a good balance between the family and school. I would also like to thanks my kids Madhusha and Harish for their love and support. They always make the life enjoyable and fun. 1 am always very grateful to my parents, siblings and friends for their continuous support and encouragement in making my career successful.

Table of Contents

List of Acronyms and Abbreviations

BER	Bit Error Ratio
CCO	Current Controlled Oscillator
CDR	Clock and Data Recovery
CML	Current Mode Logic
ClkMUX	Clock MUX
CP	Charge Pump
CS	Clock Selection
DFF	Data Flip Flop
DLL	Delay Locked Loop
DFE	Decision Feedback Equalizer
DJ	Deterministic Jitter
DMUX	Data MUX
DSCP	Data Sampling Clock Phase
FIFO	First In First Out
GPON	Gigabit Passive Optical Network
IEEE	Institute of Electrical and Electronic Engineers
ISI	Inter Symbol Interference
NEXT	Near End Cross Talk
NRZ	Non-Return-to-Zero
OC	Optical Carrier
OSR	Oversampling Ratio

PD	Phase Detector
PDF	Probability Density Function
PLL	Phase Locked Loop
PON	Passive Optical Network
PR	Phase Rotator
PRBS	Pseudo Random Binary Sequence
PSD	Power Spectral Density
RJ	Random Jitter
RSG	Rotating Signal Generator
RZ	Return-to-Zero
SNR	Signal to Noise Ratio
SONET	Synchronous Optical Network
TW	Time Window
UI	Unit Interval
VCO	Voltage Controlled Oscillator

List of Figures

List of Tables

List of Symbols

_ J W r -	Resistor
4>	Current Source
II	Capacitor
I	NMOS Transistor
3>	MUX
DFF ->	DFF
DFF -> 1	DFF with Set/Reset
l>	AND gate
3D^~	XOR gate
£>	OR gate
- >	Buffer
T	Vdd
-X-	Ground

Chapter 1

Introduction

1.1 Chapter overview

This chapter provides a description of a high-speed serial data communication system and explain the motivation and objectives. Chapter is concluded with the organization of the thesis.

1.2 Serial data communication systems

Today's data communications systems require higher bandwidth in order to support broad band applications such as video conferencing, e-commerce and other digital multimedia applications. The requirement for more bandwidth demands a communications system be capable of operating at multi-gigabits per second (Gbps) of data rates. Fig. 1.1 is a block diagram of a typical high speed serial data communications system such as Synchronous Optical Network (SONET), chip to chip interconnects and backplane links. At the transmitter side the data signal and the clock signal are combined to one single line in order to preserve timing information at these high data rates. The data signal is synchronized with the transmitter clock and launched into the dispersive channel. As the signal travels through the dispersive channel, the signal is subjected to channel impairments such as frequency dependant loss, reflections, and cross talk from adjacent channels.

A pre-emphasis equalization block is used to compensate for the loss introduced in the channel. The Pre-emphasis block pre-distorts the signal by amplifying the high frequency signal components because the channel attenuates the signal at a high frequency more severely than that at a low frequency [26] (see Fig. 1.2). At a relatively lower gigabits data transmission, a pre-emphasis equalizer alone may be enough to compensate for the channel impairments.

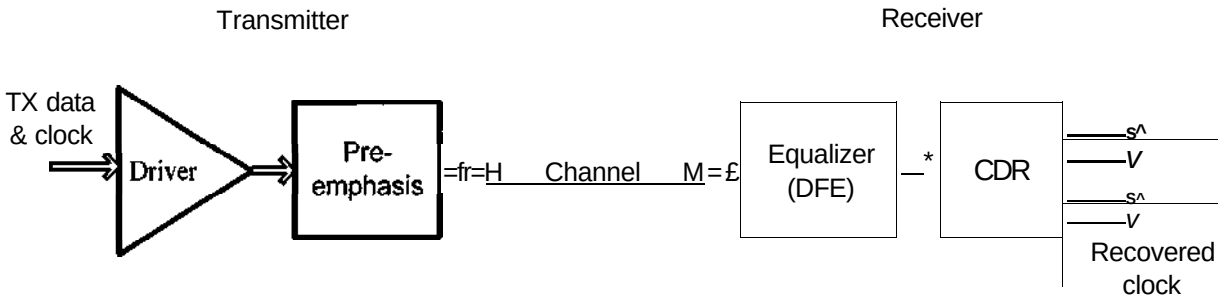


Fig. 1.1 Block diagram of high speed serial link

When the data rate is further increased, these channel impairments will become more significant and create severe inter-symbol interference (ISI), causing the signal power to degrade further. A pre-emphasis equalization will not be able to compensate for the losses and distortion at higher data rates, thus an equalizer is needed at the receiver side. A decision feed back (DFE) equalizer is used to remove the ISI introduced by the channel. A DFE usually estimates the ISI from previously detected symbol. The estimated ISI is subtracted from the currently detected data symbol.

In addition to the ISI problem at these higher data rates, the near end cross talk (NEXT) from adjacent aggressor lines becomes a major noise contributor. This is because the signal power decreases with frequency but the NEXT magnitude increases with frequency [26] (see Fig. 1.2). This cross talk effect will reduce the signal-to-noise (SNR) at the receiver. Moreover NEXT introduces a bounded uncorrelated jitter to the data signal by altering the zero crossing points; the jitter causes the data eye opening to

be narrower consequently increasing the bit error ratio (BER). A NEXT cancellation technique should be included at the receiver to make sure signal integrity at such high data rates is preserved.

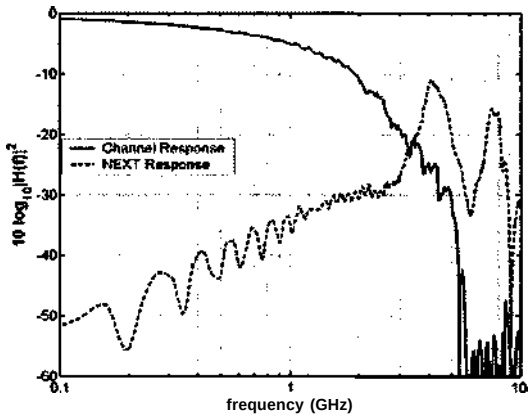


Fig. 1.2 Frequency response of a backplane channel [26]

The data received in serial data communication systems are both asynchronous and noisy, requiring that a clock be extracted to allow synchronous operations [6]. Furthermore, the data must be "retimed" such that the jitter accumulated during transmission is removed [6]. A circuit that is capable of extracting the clock timing information and recovering the data is essential at the receiver.

1.3 Clock and data recovery (CDR)

Narrow band filtering and phase locked loops (PLL) are two conventional methods used to extract the clock signal. Since non-return-to-zero (NRZ) signalling, which has zero energy at the clock frequency, is used in serial data communication, narrow band filtering requires a non-linear circuit to extract the spectral component at the clock frequency.

Analog PLL is widely used in conventional CDR circuits. An analog PLL circuit consists of a phase detector, a charge pump, a loop filter and a voltage controlled oscillator (VCO). The loop filter used in a PLL based CDR circuit usually has a narrower bandwidth to reduce the jitter, thus providing better BER performance. However, the

narrow bandwidth will result in a longer acquisition time. A longer acquisition time requires an increased number of preamble bits and this increased number of bits will reduce the efficiency of the CDR [4]. Longer acquisition time makes the PLL based CDR unsuitable for burst-mode applications such as gigabit passive optical networks (GPON), which require fast acquisition [30]. Also the passive components used in analog PLL based CDR require a larger chip area and consume more power [30]. Analog CDRs are susceptible to leakage and prohibit quick production-level testing [14].

An oversampling CDR may be a good alternative to address the issues encountered in analog PLL based CDRs. In an oversampling CDR, a free running clock with a number of phases is used to generate a certain number of data samples within the Baud period. These data samples are digitally processed to select the clock phase that is closest to the center of the data eye. An oversampling CDR replaces the VCO and loop filter with digital equivalent circuits that are easily portable to different technologies. In recent years, oversampling CDR design has caught the attention of researchers and some significant findings have been published showing successful implementation [1] [2] [4] [18] [19].

*1.4 Thesis **motivation***

Therefore an oversampling CDR that has a short acquisition time would be a good candidate for burst-mode high speed serial data communications applications such as GPON. Designing a CDR that can operate at more than a single data rate is an attractive option because it can be used for different applications, even in different system standards such as OC-48, OC-192. This thesis presents a fully differential fast acquisition 3X oversampling CDR circuit that is capable of operating from 2 to 5 Gbps. This CDR, which replaces analog components used in a conventional CDR with digital circuits, consists of a phase detector, a rotating signal generator, a phase rotator and two MUXs.

1.5 Thesis objectives

- Design an oversampling CDR that can work up to a data rate of 5 Gbps with lower power consumption and fast data acquisition.
- Develop an event-driven simulation model for the CDR in Matlab to estimate the jitter tolerance of the CDR. Also compare the simulation time of both event-driven simulation and conventional fixed-time-step simulation.
- Derive the jitter tolerance of the CDR analytically and verify with the simulation results.
- Execute a complete design flow of the CDR in 65 nm CMOS process technology, which includes: presenting the CML based circuit details and operation of each CDR functional blocks, completing the CDR layout and verifying the CDR operation by post-layout simulation.

1.6 Thesis organization

This thesis is organized as follows; chapters 2 and 3 describe the background theory related to CDR circuits. Chapter 2 focuses on CDR characteristics including its jitter performance parameters, data format. In chapter 3, different CDR architectures, including oversampling CDR and their working principles, are discussed. Chapter 4 presents current mode logic (CML) circuit overview, basic CML buffer design and application of CML logic in high-speed serial link. Chapter 5 describes the system level analysis of the proposed CDR, which includes analytical derivation of CDR jitter tolerance and acquisition time. Chapter 5 also presents the jitter tolerance estimation of the CDR using an event-driven model developed in Matlab. Chapter 6 presents the details of CMOS implementation of the proposed CDR and operating principles, and also reports the post-layout simulation results and a comparison of CDR performance parameters with recently published digital CDRs. The thesis concludes in chapter 7, with the summary of this research work (conclusion) including contributions and some suggestion for future works.

Chapter 2

Characteristics of Clock and Data Recovery Circuits

2.1 Chapter overview

This chapter describes the data format used in the CDR, and CDR performance parameters such as jitter generation, jitter transfer, jitter tolerance, acquisition time, and power consumption and chip area.

2.2 Data format

Binary data can be represented by a return-to-zero (RZ) or non-return-to-zero (NRZ) format. An example of the binary sequence '1011001' is represented using both formats in Fig. 2.1. In an RZ data format, a bit T is represented only for the first half Baud period *(T)* and the bit '0' stays the same at zero for the full period. Shorter pulse duration in RZ data format causes larger bandwidth requirement. The advantage of RZ format is that it requires a lower signal-to-noise ratio (SNR) for detection at the receiver compared to NRZ, thus the RZ format is preferred in high speed long-haul transmission [6]. On the other hand, in an NRZ data format, a bit '1' and '0' are represented for the full bit period. In the NRZ format, bandwidth requirement is smaller than that of the RZ format, which makes NRZ transmission the most commonly used data format in a high

speed serial data communication.

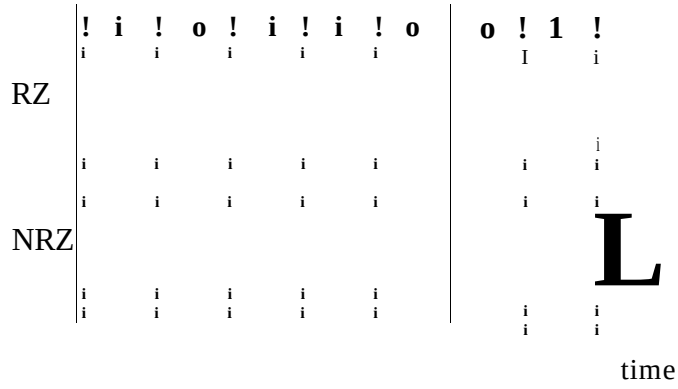


Fig. 2.1 RZ and NRZ data formats

However, the probability of NRZ data containing consecutive ones or zeros is high, so the CDR at the receiver needs to produce the clock continuously. The NRZ data do not have any spectral components at the Baud rate, *1/T,* and at any frequency that is an integer multiple of the Baud rate, which is illustrated by its power spectral density (PSD) as shown in Fig. 2.2. The expression for PSD is given by the following equation:

$$P_{NRZ}(f) = \frac{\sin(nTf)}{nTf} = Sinc(Tf) \qquad \textbf{(2.1)}$$

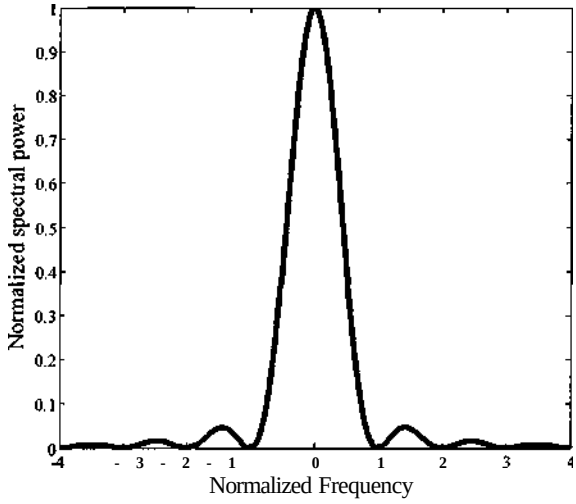


Fig. 2.2 Power spectrum of NRZ data

The absence of energy at the Baud rate makes NRZ data detection more difficult, and NRZ formatted data require regeneration of spectral energy at the Baud rate by means of a non-linear operation.

2.3 CDR jitter characteristics

2.3.1 Jitter

Jitter is defined as the deviation of a signal's transition point from its ideal position in time [13], thus the period of the signal varies from its nominal period. The effect of jitter on the edge of a clock signal is shown in Fig. 2.3. If the timing variation is faster than 10 Hz it's called jitter. If it's slower than 10 Hz, it's called wander [22].

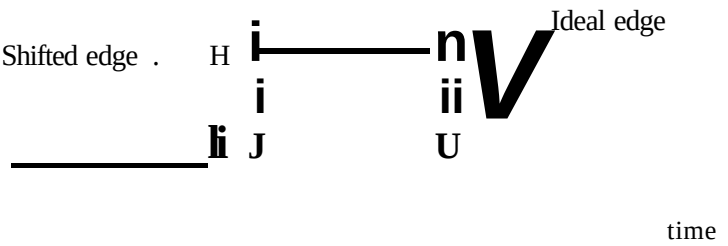


Fig. 2.3 Effect of jitter on clock signal

Commonly jitter is divided into two categories: random jitter (RJ) and deterministic jitter (DJ) [13]. The probability density function (PDF) of random jitter is considered to be Gaussian distributed because the major contributor of random jitter is thermal noise, which has a Gaussian distributed PDF [22]. Deterministic jitter is composed of three components. The first component is periodic jitter, which refers to the periodic variation of the zero crossing point with respect to the ideal position. Major sources of periodic jitter are power supply noise and ground bounce. The second DJ component is data dependant jitter (DDJ). As the name implies, any jitter that is correlated with the data pattern is considered to be DDJ, which includes inter symbol interference (ISI) and duty-cycle distortion [23]. The third component is the bounded uncorrelated jitter that is caused by

signal coupling from adjacent channel. The bounded uncorrected jitter is uncorrected with the channel's own data, but it has a direct correlation with the adjacent data stream. There are three jitter performance parameters defined for the CDR, those are jitter generation, jitter transfer and jitter tolerance.

2.3.2 Jitter generation

Jitter generation of a CDR is the measure of intrinsic jitter generated by the CDR, and it is measured at the output of the CDR. Jitter generation is preferred to be specified in peak-to-peak value because its transient's exhibit steep amplitude variation, which often causes the error [22]. Root mean square value can also be used to measure jitter generation, but it fails to provide information on peak value. The maximum jitter generation in a clock or data signal for the SONET OC-192 standard, which has the data rate of 10 Gbps, is specified as 0.1 UI peak-to-peak or 0.01 UI rms.

Jitter generation can be measured by applying a data signal synchronized with a jitter free clock and measuring the amount of jitter at the output of the CDR. The jitter measurements may not be made over all frequencies as jitter at all frequencies does not affect the signal. In a PLL based CDR any jitter above the loop bandwidth does not affect the signal. Recommended bandwidth measurement for SONET OC-48 and OC-192 standards are specified as 12 kHz-40 MHz and 50 kHz-80 MHz respectively [22].

In a PLL based analog CDR, jitter generation is mainly caused by the phase noise of the VCO and the ripple on the VCO control line. Therefore this specification mainly targets closed loop systems. Reduced loop bandwidth reduces the jitter generation in a PLL based CDR, but it would degrade the jitter tolerance performance. But in an oversampling CDR, the absence of a feedback loop relaxes the jitter generation specification requirement compared to a phase locking CDR.

2.3.3 Jitter Transfer

Jitter transfer refers to a ratio of the amplitude of jitter at the output of a CDR to the amplitude of jitter at the input; it is a measure of how the CDR attenuates the jitter at different frequencies. The jitter transfer parameter of a CDR needs to satisfy the jitter transfer specification (mask) of the communication standard used in order to attain the required BER performance. However a high speed transmission system usually does not verify whether jitter transfer of the CDR meets the specification, instead it is ensured that the BER requirement of that communication standard are met. However, Jitter transfer measurements are more important for a cascaded CDR in long transmission systems with regenerators, where additional jitter in one device can accumulate and cause error on a subsequent device [22].

The jitter transfer of the SONET specification is shown in Fig. 2.4. The corner frequency (f_c) and maximum jitter transfer (gain) are the required jitter transfer specification of a CDR, and the roll-off beyond f_c is 20 dB/decade. The attenuation of jitter will be lower for jitter at a low frequency than at a high frequency, because of the intrinsic jitter of the system [24]. The CDR operation must fall within the acceptable region where jitter gain is below 0.1 dB at the flat region, and the 3-dB bandwidth of the CDR is less than the mask cut-off frequency.

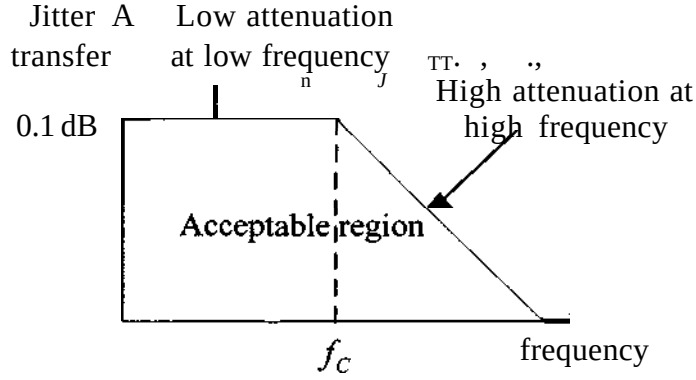


Fig. 2.4 Jitter transfer mask

In a PLL based CDR, closed loop behaviour produces a low pass filtering effect on the incoming jitter. Therefore the jitter frequencies above the loop bandwidth is attenuated significantly thus lower loop bandwidth reduces the jitter transfer. However lower loop bandwidth will degrade the jitter tolerance performance.

2.3.4 Jitter Tolerance

Jitter tolerance can be referred to as the measure of the ability of a CDR circuit to detect the input data signal using the required BER specification in the presence of worst-case jitter. Jitter tolerance of the CDR needs to satisfy the jitter tolerance specification in order to recover the data with satisfied BER performance. Usually jitter tolerance is measured as the maximum peak-to-peak amplitude of a sinusoidal jitter that can be accommodated by the CDR without exceeding the BER specification; it can be interpreted as a phase modulation of the input signal by sinusoidal jitter. The allowable amplitude varies with the frequency of the sinusoidal jitter; different communication standards provide different jitter tolerance mask over a frequency range.

The jitter tolerance characteristics of a CDR with the OC-192 jitter tolerance mask is shown in Fig. 2.5. The dotted line refers to the jitter tolerance of the CDR. When the jitter frequency is higher than the jitter corner frequency ($/_{C1}$) the CDR is unable to track the phase changes in the received data. Thus maximum allowable jitter that remains constant (A UI) is referred to as the jitter tolerance at high jitter frequency. Jitter at a frequency lower than f_{cl} can be tracked up to a certain peak-to-peak jitter amplitude, and because of the CDR loop characteristics the amplitude variation can have a slope of 20 dB/decade [4]. The solid curve in Fig. 2.5 represents the SONET OC-192 mask. For a given frequency any tolerable jitter larger than the corresponding mask value is acceptable as illustrated in Fig. 2.5.

In phase tracking CDRs, jitter tolerance can be increased by increasing the corner frequency, which will move the jitter tolerant curve towards the acceptable region. However increased f_{ci} will result in increased jitter generation and increased jitter

transfer and it may also cause the loop to be unstable [25].

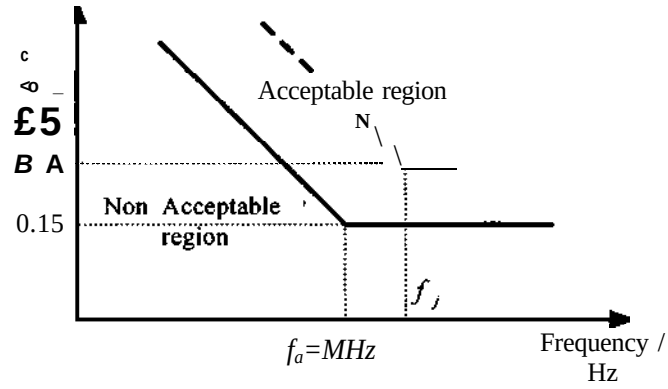


Fig. 2.5 Jitter tolerance mask of OC-192

In an oversampling CDR, the oversampling ratio *(N)* is the key parameter that determines the jitter tolerance performance. The smallest possible phase step is *UI/N,* which can also be interpreted as the quantization error, for the oversampling ratio of *N.* At high jitter frequencies, data transition may not appear within a jitter period. At these high frequencies, the amount of jitter that can be tolerated ($JTol_{p\_p}$) is equal to the difference between the Baud period and the quantization error. If the jitter amplitude is larger than $JTol_{p\_p}$ the clock may sample the adjacent data, which produces erroneous results.

$$JTol_{p\text{-}p} \quad 1 \quad \frac{}{N} \quad \text{UI} \qquad \textbf{(2.2)}$$

At low frequencies (below/C], the CDR can track the jitter by updating the sampling phase up to a certain jitter amplitude. The jitter tolerance at low frequencies depends on the phase updating speed of the CDR, faster phase updating will increase low frequency jitter tolerance. A higher oversampling ratio reduces the phase updating speed, thus degrading the low jitter tolerance performance. A higher oversampling ratio also increases the complexity of the circuits. However, according to (2.2), a higher oversampling ratio increases the jitter tolerance at high jitter frequencies. So there is a

trade off between selecting the proper oversampling ratio and satisfying both low and high frequency jitter tolerance requirements, and keeping reasonable circuit complexity.

