Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-MS-2017-01

M.Sc. Thesis

Source-Synchronous Interface with All-Digital Data Recovery

Shizhao Zhang B.Sc.

Abstract

This thesis proposes a low-cost high-efficiency source-synchronous interface for high-speed inter-chip communication. The interface is composed of LVDS transceivers as external I/O buffers, and an all-digital data recovery, which can calibrate the received data phase to be aligned to the 90° phase of the received half-rate reference clock, for error free data sampling. The proposed data recovery adopts a full-digital scheme, which uses time-to-digital converters (TDC) as phase acquisition, a digitally-controlled delay line (DCDL) to calibrate the phase, and a finite-state machine (FSM) as the control unit. Reference clock generated from phase-locked loops (PLL) or delay-locked loops (DLL) is not needed for the proposed data recovery. The interface is implemented in UMC 65 nm Low-leakage technology, with circuits designed at both transistor-level and RTL-level. The post-layout simulation shows the proposed interface works properly with data rates from 412.4Mbps to 1.25Gbps in all process corners. The total layout area is 688 µm × 87 µm, and the total power consumption is 16.74 mW.

TU Delft

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Source-Synchronous Interface with All-Digital Data Recovery

A Low-cost Efficient Design

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Shizhao Zhang B.Sc.
born in Changchun, China

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Copyright © 2017 Circuits and Systems Group
All rights reserved.

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Source-Synchronous Interface with All-Digital Data Recovery"** by **Shizhao Zhang B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 28, 2017

Advisor:

dr.ir. Rene van Leuken

Committee Members:

dr. Amir Zjajo

dr.ir. Stephan Wong

Abstract

This thesis proposes a low-cost high-efficiency source-synchronous interface for high-speed inter-chip communication. The interface is composed of LVDS transceivers as external I/O buffers, and an all-digital data recovery, which can calibrate the received data phase to be aligned to the 90° phase of the received half-rate reference clock, for error free data sampling. The proposed data recovery adopts a full-digital scheme, which uses time-to-digital converters (TDC) as phase acquisition, a digitally-controlled delay line (DCDL) to calibrate the phase, and a finite-state machine (FSM) as the control unit. Reference clock generated from phase-locked loops (PLL) or delay-locked loops (DLL) is not needed for the proposed data recovery. The interface is implemented in UMC 65 nm Low-leakage technology, with circuits designed at both transistor-level and RTL-level. The post-layout simulation shows the proposed interface works properly with data rates from 412.4Mbps to 1.25Gbps in all process corners. The total layout area is 688 µm × 87 µm, and the total power consumption is 16.74 mW.

Acknowledgments

I would like to express my gratitude toward my parents who have supported me with my study in the Netherlands for two years, as well as all my previous study in the past years.

I would like to thank my advisor dr.ir. Rene van Leuken, and dr. Amir Zjajo for their assistance during my thesis project, as well as all other people in Circuits and System group and Computer Engineering group who have provided me with the resources for carrying out my work.

Shizhao Zhang B.Sc.
Delft, The Netherlands
August 28, 2017

Contents

Abstract v

Acknowledgments vii

1 Background and Introduction 1
- 1.1 Overview on Modern High Speed Links . . . 1
- 1.2 Differential Signaling . . . 2
- 1.3 Overview of Clock and Data Recovery Architectures . . . 4
  - 1.3.1 Conventional CDR Architectures . . . 5
  - 1.3.2 CDR for Source-Synchronous Links . . . 6
- 1.4 Motivation and Thesis Overview . . . 7

2 All-Digital Data Recovery 9
- 2.1 Proposed Architecture . . . 9
- 2.2 Digitally-Controlled Delay Line . . . 10
  - 2.2.1 DCDL Based on Tri-state Inverter . . . 11
  - 2.2.2 The Design of the Inverter and Tri-state Inverter . . . 12
  - 2.2.3 DCDL Layout and Simulation . . . 12
- 2.3 Time-to-Digital Converter . . . 14
  - 2.3.1 TDC based on D-latch Chain . . . 15
  - 2.3.2 Double-edge-triggered D Flip-flop . . . 16
  - 2.3.3 D-latch Chain . . . 16
  - 2.3.4 Single-complementary Conversion and Buffering . . . 18
  - 2.3.5 TDC Layout and Simulation . . . 19
  - 2.3.6 TDC Measurement Range . . . 19
- 2.4 Finite-State Machine . . . 22
  - 2.4.1 From Algorithm to RTL . . . 22
  - 2.4.2 From RTL to Layout . . . 23
- 2.5 Implementation of the Data Recovery Architecture . . . 23
- 2.6 Mathematical Modeling of the Data Recovery Architecture . . . 26
  - 2.6.1 Operating Range . . . 26
  - 2.6.2 Convergence . . . 27
  - 2.6.3 Fine Tuning the DCDL resolution . . . 29
  - 2.6.4 Phase Error . . . 31
- 2.7 Simulation of the Data Recovery . . . 32
- 2.8 Summary of the Data Recovery . . . 32

3 LVDS Transceiver 37
- 3.1 LVDS Driver . . . 37
  - 3.1.1 H-bridge Based Driver . . . 37
  - 3.1.2 Pre-driver . . . 39
  - 3.1.3 LVDS Driver Layout . . . 39

3.1.4 LVDS Driver Simulation . . . 39
3.2 LVDS Receiver . . . 43
3.2.1 Hysteresis Comparator Based Receiver . . . 43
3.2.2 LVDS Receiver Simulation . . . 43

4 Top-level Design and Simulation **49**
4.1 Top-level Design . . . 49
4.2 Top-level Simulation . . . 50

5 Conclusion and Future Work **55**
5.1 Conclusion . . . 55
5.2 Future Work . . . 56
5.3 Comparison with State-of-the-art . . . 56

A Mixed-signal Design Flow **63**
A.1 Schematic-driven Design Flow . . . 63
A.2 Netlist-driven Design Flow . . . 64

B Verilog RTL Code of the FSM **65**

C Script for Synthesizing the FSM **69**

D Script for Placing & Routing the FSM **71**

E Bit Sequence Comparison **75**

F Source-Synchronous Interface Application Note **77**
F.1 Overview . . . 77
F.2 Physical Information . . . 77
F.3 Port Description . . . 78
F.4 Configuration . . . 78
F.5 Data Recovery . . . 79
F.6 Port Input Capacitance . . . 80
F.7 Constraints . . . 80
F.8 AC Characteristics . . . 81
F.9 DC Characteristics . . . 81

List of Figures

1.1 Data rate per pin trend . . . 1
1.2 Block diagram of a typical HSL . . . 2
1.3 LVDS transceiver configuration [25] . . . 3
1.4 Signal "high" and "low" time comparison between single-ended and differential signals in presence of unequal signal rise and fall times . . . 4
1.5 Clock recovery and data retiming of a CDR circuit . . . 4
1.6 Sampling clock transitions should be aligned to the center of the data eye 5
1.7 Block diagram of the PLL based CDR . . . 5
1.8 Block diagram of the DLL based CDR . . . 6