2.4 Acquisition time

For a PLL based CDR, the acquisition time refers to the time taken by the loop to achieve lock without any cycle slip for a given initial frequency offset (A/) and phase offset ($) between the local clock (VCO) and incoming data [12]. The acquisition time of a linear PLL based CDR, with an initial frequency offset of A/ and a zero initial phase offset, is given by the following equation [12],

(2.3)

where £, and co_n are the damping ratio and natural frequency of the denominator of the closed loop transfer function respectively, $(j>_f$ is the maximum peak-to-peak oscillation amplitude. Increased loop bandwidth may also speed up the acquisition for a given initial A/ [5].

In an oversampling CDR, the acquisition time can be expressed as the time taken by the CDR to update the phase of the sampling clock without making any errors. So it depends on how often the CDR circuit is designed to update the phase. It is possible to have the shortest acquisition time of one bit period by updating the phase correctly in each period. In general, for any type of CDR, acquisition time is one of the important figure-of-merit parameters; shorter acquisition time is preferred as it reduces the required number of pre-amble bits, thus increasing the efficiency.

*2.5 Power consumption and chip **area***

Power consumption and the chip area of CDR are two secondary parameters to be minimized. As the data rate increases saving power is critical. CML based circuit has an opportunity to trade power for speed. In an oversampling CDR, the oversampling ratio

(OSR) is a key parameter that determines the performance parameters. Increasing the OSR will increase the complexity of the CDR, thus increasing the power consumption and chip area.

Chapter 3

Clock and Data Recovery Circuit Architectures

3.1 Chapter overview

This chapter first describes the classification of CDRs based on the phase relation between the data and the clock signals. Then some commonly used CDR architecture and its operating principles are discussed.

3.2 Classification of CDRs

CDR architectures can be classified into three categories according to the phase relationship between the received input data and the local clock at the receiver [30]. Those three categories are listed below:

- Feedback phase tracking topologies, PLL based CDR, delay locked loop (DLL) based CDR are two main architectures in this category,
- Oversampling based CDR architecture,
- Topology that uses phase alignment but not feedback phase tracking, including a gated oscillator and high-Q band pass filtering.

*3.2.1 Phase locked **loop** (PLL) based **CDR***

Fig. 3.1 shows a block diagram of a conventional PLL based CDR architecture that consists of a phase detector (PD), a charge pump (CP), a loop filter, a voltage control oscillator (VCO) and a retiming circuit. The phase detector (PD) compares the phase of the data signal to that of the internal clock signal and generates the difference as the phase error, and the phase error is applied to the charge pump (CP). The CP adds charges to, or subtracts charges from, the loop filter depending on whether the phase error is positive or negative, respectively. The output voltage of the loop filter is used to set the frequency of the voltage controlled oscillator (VCO). At the phase locked condition, the zero crossing of the incoming data signal and internal clock signal coincide, which means that the VCO frequency is kept constant. The internal clock signal from the VCO is sent back to the PD as well as to the retiming circuit, where the incoming data is retimed by the clock signal to recover the data.

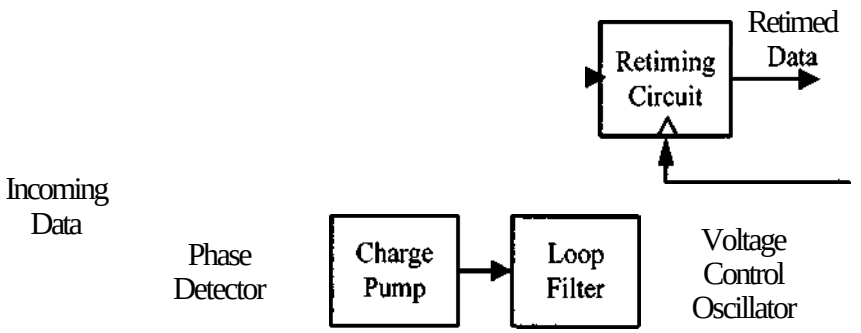


Fig. 3.1 Analog PLL based CDR architecture

There are two types of phase detectors used in CDR circuits, one is the linear phase detector and other is the binary phase detector. The output signal of a linear phase detector is proportional to the magnitude of the phase difference between the data signal and the clock signal.

At the phase locked condition, a proportional response of the phase detector creates low activity on the charge pump and control voltage of the VCO, which

leads to better jitter performance [5]. The linear nature of the phase detector allows us to analyze the circuit in the system level easily using linear PLL theory. A major drawback of linear phase detectors is that they fail to uniquely represent the phase error information for various data patterns [6]. Fig 3.2 shows the architecture of the Hogge's PD, which is the most widely used linear PD in CDRs. In Hogge's PD, the above problem is solved by having two pulses; one pulse is proportional to the phase error (Y), and the other pulse has a constant width *ofT/2* (reference pulse, X). However due to the finite *Clk-to-Q* delay *{AT)* in the flip-flop under the locked condition, width of the proportional pulses are *AT* larger than the reference pulses. This phase offset is a serious issue at high speed because *AT* is not negligible compared to Baud period (T). There are linear phase detectors such as modified Hogge's phase detector that does not have above mentioned problem. However modified Hogge's phase detector is more complex. Binary phase detectors are considered a good candidate at high speed because of its quantization and less complex architecture, thus most of the commercial high speed CDR circuits use binary phase detectors.

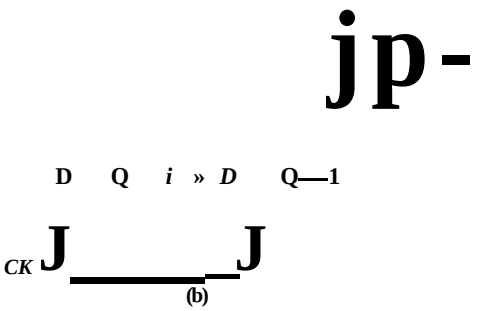
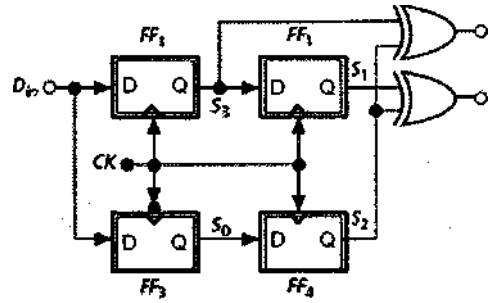
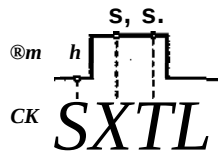


Fig. 3.2 Hogge's phase detector [6]

Binary phase detectors generate an output signal that has only two stages depending on whether the edge of the clock signal arrives *early* or *late* with respect to the edge of the data signal, regardless of the absolute value of the phase difference between them. Continuous movement of the clock edge around the zero crossing of the data signal leads to a higher charge pump activity, thus increasing the clock jitter. Fig. 3.2 shows the architecture of the Alexander phase detector, which is a commonly used binary phase detector in CDR circuits [5]. As shown in the Fig. 3.3, three different samples are taken:

sample SI is the previous data bit, sample S2 represents the current data bit sample at the zero crossing point, and sample S3 is the current bit. An early signal (X) is generated when *S\ * S2 = S3* shown in Fig. 3.4. A late signal (Y) is generated when51 = *S2 * S3*, as shown in Fig. 3.5.





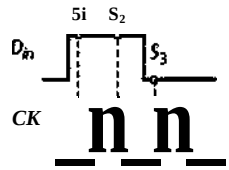


Fig. 3.3 Alexander phase detector [6] Fig. 3.4 clock early [6] Fig. 3.5 clock late [6]

Even though binary phase detectors are not easy to do system level analysis, in [29] a harmonic balance analysis is used to model a bang-bang type binary phase detector. Fig. 3.6 shows the model of the bang-bang type phase detector in the CDR loop. Statistically, the transfer function of bang-bang type phase detector has a finite slope due to the dithering of the clock jitter [29] as shown in Fig. 3.7. The slope (K_{PD}) of the transfer curve is $nl\$_{lin}$, where $</>_{lin}$ can be obtained by statistical analysis of the RMS sum of random jitter for the serial data and the clock [29].

The above phase detector model is used to derive jitter tolerance of the CDR. In the jitter tolerance test, the phase error between the data sampling clock and data signal is assumed to be sinusoidal$[0_e(t) = 2rtA_{sj}$ sine* 7). The Fig. 3.7 shows the quantized phase error $(Q\{\wedge_e(t)))$ in time domain, where the phase error value is clipped when the phase error amplitude is larger $thanK_{PD}0_{liri}$. A harmonic analysis is used to obtain the analytical expression for the quantized error using Fourier series. Only the first order harmonic is taken into account given that the error due to ignoring the higher order harmonic is less than 15% [29]. As a result the quantized error is approximated as,

$$e(\wedge(0)*6,\sin fl>_{jy}(0 \tag{3.1}$$

Where b_x is a constant depends on K_{PD}, jitter frequency ($a>$.) and amplitude (A_{sj}). Using the above equations, the describing function of the bang-bang phase detector is derived as [29],

$$N[A_{sj})=Q\{<l>_e)l<t>_e\{t). \tag{3.2}$$

This describing function is used to derive the CDR open loop transfer function as

$$H_{ol}(z)=N(A_{sj})LF(z)LO(z) \tag{3.3}$$

where $LF(z), LO(z)$ are transfer function of loop filter and local oscillator. Therefore the phase error is given by,

$$\frac{1}{1 + HAz)} \tag{3-4}$$

(3.4) is used to derive the jitter tolerance (J_{Tol}) as,

$$J_{To}, = (1 + \wedge(z))\max(A) = (1 + H_{ol}(z))A_{sjn} \tag{3.5}$$

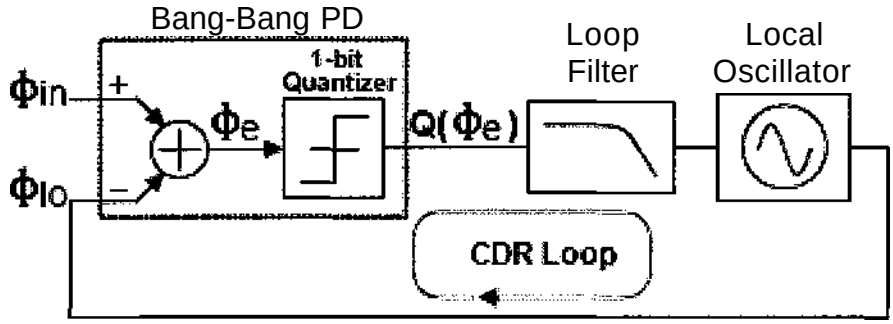


Fig. 3.6 CDR based on bang-bang phase detector [29]

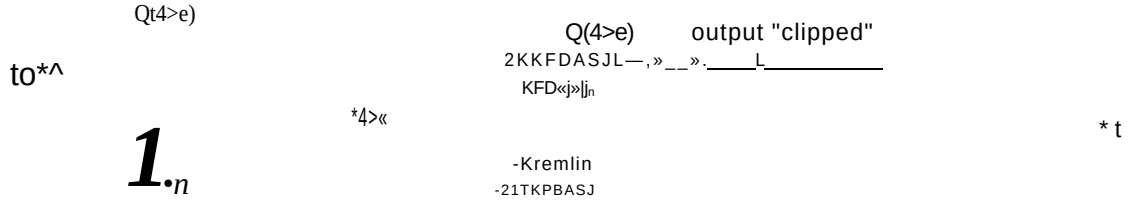


Fig. 3.7 Statistical transfer function [29] Fig. 3.8 Quantized phase error [29]

The PLL based CDR can be further divided into two major groups depending on whether the data rate and clock frequency are equal or not, those are full-rate and non-full-rate PLL based CDRs.

3.2.1.1 Full-rate PLL based CDR

In a full rate PLL based CDR, the data edges are compared with the rising or falling edge of the clock in order to generate the phase error. Thus the clock frequency is equal to the data rate. Fig. 3.9 shows a full-rate PLL based analog CDR reported in [7]. This CDR employs dual loop architecture, a frequency acquisition loop (Loopl) and a phase locked loop (Loop2). The frequency acquisition loop compares the VCO frequency to the reference frequency. When the difference falls below a certain value the loopl is disengaged, and the frequency is considered to be acquired. The frequency acquisition loop facilitates to achieve a large frequency acquisition range. After the VCO frequency acquires with the reference clock, the CDR switches to normal operation where the phase locked loop takes control and locks into the incoming data stream. The precision phase alignment is achieved by the phase locked loop, which provides better jitter tolerance performance. The lock detector controls the switching between the frequency acquisition loop and the phase locked loop. Data is recovered at the DFF that retimes the data using the extracted clock from the VCO.

The major advantages of this full-rate architecture are: simple structure, robust to operate with different data patterns, and lower high frequency jitter due to the increased loop bandwidth. However it has some disadvantages such as: CDRs require higher clock speed that results in a VCO with a large power consumption and dual loop architecture adds extra hardware cost.

3.2.1.2 Non full-rate PLL based CDR architectures

The major drawback of a full-rate CDR is that it requires a high speed VCO clock signal at a high data rate. A CDR, in which the VCO frequency can be reduced to half of the data rate by sampling the data at the rising and the falling edges of the clock, is

defined as a half rate CDR architecture. A half rate CDR employing linear and binary phase detectors is reported in [8] and [9] respectively. The half rate CDR shown in Fig. 3.10 reported in [8] has a similar architecture as that shown in Fig. 3.1, except the phase detector.

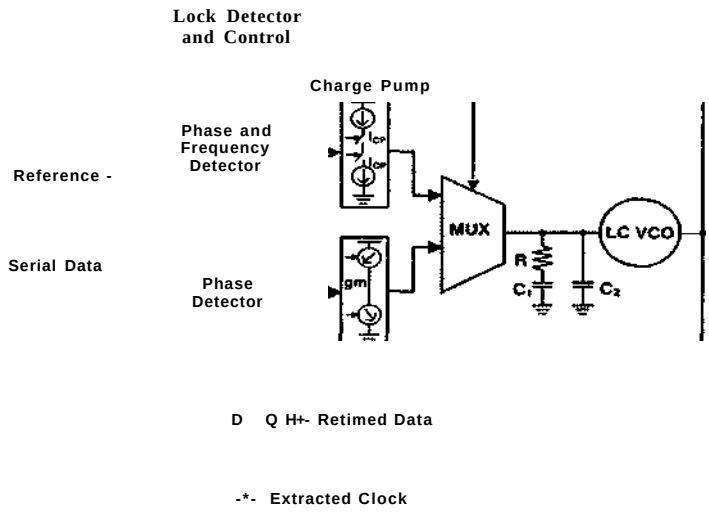


Fig. 3.9 Dual loop phase tracking CDR [7]

Fig. 3.11 shows the phase detector, which compares the edges of 10 Gbps of data to both the rising and falling edges of the 5 GHz VCO clock signal, generating two pulses. One pulse width is proportional to phase difference between the data and clock and the other pulse width is kept constant at *T/2.* Data is retimed as D_{5GA} and D_{5GB} at the phase detector, and a multiplexed full rate data D_{10G} is produced at the PD.

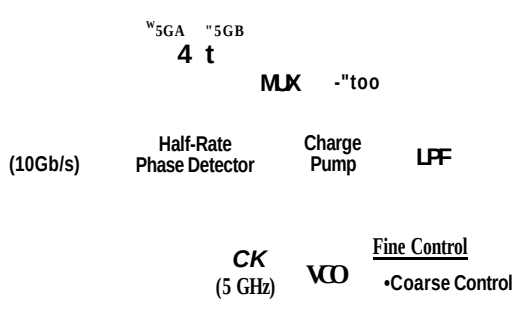


Fig. 3.10 Block diagram of half rate CDR [8]

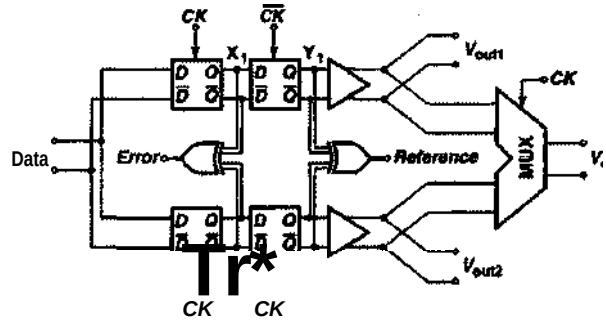


Fig. 3.11 PD used in half-rate CDR [8]

Advantages of this CDR are reduced power consumption and chip area, resulting from reduced VCO speed and design simplicity. The major drawback of the half-rate architecture is the clock duty cycle distortion, defined as the deviation of the clock duty cycle from 50%. This duty cycle distortion will cause both the falling and rising edges to move away from the center of the data bit because both edges are used to sample the data. Also the half-rate CDR usually does not employ any frequency acquisition loop, therefore the CDR has shorter frequency acquisition range.

3.2.2 Oversampling based CDR

In oversampling techniques, a free running internal clock with a number of phases is used to generate a certain number of data samples within a Baud period. These data samples are digitally processed to select the clock phase that is more close to the center of the data eye. Oversampling CDR circuits can be classified in two major groups according to whether the clock phase is adjusted to track the data eye or not: Synchronous oversampling CDR and the blind oversampling CDR. In a synchronous oversampling CDR, the sampling clock is adjusted step by step to track the data eye. On the other hand, blind oversampling CDR does not adjust the clock phase to track the data eye. Instead the clock phase that is aligned closer to the data eye is used to sample the data.

3.2.2.1 Synchronous oversampling CDR

A synchronous oversampling CDR can be considered as a digital version of the phase tracking analog CDR, because it adjusts the clock phase step by step in a cyclic manner to track the center of the data eye. Synchronous oversampling CDR becomes a good candidate to alleviate the drawbacks of the analog PLL based CDR such as, larger power and chip area

A 2.5 Gbps synchronous oversampling CDR with a reduced power consumption and chip area was reported in [1]. The circuit description and operation are briefly explained below. Fig. 3.12 shows the block diagram of the CDR circuit, which consist of four functional blocks: a bang-bang type phase comparator (PC), an *UP/DOWN* decision

circuit, a clock phase pointer, and a clock interpolator/clock selector. The phase comparator contains two delayer elements that have a fixed delay of 774 is used to detect the data transition. The incoming data is processed for group *of ND* bits. Within a group of 8 bits *(ND =16),* the first four odd clock edges are compared to the corresponding data edges, and PC generates an *UP* or *DOWN* signal when the edge of the data leads or lags that of the clock respectively.

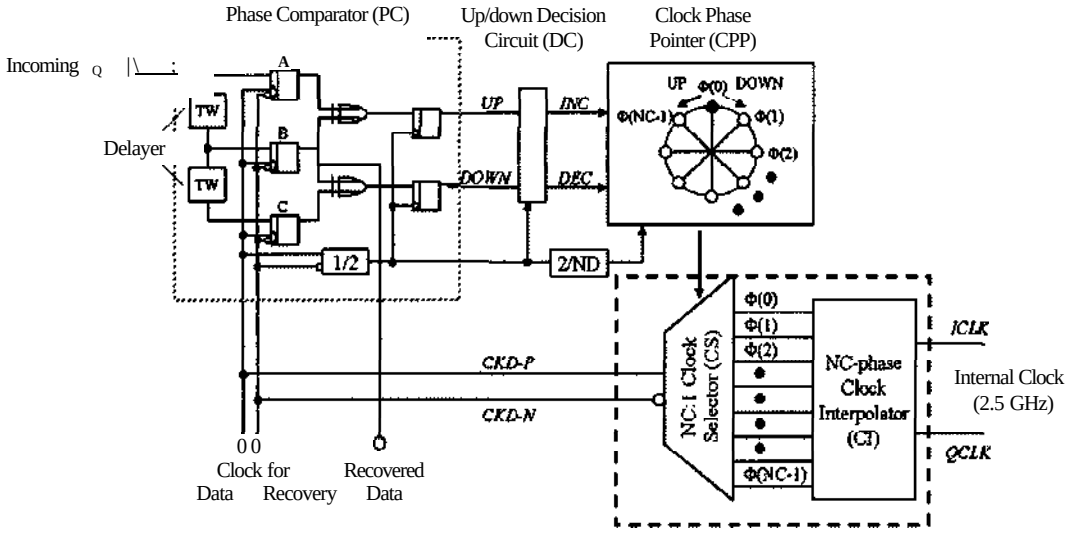


Fig. 3.12 Synchronous oversampling CDR [1]

The major components of the *UP/DOWN* circuit are the temporary buffers. Buffers stores the *UP* and *DOWN* request signal received from phase comparator for a group of data. Within the group of *ND* data bits, when the buffers receive only an *UP* or *DOWN* signal an increment *(INC)* or decrement *(DEC)* signal is generated respectively. If the buffer collects both an *UP* and *DOWN* signal, neither an *INC* nor a *DEC* signal is outputted.

When the clock phase pointer (CPP) receives an *INC* signal it increases the count to delay the clock phase. On the other hand if it receives a *DEC* request the count is decreased to advance the clock phase. The cyclic structure of the CPP allows for switching of the clock phase infinitely beyond one UI, which helps to trace excess data wander. The CPP block is clocked by *1/ND* time of the recovered clock, so the CPP

updates the phase once for *ND* data bits. The last block consists of two sub block. The first block is the clock interpolator that generates an *NC (NC=8)* number of clock phases from the internal in-phase (ICLK) and quadrature phase (QCLK) clocks. The clock selector block receives the command from the CPP and selects one clock phase as the recovered clock from an *NC* number of clock phases.

This synchronous architecture has some major advantages. The first advantage is the reduced power and chip area, which is achieved by the increased phase updating period of *ND* bits and the reduced clock distribution network. The clock phase aims to track the center of the data eye step by step. This eye tracking technique helps to achieve a better jitter tolerance performance and makes the CDR able to operate in noisier environment. However this CDR has some drawbacks. It has a longer acquisition time *(ND* bits) because the sampling clock phase is updated for every *ND* bits, the fixed delay element prohibits the CDR circuit to operate at different data rate, and the design requires a different clock rate for each block, which results in more clock dividers, leading to higher hardware cost.