2.1 Block diagram of the proposed data recovery architecture . . . 10
2.2 Circuit diagram of the tri-state inverter based DCDL . . . 11
2.3 Layout of 1, 2 and 4 time sized inverters . . . 13
2.4 Circuit diagram of the tri-state inverter . . . 13
2.5 Layout of the tri-state inverter . . . 13
2.6 Layout of DCDL . . . 14
2.7 Circuit diagram of a basic delay line based TDC . . . 15
2.8 Circuit diagram of the designed TDC . . . 15
2.9 Circuit diagram of DEDFF . . . 16
2.10 Layout of DEDFF . . . 17
2.11 Circuit diagram of D-latch . . . 17
2.12 Circuit diagram of the single-to-complementary converter . . . 18
2.13 Buffering the "start" and "stop" signals . . . 19
2.14 Layout of TDC . . . 19
2.15 TDC post-layout simulation results (The waveforms from the top row to the bottom row are START, STOP, TDC output thermometer code, TDC finish) . . . 20
2.16 Timing diagram of the edge detector . . . 21
2.17 TDC output vs. input START"-STOP" transition interval . . . 21
2.18 FSM state transition diagram . . . 22
2.19 Block diagram of the DLSET calculation logic in the FSM . . . 23
2.20 FSM placed & routed in *Encounter* . . . 24
2.21 Final FSM layout in *Virtuoso* . . . 24
2.22 Circuit diagram of the data recovery . . . 25
2.23 Layout of the data recovery (all metal layers are hidden, except vias and pins) (157 µm × 63 µm) . . . 25
2.24 Timing diagram of the edge detector when $T_{d-ck} < T_{min}$ . . . 26
2.25 N_e and new relative calibration error vs. relative calibration error . . . 30
2.26 Circuit diagram of the DCDL with dummy inverters . . . 31
2.27 Simulation results of the data recovery @ T_{bit}=1000 ps, TT corner . . . 34
2.28 Simulation results of the data recovery @ T_{bit}=560 , SS corner . . . 34
2.29 Simulation results of the data recovery @ T_{bit}=2373 ps, FF corner . . . 35

2.30 Moving the data phase to 90° of the reference clock @ T_{bit}=1000 ps, TT corner . . . 35
2.31 Moving the data phase to 90° of the reference clock @ T_{bit}=600 ps, SS corner . . . 36
2.32 Moving the data phase to 90° of the reference clock @ T_{bit}=2300 ps, FF corner . . . 36

3.1 Circuit diagram of the LVDS driver . . . 38
3.2 Circuit diagram of the LVDS pre-driver . . . 40
3.3 Layout of the LVDS driver . . . 40
3.4 Test bench for the LVDS driver . . . 41
3.5 Transient simulation results of the LVDS driver @1Gbps . . . 41
3.6 Eye diagram at the input of the FPGA receiver . . . 42
3.7 Eye diagram at the output of the FPGA receiver . . . 42
3.8 Circuit diagram of the LVDS receiver . . . 44
3.9 Layout of the LVDS receiver . . . 45
3.10 Layout of the LVDS receiver biasing . . . 45
3.11 Circuit diagram of the LVDS receiver test bench . . . 46
3.12 Transient simulation results of the LVDS receiver in all process corners @1Gbps . . . 46
3.13 Duty cycle of the receiver output waveform in all process corners @1Gbps 47
3.14 Eye diagram of the LVDS receiver output in all process corners @1Gbps 47

4.1 Block diagram of the top-level interface . . . 49
4.2 Top-level layout (688 µm × 87 µm) (all metal layers are hidden, execept vias and pins) . . . 51
4.3 Eye diagram of recovered data with reference clock @1Gbps, TT corner 52
4.4 Eye diagram of recovered data with reference clock @1.25Gbps, SS corner 52
4.5 Eye diagram of recovered data with reference clock @421.4Mbps, FF corner 52
4.6 Eye diagram at FPGA LVDS receiver input and output @1Gbps, TT corner . . . 53
4.7 Eye diagram at FPGA LVDS receiver input and output @1.25Gbps, SS corner . . . 53
4.8 Eye diagram at FPGA LVDS receiver input and output @421.4Mbps, FF corner . . . 53

5.1 Block diagram of the interface with multiple receiver channels . . . 58

A.1 Design flow of this work . . . 63

E.1 Comparison of transmitted and received bit sequence . . . 76

F.1 Physical dimension information of SSI . . . 77
F.2 Block diagram of the SSI configuration . . . 78
F.3 Timing diagram of data recovery . . . 79
F.4 Reference voltage vs. reference current of LVDS driver . . . 82
F.5 Output current vs. reference current of the LVDS driver @R_{out}=110 Ω . 82

F.6 Output common-mode voltage vs. reference current of the LVDS driver @R_{out}=110 Ω 83
F.7 Output current vs. output differential voltage of the LVDS driver @Iref=300 µA 83
F.8 Output common-mode mode voltage vs. output differential voltage of the LVDS driver @Iref=300 µA 84
F.9 Total output resistance vs. reference current of the LVDS driver @Iout=3.5 mA 84
F.10 Reference voltage vs. reference current of LVDS receiver 85

List of Tables

2.1 Summary of DCDL resolution 12
2.2 Summary of TDC resolution 19
2.3 Summary of TDC measurement range in all process corners 21
2.4 Summary of the operating range of the data recovery in all process corners 27
2.5 Summary of the resolution of TDC and DCDL, resolution mismatch and relative resolution mismatch 31
2.6 Area of the designed data recovery 33
2.7 Power consumption of the data recovery @ 1Gbps 33

3.1 LVDS signaling specification of Xilinx Virtex-6 and Virtex-7 FPGAs [28] [29] 37
3.2 Comparison of different LVDS driver topologies 39
3.3 LVDS driver output specifications and power consumption 43
3.4 Power consumption of the LVDS receiver in all process corners 47

4.1 Area summary of the source-synchronous interface 50

5.1 Comparison with state-of-the-art 57

F.1 Port description 78
F.2 Port input capacitance 80
F.3 Constraints 80
F.4 AC Characteristics 81

1 Background and Introduction

1.1 Overview on Modern High Speed Links

The ever-increasing functionality of modern electronic systems has been requiring a considerable amount of chip-to-chip communication. To keep up with the increasing bandwidth demand, per-pin data rate of commercial chip-to-chip interconnect standards has also been growing rapidly in recent years, as shown in Figure 1.1. Besides, according to the International Technology Roadmap for Semiconductors [2], the I/O data rate will continue to increase 20% per year in the future. Because of this, the efficiency of traditional parallel data bus is limited by [4]:

- High IC package pin-count;
- Simultaneous switching noise;