3.2.2.2 ***Blind oversampling CDR***

Unlike an analog or digital PLL based CDR, the blind oversampling CDR does not have a feedback system to track the center of the data eye. But its feed forward architecture determines the center of the data eye from the transition information extracted from the multiple samples in the phase detector. The absence of a feed back loop provides a faster response to a change in clock phases, unlike the phase tracking CDR where the feedback loop bandwidth constraint puts a limit on how fast the CDR responds to phase change. The digital architecture simplifies the design process and allows the design to transfer easily to another modern process technology. The digital process also has the advantage of having better immune to noise [17], thus generating less jitter.

A block diagram of a blind oversampling CDR reported in [16] is shown in Fig. 3.13. First the received signal is sampled by a multiphase internal clock, generating a

certain number of samples *(N)* within a Baud period, where *N* is the over sampling ratio.

The phase detector detects the data transition from the set of samples expecting a transition within the set of samples for each Baud period. In the absence of a transition the data is selected according to the phase that is one Baud period after the previous phase information. The transition phase information is sent to a multiplexer that extracts the corresponding data samples as the recovered data. The CDR consists of a first-in, first-out (FIFO) block that is used to absorb any cyclic slip between the local clock and the clock information embedded in the received signal. This is because the absence of a feedback architecture there is no phase tracking between the local clock and the received clock information, which may result with a phase offset of many *UI.*

Even though the blind oversampling CDR has the above mentioned advantages the CDR has the following disadvantages: Multi-phase clock phases causes extra hardware cost to the design, A low over sampling ratio (3) degrades the high frequency jitter tolerance as higher phase resolution results in a larger static phase error. On the other hand very high oversampling ratio leads to a reduction the phase tracking ability at low jitter frequency. There is a trade off in choosing a suitable oversampling ratio, which is limited to 3 or 5 as the typical values [17].

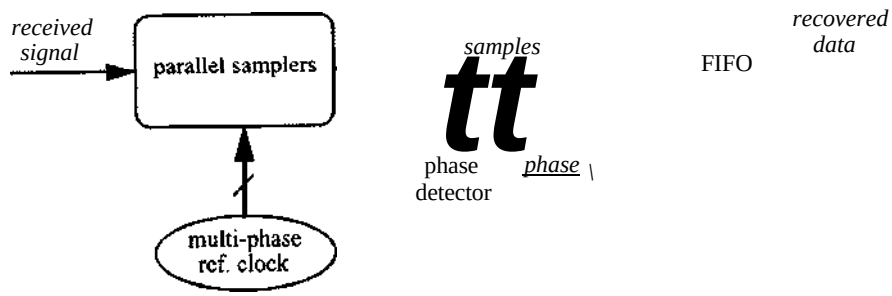


Fig. 3.13 Blind oversampling CDR [16]

Fig. 3.14 shows a 5X oversampling CDR that uses an eye tracking technique reported in [4] that consists of a phase detector, a rotating clock generator, and a phase point rotator. Multi-phase clocks are generated from a local clock in DLL. The phase detector is made of 3 sub blocks. The first block is the sampler, which samples the

incoming data by 5 clock phases equally spaced by *0.2T* time intervals, and generates 5 data samples within each Baud period. The transition detection block detects the data transition information from the over sampled data and send them to the digital threshold detection (DTD) block. The phase point rotator block generates 5 phase selection signals, *a, b, c, d, e,* which correspond to the 5 clock phases, *clkl, clk2, clk3, clk4, clk5* respectively. Of the 5 phase selection signal, one is active *{high)* for each Baud period, and the active signal is called the 'hot-coded signal'. The other 4 signals are set to be inactive *(low).* The data sampling phase position is compared with the data transition information in the digital threshold detection block, and generates the phase error. The phase error value will be a multiple of the phase resolution, which is *0.2T* for an oversampling ratio of 5. When the data transition occurs *earlier* (or *later)* than the sampling phase position by *k* phase resolution the phase error will be - *k* x *0.2T* (or + *kx0.2T).*

As the name implies the DTD block compares the phase error with a threshold value, and if the magnitude of the phase error exceeds the threshold value of 2x0.27, it outputs *L* or *R* signal according to whether the phase error is negative or positive respectively. Otherwise no signal will be outputted. Even though the *L* and *R* signals are estimated in each Baud period phase updating is not performed at each Baud period. The rotating clock generator works as a filter that keeps track of phase error information within the previous 8 Baud period. The phase rotating signal *(ROT)* is outputted from the rotating clock generator if an *L* or *R* signal is received provided that no *R* or *L* signal appeared within the previous 8 symbols respectively.

Assume that initially phase selection signal 'c' is set to be active. When the phase pointer receives the *ROT* signal, as well as the *L* or *R* signal from DTD, the phase pointer advances *(c to b)* or retard (c *to d)* the hot coded signal, depending on whether the received signal is *L* or *R,* by a one phase step *(0.2T).*

The advantages of this CDR are: simple and low complex digital logic circuit, reduced power consumption and chip area, ability to operate at a variable data rate up to 2-3 Gbps, and the digital eye tracking techniques allows the CDR to achieve high jitter

tolerance performance. However the multi-phase clock distribution contributes to a large portion of power consumption.

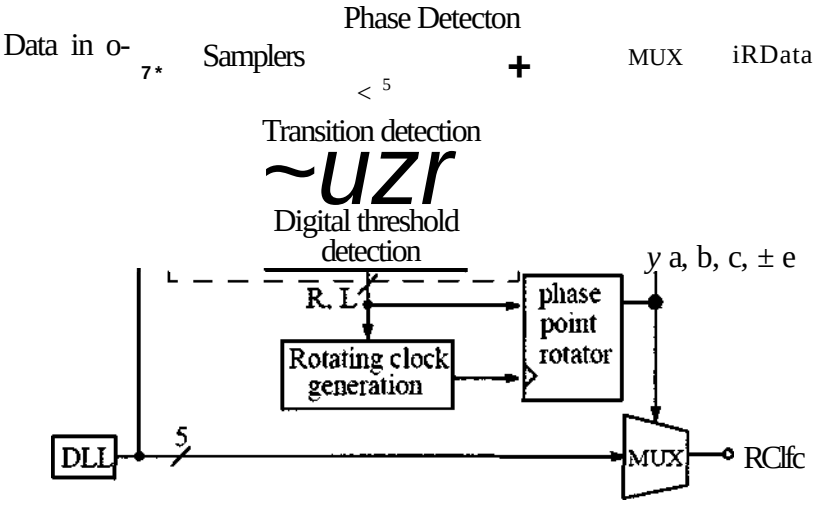


Fig. 3.14 5X oversampling CDR [4]

Another blind oversampling CDR reported in [19] that uses a phase averaging technique is shown in Fig. 3.15. First the incoming data is retimed, and then the oversampled data is sent through two different paths. One path is the data path where the data is saved in buffers, and the other path is the control path where the phase information is extracted. Phase averaging is done in two stages. In the first stage, the phase information of a block of 10 bits of data is calculated by taking the average of all the edge phase values, and compared with the previous phase pointer to generate an *Up/Down* signal. A larger number of bits of data would lead to better receiver performance at the expense of increased hardware cost.

In the second stage of phase averaging, two different voting algorithms are used, and those are unanimous and majority voting algorithms. Most of the high frequency jitter is filtered out at the first stage, and the unanimous voting algorithm reduces the phase error better than the majority voting algorithm. The architecture includes a forward path at the second stage of phase averaging, through which the phase information of the latest block is sent directly to the phase pointer, providing latency of one cycle. This

latency can be easily matched in the data path by adding a buffer that has one cycle of latency, and avoiding the phase point lag caused by the latency introduced by the two averaging stages in the control path. So the possible instability at intermediate frequencies is avoided.

The advantages of this CDR are: The higher oversampling ratio reduces the quantization error and achieves better high frequency jitter performance, and the power consumption is moderately low. However this CDR may have longer acquisition time due to the phase averaging in each window size. Increased window size will result in a longer acquisition time.

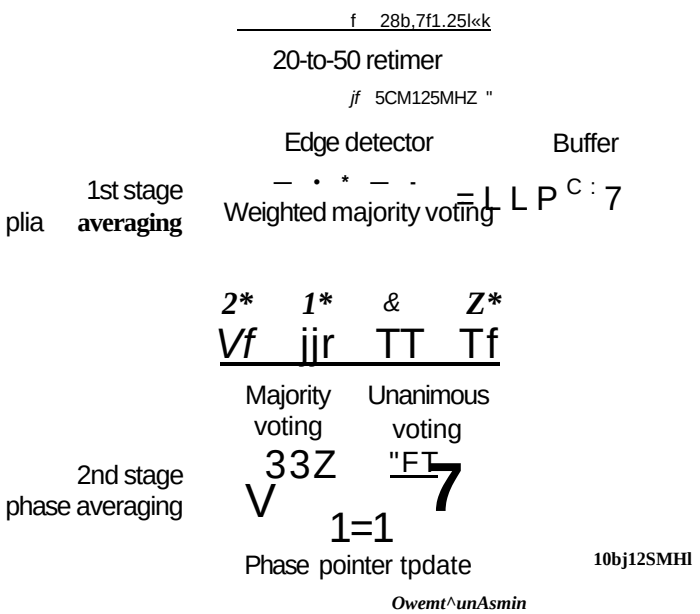


Fig. 3.15 Phase averaging CDR [19]

3.2.3 Gated-oscillator based CDR

Fig. 3.16 shows a gated-oscillator based CDR architecture reported in [30] that consists of a current controlled oscillator (CCO) and an edge detector. The edge detector block generates an *ED* signal at each data edge based on the delay line and the XOR gate. At a data edge, the *ED* signal is set to low for a duration determined by the delay line and the CDR locks the output clock *(Clk_{out})* to high through the first stage of the oscillator.

The oscillator releases and returns to its free oscillation frequency as determined by the controlling current and last received data edge. The delay introduced by the delay line is eliminated by sampling the delayed data *(DD_{jn})* instead of the incoming data *(D_{in})*. The delay at the XOR gate and the delay mismatch between the NAND gates are compensated by the dummy gates. The sampler uses the clock output to sample the compensated delayed data to recover the data.

The gated-oscillated based CDR topology is commonly used in passive optical network applications that require fast clock recovery and data acquisition [30]. This topology is simpler and the less complex design has lower power. The major drawback of the gated-oscillator topology is that it has no jitter rejection because its broadband open loop design has no bandwidth filtering [30]. Another disadvantage is that phase alignment between the incoming data and recovered clock is sensitive to process, temperature and supply voltage variation. Another drawback is that the gated controlled based CDR is more difficult to transfer from one process technology to another [30].

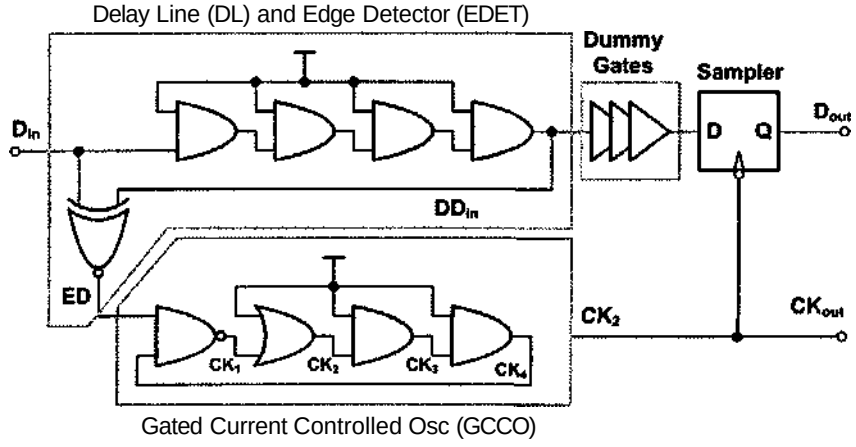


Fig. 3.16 Gated oscillator based CDR [30]

Chapter 4

Current Mode Logic (CML) Circuits

4.1 Chapter overview

This chapter first describes the CML architecture and working principles. Then a CML buffer design and the design consideration are presented. The chapter also reports the advantages of CML based CDR design at high speed serial data communications systems.

4.2 CML architecture

A common CML architecture is shown in Fig. 4.1, where the input and output signals are in differential format, and the current is steered between two pull-up resistors. The voltage swing is determined by the current *(I)* and the resistor *(R)*. Generally a voltage swing of 200 mV- 400 mV is considered to be good value [15]. Unlike a standard CMOS logic circuit, where the voltage swing is rail-to-rail, the CML circuit voltage does not swing rail-to-rail. Therefore CML logic is considered one of the fastest logic styles, and this is considered to be the prime advantage of the CML circuit. Some other advantages of CML circuits, compared to the standard CMOS logic are: because of differential signalling the CML circuit is immune to common-mode noise, the constant dc current leads to less switching noise [15], since the signal is always accompanied by its complementary signal, no discrete inverters are required. The CML circuits normally

consume static power but its total power dissipation at high frequencies is lower than that of the standard CMOS logic circuit.

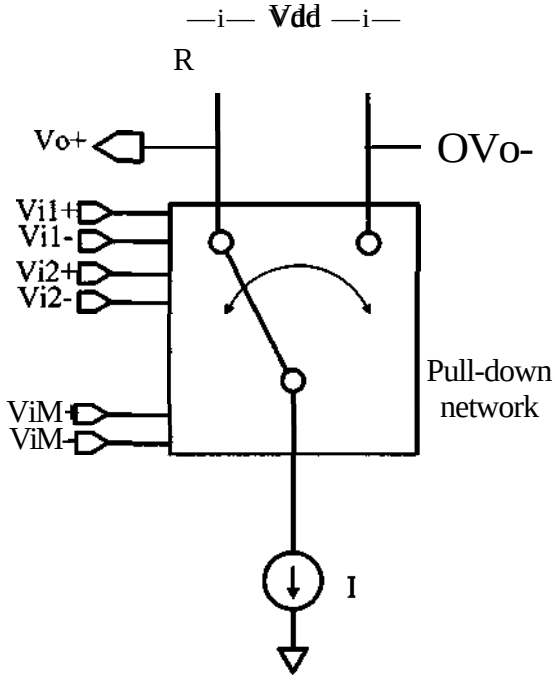


Fig. 4.1 Basic CML architecture

*4.3 **CML operating principle***

In the CML architecture shown in Fig. 4.1, there are M numbers of differential inputs (Vil+/Vil-. . . . . ViM+/ViM-) and a differential output (V0+/V0-). The current *(I)* is switched between one of the two output branches depending on the pull-down network.

The highest output voltage (V_{oh}) and lowest output voltage (V_{ol}) are given by,

$$V_{oh}=Vdd, \tag{4.1}$$

$$V_{ol}=Vdd-AV, \tag{4.2}$$

$$\text{Voltage swing } (AV) = IR \tag{4.3}$$

Since the current is steered continuously the total power dissipation can be written as

$$P_{CML} = IVdd \tag{4.4}$$

The power dissipation of standard CMOS circuit is given by the following equation [31]

$$P_C Mos = C_L (Vdd)^2 f \quad (4.5)$$

where the C_L is the load capacitance and/is the frequency of operation.

According to equations (4.4) and (4.5), the power dissipation variation against the operating frequency of CML and standard CMOS circuits are plotted in Fig. 4.2. The standard CMOS circuits have lower power dissipation at low frequencies, however their power increases linearly with the frequency. The CML power dissipation does not change with the frequencies. Therefore at high frequencies, any frequency above f_0, the CML circuits dissipate less power than the standard CMOS circuits. The propagation delay of a CML circuit (T_{CML}) can be derived as [32]

$$"_{CML} \; -^{C}\text{-}\wedge\text{-}C_L R \quad \textbf{(4.7)}$$

According to (4.7), to reduce the propagation delay of a given CML circuit (to increase the speed), value of R needs to be reduced since the C_L is already fixed by the circuit that is driven by the CML circuit. If the value of R is reduced the current needs to be increased in order to maintain the voltage swing. Increasing current will increase the power dissipation. So there is a trade-off between power and speed. This trade-off can be considered as one of the advantages of CML, i.e. it is possible to trade power for speed.

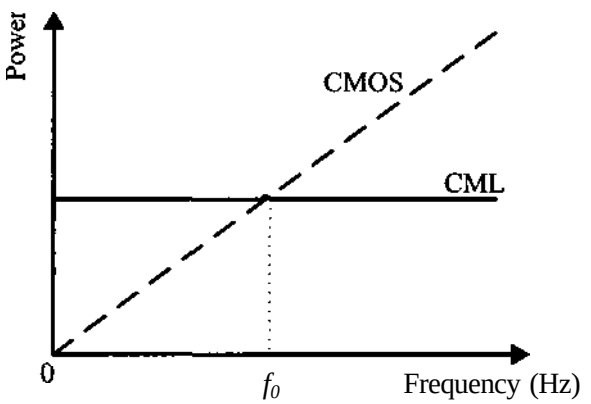


Fig. 4.2 Power dissipation of CML and CMOS

4.4 CML Buffer/Inverter design

Fig. 4.3 is a circuit diagram of a CML buffer that operates as an inverter when the output terminals are interchanged (shown within the parenthesis). Both differential pair transistors are identical and have the width to length ratio of *W/L*. The length of the differential pairs is usually kept at a minimum for the process technology used. The current source transistor's (M3) length is kept at a length larger than the minimum value.

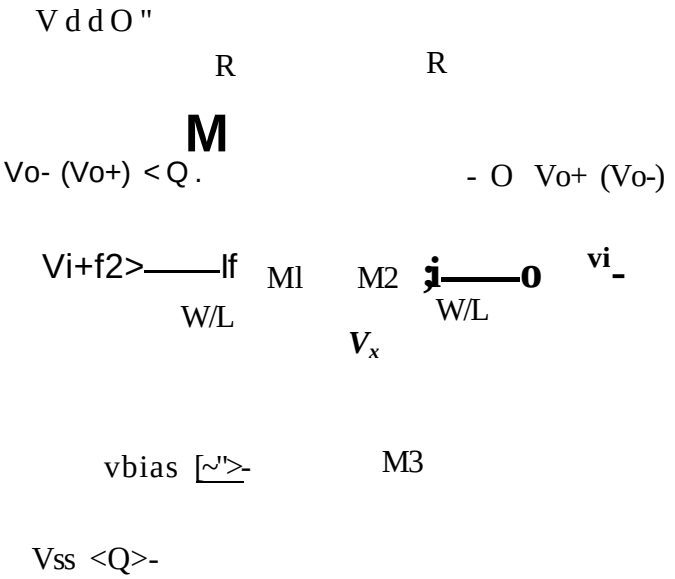


Fig. 4.3 CML buffer (inverter)

As shown in the Fig. 4.3, assume the current through the transistors Ml and M2 is /, and $/_2$, respectively. When Ml and M2 are at saturation, ignoring the effect of channel length modulation, the current /, and I_2 can be expressed as follows

(4.8)

(4.9)

where V_{GSI} and V_{GS2} are the gate to source voltage of transistors Ml and M2, and K is constant, as given by the following equation,

$$K = fi_N C_{ox} \quad W \tag{4.10}$$

where $/J_N$ is the permeability of the NMOS transistors and C_{ox} is the oxide capacitance per unit area. V_T is the threshold voltage of differential pair transistors. Since Ml and M2 are identical transistors their K values are the same. The differential input voltage V_j and total current *(I)* can be written as

$$V, = V,^{+} - V,' \tag{4.11}$$

$$I = I_{l+} I_2 \tag{4.12}$$

The gate to source voltages can be expressed in terms of input voltages V_t^{+}, V_i and V_x as follows, where V_x is the common source voltage,

$$*GS\ \sim * i \quad * X \tag{4.13}$$

$$*GS2 \sim "i \quad *X \tag{4.14}$$

Using (4.13) and (4.14), the V_{in} can be rewritten as

$$V = V - V \tag{4.15}$$
$$y\ i \quad v\ GS\ \quad v\ GS2$$

Using (4.8), (4.9), (4.12) and (4.15), a quadratic equation for I_x is obtained and shown in (4.16).

$$1,-11,+ \quad (1 \;,, \quad \frac{K}{4} iv; + \quad \frac{K}{I\ 4}\ a \quad v, \quad \backslash = 0 \tag{4.16}$$

(4.16) is solved to get solution for /, and $/_2$, which are shown below

$$h = {}^{!}-/-_2 . w- \quad {}_{,2} \tag{4.17}$$

$$\wedge k-\backslash^{m} \quad -V-\wedge K \tag{4.18}$$

Using (4.17) and (4.18), the variation of current /, and I_2 against the input differential voltage *(V,)* are shown in Fig. 4.4 (DC transfer characteristics). In order to find the maximum or minimum value of / j , (4.17) is differentiated with respect to V_i and equated to zero

$$\frac{dL}{dV,}\text{1L.I} \quad l_B\text{-} \quad £* \quad \wedge \quad \frac{1}{KI\text{-} \quad K_{,}\text{-}K} \quad i\text{-}V,K) = 0 \tag{4.19}$$

From (4.19), value of V_t , and the corresponding/), I_2 values are obtained as,

$$V_i|_M = \pm\sqrt{\frac{2I}{K}} = \pm\sqrt{\frac{2I}{\mu_{N^{\wedge}OX}\frac{W}{L}}} \tag{4.20}$$

$$h=i,o$$

When $V,$ is equal to $V\backslash_M$ (or $-V\backslash_M$), /, = / (or $I_2 = I$) and $I_2 = 0$ (or /, = 0), where the current is steered completely by one of the branches. This behaviour is clearly shown in the transfer characteristics (see Fig. 4.4). It can be noted that, in order to switch the transistors completely, the differential input voltage should be greater thanJ^I .

$$\backslash V\backslash > \frac{21}{lf_{NN^{\wedge}OX}\frac{W}{}} \tag{4.21}$$

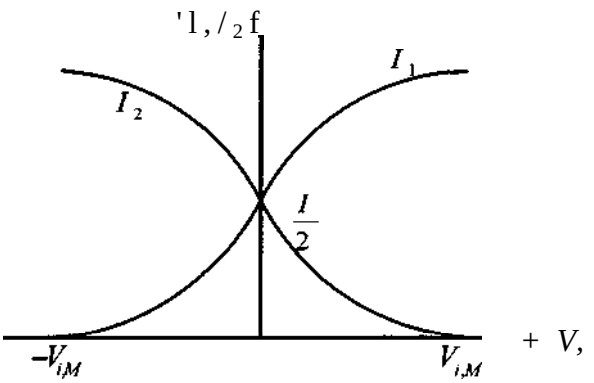


Fig. 4.4 Transfer characteristics of CML buffer

4.5 CML buffer design considerations

CML circuits are designed with the voltage swing of 400 mV to ensure complete current-switching as the CDR has cascaded logic gates. When designing a CML buffer with a voltage swing of 400 mV for 5 GHz signal, the following analysis is considered. Assume the capacitive load seen at the output terminal of the buffer due to a single fan out is C_u, which is the addition of input capacitance of the gate that is driven by the buffer, transistor parasitic capacitance and layout capacitance. For a typical circuit, it is assumed that the buffer is driving a fan out of 2, and thus total load capacitance (C_L) would be $2C_U$. Fig 4.5 shows the total capacitive load seen at the output terminal of the buffer and Fig. 4.6 shows the equivalent RC network.