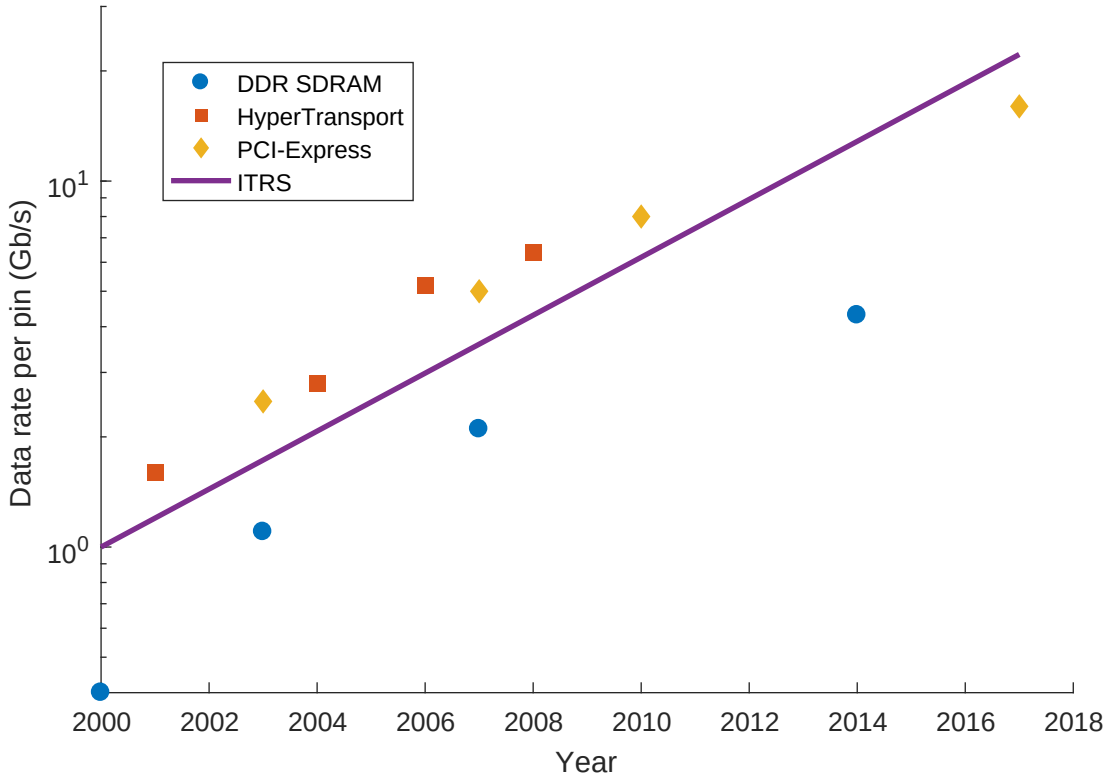

Figure 1.1: Data rate per pin trend

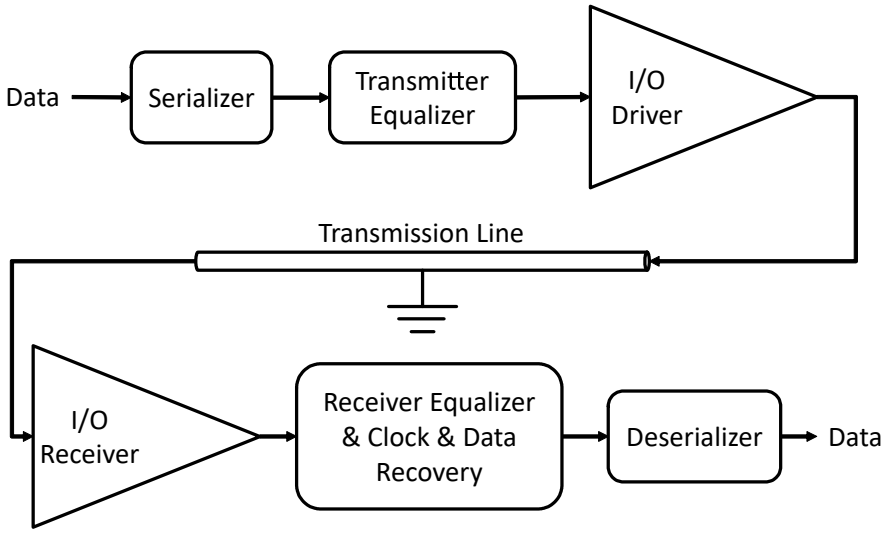

Figure 1.2: Block diagram of a typical HSL

- Electro-magnetic interference (EMI);
- High power consumption;
- Complex external connection.

Therefore, modern high-speed links (HSL) adopt a highly serial scheme to reduce the number of external connections to avoid such problems. A typical HSL block diagram is shown in Figure 1.2.

Serializer and deserializer (SerDes) are common functional blocks in HSLs where the conversion between parallel data and serial data takes place. In an ideal baseband digital transmission system, the signals contain an infinite amount of frequency components. However, real transmission lines always have a certain frequency response, which can significantly distort the transmitted signals. Equalizers can be designed to have a reverse frequency response to the transmission line to recover the waveform as much as possible [20].

1.2 Differential Signaling

There are various I/O signaling standards, most of which can be classified into single-ended and differential signaling. Single-ended signaling is common in low-speed short-range applications because of the low-cost I/O circuits. However, single-ended signals are susceptible to noise. This can be resolved by increasing the signal swing, but at the cost of higher power consumption and EMI generation, which is especially impractical at high bit rates. Consequently, differential signaling prevails in modern HSL.

Low-Voltage Differential Signaling (LVDS) is a widely used signaling standard for low-power, high-speed and low-noise data transmission [6]. A typical LVDS transceiver configuration is shown in Figure 1.3. The driver generates a constant 3.5 mA output

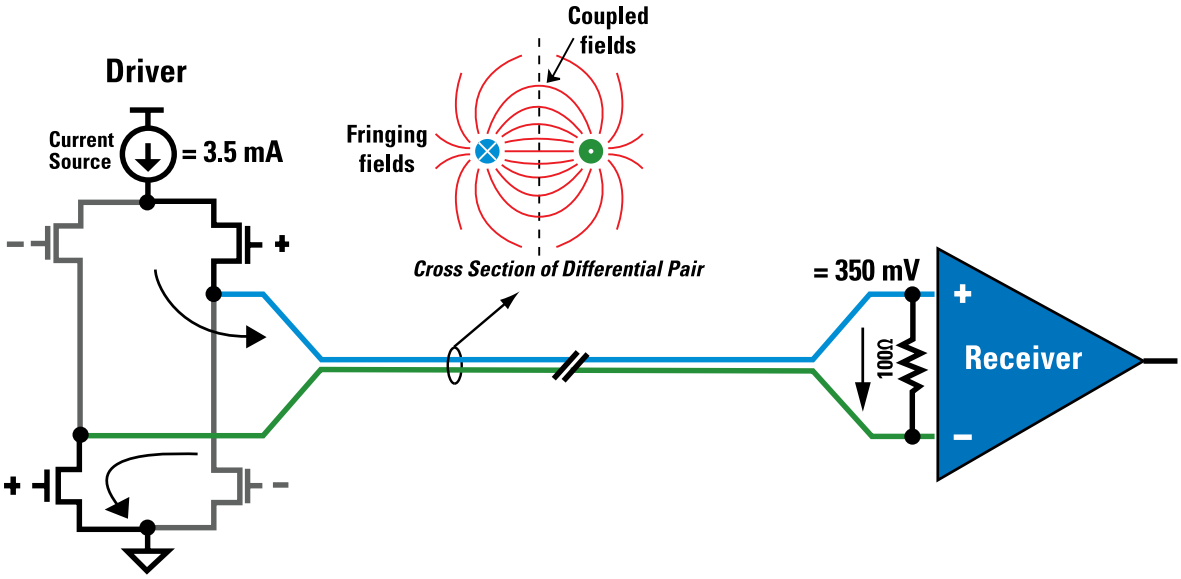

Figure 1.3: LVDS transceiver configuration [25]

current with switchable current direction. The current flows through 100 Ω termination resistor at the receiver input, and generates ±350 mV input voltage according to the current direction.

Like other differential signaling standards, LVDS is resistant to common mode noise injected into the transmission channel. As the receiver senses the difference between the two signals in the differential pair, the common mode noise is removed from the output signal. This feature enables differential signaling standards, such as LVDS, to use a small signal swing while maintaining a high signal integrity.

The coupled differential pair of transmission channel also generates less noise. Since the current in the differential pair always changes in the opposite direction, the generated electro-magnetic field is also in the opposite direction, which tends to cancel each other. Additional, the small signal swing of LVDS also reduces noise generation compared to other large signal swing standards.

Differential signals are less sensitive to unequal signal rise and fall times. In single-ended signaling, unequal signal rise and fall times cause duty-cycle distortion, which is particularly undesirable in double data rate (DDR) transmission. This is because the threshold voltage between "high" and "low" signals is usually at the middle of the signal swing, while for differential signals, the threshold voltage is the "0" differential voltage, i.e. the crossover point of the two signals. As a result, the effect of unequal rise and fall times is canceled.

Single-ended I/O drivers based on push-pull structures inject a lot of noise into the power supply due to the parasitic resistance and inductance presented in the power connections. It is also known as the "simultaneous switching outputs", "simultaneous switching noise", or "ground bounce" effect [23]. Due to the inherent constant current property of LVDS, the AC switching current is small and is a result of the non-ideal current source.