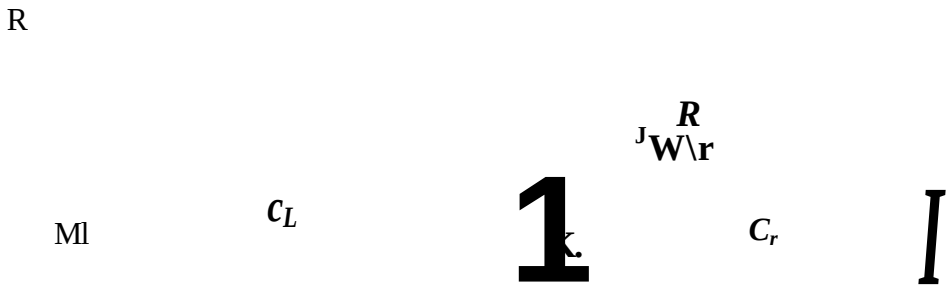


Fig. 4.5 MOSFET Capacitive load in CML Fig. 4.6 RC network

Consider that a 5 GHz square wave signal (V_{in}) is applied at the input of the RC network. For initial estimation, the $C_{L1}$is assumed to be 10 fF. Therefore the total load capacitance is 20 fF. The time constant of the RC network would be RC_L. The transient output voltage (V_{oul}) can be expressed as

$$V_{out} = V_{in}\left(1 - e^{-t/RC_L}\right) \qquad (Ml)$$

where / is the time.

Equation (4.22) shows the voltage variation across the capacitor during charging. Assume that time to switch the logic from logic 'low' to 'high' is as $5RC_L$ [15], where the voltage is charged to 99.3% of the final voltage. However the circuit will start to switch when the input is half of the final voltage. The time to switch the logic level $(5RC_L)$ should be less than half of the signal period *(T12)* in order to ensure that the output voltage reaches the maximum swing [15]. This condition can be used to calculate the maximum operating frequency for a given R and C_L value.

$$5 * Q < | \tag{4.23}$$

Since the frequency of operation is already taken as 5 GHz, T is equal to 200 ps. From (4.23) the maximum value of R that provides a satisfactory rise/fall time can be estimated as follows,

$$** \quad \frac{I_r}{2 \text{ x } 5 \text{ x } C_L} \tag{4-24}$$

Equation (4.24) gives the maximum value of R as 1 *kQ*. Now the current required *(I)* to drive this load can be calculated for the given voltage swing of 400 mV.

$$\bullet \quad \frac{\circ - ^{4\,F}}{1000Q} = 4\,0\,0\,\wedge$$

These values are used as the initial parameter in CML buffer design. All the other gates are designed based on the buffer design. If any gates have to drive a double fan out $(2C_L)$, the parameters are scaled accordingly, and the value of the resistor *(R)* is halved to make sure the condition stated in (4.21) is met. Since the resistor value is halved, the current *(I)* has to be doubled in order to maintain the voltage swing of 400 mV. The sizes of transistors are also doubled to handle the doubled current.

In a real circuit the charging time could be assumed to be less than $5i?C_L$ and eventually that would lead to switching failure unless the current is increased.

4.6 CML circuits in high-speed CDR design

The whole CDR circuit design is performed using CML circuit. Fig. 4.7 shows the circuit diagram of CML based AND/NAND gate that can also be used as OR/NOR combination by interchanging the inputs. A CML based XOR gate and D-flip flop are shown in Fig. 4.8 and Fig 4.9, respectively.

There are several benefits of using CML circuits. The first benefit is that the differential nature of CML circuits gives them excellent common mode noise immunity, and their current steering nature allows for greater performance than any other logic family [21]. Since the differential signals are defined with respect to each other, any noise affecting both signals will be cancelled out. The second benefit is that CML circuits can operate with a lower signal voltage and a higher operating frequency at lower supply voltage than static CMOS circuits [20]. Therefore CML logic is a better choice when designing multi-gigabit CDRs. The next advantage is that the constant current flow in CML causes a very small amount of noise to be injected into the substrate, whereas the single ended logic family, like standard CMOS, injects a lot of noise into the substrate. Finally, with a CML circuit the signal is always accompanied by its complementary signal. Since the inverted signal is always available in CML logic no separate inverter is needed to produce the inverted signal. However single ended logic, such as standard CMOS, requires a separate inverter to produce an inverted signal, and that extra inverter introduces a phase offset to the inverted signal with respect to the original signal.

One of the drawbacks of CML logic is the constant current flow, and therefore CML circuits consume more power than the other logic family. Another disadvantage is that the CML design is considered to be more challenging than the static CMOS circuit design.

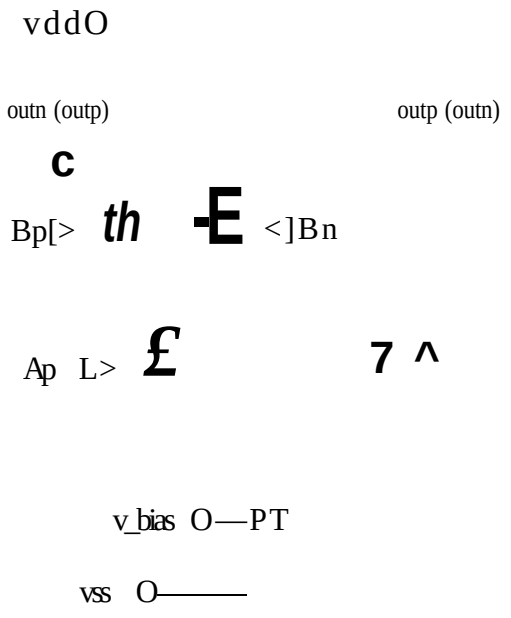


Fig. 4.7 CML AND (NAND)

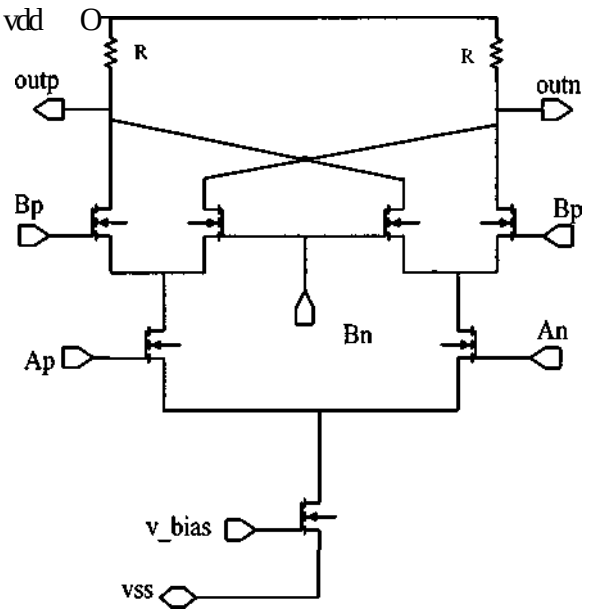


Fig. 4.8 CML XOR

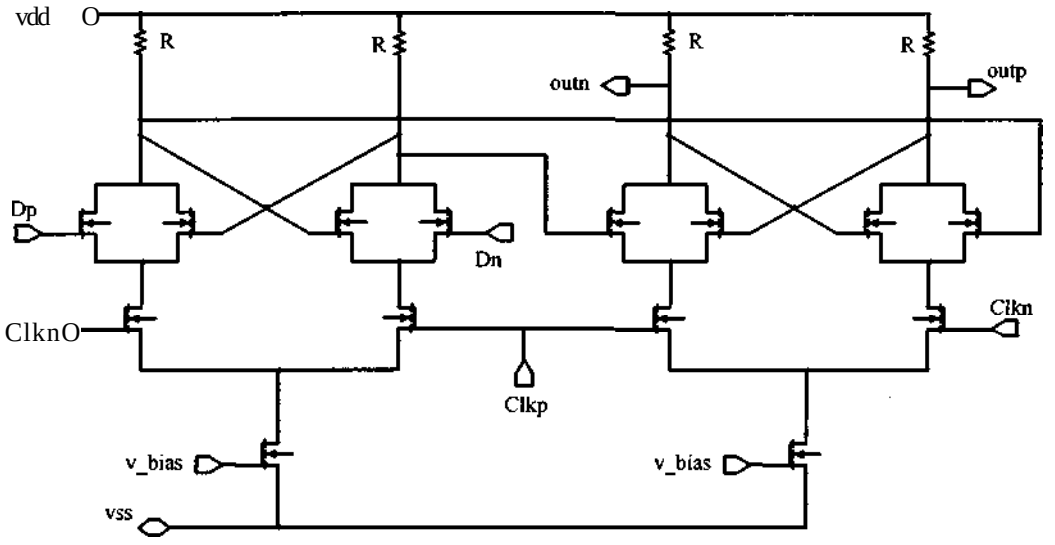


Fig. 4.9 CML D-Flip Flop

Chapter 5

System level Analysis and Simulation of proposed CDR

5.1 Chapter overview

In this chapter, I propose a 3X oversampling Clock and data Recovery circuit (CDR) that uses a digital threshold decision technique to achieve high jitter tolerance performance. First, the system level operating analysis of the CDR is described using functional block diagrams. Then mathematical analysis and derivation of the CDR jitter tolerance are presented. Next, in order to derive jitter tolerance of the CDR, an event-driven modelling of the CDR is developed and presented. The chapter concludes with a presentation of event driven simulation results.

5.2 System level analysis of proposed CDR

Fig. 5.1 shows the block diagram of the proposed 3X oversampling CDR that consists of a phase detector (PD), a rotating signal generator (RSG), a phase rotator (PR), a MUX for data recovery (DMUX), and another MUX for clock extraction (ClkMUX). The CDR has a fully differential architecture, in Fig. 5.1, all the signal paths, which are shown in single line, represent the differential signal.

The data is sampled in each Baud period by three clock phases *{Clkl, Clkl* and *Clk3)* separated by a *T/3* time interval. One of the three clock phases is dynamically selected as the Data sampling clock phase (DSCP), which is the clock phase from which the sample is taken as the recovered data. The DSCP is continuously adjusted by the CDR so that the DSCP is kept close to the middle of the data eye, so as to reduce the BERoftheCDR.

In order to achieve such a dynamic adjustment of the DSCP point, the rising edge of the DSCP is compared with the data transition edge to generate a phase error between them at the phase detector. The phase error is defined as the phase difference between the DSCP and the middle phase, which is the 2nd phase after the data transition, and thus the phase error is produced as a multiple of *T/3.* In an ideal case, the middle phase is aligned with the DSCP (see Fig. 5.4), which is also the center of the data eye, thus producing a zero phase error. So in the case of zero jitter, an ideal case, the DSCP will be 2^n clock phase after the data transition. The phase error is assumed to be zero in the absence of a data transition.

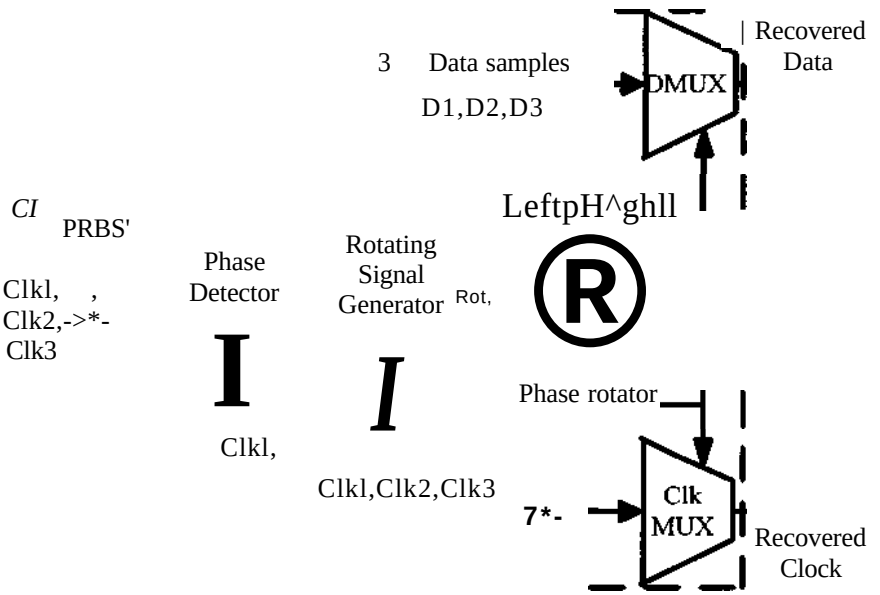


Fig. 5.1 Functional Block Diagram of proposed CDR

The phase detector generates a request signal *L*, if the data transition point is leading in time with respect to that of the zero phase error position, as shown in Fig. 5.2, thus generating a phase error of -773. A request signal *R* is outputted if the data transition point is lagging in time with respect to that of the zero phase error point and generating a phase error of *T/3* (see Fig. 5.3). The *L* and *R* signals indicate that the current DSCP is to be rotated left (advanced in time), and right (retarded in time) in order to keep the DSCP closer to the center of the data eye. According to the proposed technique, *L* and *R* signals are generated only if the magnitude of the corresponding phase error is larger than *T/3*. If the magnitude of the phase error is less than *T/3*, the current DSCP will remain unchanged. The *T/3* is considered as the decision threshold of the CDR.

However, in order to improve high frequency jitter tolerance performance, the CDR does not update the current DSCP in every Baud period, even though *L* and *R* request signals are produced. The DSCP position is updated once within each timing window (TW) of an eight Baud period. Updating in each eight Baud period will force the CDR to accumulate low frequency jitter, thus degrading the low frequency jitter tolerance performance. However updating the DSCP once in each TW leads to high frequency jitter averaging, which is the most common way to improve high frequency jitter tolerance performance.

Since the clock phase is updated at the end of the 8 Baud period interval, the CDR has a data acquisition time of an 8 Baud period. The suggested specification for a data acquisition time of 2.5 Gbps Gigabit passive optical networks (GPON) is 44 bits [11]. Therefore the proposed CDR is well suited for a GPON application.

The RSG collects the *L* and *R* signals within each TW and stores them, and generates a command signal according to the following conditions,

(1) generate *RotL,* if RSG finds at least one *L* signal but no *R* signal within a TW

(2) generate *RotR,* if RSG finds at least one *R* signal but no *L* signal within the TW

If both *L* and *R* signals are present within any TW, no command signal will be generated.

The command signal *RotL* or *RotR* is sent to the phase rotator. The phase rotator outputs three clock selection signals *(CS1, CS2* and *CS3).* At any given time, one of three clock signals will be active (logic '1'), which corresponds to the DSCP point. For example if *CS2* is high the DSCP will be *Clk2.* When the phase rotator receives a request signal *L* (or *R)* and an active *RotL (or RotR)* signal, it rotates the clock selection signal left (or right), thus rotating the DSCP point left (or right) by *T/3.*

Figures 5.2-5.4 show the three different scenarios that are possible between the data transition point and the DSCP. In the case where binary data of '0100', with a Baud period of *T* is assumed, the phase relation between the data transition and the DSCP, from the second bit, is considered. All three clock phases are marked as 1, 2 and 3, and the DSCP point is shown by a solid arrow. The first scenario is shown in Fig. 5.2, where the data transition is leading and generating a phase error of -T/3, and therefore the DSCP is switched from *Clk2* to *Clk 1* (rotated left) at the next bit. Then the DSCP is kept as *Clkl* as there is no phase error is detected.

In Fig. 5.3, a phase error of *T/3* is detected. Thus the DSCP is rotated right from *Clk2* to *Clk 3,* and then kept unchanged as no phase error or data transition is detected. The ideal case is shown in Fig. 5.4 where the DSCP and the middle phase are aligned resulting in a zero phase error. Therefore the DSCP remains as *Clk 2.*

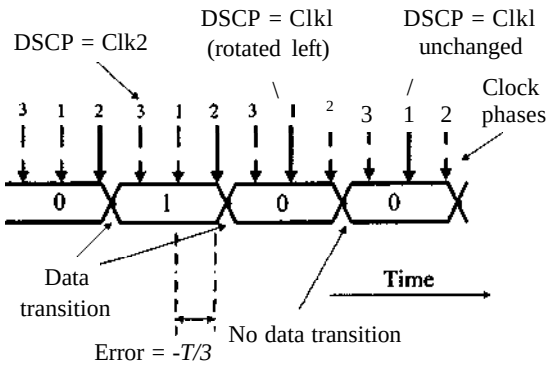


Fig. 5.2 Phase error of-773

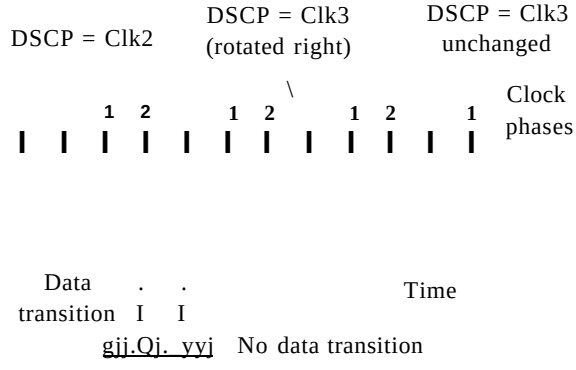


Fig. 5.3 Phase error of *T/3*

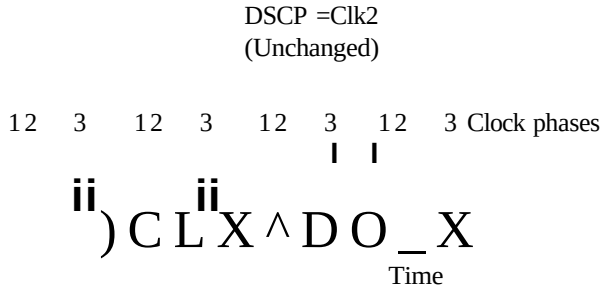


Fig. 5.4 Phase error of zero

5.3 Mathematical Analysis of CDR Jitter Tolerance

The incoming data is accompanied with jitter that is assumed to be sinusoidal. A sinusoidal jitter at jitter frequency /. can be expressed as

$$</>j = 27TAJ\ sin[2nfjt) \quad \text{rads}, \tag{5.1}$$

where Aj is jitter amplitude in Unit Interval (UI).

From (5.1), the phase change speed (PCS) can be obtained by differentiating the jitter function with respect to the time.

$$PCS = \ ^{\wedge J \wedge "} = Aj\{in)^2 fj\ cos(2nf/) \ \text{rad/sec}, \tag{5.2}$$

From (5.2), the maximum phase change speed (MPCS) is evaluated in terms of normalized jitter frequency, which is the ratio between jitter frequency and Baud rate.

$$MPCS = Ajiln)^2 fjrad\ I\ \text{sec} = A_j 2nF_j \quad \text{UI/UI} \tag{5.3}$$

Since the phase is updated by 1/3 *UI* once in eight Baud periods, the minimum phase updating speed (MPUS) can be expressed as,

$$MPUS = \{\backslash l3\backslash D_{Trans}) \tag{5.4}$$

Where D_{Trans} is the minimum transition density of the incoming data, where the transition density is defined as the number of transition per number of baud periods.

This minimum phase updating speed should be larger than the MPCS determined in (5.3) in order to avoid bit errors. The maximum peak-to-peak low frequency jitter amplitude $[j_{T0L\ L0W}$) that can be tolerated by the CDR is obtained as,

$$JTOL\text{-}LOWV\sim'^{*})\ ^{=}\ \sim z\quad 7T\sim \qquad \text{P--7}$$

If PRBS* data is used as the source of data to the CDR, the low frequency jitter tolerance can be expressed by the following equation,

$$^{J}TOL\text{-}WW\backslash_{PRBS}\ -\ \frac{\boldsymbol{D_T}\,\mathbf{I}}{\wedge\ \underline{Trans\ \backslash_{PRBS}}\ F}$$

In this analysis PRBS of length 2^7 -1 (PRBS7) is used as incoming data to the CDR. The minimum transition density of the PRBS7 data is 1/13, and it occurs at once in 127 bits [10]. The maximum peak-to-peak low frequency jitter tolerance of the CDR $u_{Tnr\ \ ,nw}$ I $_{V\ \ TOL\_LOW\ \backslash_{PRBS7}\ )}$) for PRBS7 data can be expressed as,

$$\underset{^{J}TOL\text{-}WW\backslash_{PRBS1}}{\boldsymbol{J}\ \ \mathbf{I}} = \underset{\wedge p}{<}\ \wedge\ \underset{U\ i}{\mathbf{UI}} \qquad (5\ 7)$$

At high jitter frequencies, the jitter period is smaller than at low jitter frequencies, so data transition may not occur within one jitter period. Between two transitions, which is larger than a jitter period, the average phase change is equal to A_j, which should be less than the phase change limit of 1/3 UI for a 3X oversampling CDR. So the peak-to-peak high frequency jitter $J_{tol}\,(ill)$ can be determined as

$$J_{tol}(Ul) = \backslash \qquad (5.8)$$

* Pseudo random binary sequence (PRBS) is used as the source of random data in digital communication circuits. The longer the length of the sequence the better the approximation of a truly random sequence.

From the low frequency jitter function (5.7), and the high frequency jitter function (5.8), the normalized jitter corner frequency $(F_c)_{PRBS7}$ can be calculated as.

$$F_I = \text{——} \quad (5\ 9)$$

Equation 5.7 can be rewritten in log scale as,

$$M^J TOL\_Low)_{PRBS1} = \log \wedge \text{—} J - \log F, \quad (5.10)$$

From equation 5.10, the slope of the low frequency jitter variation is found as -1 (20 dB/decade).

5.4 Simulation Setup for CDR Jitter Tolerance Estimation

Jitter tolerance is evaluated as the amount of peak-to-peak sinusoidal jitter that can be tolerated by the CDR without an onset of errors or exceeding the specified BER [3]. Fig. 5.5 shows the simulation setup to find the bit error produced by the CDR. First, sinusoidal jitter is generated for a given jitter frequency and amplitude which is then combined with the PRBS data to produce the jittered data. The CDR receives the jittered data and three clock signals, and outputs the recovered data. The recovered data is compared with the original data at a bit error tester to produce the bit errors.