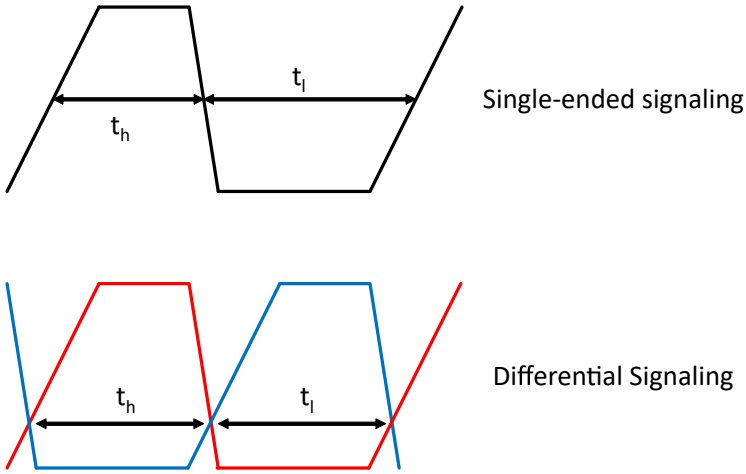

Figure 1.4: Signal "high" and "low" time comparison between single-ended and differential signals in presence of unequal signal rise and fall times

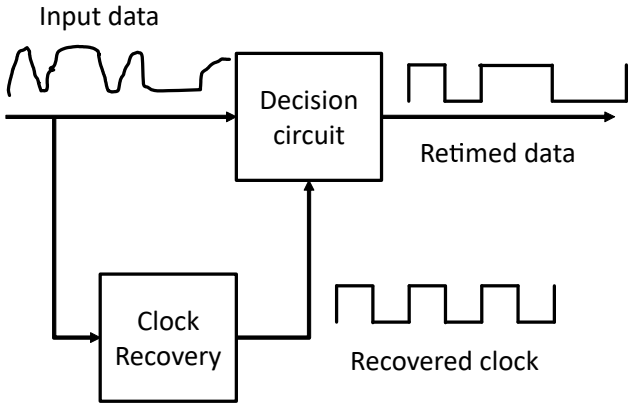

Figure 1.5: Clock recovery and data retiming of a CDR circuit

1.3 Overview of Clock and Data Recovery Architectures

Due to many non-idealities of the external transmission channel, such as the IC packaging parasitic RLC, lossy transmission line, inter-symbol-interference (ISI), cross talk, and EMI noise, the received data in an HSL can be severely distorted and exhibits significant jitter and skew. Clock and data recovery (CDR) is a critical functional block in modern HSLs, which can extract the data sequence and clock information from the received data signal. Figure 1.5 shows the basic principle of a CDR circuit [22]. The clock recovery part is used to generate a clock signal that has the same or half frequency as the data rate, and must have a certain phase relationship to the data signal. The decision circuit samples the data signal using the recovered the clock, and can be a D flip-flop. In order to leave sufficient timing margin for the jitter presented in the data, the optimum sampling point is chosen to be at the center of the data eye as shown in Figure 1.6.

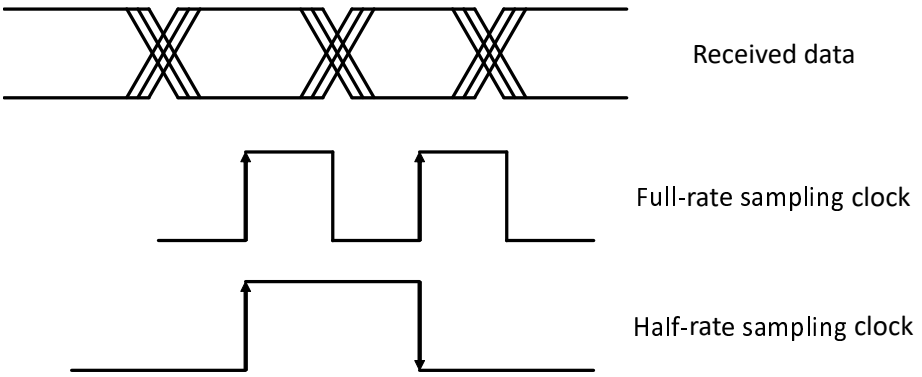

Figure 1.6: Sampling clock transitions should be aligned to the center of the data eye

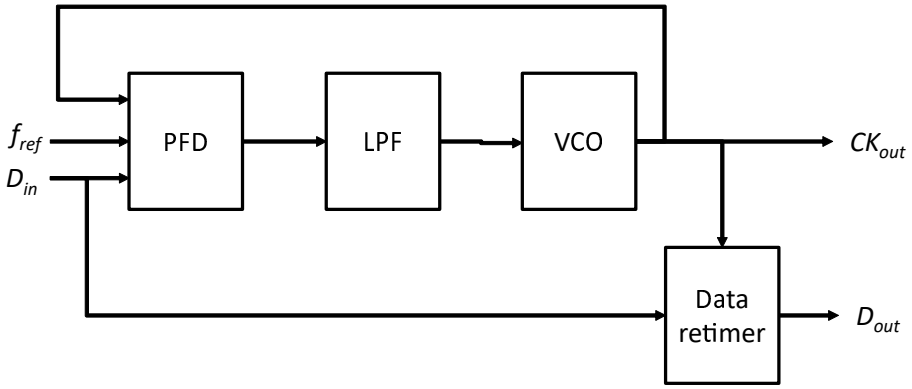

Figure 1.7: Block diagram of the PLL based CDR

1.3.1 Conventional CDR Architectures

Depending on whether an accompanying clock is transmitted with the data, communication links can be mainly divided into two categories, i.e. source-asynchronous and source-synchronous. In source-asynchronous links such as optical communication, Ethernet and SATA, the data is transmitted without an accompanying clock. Therefore, a proper clock signal must be generated to sample the data, and most CDRs in this type of interface employ some form of phase locked loops (PLL). Figure 1.7 shows the block diagram of a PLL based CDR [24].

The operation of a PLL based CDR is described as follows. The input data D_{in} is phase-compared with the generated clock signal CK_{out} by the phase and frequency detector (PFD), and the comparison result signal is filtered by the following low-pass filter (LPF) as the control voltage signal for the voltage-controlled oscillator (VCO). Whenever the PFD detects a phase mismatch between CK_{in} and D_{in}, it generates up or down signal for the LPF which changes the control voltage of the VCO. The VCO then speeds up or slows down its output oscillation frequency to move its phase closer to the phase of the data. If the phase mismatch between CK_{out} and D_{in} is sufficiently small, the PLL based CDR locks.

In a source-synchronous link, an accompanying clock is transmitted with the data

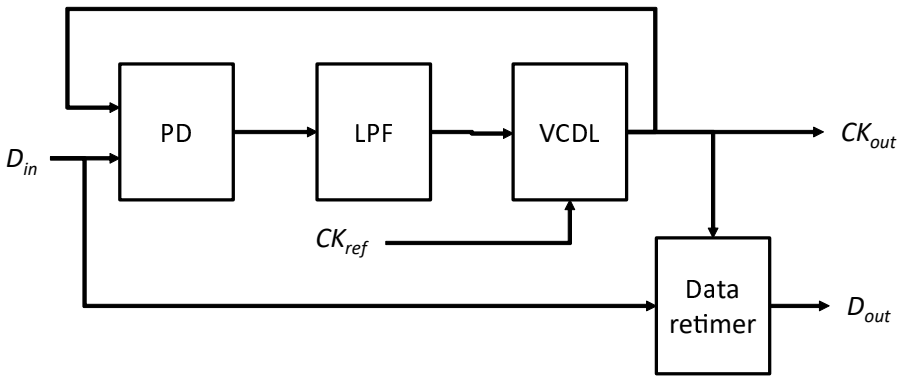

Figure 1.8: Block diagram of the DLL based CDR

which has the same frequency as the data. Most chip-to-chip communication uses source-synchronous links, for example, Intel Quick Path Interconnect (QPI), AMD Hyper-transport (HT) and PCI Express. CDRs in this type of link only need a certain range of phase tracking capability, as the clock frequency information is explicitly provided. As a result, delay locked loop (DLL) based CDRs are more popular in source-synchronous interfaces because of the improved jitter performance, improved stability, and reduce area compared to PLL based CDRs [24]. Figure 1.8 shows the block diagram of a DLL based CDR.