Since the jitter tolerance estimation is performed for the sinusoidal jitter (not random jitter) simulation runs until the occurrence of the first bit error. To find the jitter tolerance at a given jitter frequency (/.), a jitter amplitude is set at an initial value (4), and the bit error tester output is observed. If any bit errors are detected, the jitter amplitude is reduced by a small step, A/4, and the simulation runs again. The amplitude at which errors stop to appear would be the jitter tolerance at that jitter frequency. Alternatively, if the initial amplitude produces no bit error, the jitter amplitude is stepwise increased by *M.,* until the CDR produces a bit error. The corresponding amplitude would be the jitter tolerance at that jitter frequency. In order to obtain the

jitter tolerance over a frequency range, the frequency is changed by small frequency steps, A/] to define a set of discrete frequencies. The above procedure is repeated for each discrete frequency value to get the corresponding jitter tolerance.

5.5 Conventional simulation

Conventional simulation tools use a time sweep with a certain time step to calculate the signal level at each time point [28]. This type of simulation is called fixed-time-step simulation [28]. A simulation requires a sufficiently small time step to ensure that good timing resolution is achieved. A small time step results in a large number of signal points that needs to be evaluated. Therefore the simulation time will be dramatically increased for a simulation such as the one used to find jitter tolerance [4].

In the jitter tolerance simulation, the accuracy of the jitter tolerance value depends on the amplitude and frequency step sizes used. The smaller the step size, the better the accuracy. However, a smaller step size will require more iteration, thus increasing the simulation time.

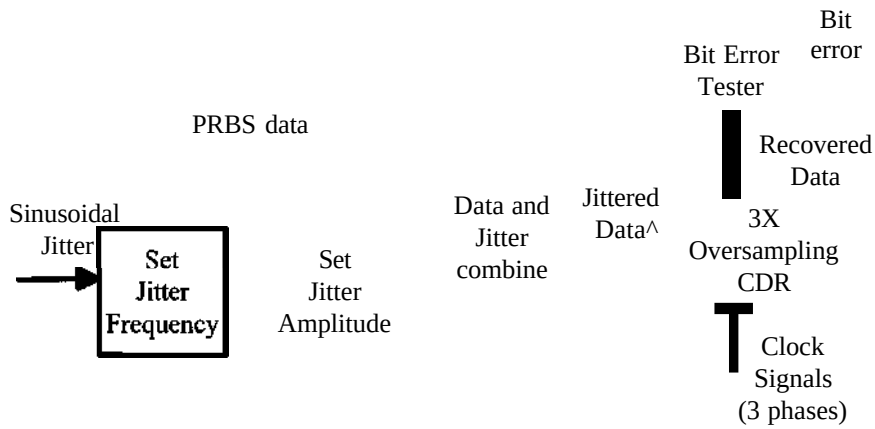


Fig. 5.5 Simulation setup to find bit errors

5.6 Concept of Event-Driven Modelling and Simulation

In an event-driven simulation, only the time instants of interest are used. In an oversampling CDR, even though the real data and clock signals have values at all instants

of time, the transition time points are considered to be the time points of interest. By using only the time point of interest, the simulations are no longer driven by a regular increment of time, but rather they are driven by the events at the transition points. Event-driven simulations require much less time than conventional fixed-time-step simulations to perform the same task [28].

In an event-driven simulation model, each event is defined by three fields: event time, event index, and event parameter [27]. Event time is the exact time instant at which the event is to be executed. Event index indicates the functional block in which the event is generated and executed. For example, the event index "PDindex" indicates that events are from the phase detector (PD). An event parameter is used to control different event tasks by assigning different event parameter values.

An event driven simulation model has three major time blocks, the event generator, event dispatcher and event handler. The data flow among these blocks is shown in Fig. 5.6 [27]. All the functional blocks are modelled within the event handler. For example, in a CDR system, the phase detector, and phase rotators are some of the functional blocks modelled in the event handler.

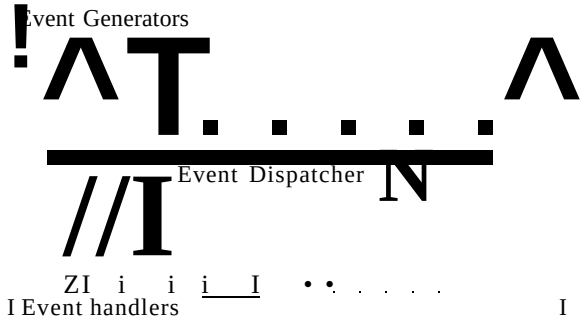


Fig. 5.6 Event driven program concept [27]

5.6.1 Event Generators

Events are generated by one of the functional blocks represented by the event handler, except for the events that are generated at the start of simulation by initializing each functional block at time zero with a parameter of zero. The Matlab code for

initializing the functional blocks is shown in List 1 [27]. The Matlab function *iAdd_Evenf* is called to generate the event at time zero by passing the parameter of zero.

List 1: Matlab code to initialize the functional blocks [27]

```
event('initialize) % reset all functional block by passing parameter ofO at time ofO
event ('Add_Event',@time=0, PRBS_generator, parameter = 0)
event(Add_Event', @time =0, Phase Detector, parameter =0)
event ('Add_Event',@time-0, Phase Rotator, parameter = 0)
event(Add_Event', @time =0, Data_MUX, parameter =0)
end
```

5.6.2 Event dispatcher

The event dispatcher receives all the event requests in the order in which they are generated according to the functional operation of the CDR. Within the event dispatcher, an event management function sorts the events in an order in which the events are to be executed and stores them in a queue.

5.6.3 Event handler

The event handler executes the events in the order in which they appear in the queue. Once an event is executed by the event handler that event will be deleted from the queue. In addition to executing events, the event handler is also responsible for generating the subsequent event according to the CDR operation.

Each functional block is modeled by writing a Matlab function. A Matlab code for a functional block is shown in List 2 [27]. Each function is defined by two arguments: time and parameter. Different parameters (cases) are used to perform different event tasks. For example, in a clock generation function, at case 0 the first clock edge is generated at time zero (initialization) and at case 1 the next clock edge is generated after a time of clock period *(T).*

List 2. Matlab code for a functional block [27]

```
function Block_Name(time, parameter)
persistent vor 1, var2
switch parameter
case 0 %initialize the block
    < initialization>
case 1
    <processing of event type 1>
case 2
    <processing of event type 2>


Case N
    <processing of event type N>
End
```

5.7 Event-driven modelling of the proposed CDR

The event driven model is developed for each functional block of the CDR system that is used to perform the jitter tolerance performance test of the CDR. The functional blocks are the Sinusoidal Clock Generator, Jittered PRBS generator, Phase Detector (PD), Rotating signal generator (RSG), Phase Rotator (PR) and data multiplexer (DMUX). The Matlab simulation model of each block is discussed in the following sections and finally the simulation results are presented. The detailed Matlab code is presented in appendix I.

5.7.1 *Sinusoidal jittered clock generator*

In order to find the data edges, modulated by sinusoidal jitter, a phase modulated sinusoidal signal is analysed, which can be expressed as.

$$A(t) = Sin(2nf_d t + 2TIA_J \sin(2\#"/)), \quad (5.9)$$

where $A\{f)$ is sinusoidal phase modulated signal and f_d is the Baud rate.

List 3 shows the Matlab code of an event-driven functional model that outputs the jittered edges of the signal $A(f)$ as the global parameter 'period'. The function 'Clock' has two fields, "time" and "para". The case statements are used to control the process, in

'case 0', the event is initialized by generating an event with time zero. In 'case 1', the zero crossing point (zerocross) and even numbered zero-crossing (even_zero_cross) points are identified and the jittered period is calculated.

List 3: Matlab algorithm for jittered clock

```
function Clock (time, para)
global Clocklndex, period, aj, fj
persistent pointer, pointer!, zero cross, even_zero cross, PrevP
switch para,
case 0          % intialize the event
        AddEvent (0, clock index, 1)
        pointer =1; pointer2=0; pointer3=0;

case 1          %find the zero crossing points and calculate jittered clock period (period)
        for time=0:le-12:2*endtime
                pointer = pointer+1;
                P=sin((2 *pi *(fj) *time) + pi *aj*sin(2 *pi *(fj) *time)))
                if((P<0) && (Prev_P>0)) \\((P>0) && (Prev_P<0))
                        pointer 2 =pointer2+1;
                        zero _cross=time;
                        if (rem(pointer2,2) ==0)
                                pointer 3 =pointer3+1;
                                even_zero_cross(pointer3)=zero cross; end
                        if (pointer 3> 1)
                                period(pointer3-)=even_zero_cross(pointer3)-
                                                even_zero_cross(pointer3-l); end
                end
                Prev_P=P;
        end
end
```

5.7.2 Jittered PRBS generator

After generating the PRBS data pattern using a Matlab function (PRBS), jitter is added to the data by setting duration of each data bit by the corresponding jittered period. The Matlab code for functional block "JPRBS", shown in List 4, generates a PRBS data stream that is modulated by sinusoidal jitter at the data rate for a given period of time (endtime).

First, in 'case 0', the event is initialized by generating an event at time zero. Also in 'case 0', the Matlab function 'PRBS' is called to make PRBS data bits available so that they can combine with jitter. A pointer is used to assign each data bit for the duration of the jittered period. This is done by generating (adding) a subsequent event at the time of "time+period(pointer)". The value of the 'period' (pointer) refers to the jittered period of the corresponding data bit. In other words, if one data edge is set at a time instant, the next data edge is set after the jittered period of time.

List 4: Matlab code for Jittered PRBS data

```
function J_PRBS(time,para)
global J PRBS Index, Clocklndex, period, Datain, endtime
persistent pointer, Data
switch para
case 0 % intialize the event and call the PRBS function to generate data
        pointer = 1;
        AddEvent (0, JPRBSIndex, 1);
        Data = PRBS(initial, endtime);
case 1 % Generate the Jittered PRBS Data
        Datain(pointer) — Data(pointer);
        Add_Event(time+period(pointer), JPRBSIndex, I);
        pointer = pointer+1;
end
```

5.7.3 Phase Detector

The event-driven Matlab code for the phase detector is shown in List 5. In 'case 0', events are initialized at time zero and a parameter of zero. In case 1, three data samples, separated by T/3, are generated in each nominal Baud period (7), and adjacent samples are compared to detect the transition. If a transition is detected an event is triggered to case 2 where the center of the data eye (centerpoint) is compared with the DSCP point (samplepoint), and the phase error is calculated. In Case3, the current DSCP point is captured as a 'samplepoint' by the event that is triggered by phase rotator in order to compare the transition point with the DSCP point. Once the phase error is calculated an event is triggered to case 4, where the *L* and *R* signals are generated according to the

sign of the phase error. The phase error and *L* and *R* signals are calculated only if a data transition is detected.

List 5: Matlab algorithm for phase detector

```
function  PD(time,para)
global PRBSIndex, PD  Index, PRIndex, Datain,  Data  samples, L, R ;
persistent pointer, pointer2, T, phase  error, i, centepoint, samplepoint; state;
switch para
case 0      % intialize the events
        pointer  = 1; pointer2 =1;
        Add_Event (0, PD  Index, I);
case 1  % generate the data samples
        Data_samples(pointer2)  = Data_in(pointer);
        if (rem(pointer2,3) ~= 0) begin
                Add_Event(time+T/3, PD  Index, I);
        end
        if (Datasamples(i) ~= Datasamples (i-1) % detect the transition
                AddEventftime,   PD_Index,2);
        end
        pointer2  = pointer2+l;

case 2 %  Compare the data eye center (centerpoint) with samplepoint
        centerpoint= time;
        if state  ==I
                phase  error = centerpoint -samplepoint;
                Add Event (time, PDIndex, 4);
        else
                state =2;
        end

case 3 % receives the timing information of DSCP from Phase Rotator (PR)
        samplepoint=time;
        if state  ==2
                phase  error =centerpoint-samplepoint;
                Add_Event(time, PD  Index, 4);
        else
                state =//
        end
        Add_Event(time,RSG  Index, 1);      % send the current DSCP edge to RSG

case 4 % generating L and R signals
        if fphase error >0)
                R = round (phase_error/(T/3));
```

```
        else if(phase  error <0)
                L = round (-phase_error/(T/3));
        end
end
```

5.1 A Rotating signal generator (RSG)

The phase rotation action is carried out once in a timing window (TW) of eight symbols. List 6 describes the Matlab code for RSG. *Ls* and *Rs* are the total number of request signal *L* and *R* within given TW respectively. The signal *Ls, Rs,* and command signals *RotL* and *RotR* are initialized in case 0. In case 1, the number of request signals *L* and *R* are counted within the TW and total number is stored as *Ls* and *Rs* respectively. The command signals are generated at the end of each TW according to the proposed decision technique. An event is triggered to the phase rotator to send the command signal *RotL* and *RotR* to the phase rotator to update the phase.

List 6: Matlab code for Rotating signal generator

```
function RSG (time,par a)
global RSG  Index, PD  Index, PRJndex, L, R, RotL, RotR;
persistent pointer, Ls, Rs;
switch para
case 0      % initializes events
        pointer  = 1; Ls=0; Rs=0;
        RotR  =  0;  Rotj=0;
        Add_Event (0, RSG  Index, 1)
case 1
        if (rem(pointer,8) ==1) % reset the L, R count after each timing window (TW)
                Ls=0; and Rs=0;
        end
        Ls = Ls+1;              %counting number ofL within TW
        Rs= Rs+1;
        if(rem(pointer,8) == 0)
                if((Rs~=0) && (Ls ==0)
                        RotR  =  I;
                        Add Event (time, PRJndex, I); % send the DSCP to PR
                elseif((Rs ==0) && (Ls~=0))
                        Rot_L=  I;
                        Add Event(time, PRJndex, 1); % send the DSCP to PR
                Else
```

```
                RotL=0;
                RotR  =0;
                Add_Event(time,PRJndex,l);  % send the DSCP  to PR
            end

        else
        Rot J =0; Rot_R=0;
        Add_Event(time,PRJndex,l);  % send the DSCP  to PR
        end
        pointer  = pointer+I;
end
```

5.7.5 Phase rotator (PR)

The Matlab code for the phase rotator module is shown in List 7. 'Case 0' is used to initialize the event by passing a parameter of zero at time zero. The phase rotation action is performed at case 1. For example, when PR receives a non-zero command signal *RotR,* the current DSCP point *(Clk2)* is retarded by adding '7/+773'. The updated DSCP point *(Clk3)* for the next symbol is defined by moving the current sampling phase by *4T/3.* In the case of the request signal *L,* the next DSCP point is set by adding ' 7-773' to the current DSCP point. The DSCP point information (DataSamplesPt) is sent to the DMUX by triggering an event at case 2 to extract the corresponding data sample as the recovered data.

List 7: Matlab code for phase rotator,

```
function  PR(time,para)
global PRJndex, RSGJndex, L, R, RotL, RotR,, DataSamplePt, DMUXJndex;
persistent pointer2, phase step;
switch para
case 0      % intialize events
        pointer  = 1; phase  step =T/3;
        AddJEvent (0, PRJndex,  1)
case 1
        if (rem(pointer,8) == 0)
                if((Rot_L==0)  &&  (RotR ~=0))
                        Add_Event(time+phase_step+T, PD  Index, 3);
                        Add_Event(time+phase_step+T,  PRJndex,  2);
                        Rot_R=0;
                elseif((RotJ~=0)  &&  (Rot_R==0))
                        Add_Event(time-phase_step+T, PD  Index, 3);
```

```
                    Add_Event(time-phase_step+T, PR  Index, 2);
                    Rot_L=0;
            else
                    AddJEvent (time +T, PDIndex, 3);
                    Add_Event(time+T,  PRIndex,  2);
                    Rot_R=0; Rot_L=0;
        else
                Add_Event(time  +T,  PDIndex,  3);
                Add_Event(time+T,  PRIndex,  2);
                Rot_R=0; Rot_L=0;
        end
        pointer = pointer+1 ;
        L =0; R =0; % reset L and R to zero after phase rotation
case 2 % send the current sampling clock phase information to the Data Recovering
module (MUX)
        DataSamplePt(pointer)=      time;
        Add_Event(time,DMUX_Index, I);
end
```

5.7.6 Data MUX (DMUX)

DMUX receives the current DSCP point (Data_Sample_Pt) for each symbol as the control signal and the sampled data. The data sample that is sampled by the DSCP is output as the recovered data. List 8 shows the Matlab code for DMUX where case 0 is used to initialize the event. In case 2, the DSCP point information is received by an event that is triggered at the phase rotator. The DSCP timing information is captured as 'samplepoint', which is processed to find the corresponding sample number in the data stream. The detected sample for each symbol is extracted as the recovered data (RecoveredData). Recovered data is compared with original data, which is the input data to the CDR, to generate the bit error and (BER).

List 8: Matlab code for DMUX

```
function  DMUX(time,para)
global    Datasamples   RecoveredData   DMUXIndex

persistent pointer, T, datasample  time;
switch para
   case 0 % Initialization of events
        pointer=I;
```

```
        Add_Event(0,DMUX_Index, I);
  case 1
        samplepoint = time % receives the data sampling points
        i=(round((samplepoint-prev_samplepoint)/(T/3)));%identifythe    corresponding
                                                        %data samples within the symbol
       j=j+i;                                           % defining the exact location of
                                                        % the sample


        Recovered_Data(pointer)=Data_samples(j);
        pointer =pointer+1;
        prev   samplepoint=samplepoint;
        ij>rev=i;
end
```

5.8 Simulation Results

First the simulation time of the conventional fixed-time-step model and the event-driven model are estimated and compared. Table 1 compares the simulation time for both models. The fixed-time-step model uses a time step value of 1/100 of the Baud period. In both cases the simulation is run for 1000 Baud period intervals. The event-driven simulation is found to be 30 times faster than conventional simulation with fixed-time-step of 1/100 of baud period

Table 5.1 Simulation time comparison

	Conventional fixed-time-step	Event-driven
Simulation time	120 s	4 s

To obtain the jitter tolerance of the CDR, the event-driven simulation is executed for 20,000 symbols of PRBS7 data. Since the 3X oversampling is employed total of 60,000 samples are processed. The variation of jitter tolerance against jitter frequency is obtained. The simulated result and theoretical values based on (5.7) and (5.8) are

presented in Fig. 5.7. The simulation results show a very close match to the theoretical values. The simulated high frequency jitter tolerance of 0.66 is very close to the theoretical value of 0.67 from (5.8). The normalized jitter corner frequency obtained from simulation (0.014) is close to the calculated value of 0.012 from (5.9). For a longer PRBS sequence (here PRBS 7) the error might be larger.

The Fig. 5.7 also shows the theoretical jitter tolerance for PRBS 15 (2^{15} -1) and PRBS23 (2^{23} -1). It can be noted that jitter tolerance at high frequency is constant for the CDR regardless of the PRBS data pattern. However the low frequency jitter tolerance degrades for longer PRBS. This is because the longer PRBS has lower transition density thus reducing the updating speed of the CDR.

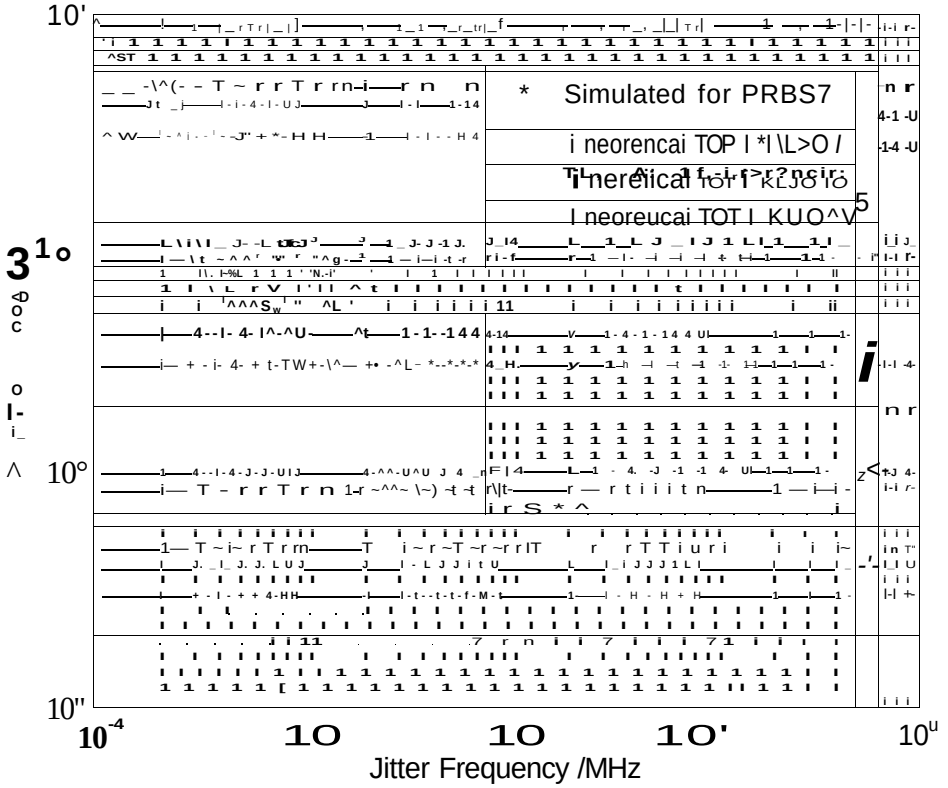


Fig 5.7: Jitter tolerance of proposed CDR

Chapter 6

Design and Implementation of the Proposed CDR

6.1 Chapter overview

In this chapter, the circuit detail and working principle of each functional block of the proposed CDR are presented. All the functional blocks that are inside the dotted line (see Fig. 6.1), are integrated on a chip. The post-layout simulation results and the

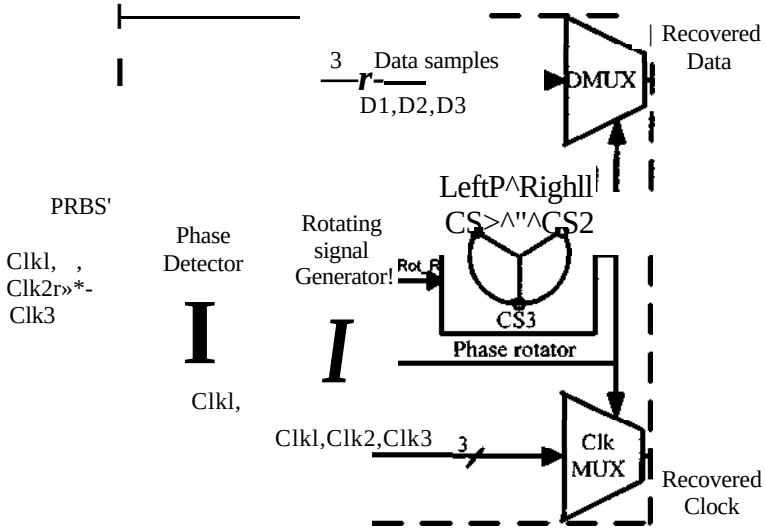


Fig. 6.1 Functional Block Diagram of proposed CDR

(Fig. 5.1 is repeated for convenience)

comparison of the CDR performance parameters are also reported in this chapter. This chapter also reports the critical path failure analysis of the CDR and proposed circuit modifications to avoid circuit failures.

6.2 Details of the proposed CDR circuit design

The proposed CDR is implemented using CMOS 65nm process technology. The circuit details and operations of all the functional blocks of the CDR are explained in this section.

6.2.1. Phase detector

The phase detector can be divided into two sub-blocks, the transition detector and the L-R generating logic circuit, as shown in Fig. 6.2. At the transition detector, data is first sampled by the three clock phases *Clkl, Clk2* and *Clk3* and generated the data samples *Dl, D2* and *D3* respectively. XOR compares the adjacent samples in order to detect the transition. The output of the XOR and the clock signal whose data sample is not used as an input to the XOR, are applied to an AND gate and transition signals are produced. For example, when samples *Dl* and *D2* are applied to the XOR gate, the output is ANDed with *Clk 3* and generates transition signal *T12*. The AND operation removes the unwanted pulses and limits the pulse width of transition signals to *T/2*.

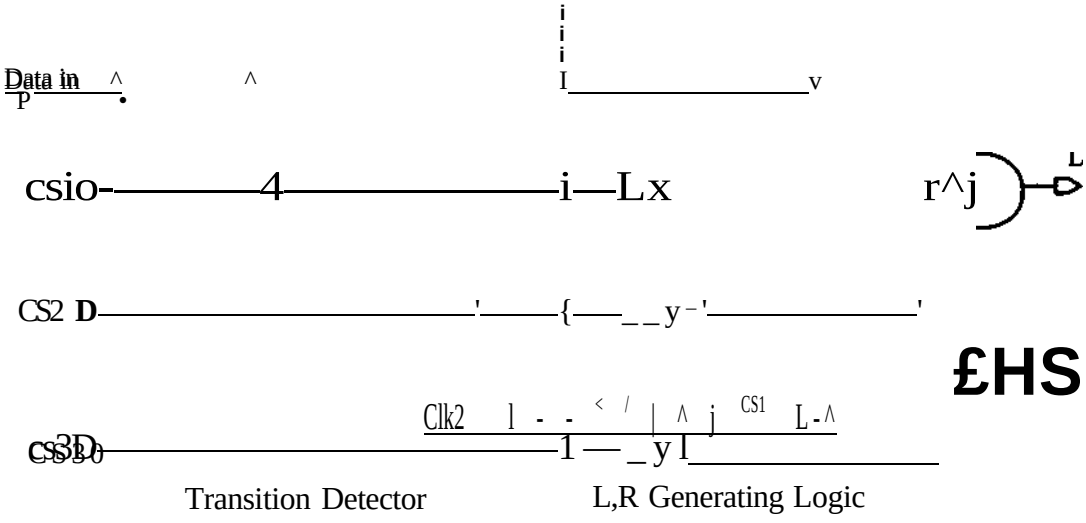


Fig. 6.2: Circuit diagram of the phase detector

The L-R generating logic receives the transition signals and processes them with clock selection (CS) signals. The CS signals are the output from the phase rotator; at a given time instant only one of three CS signals will be active (high). The active CS signal selects the corresponding clock phase signal as the current DSCP. For example, if the *CS1* is at logic 'high', the DSCP will be *Clkl.* The transition signals are ANDed with corresponding CS signals to determine the direction of rotation that has to be performed to keep the DSCP closer to the data eye. For example, the presence of a transition signal 772 implies that there is a data transition between *Clkl* and *Clk2,* and therefore *Clk3* will be closer to the data eye. If *Clk3* is the DSCP, there is no phase rotation action needed, i.e. neither *L* nor *R* signal is to be generated. Referring the L-R generating logic in Fig. 6.1, the transition signal *T12* is ANDed with *CS2* and *CS1,* and since *CS1* and *CS2* are not active signals, both *L* and *R* signals at zero. If the DSCP is *Clkl,* in order to move the DSCP closer to center of the data eye, the DSCP has to be moved away from the data transition point. Since the current DSCP *(Clkl)* is located on the left side of the data transition, the DSCP has to be rotated left to *Clk3.* The *L* and *R* signal is expressed in terms of transition signal and CS signals

$$L = Tl2\text{-}CSl + T23\text{-}CS2 + T3l\text{-}CS3 \quad (6.1)$$

$$i? = n2\text{-}CS'2 + 7'23\text{-}CS3 + 7'31C5'l \quad (6.2)$$

Table 6.1 shows the required request signal *(L* or *R)* and the updated DSCP for all the combination of transition signals and CS signals. Depending on the incoming data and its associated jitter, the transition point may vary, i.e. the transition signal may vary.

6.2.2 Rotating signal generator (RSG)

A gate level circuit diagram of the RSG is shown in Fig. 6.3. The circuit shown below the dotted line is a frequency divider and the output of the DFF7 has a frequency that is 1/8 of the input clock frequency. Two AND gates and an OR gate are used to generate a narrow pulse *(TW_pulse),* with a pulse width of *T,* once in an eight Baud period. The OR gate allows the CDR to synchronize with reset signal *(RESET).*

The timing window is defined by the time duration between the two adjacent *TW_pulses.*

Table 6.1: Phase updating pattern of proposed CDR

Transition Signal	CS	DSCP	L,R signal	Updated CS	Updated DSCP
772	*CS1*	*CM*	*L*	*CS3*	***cm***
	CS2	*Clk2*	*R*	*CS3*	***cm***
	CS3	***cm***	-	*CS3*	***cm***
T23	*CS1*	*CM*	-	*CS1*	*CM*
	CS2	*CM*	*L*	*CS1*	*CM*
	CS3	*Clk3*	*R*	*CS1*	*CM*
T31	*CS1*	*CM*	*R*	*CS1*	*CM*
	CS2	*CM*	-	*CS1*	*CM*
	CS3	***cm***	*L*	*CS1*	*CM*

The *TW_pulse* is used to reset the DFF1 and DFF2, which are clocked by request signals *L* and *R,* respectively. For example, when the DFF1 receives *L* signal the flip-flop triggers logic T at the output, and keeps the logic until the DFF1 is reset by the next *TW_pulse.* At the start of each timing window, the DFF1 and DFF2 outputs are reset, which allows the circuit to collect the request signals *L* and *R* within that timing window. The outputs of DFF1 and DFF2 are applied to a XOR gate that outputs logic '0' if both DFF outputs are logic '1'. Otherwise it outputs logic '1'. XOR output is sent to two flip-flops DFF3 and DFF4. DFF1 and DFF2 outputs are reset from logic '1' to logic '0' if the DFFs receive any *L* or *R* signals. Rising edge of the inverted output of DFF 1 and DFF2 is used to clock the DFF3 and DFF4, respectively. The RSG outputs the command signal *RotL* or *RotR* at the end of the timing window.

For example if only an *L* (or *R)* signal is received, DFF3 (or DFF4) receives logic '1' at its data input and outputs a logic '1' at *RotL* (or *RotR).* The *RotL* and *RotR*

signals are reset to logic '0' at the middle of the timing window to enable the DFF3 and DFF4 output line to capture the request for the next timing window, respectively.

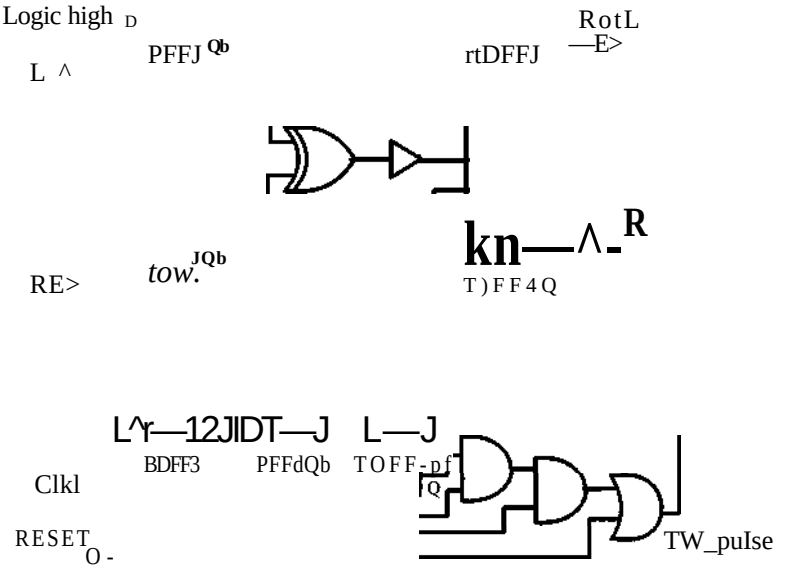


Fig. 6.3 circuit diagram of the rotating signal generator

6.2.3 Phase rotator

Fig. 6.4 shows the top level diagram of the phase rotator, which has three rotating cells. A detailed circuit diagram of the rotating cell is shown in Fig. 6.5. Celll and cell3 uses a flip-flop with reset (DFF-R), and cell2 uses a flip-flop with set (DFF-S). When the CDR reset is asserted the *CS2* will be triggered to logic '1', and *CS1* and *CS3* will be set to logic '0'.

When the phase rotator receives command signal *(RotL* or *RotR)* the phase rotator rotates the active clock signal depending on whether the request is left or right. For example assume that the current clock selection signal is *CS2,* and the phase rotator receives *RotL.* At rotating celll, *VL* is high and *VR* is low, so NAND1 outputs a logic '1', and thus NAND3 produces a logic '1'. The rising edge of the *RotL* signal triggers the DFF-R and switches CS1 to logic '1' (active). Rotating cell2 has both *VL* and *VR* low, which makes the NAND1 and 2 outputs to be logic '1', and so the DFF-S data input will be at logic '0'. The rising edge of the *RotL* signal triggers the DFF and passes the

logic '0' to its output, i.e. CS2 will turn to logic '0' (inactive). At Cell3, even though VR is high, outputs of NAND1 and 2 are still logic '0'. Therefore it produces the same result as cel!2. *CS3* will be kept unchanged at logic '0'. In this example, when the phase rotator receives a *RotL* signal, the active clock selection signal is switched from *CS2* to *CS1,* i.e., rotated left.

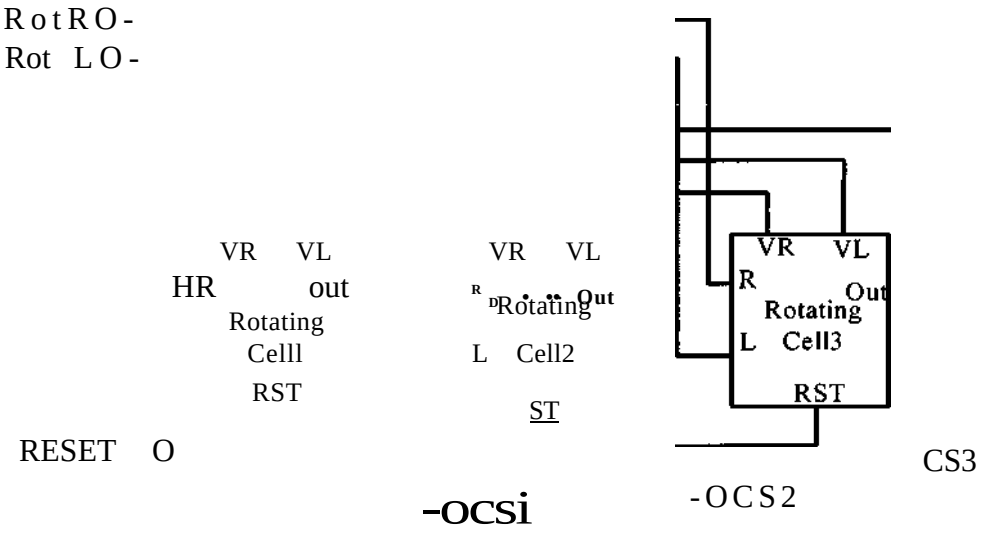


Fig. 6.4 Block diagram of the phase rotator

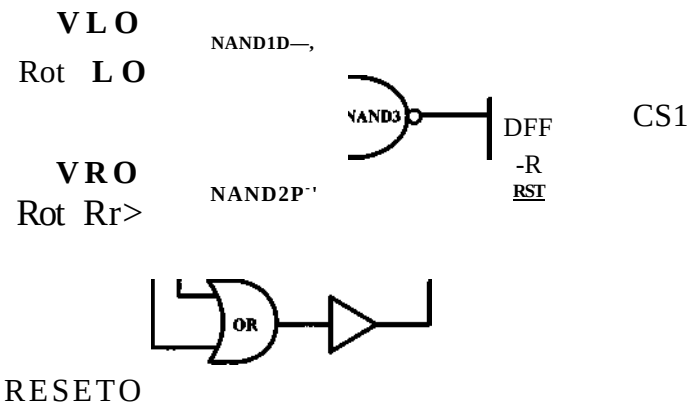


Fig. 6.5 Circuit diagram of a rotating cell (Celll)

The caption at Fig. 6.6 shows the post layout simulation result, where the top waveform *{TW_pulse)* is the pulse that is generated at the beginning of each timing window. The other three waveforms are the phase rotator output. To observe the phase tracking behaviour, the frequency of the clock signal (5 GHz) and the data signal (4.77 Gbps) are set at slightly different values. It can be observed that the clock selection signal is continuously rotated.

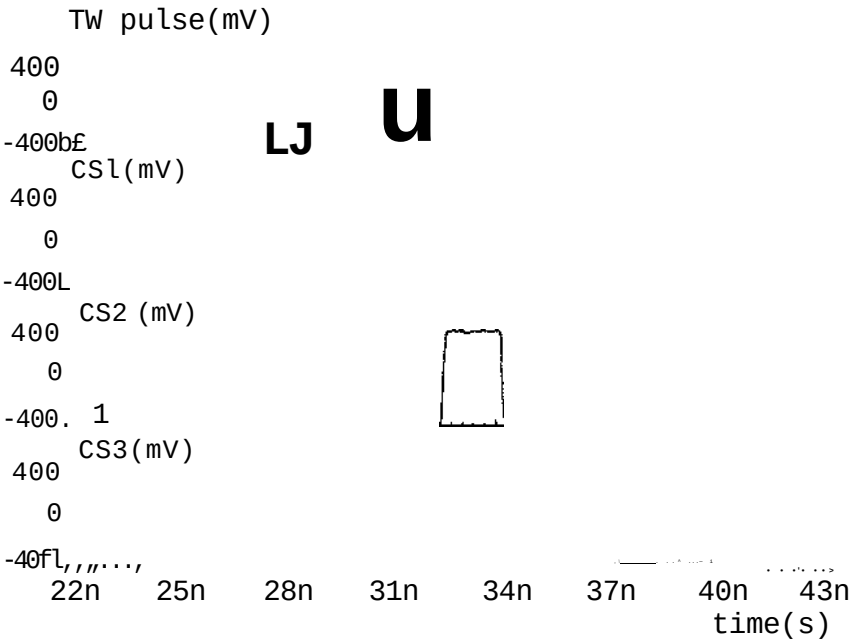


Fig. 6.6 Rotation of the CDR phase

6.2.4 MUX

The CDR has two MUXs, one to extract the clock (ClkMUX), and one to recover the data (DMUX). The CML based transistor level MUX circuit is shown in Fig. 6.7. As a DMUX, all three pairs of differential data samples *(Dlp/Dln, D2p/D2n* and *D3p/D3n),* from the phase detector are sent to the MUX inputs. The positive clock selection signals *(CSlp, CS2p* and *CS3p)* are used as the control signals. For example, if we assume that CS2 is active at a given Baud period, the corresponding transistor will turn on, and thus D2 will be passed to the output (Dout) as the recovered data. ClkMUX works in the same way as the DMUX, as it picks up the corresponding clock phase. For example, when *CS1* is active, *Clkl* will be selected as the extracted clock.

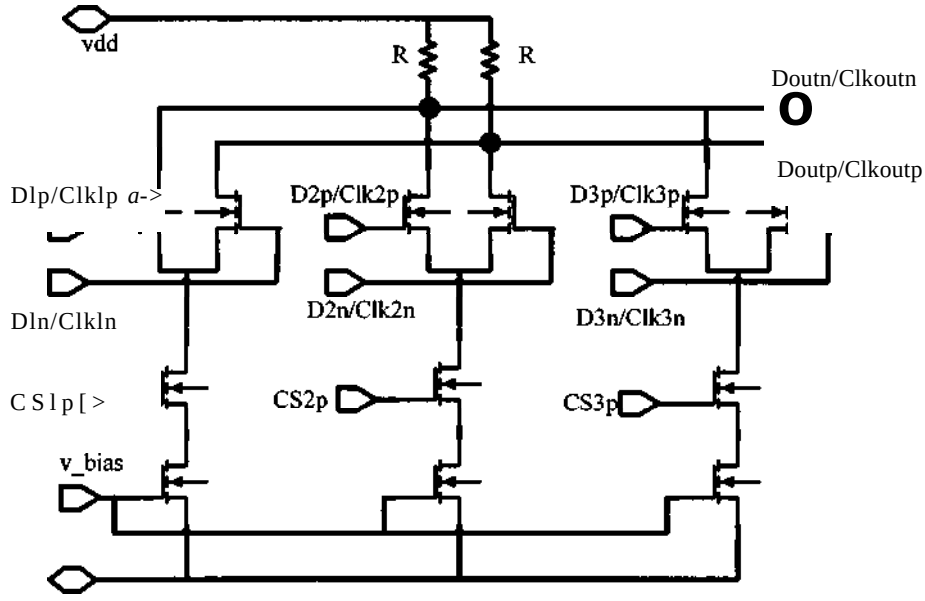


Fig. 6.7 Circuit diagram of CML based MUX

6.2.5 PRBS generator

The input data to the CDR is generated by a pseudo random bit sequence (PRBS) generator. The circuit diagram of the PRBS generator, as shown in Fig. 6.8, consists of flip-flops (DFF) and a XOR gate. All the flip-flops are clocked by buffered version of one of the clock phase signals *(Clkl).* The first DFF has a SET input, and the output of the first DFF is set to logic ' 1' by the CDR reset signal. The outputs of the first and last DFF are applied to the XOR input, initialling the sequence by logic and generating the PRBS data of length 127 (2^7 -1). The PRBS does not generate only one sequence that is 00000000, because if all the DFF outputs are at logic '0' it stays at '0' forever and fails to generate a random bit sequence.

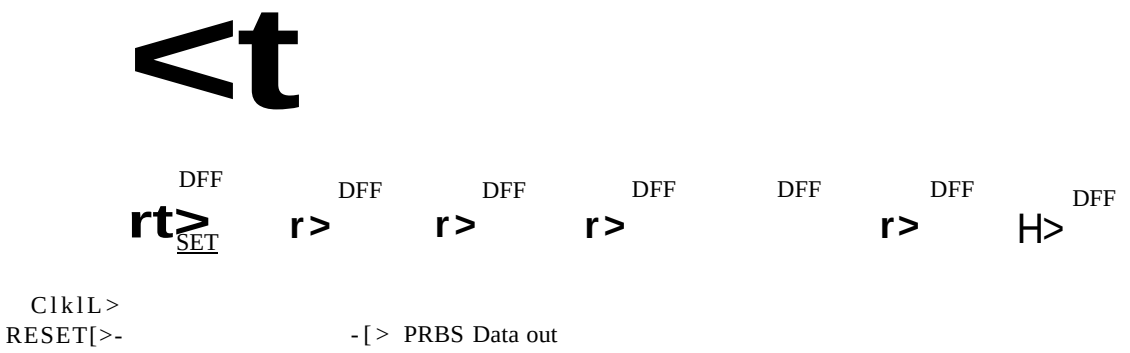


Fig. 6.8 Circuit diagram of the PRBS (2^7-1) generator

6.3 Post-layout simulation results

The complete CDR layout and post-layout simulation are performed in 65nm CMOS with a supply voltage of 1.1V. Fig. 6.9 shows the layout of the CDR, which contains more than 400 transistors. Functionality of the CDR is verified by post layout simulation with the incoming PRBS data of 2^7-1. The simulation results of incoming data, *data,* the extracted clock, *Recov clock,* and the recovered data, Recovdata at 5 Gbps, are presented in Fig 6.10. The CDR consumes 39 mW of power at 5 Gbps, and occupies a core area of 0.013 *mml.* Table 6.2 compares the important CDR performance parameters with recently reported digital CDRs.