The operation of a DLL based CDR is similar to that of a PLL based CDR, except that the frequency acquisition part is omitted. Instead of generating a clock by the VCO, the DLL-based CDR delays the input reference clock CK_{ref} by the voltage-controlled delay line (VCDL) to match the phase of the data. Since the VCDL has limited delay range, the DLL based CDR only has a certain phase capturing range and no frequency tracking capability. As a result, DLL based CDRs are mostly used in source-synchronous links, where the reference clock has the same frequency as the data.

1.3.2 CDR for Source-Synchronous Links

Various CDR architectures for source-synchronous links have been proposed and studied including the conventional PLL/DLL based CDR [17], phase interpolator (PI) based CDR [11] and oversampling based CDR [12], as well as injection-locked oscillator (ILO) base CDR which has become popular in recent years [8].

The DLL based CDR [17] uses a coarse and fine DLL structure to achieve both wide operating range and high accuracy. The coarse DLL is implemented with time-to-digital converter and digitally-controlled delay line (DCDL) to achieve fast long-range phase acquisition, and the fine DLL is a conventional analog DLL for accurate locking. This DLL based CDR has the advantage of fast locking and wide operating range, as well as the high accuracy as the conventional analog DLL based CDR. However, it requires a global PLL or DLL to generate quadrature phase clock, and combined structure takes a large area.

The PI based CDR [11] uses PI to generate the sampling clock. The PI generates

an output clock phase by taking the weighted average of two quadrature clock signals, and the weight information is stored in a digital format. The reported CDR uses less analog components, and has smaller area. However, as the DLL based CDR, it also needs a global PLL or DLL to generate the quadrature phase clock. The oversampling based CDR [12] also uses PI to generate the sampling clock, but with a different phase detection scheme, which is to oversample the data for multiple bits. This reported CDR uses an open-loop delay line and PIs to generate the quadrature phase clock, which has smaller area and power consumption, but the operating speed is limited at 10Gbps.

1.4 Motivation and Thesis Overview

The motivation of this thesis is to design an source-synchronous interface for chip-to-chip communication in cost-constraint chips, and to explore possibilities for low-area and energy-efficient clock and data recovery. The target bit rate is 1Gbps. As discussed in previous sections, the fundamental principle for flip-flops to sample the data correctly is that the sampling clock edges must be located at the center of the data eye as shown in Figure 1.6. In source-synchronous links, a half-rate reference clock is much more common than using the full-rate clock, because of the utilization efficiency of the transmission channel bandwidth. Therefore, one of the goal of the thesis is to find a way to have a clock and data signal that is 90° phase shifted.

Using the CDR architectures discussed in the previous sections is a possible solution. However, the large silicon area and high power consumption due to the inherent properties of analog PLL and DLL is not suitable for cost-constraint chips. Additionally, most conventional CDR architectures need a global PLL or DLL to generate quadrature phase clock, which also takes large area. This thesis proposes a novel data recovery scheme by delaying the received data to match the phase of the received half-rate reference clock, instead of adjusting the clock phase as in conventional CDR, which eliminates the need of any global/local PLL/DLL. The proposed data recovery adopts a fully digital scheme, with time-to-digital converters (TDC) in phase acquisition, digitally-controlled delay line (DCDL) in delay generation and finite-state machine (FSM) as control unit. The proposed data recovery takes much smaller silicon area and consumes much less power compared to the conventional CDR, but effectively solves the data sampling problem in a source-synchronous interface operating at around 1Gbps, which is specially suited for cost-constraint chips.

Besides the data recovery, this thesis also designs LVDS transceivers as high-speed external I/O buffers. The rest of the thesis is organized as follows:

- Design of the all-digital data recovery (Chapter 2);
- Design of the LVDS transceiver (Chapter 3);
- Integration and verification of the entire source-synchronous interface (Chapter 4);
- Conclusion and future work (Chapter 5)
- Mixed-signal design flow (Appendix A);

- Verilog HDL code and TCL scripts with explanation (Appendix B, Appendix C, Appendix D);
- Comparison of transmitted and received bit sequence (Appendix E);
- Application note for the designed source-synchronous interface (Appendix F).

2 All-Digital Data Recovery

In a real source-synchronous link, the phase relationship between the received data and clock is unknown due to a lot of non-deterministic factors such as transmission channel mismatch, clock buffering, and PVT variations. The use of the data recovery is intended to adjust the unknown phased relationship to 90°, which is an optimum phase shift in DDR communication.

In this Chapter, the proposed and implemented all-digital data recovery will be discussed in detail. The data recovery is a mixed-signal block, where both transistor-level and RTL-level design methodologies are used. The transistor-level part is designed in *Cadence Virtuoso Design System* through the "schematic-driven flow" process, and the RTL-level part is designed in Verilog HDL, synthesized in *Synopsys Design Compiler*, and placed & routed in *Cadence Encounter Digital Implementation* through the "netlist-driven flow" [19].

All the circuits are implemented in UMC 65 nm Low-Leakage technology, using 1.2 V thin-gate-oxide devices with 1.2 V operating voltage. Verification through DRC, LVS and parasitic extraction are performed in *Mentor Graphics Calibre*. The simulations are performed in *Virtuoso Analog Design Environment with Spectre simulator*, and the results are plotted and analyzed in *Virtuoso Visualization & Analysis*.

2.1 Proposed Architecture

The proposed data recovery consists of three key components: digitally-controlled delay line (DCDL), time-to-digital converter (TDC), and finite state machine (FSM). Figure 2.1 shows the block diagram of the proposed data recovery block.

The data recovery works as follows. The inputs to the data recovery are a data stream and a half-rate reference clock coming from the input I/O buffers. The input data stream first goes through the DCDL, and will be delayed for a certain amount of time depending on the delay line settings. The delayed data signal then goes to the TDC. The clock signal will normally go through the clock buffer tree, and then reach the TDC.

The TDC measures the time interval between a transition on the data signal and the immediate transition on the clock signal, as well as the time interval between two clock transitions. The measured results are represented in a digital format, and are then sent to the FSM for further processing.