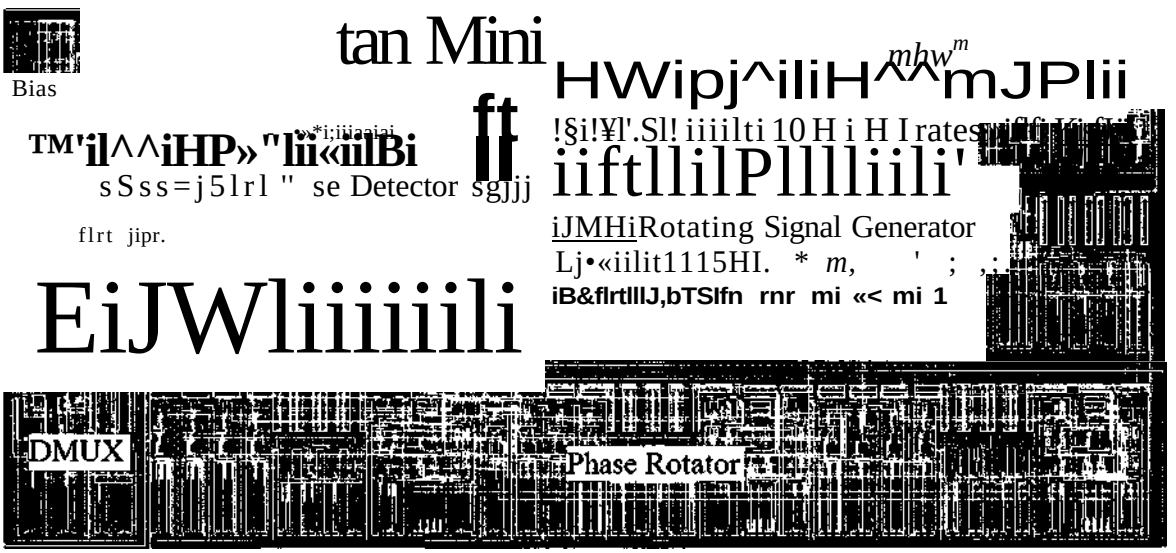


Fig. 6.9 CDR Layout

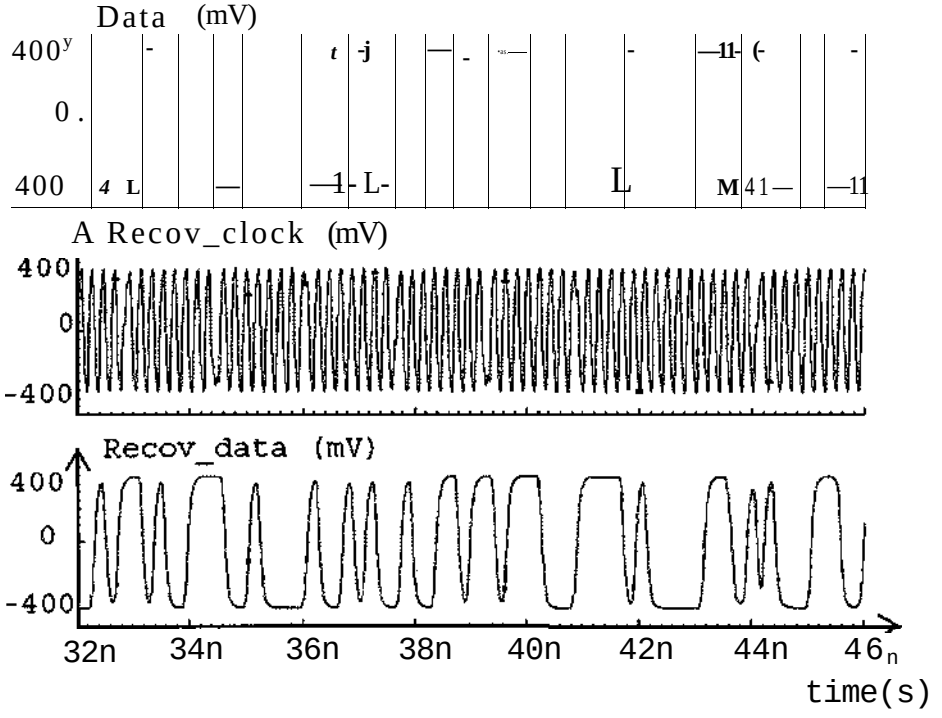


Fig. 6.10 Recovered data and clock

Table 6.2 Comparison of CDR performance parameters

descriptions	[1]	[4]	[18]	[19]	This work
CMOS [nm]	180	90	180	130	65
Supply [V]	1.8	1.2	1.8	1.2	1.1
Data rate robpsi	2.5	2 - 3	2.5, 1.25 0.62,0.15	3.125	2-5
Power [raW]	50	5	70	49	39
Jitter tolerance [UI]	0.7	0.8		0.67	0.67
Acquisition time [bits]	16	1	< 7	40 or 80	8
area *[mm^2]*	0.02	0.017*	0.41*	1.95	0.013*

* Core area

6.4 Critical path analysis of the CDR

The proposed CDR functionality is verified from 2 to 5 Gbps. When the data rate is increased beyond 5 Gbps the CDR starts to fails; this may be due to a single node or multiple node failures. The node at which the signal starts to fail are considered to be a critical node. Along the data path, the node that has the highest capacitive load to drive the following gates is more likely to fail at first. Once a failure occurs along the data path at a front end of the CDR (for example phase detector), it is obvious that the logic does not propagate as expected along the data path to the back end of the CDR (phase rotator). So it is important to find the critical nodes from the front end to the back end of the CDR along the data travelling path. For example when the signal fails at a node in the phase detector, that node has to be fixed. Once it is fixed we need to find the next node at which the signal will fail along the data travelling path.

When the data was increased from 5 to 6.25 Gbps, and the first failure node was identified. The output nodes of the transition detector failed first, and caused the phase detector to fail. A failure is expected at this node at first since the transition signals have a pulse width of *T/2,* this time duration will not be enough to charge the capacitor to the final value. The time constant of the circuit is unchanged but the Baud period is reduced, which increases the ratio between the rise/fall time and the Baud period, and therefore the signal levels are degraded.

Fig. 6.11 shows the waveform of data signal (Dl) and the transition signal (T12) at the failing point. The charging curve of T12 is extended to find the time taken to charge the final value. Time taken to charge to the final value is found as 110 ps, which is close to *2.5*RC.* Since the T12 signal has the pulse width of only 100 ps so the signal level is degraded significantly and cause the following logic circuit to fail.

One of the major benefits of the CML circuit is that the speed can be traded for power to a certain limit. The failure nodes can be fixed by increasing the current through the CML circuit in transition detector block. Increased current allows the resistor value to reduce in order to keep the voltage swing unchanged, thus the time constant *(R.* Q will

also be reduced. In other words rise/fall time is reduced to keep the ratio between the rise/fall time and Baud period unchanged.

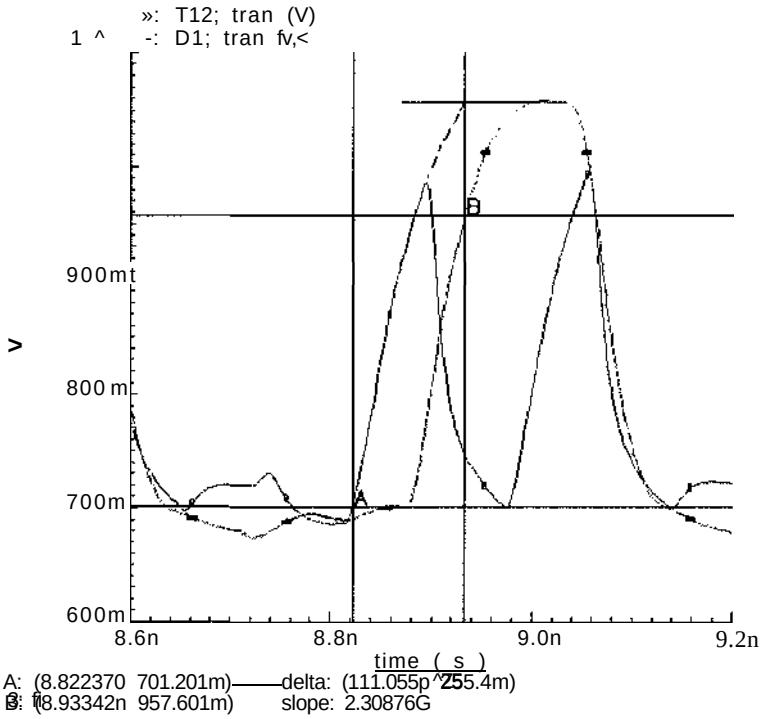


Fig. 6.11 Waveform of Transition signal and data signal at 6.25 Gbps

For example, when the data rate is increased by 25% i.e. data rate increased from 5 Gbps to 6.25 Gbps, therefore the Baud period is now reduced by 20%, from 200 ps to 160 ps. The current through the circuit is increased by 25%, from 0.5 to 0.625 mA. The transistor sizes are also increased by approximately 25% to handle the increased current. The resistance value (R) is reduced by 20% in order to keep the voltage swing *(I.R)* unchanged. Since there is no change in the load, the time constant RC is reduced by 20%, and thus the ratio between the rise/fall time and the Baud period is kept unchanged.

However the result of the above procedure is successful up to a certain increment of current. Referring the variation of f_T against the current of the transistor, as shown in Fig. 6.12, initially it is preferable to operate at point A to keep some room to allow to

trade speed for power. We have some room to increase the current until it reaches the maximum f_T point (B), but after reaching point B the f_T decreases when the current increases. Therefore only a certain amount of current can be increased (up to the point B) in order to increase the speed.

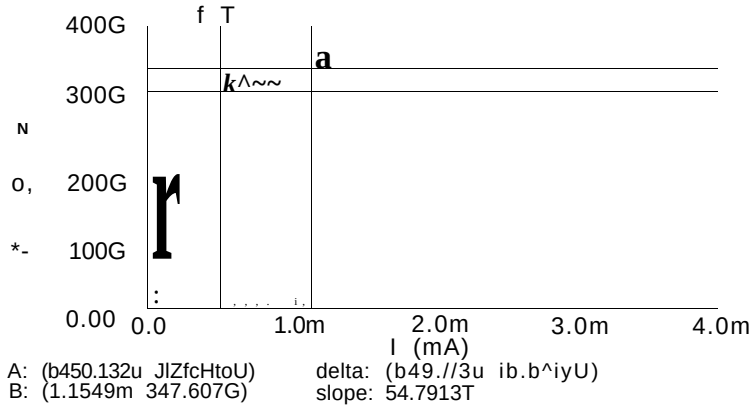


Fig. 6.12 f_T variation against current of a NMOS transistor $\frac{W}{60nm}^{2/Jm}$ in 65 nm CMOS

6.5 CML circuit configuration for speed/power

Some modifications needed to be made to the phase detector CML circuits to increase the current and reduce the resistance to avoid the critical node failures. These modifications can be done by using more than one approach. In the first approach an external signal can be used to turn on some transistors and modify the circuit, which may change the current and resistance according to the requirements. The modification can also be done to the layout by metal fix at higher metal layers that would modify the circuit to increase the current and reduce the effective resistor values. A metal fix may be expensive, however an increase in cost would be acceptable for a larger volume of products.

For example assume that the CDR operates at 5 Gbps. When the CDR is used for an application where the data rate is at 6.25 Gbps, some part (phase detector) of the CDR circuit has to be modified so that the critical node failures can be avoided. In other words, the current has to be increased by 25%, and the effective resistance has to be reduced

by 20% in the circuits that encounter critical node failures.

In the first approach, as shown in Fig. 6.13, the PMOS transistors (Ml, M2) are turned on by the control signal *(Ctl).* Therefore the effective resistance of the differential pair would be *RlRl* • Here the resistance of the PMOS transistor is ignored as it is small compared to i?,. The *Ctl* signal will also turn on the PMOS transistor M3 that will pull extra current through the CML circuit. The number of parallel transistors are now increased by 25% so the current through the differential pair increases by 25 %.

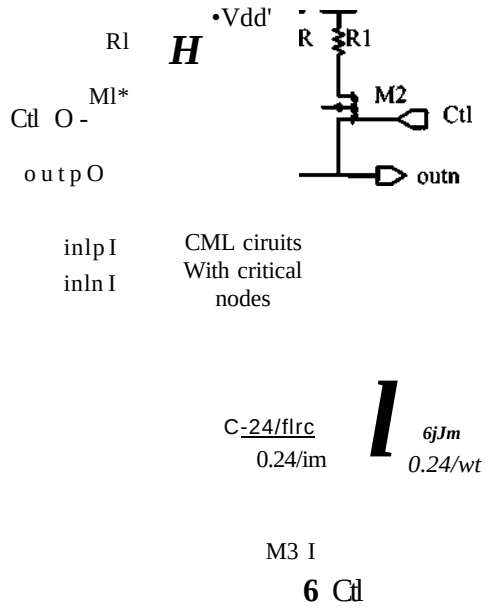


Fig. 6.13 Modification of critical CML circuits using external signal

In the second approach, the idea is the same, but there is no external signal to change current or resistor values. When the original CDR design is performed it has to be pre-planned to adopt a circuit that would make the CDR to operate at a specified higher data rate. So in the second approach the CDR designed to work at 5 Gbps would be configured during design process to be able to avoid critical path failures at 6.25 Gbps. This configuration allows us to make the modifications by using metal fix at the higher

metal layers to change the current and resistance. Fig. 6.14 shows the design configuration of the critical path. The current source transistor M5 is always used, the parallel transistor M6 is also connected same as M5 except its drain is not connected to the bias circuit, therefore the pull down current is only *I* (current drawn by M5 only). But if the connection X is made by the metal fix, M5 and M6 are in parallel, so now M6 also draw current (0.257). The total current through the CML circuit would be 1.25 times of *I.*

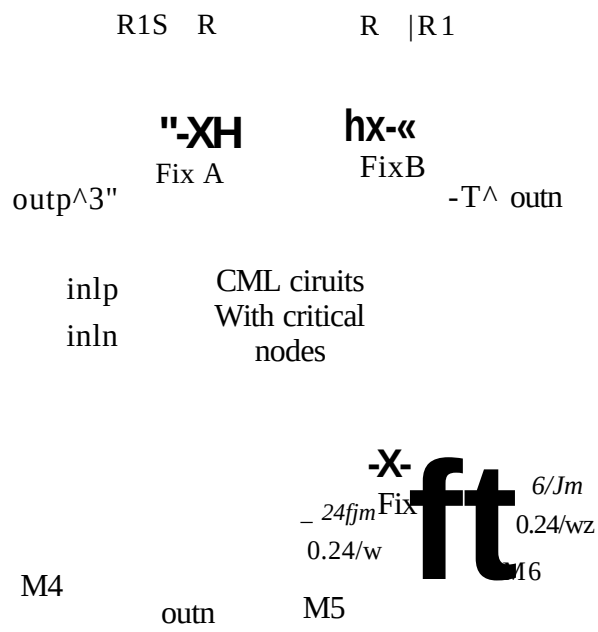


Fig. 6.14 Modification of critical CML circuits using metal fix

The resistors *Rl* are placed parallel to *R* however there is no connection made between them for a normal operation. Connections are made at a higher metal layer (layer 3) where the resistors *Rl* and *R* are parallel. Thus the effective resistance would be *R.\\R*, which is *0.8R.* In both approaches the current is increased by 25%, which makes the new current 1.251, and the resistance is reduced by 20%, which makes the new resistor value 0.8R. Therefore the new time constant would be 0.8RC (20 % reduction), and the rise/fall time would be reduced by 20%. The Baud period is also reduced by 20%, and therefore the ratio between rise/fall time and Baud period is kept constant.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this paper a 2-5 Gbps fully differential 3X oversampling CDR is proposed for use in burst-mode serial data link applications. First the jitter tolerance and acquisition time of the proposed CDR are derived analytically and the theoretical values are presented. The jitter tolerance the CDR is evaluated by an event-driven simulation model developed in Matlab. The simulation result shows a close match to the theoretical values. The proposed CDR has a high frequency jitter tolerance of 0.67 UI and an acquisition time of an 8 Baud period. It was also found that event-driven simulation is 30 times faster than conventional simulation with a fixed-time-step of 1/100 of Baud period.

The most of the existing digital CDR are capable of working up to 3.25 Gbps and consumes more power, and also they have longer acquisition time that is not suitable for burst mode application. A fully differential architecture is employed in the proposed CDR design in order to achieve the speed up to 5 Gbps. Complete design flow for the CML based CDR is executed in 65 nm CMOS process technology. The functionality of the CDR is verified by post-layout simulation with 2^7 -1 PRBS data. The CDR consumes 39 mW of power from 1.1 V supply at 5 Gbps and occupies a core area of 0.013 *mrrd*. The proposed CDR performance parameters are compared with reported digital CDRs. The CDR has low power consumption than other reported CDR except the

CDR reported in [4], which can operate up to 3 Gbps. The acquisition time of the proposed CDR is shorter and satisfies the requirements for the GPON applications. Since the CDR is capable of operating at any data rate from 2 - 5 Gbps, it may be used for different data rate applications and even different standards such as OC-48, OC 192. The following contributions have been made during this research studies,

- This work is an extension to the work done by Carleton's VLSI group [4]. There are a number of areas in which the CDR in this thesis is significantly different from the previous work. This CDR uses CML based fully differential architecture while previous work used standard CMOS technique, which allows to increase the speed up to 5 Gbps. Different circuit techniques are introduced, especially the rotating signal generator uses completely new technique to do phase rotation once in every 8 Baud period.
- Even though the reference [4] reported that event-driven simulation is used to obtain the jitter tolerance of the CDR no detail information was reported. First time, the details of event-driven modelling of an oversampling CDR is reported in this research work. As a result the following conference paper has been successfully published in IEEE conference proceedings,

  [33] Nathan Kiddinapillai, Tad Kwasniewski, "Jitter tolerance estimation of 3X Oversampling CDR using event-driven simulation", 2nd Microsystems and Nano-electronics conference (MNRC), pp. 136 - 139, Oct. 2009.

- Design flow of a CML based 3X oversampling CDS is executed in modern CMOS process technology (65nm). The layout of the CDR, which contains more than 400 transistors, is successfully completed and the functionality of the CDR is verified by post layout simulations. The CDR is able to operate to at a higher data rate compared to the existing reported CDR, and consumes lower power than the reported CDR at data rate of 5 Gbps. The CDR has shorter acquisition time than most of reported CDR.

This work has been has been successfully published in following IEEE conference proceedings,

[34] Nathan Kiddinapillai, Tad Kwasniewski, "A 2-5 Gb/s Fully Differential 3X Oversampling CDR for High-Speed Serial Data Link", Accepted for publication in Proceeding, 27th IEEE NORCHIP conference, Trondheim, Norway, Nov. 2009.

7.2 Potential for future work

This work can be further developed in a number of directions. Some suggestions for future works are listed below:

- continue with the fabrication of the CDR followed by test and measurements.
- explore some techniques to increase the operating speed of the CDR, passive or active inductors could be considered to increase the bandwidth
- explore some techniques that would switch off the CDR when it is idling when the CDR is targeted for burst mode applications to save some energy.

References

[1] 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 J. of Solid State Circuits, Vol. 39, No.4, pp. 613-621, April 2004.

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

[3] R. Stephens, "Analysing jitter at high data rates", IEEE Communication magazine, vol. 42, pp. S6-10, Feb. 2004.

[4] Q. Du , "A CDR with digital threshold decision technique and a cyclic reference injected DLL frequency multiplier with a period error compensation loop", Ph.D. Thesis, Carleton University, Canada, 2008.

[5] J. Savoj, B. Razavi, "High-speed CMOS Circuits for optical receivers", New York, Kluwer2001.

[6] B. Razavi, "Challenges in the design high-speed clock and data recovery circuits", IEEE Communication magazine, Volume 40, Issue 8, pp. 94 - 101, Aug. 2002.

[7] J. Cao, M. Green, A. Momtaz, K. Vakilian, D. Chung, J. Keh-Chee, M. Caresosa, X. Wang, T. Wee-Guan, C. Yijun, L. Fujimori, A. Hairapetian, "OC-192 transmitter and receiver in standard 0.18-um CMOS", IEEE J. of Solid-State Circuits, Volume 37, Issue 12, pp. 1768 - 1780, Dec. 2002.

[8] J. Savoj, B. Razavi, "A 10-Gb/s CMOS clock and data recovery circuit with a half-rate linear phase detector", IEEE J. of Solid-State Circuits, Volume 36, Issue 5, pp. 761 - 768, May 2001.

[9] J. Savoj, B. Razavi, "A 10-Gb/s CMOS clock and data recovery circuit with a half-rate binary phase/frequency detector", IEEE J. of Solid-State Circuits, Volume 38, Issue l,pp. 13-21, Jan. 2003.

[10] Rong-Jyi Yang, Shang-Ping Chen, Shen-Iuan Liu, "A 3.125-Gb/s clock and data recovery circuit for the 10-Gbase-LX4 Ethernet", IEEE J. of Solid-State Circuits, Volume 39, Issue 8, pp. 1356 - 1360, Aug. 2004.

[11] "G. 984-2 Gigabit-capable Passive Optical Networks (GPON)", Physical media dependent (PMD) layer specification, ITU-T, March 2003.

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

[13] S.-J. Jou, C.-H. Lin, " Design and analysis of digital data recovery circuit using oversampling", IET J. of Circuits, Devices and Systems, Volume 1, Issue 1, pp. 95-101, Feb. 2007.

[14] Pavan Kumar Hanumolu, Min Gyu Kim, Gu-Yeon Wei, and Un-ku Moon, "A 1.6 Gbps Digital Clock and Data Recovery Circuit", IEEE 2006 Custom Integrated Circuits Conference, pp. 603-606, Sept. 2006.

[15] J. Rogers, C. Plett, "Integrated circuit design for high-speed frequency synthesis", Artech House, Boston, USA, 2006.

[16] M. Van Ierssel, A. Sheikholeslami, H. Tamura, W.W. Walker, "A 3.2 Gb/s CDR Using Semi-Blind Oversampling to Achieve High Jitter Tolerance", IEEE J. of Solid-State Circuits, Volume 42, Issue 10, pp. 2224 - 2234, Oct. 2007.

[17] Jaeha Kim, Deog-Kyoon Jeong; "Multi-gigabit clock and data recovery using blind oversampling", IEEE Magazine of Communications, Volume 41, Issue 12, pp. 68-74, Dec. 2003.

[18] C. F. Liang, S. C. Hwu, "A Multi-band burst-mode Clock and Data Recovery circuit," *Trans.*, IEICE Electron, Vol. E90, pp. 802-810, April 2007.

[19] Bong-Joon Lee, Moon-Sang Hwang, Jaeha Kim, Deog-Kyoon Jeong, Wonchan Kim, "A quad 3.125 Gbps transceiver cell with all-digital data recovery circuits", IEEE Symposium of VLSI Circuits, pp. 384-387, June 2005.

[20] P. Heydari, R. Mohanavelu, "Design of ultrahigh-speed low-voltage CMOS CML buffers and latches", IEEE Trans, on VLSI Systems, Volume 12, Issue 10, pp. 1081-1093, Oct. 2004.

[21] D. Rennie, M. Sachdev, "A 5-Gb/s CDR Circuit with automatically calibrated linear phase detector", IEEE Trans, on Circuits and Systems, Volume 55, Issue 3, pp. 796 - 803, April 2008.

[22] "Understanding Jitter and Wander Measurements and Standards", 2nd edition, Agilent technologies available online at http://cp.literature.agilent.com/litweb/pdf/5988-6254EN.pdf.

[23] A. Kuo, R. Rosales, T. Farahmand, S. Tabatabaei, A. Ivanov, "Crosstalk bounded uncorrelated jitter (BUJ) for high-speed interconnects", IEEE Trans. On Instrumentaion and Measurements, Volume 54, Issue 5, pp. 1800-1810, Oct. 2005.

[24] "Jitter on PLL", Altera Corp., available at http://www.altera.com/common/print/prt-print.isp?page=/support/devices/ pllclock/ jitter/pll-jitter.

[25] B Razavi, "Monolithic PLL and CDR", IEEE press, New York, 1996;

[26] C. Pelard, E. Gebara, A.J. Kim, M.G. Vrazel, F. Bien, Y. Hur, Moonkyun Maeng, S. Chandramouli, C. Chun, S. Bajekal, S.E. Ralph, B. Schmukler, V.M. Hietala, J. Laskar, " Realization of multigigabit channel equalization and crosstalk cancellation integrated circuits", IEEE Journal of Sold-State Circuits, Vol. 39, Issue 10, pp. 1659 - 1670, Oct. 2004.

[27] J. Zhuang, Q. Du, T. Kwasniewski, "Event-Driven Modeling and Simulation of an Digital PLL", IEEE International Workshop in Behavioral Modelling and Simulation, pp 14-15, Sept. 2006.

[28] M. V. Ierssel, H. Yamaguchi, A. Sheikholeslami, H.Tamura, W.W. Walker, "Event-driven modelling of CDR jitter induced by power-supply noise, finite decision-circuit bandwidth, and channel ISI", IEEE Transl. on Circuits and systems, Vol. 55, pp. 1306-1315, June. 2008.

[29] Yehui Sun, Hui Wang, " Analysis of digital bang-bang clock and data recovery for multi-gigabit/s serial transceivers", IEEE 2009 Custom Integrated Circuits Conference (CICC), pp. 343-346, Sept. 2009.

[30] H. Ming-ta, G. Sobelman, "Architectures for multi-gigabit wire-linked clock and data recovery," IEEE Magazine on Circuits and systems, Vol. 8, pp. 45-57, 2008.

[31] A. P. Chandrakasan, R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits", IEEE Proceedings, Volume 83, Issue 4, pp. 498 - 523, April 1995.

[32] J. M. Musicer, J. Rabaey, "MMOS current mode logic for low power, low noise CORDIC computation in mixed-signal environments", Proceedings of the International Symposium on Low Power Electronics and Design, pp. 102 - 107, 2000.

[33] Nathan Kiddinapillai, Tad Kwasniewski, "Jitter tolerance estimation of 3X Oversampling CDR using event-driven simulation", 2nd Microsystems and Nano-electronics conference (MNRC), pp. 136 - 139, Oct. 2009.

[34] Nathan Kiddinapillai, Tad Kwasniewski, "A 2-5 Gb/s Fully Differential 3X Oversampling CDR for High-Speed Serial Data Link", Accepted for publication in Proceeding, 27th IEEE NORCHIP conference, Trondheim, Norway, Nov. 2009.

Appendix

A.l Circuit Diagrams

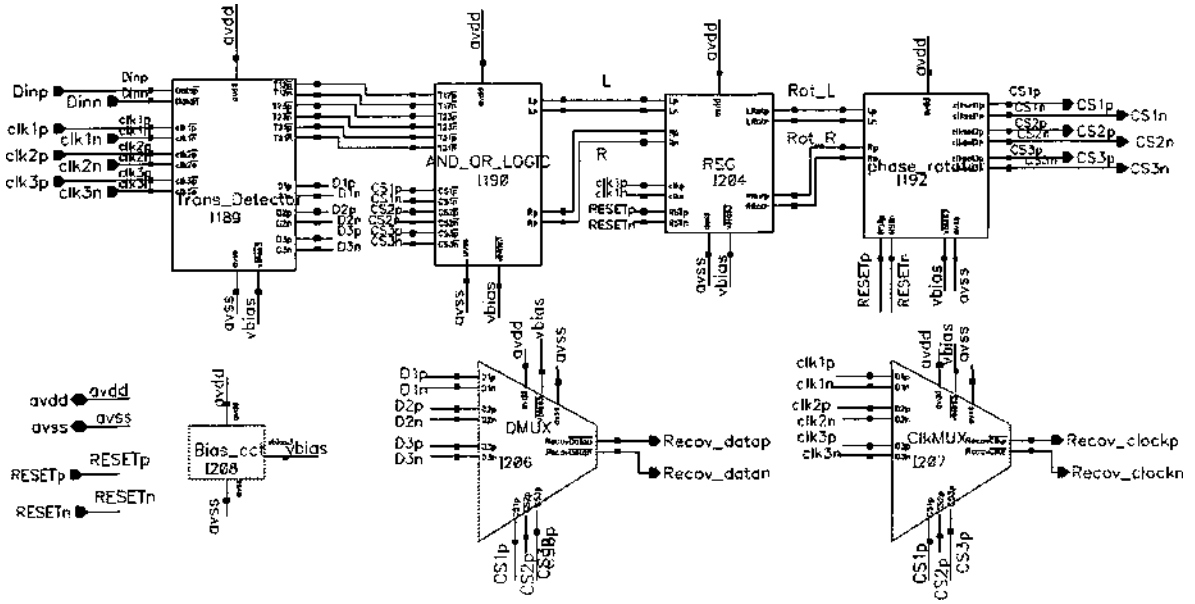


Fig. Al Top level diagram of proposed CDR

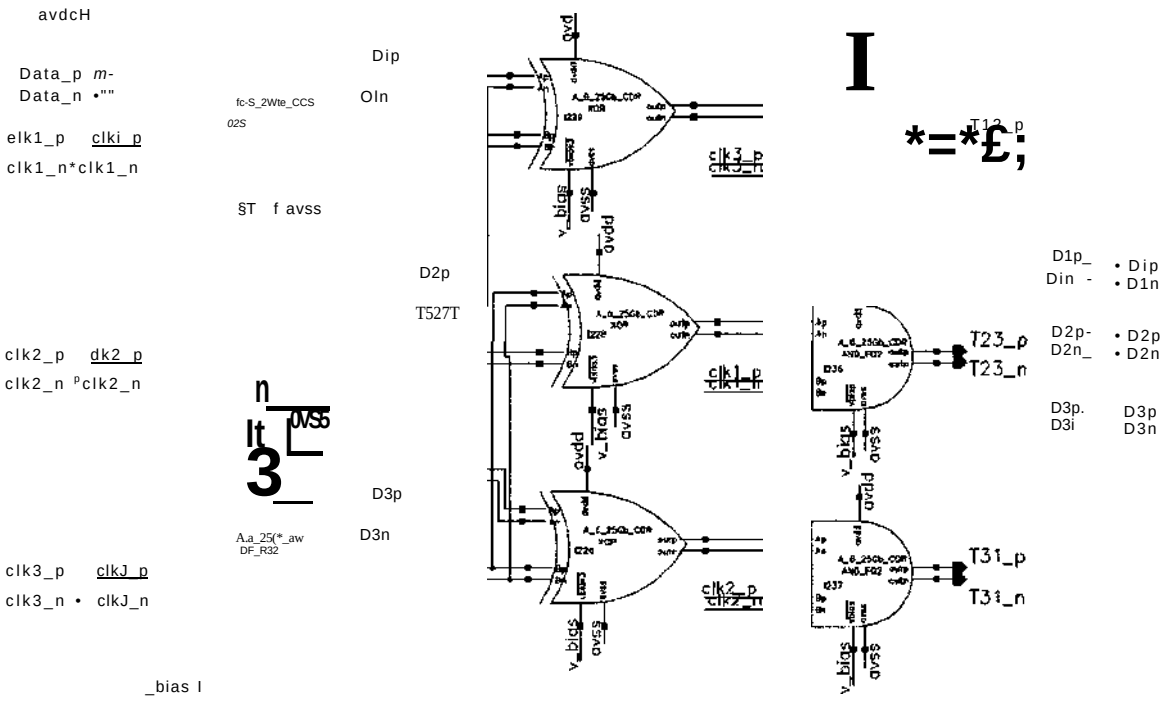


Fig. A2 Circuit diagram of the transition detector

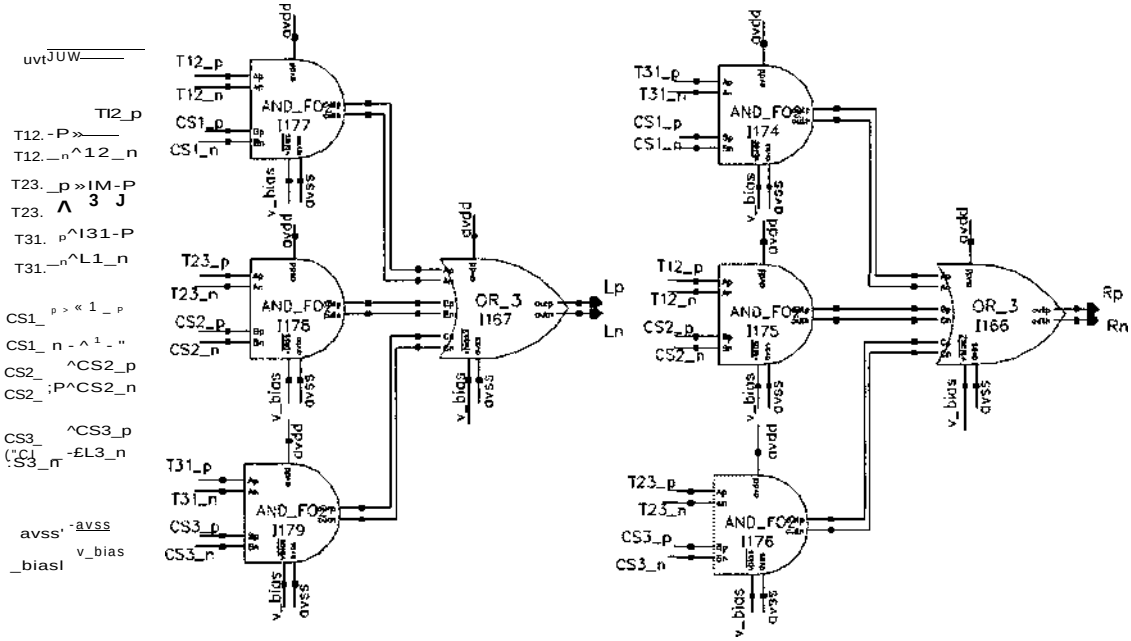


Fig. A3 Circuit diagram of the L-R generating logic

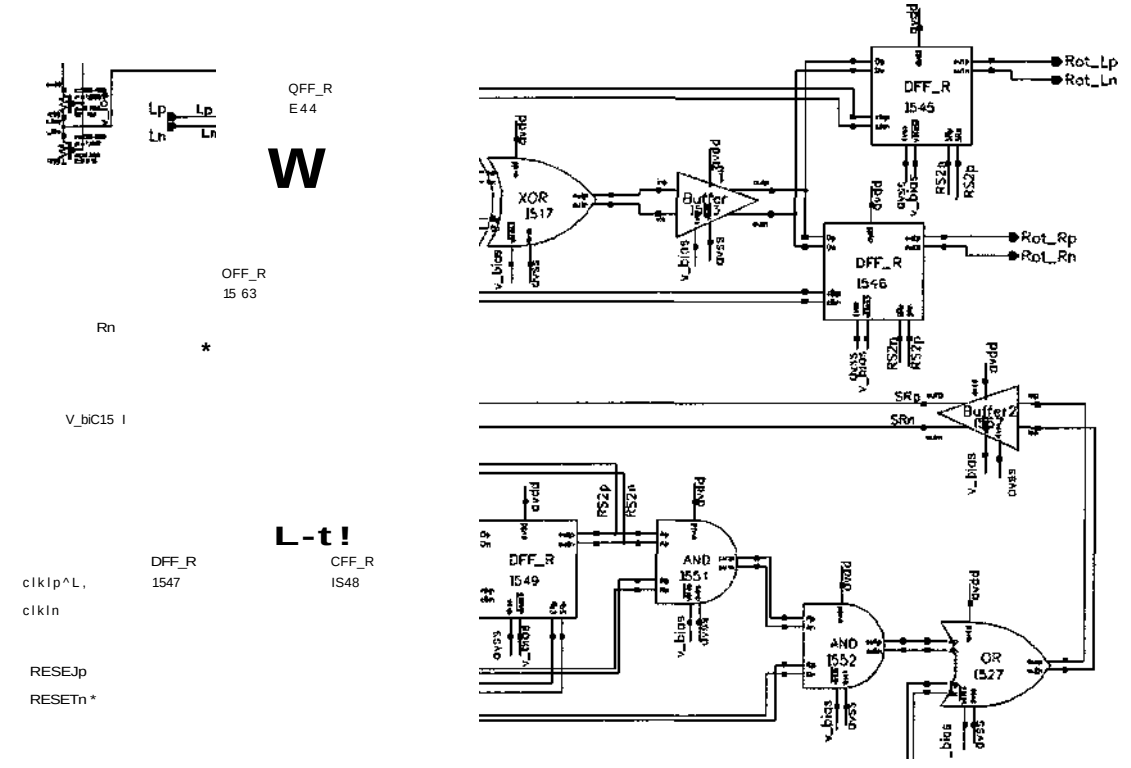


Fig. A4 Circuit diagram of the rotating signal generator

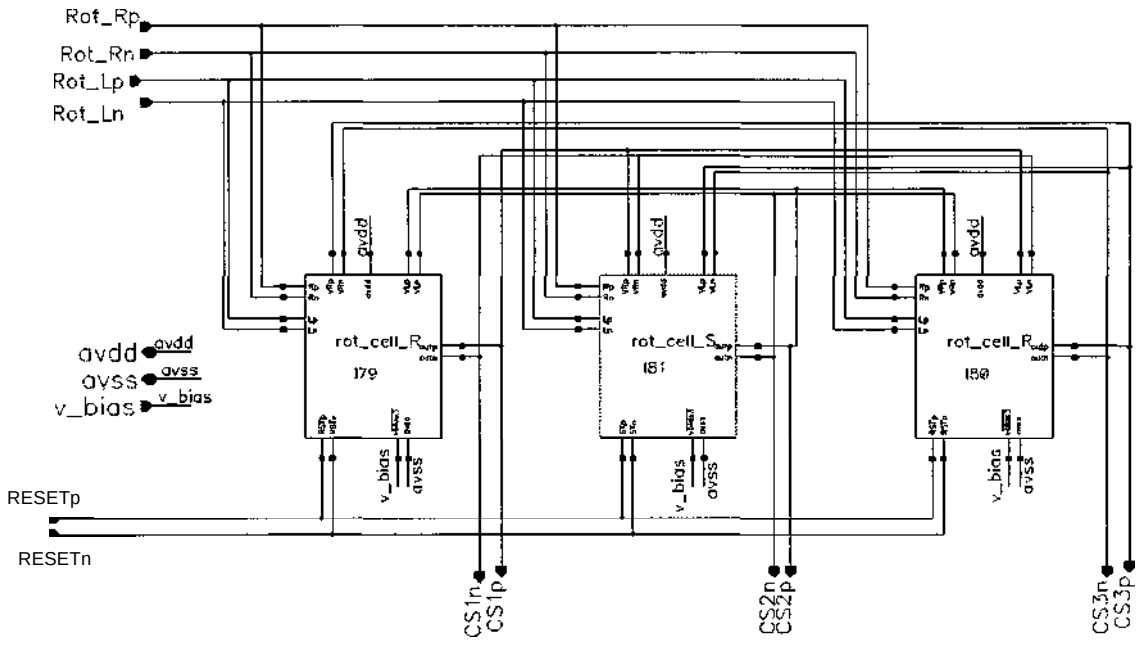


Fig. A5 Circuit diagram of the phase rotator

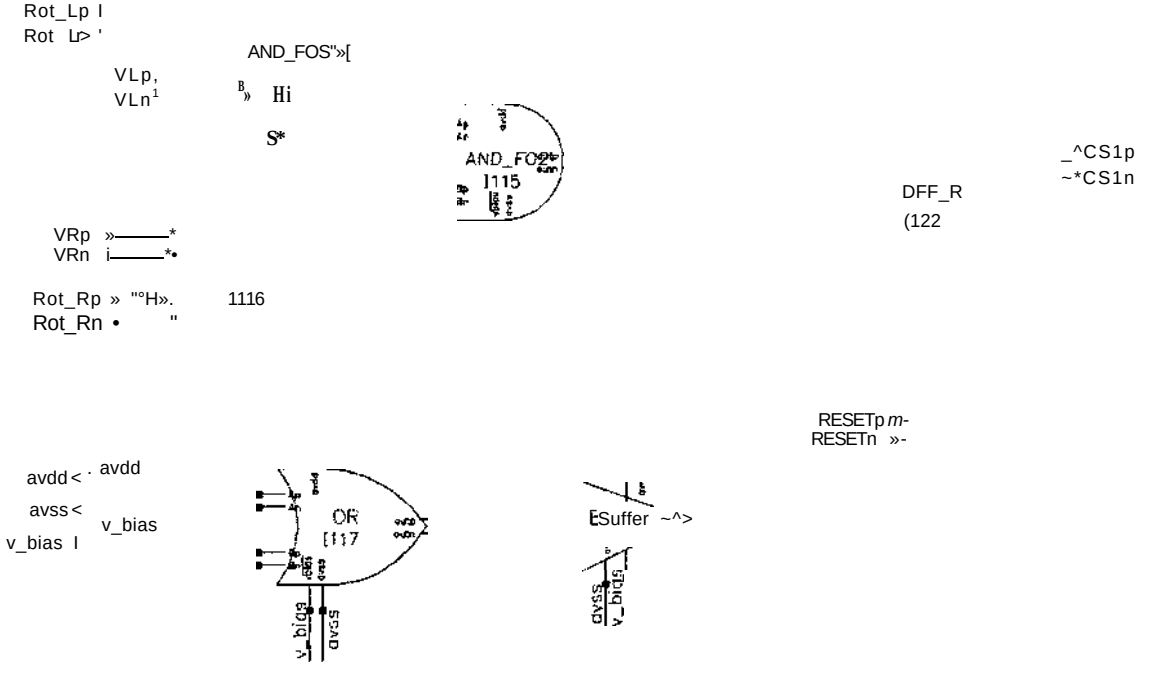


Fig. A6 Circuit diagram of the phase rotating cell

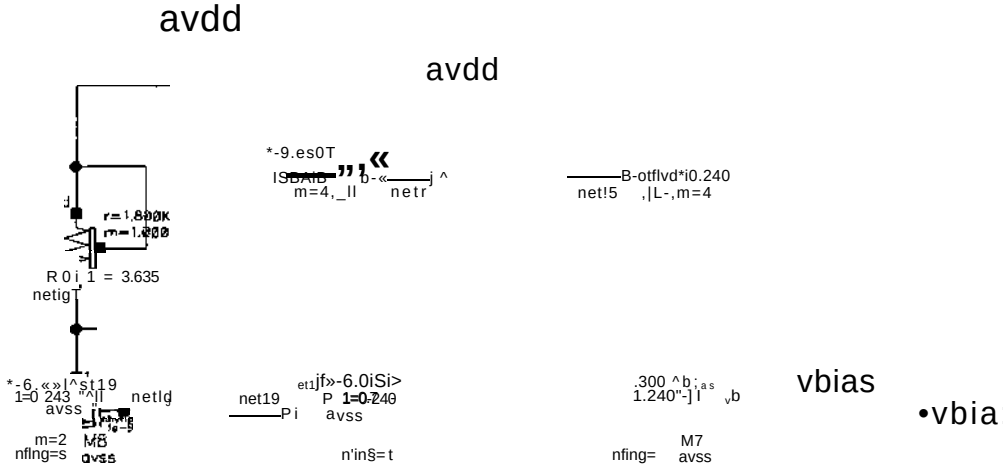


Fig. A7 Circuit diagram of bias circuit

A.2 Simulation waveforms at 5 Gbps and 6.25 Gaps

Fig. A8 Waveforms at 5 Gbps

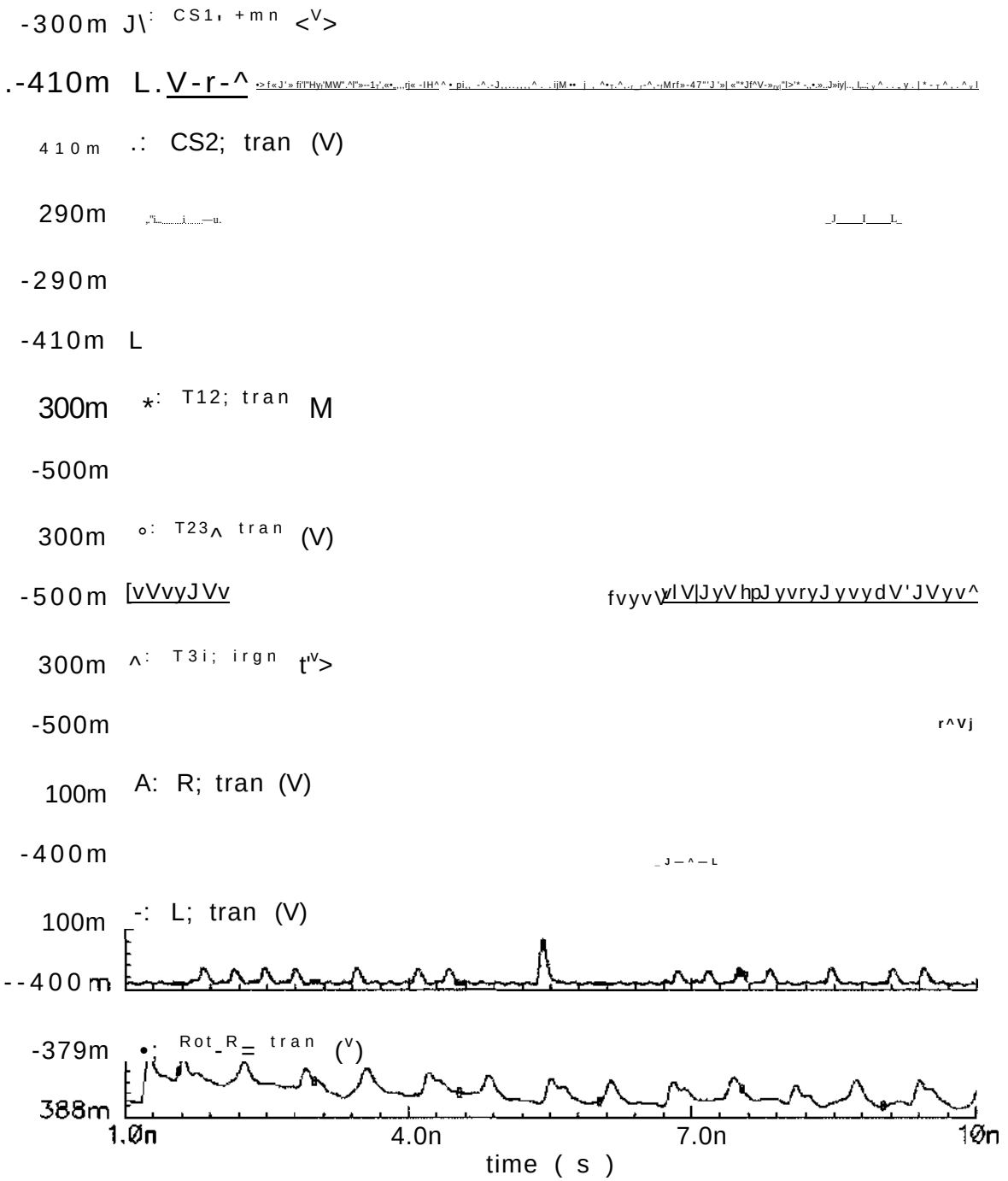


Fig. A9 Waveforms at 6.25 Gbps