The FSM will calculate the current actual phase difference between the data and the clock signal using the results from the TDC, and then adjust the DCDL settings if the phase difference is not 90°. The FSM can re-run the process for multiple iterations, until the 90° phase difference is reached. Then, the clock transitions are located at the center of the data eye, and the data can be sampled correctly by the following D

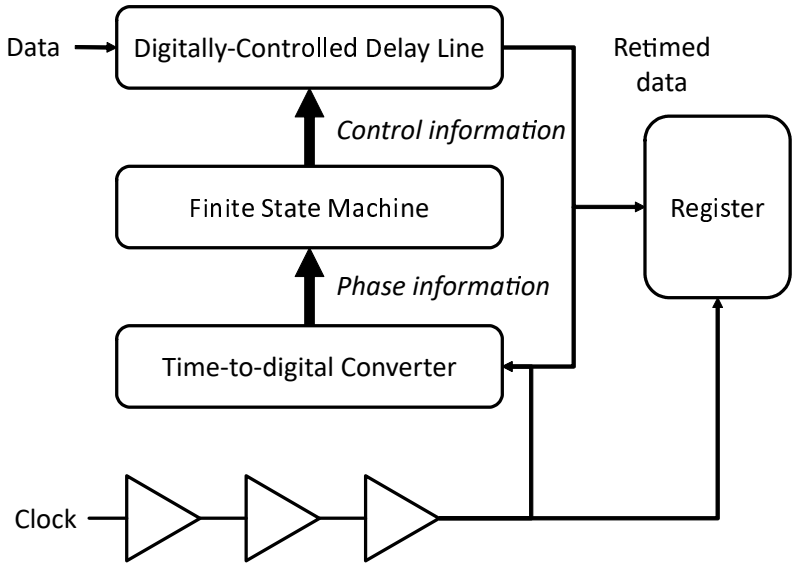

Figure 2.1: Block diagram of the proposed data recovery architecture

flip-flop.

The proposed data recovery architecture requires that the resolution of the DCDL and the TDC should be close to each other for efficient FSM operation. Since the target data speed for the data recovery is around 1Gbps, a resolution of around 100 ps is chosen for both TDC and DCDL. The detailed relationship between the resolution of the TDC and DCDL will be discussed in later sections.

2.2 Digitally-Controlled Delay Line

A delay line is a functional block that delays the input signal for a certain amount of time. Based on how the control information is presented, there are two kinds of delay lines: one is voltage-controlled delay lines (VCDL), where the delay control signal is represented in an analog from, i.e. voltage; the other is digitally-controlled delay lines (DCDL), where the delay control signal is represented in a digital binary form. Both VCDL and DCDL are actively used in delay-locked loops (DLL) and clock and data recovery (CDR), depending on different situations and requirements. For the proposed data recovery architecture, a DCDL is used as the whole block is highly digital.

There are various architectures for digitally-controlled delay line, which can be mainly classified into tapped-delay line architecture and single-output delay line architecture [3]. The tapped-delay line architectures are usually composed of a delay line with a large multiplexer selecting different delay tap stages. The disadvantage of this architecture is that the multiplexer adds up to a significant offset to the delay output, which is undesirable. Besides, the large multiplexer can also result in low linearity of the delay output.

Single-output delay line architecture varies the delay from the delay line input to output by changing the internal signal propagation speed or path. By changing the

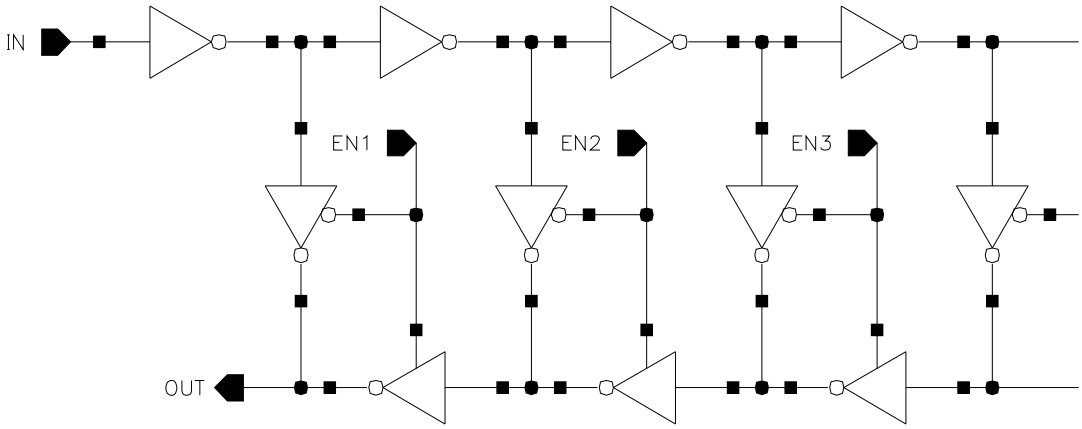

Figure 2.2: Circuit diagram of the tri-state inverter based DCDL

signal propagation speed, common techniques include using variable load capacitor at the output of each delay element, and using variable charging and discharging current for the delay element. Delay lines of these architectures can usually produce sub-gate delay resolution.

The function of the DCDL in the proposed data recovery is to generate a certain delay to the data signal under the control of the FSM with a resolution of around 100 ps. A DCDL architecture of changing the signal propagation path is used, because of the advantage of low offset, high linearity, and simple control.

2.2.1 DCDL Based on Tri-state Inverter

Figure 2.2 show the circuit diagram of the DCDL used in the proposed data recovery architecture, which is composed of only inverters and tri-state inverters. The control signal formed by EN1, EN2 ... and EN31 is a thermometer code, and controls the propagation of each tri-state inverter. According to the thermometer control code, the DCDL varies its delay from input IN to output OUT by including and excluding specific inverters and tri-state inverters in the signal propagation path.

We denote the delay of one inverter and one tri-state inverter as T_{inv} and T_{tinv} respectively, and the total delay from IN to OUT is represented as

$$D = T_{DCDL}(N + 1) \tag{2.1}$$

$$T_{DCDL} = T_{inv} + T_{tinv} \tag{2.2}$$

where N is the decimal equivalent thermometer control code, and T_{DCDL} is the resolution of the DCDL. Since all the inverters in the DCDL are identical, and so as the tri-state inverters, the exact value of T_{inv} and T_{tinv} are only determined by their output load capacitance. From the circuit diagram, it is obvious that the output load capacitance of all the inverters are equal, and so as the tri-state inverters. We can write

them as

$$C_{Linv} = 2C_i \tag{2.3}$$

$$C_{Linvt} = C_i + C_{otinv} \tag{2.4}$$

where C_i is the input capacitance of the inverter and tri-state inverter assuming they are equal, and C_{otinv} is the tri-state output capacitance of the tri-state inverter.

As a conclusion, the designed DCDL has a high linearity, and the delay offset is one delay resolution if the DCDL output is connected by an inverter or tri-state inverter the same as in the DCDL. Besides, the range of the DCDL delay range can be easily extended by cascading more stages. To satisfy the requirements of the proposed data recovery, a total number of 31 stages is used, which produces 32 different delay outputs including the 0 setting.

2.2.2 The Design of the Inverter and Tri-state Inverter

The designed DCDL is expected to provide a certain delay output irrespective to the input signal transition direction, thus the low-high and high-low propagation delay should be as close to each other as possible for both inverters and tri-state inverters. Therefore, a ring oscillator is made to test the optimum ratio between the PMOS and the NMOS transistors for the inverter and tri-state inverter, and the result shows that a ratio of 2.5 is the best for both delay matching and layout in the working technology.

The transistor channel width of a minimum sized inverter is 375 nm and 150 nm for the PMOS and NMOS respectively, while the channel length is set to 60 nm. The inverter layout is composed as a PCell, where inverters of different sizes are generated from a single layout by extending the number of gate fingers of the transistors, as shown in Figure 2.3.

The designed tri-state inverter is a C^2MOS structure, which is insensitive to clock overlaps in sequential logics, as long as the clock transition time is sufficiently small. Figure 2.4 and Figure 2.5 shows the circuit diagram and layout of the tri-state inverter. It is worth noting that in the layout, the drain of NM0 and the source of NM1 are directly connected by forming a continuous diffusion region, which saves area and reduces parasitic resistance and capacitance.

2.2.3 DCDL Layout and Simulation

Corner	TT	SS	FF	SNFP	FNSP
Resolution (ps)	105.4	141.5	81.12	110.7	101.2

Table 2.1: Summary of DCDL resolution

Figure 2.6 shows the layout the designed DCDL. The layout consists of 31 identical delay blocks as discussed in the previous section, with a dummy inverter at the left end of the loop to balance the load capacitance. A large p+ diffusion guard ring is made around the whole block to absorb any possible substrate noise. Table 2.1

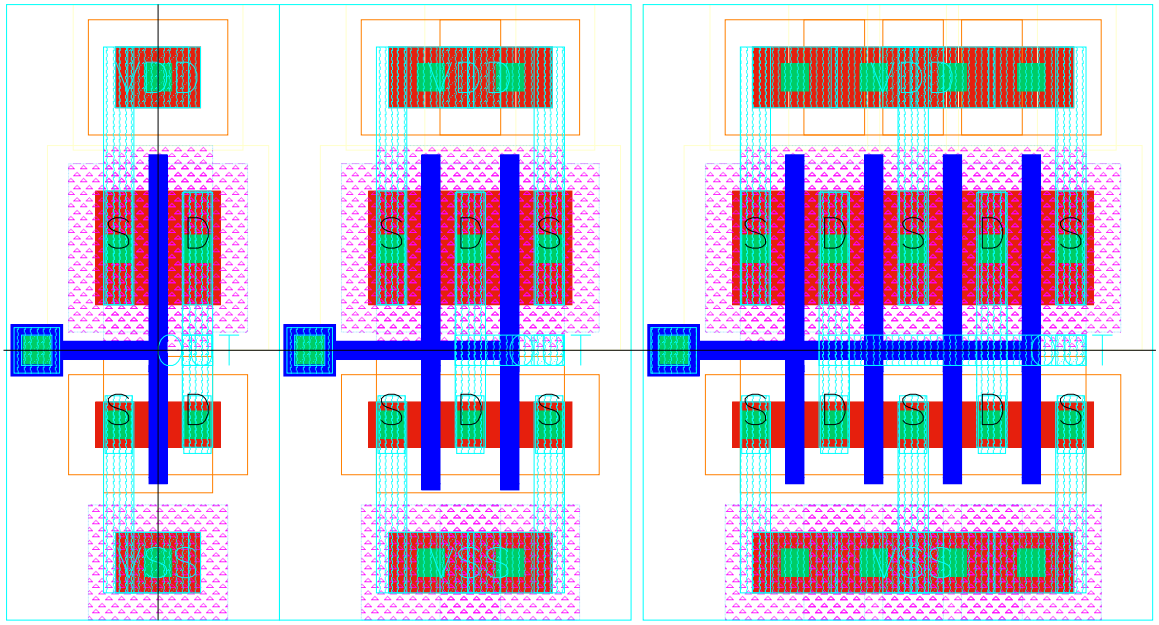

Figure 2.3: Layout of 1, 2 and 4 time sized inverters

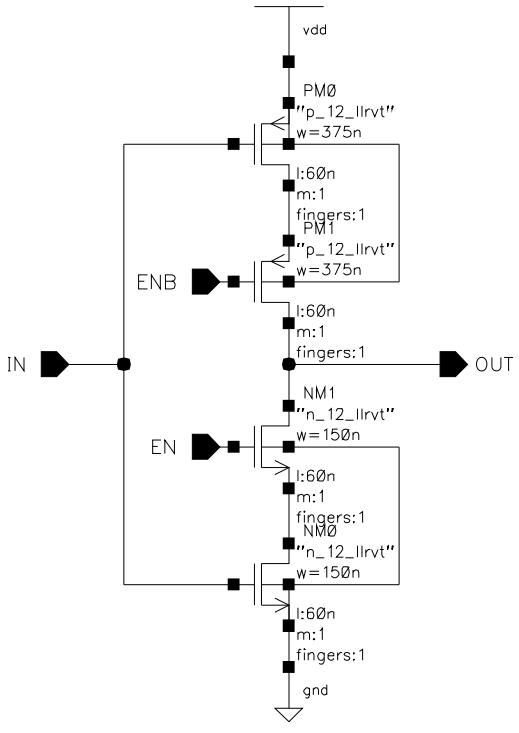

Figure 2.4: Circuit diagram of the tri-state inverter

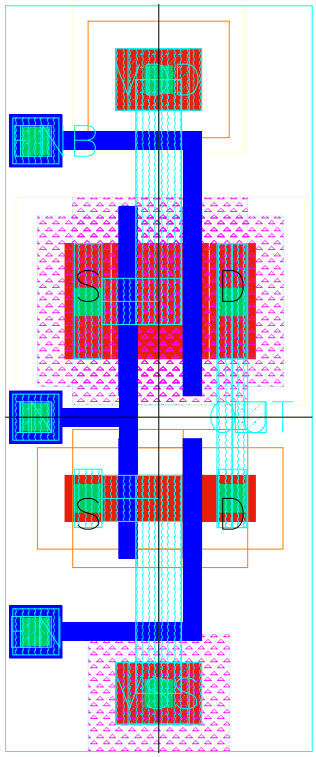

Figure 2.5: Layout of the tri-state inverter

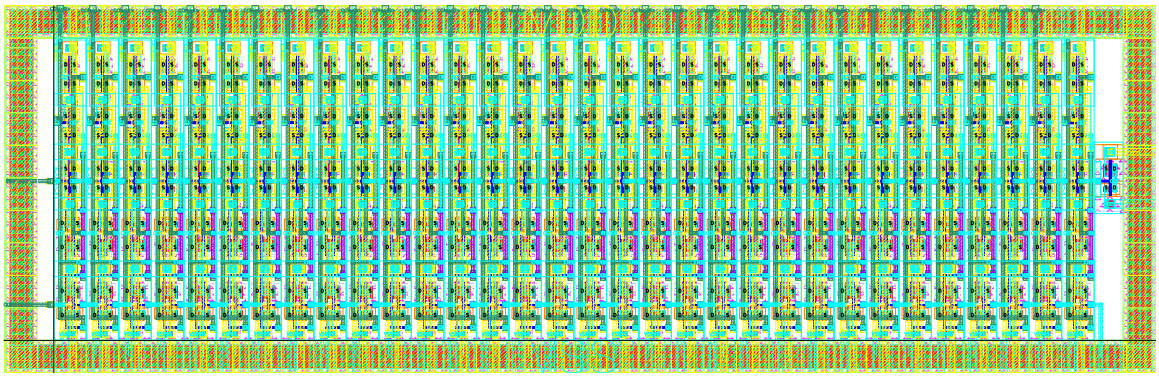

Figure 2.6: Layout of DCDL

summarizes the resolution of the DCDL in all process corners calculated from the post-layout simulation results. It is obvious that the worst-case delay output resolution is around 14% of 1ns period, which is sufficient to adjust the data phase with a maximum phase error of 141.5 ps as a rough estimation. As a conclusion, the resolution of the designed DCDL satisfies the requirement of the proposed data recovery architecture operating at around 1Gbps.

2.3 Time-to-Digital Converter

Time-to-digital converter (TDC) is a functional block that can measure the time interval between two signal transitions, usually denoted as "start" and "stop" signals, and represent the value in a digital format. Traditional TDCs use a two-step conversion process: first convert the time interval into a voltage using an current integrator, and then convert the voltage into digital format using an analog-to-digital converter. This kind of TDCs have the advantage of simple operation with common analog circuits, but they usually take large silicon area, and have poor scalability and performance in deep sub-micron CMOS technologies [10].

Digital TDCs have been studied and used widely due many advantages such as high scalability, low area, and low power consumption. The simplest form of digital TDC is essentially a counter, where a reference clock is used to synchronize the counter. The resolution of the counter based TDC is determined by the frequency of the reference clock, while the speed the flip-flops and the clock distribution can be a limitation to improve the resolution. A TDC with gate delay resolution can be easily designed by using the delay line based architecture, where the resolution is determined by the propagation delay of one delay element.

A basic delay line based TDC is shown in Figure 2.7, which is consisted of delay elements and D flip-flops. The propagation delay of each delay element in the delay line is constant and determines the resolution of the TDC, denoted by T_{LSB}. Once a step signal occurs on the "start" input, it ripples through the delay line, changing the input signal of the D flip-flops at the same time. Once a step signal occurs on the "stop" input, which is connected to the clock input of each D flip-flop, the D flip-flops read in the current value on their D inputs, and forward to the Q outputs. Combining

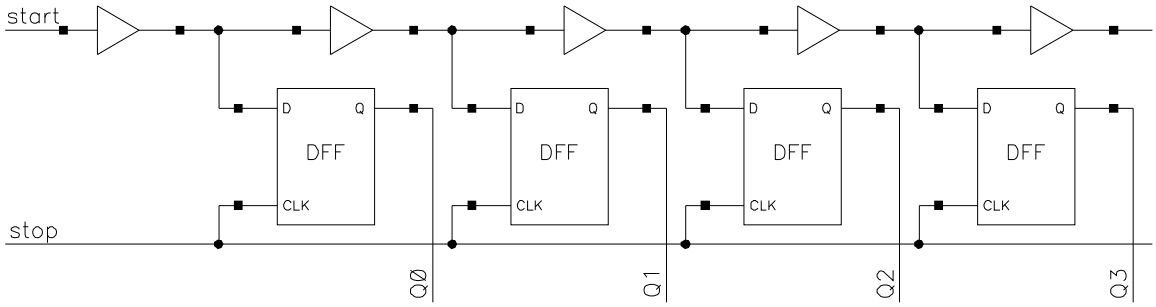

Figure 2.7: Circuit diagram of a basic delay line based TDC

all the Q outputs of the D flip flops forms a thermometer code denoting how far has the step signal propagated through the delay line. The TDC measurement result can be written as

$$T = NT_{LSB} + \varepsilon \tag{2.5}$$

where N is the decimal equivalent TDC thermometer code, and ϵ is the quantization error which is less than the TDC resolution T_{LSB}.

The function of the TDC in the proposed data recovery is to get the phase difference between the data and the clock, thus, there must be sufficient measurement range. Besides, for data rate at around 1Gbps, a resolution of around 100 ps is considered optimum, as its sufficient to reach a 90° phase with limited error. Therefore, the target specification for the TDC is chosen to be 5-bit with a resolution of around 100 ps. The TDC designed in this work is a variant of the delay line based TDC, and is discussed in detail in the following sections.

2.3.1 TDC based on D-latch Chain

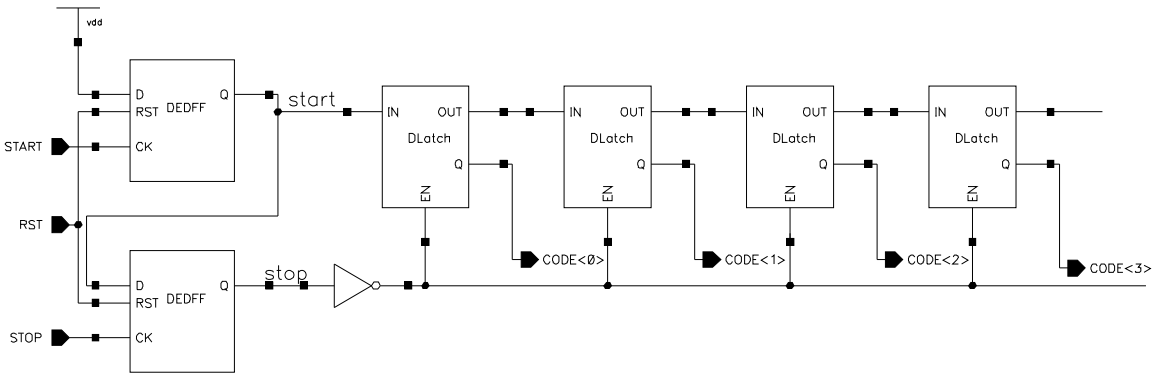

Figure 2.8: Circuit diagram of the designed TDC

Figure 2.8 shows the circuit diagram of the designed TDC. The input signal START and STOP are first connected to an edge detector, which consists of two double-edge-triggered D flip-flops (DEDFF). The edge detector converts either the rising edge or the

falling edge of the START and STOP signal to rising pulses, as in DDR transmission, both clock edges are used, and the TDC needs to start a measurement process no matter which transition occurs. The first DEDFF has an input of constant high, while its output is connected to the input of the second DEDFF, so that the stop pulse can only be generated after the start pulse.

The generated start pulse propagates through the following D-latch chain, and locks its state when the stop pulse occurs. The TDC measurement result can be expressed using the same equation as Equation 2.5.

2.3.2 Double-edge-triggered D Flip-flop

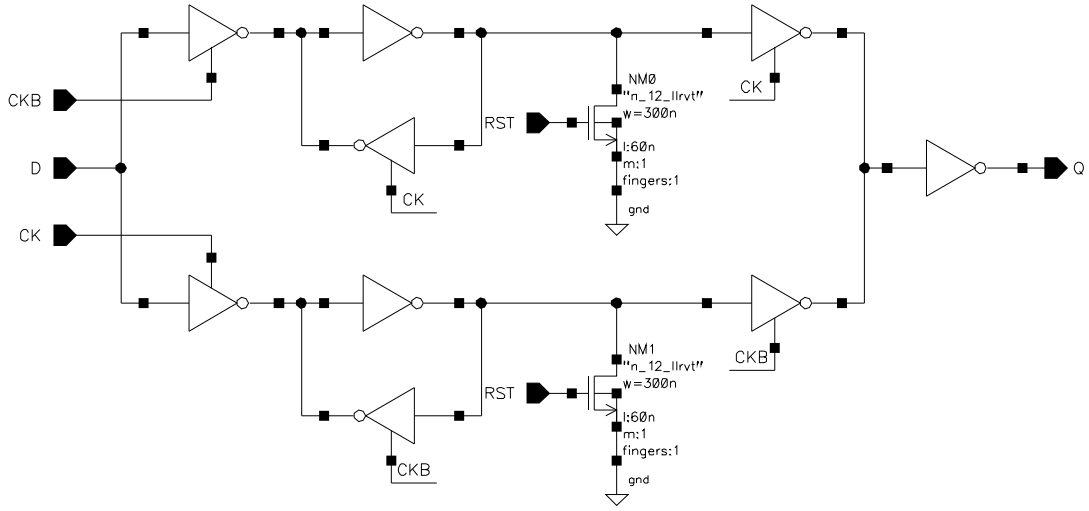

Figure 2.9: Circuit diagram of DEDFF

Figure 2.9 shows the circuit diagram of the designed DEDFF, which consists of two single-edge-triggered D flip-flops working at opposite clock edges, and with their outputs multiplexed. The C^2MOS tristate inverters are used instead of a real multiplexer, as they are less sensitive to clock overlaps. The positive feedback loop in each branch can drive the circuit out of any possible meta-stable point. Figure 2.10 shows the layout of the designed DEDFF.

2.3.3 D-latch Chain

The D-latch chain is the key part of the TDC, which functions as both delay line and measurement result storage. It directly determines the resolution of the designed TDC, as well as other performance metrics such as linearity and measurement range. Instead of using a dedicated delay line as shown in the basic delay line based TDC, the TDC in this work merges the delay line into the storage part, which is changed from D flip-flops to D-latches. As a result, the designed TDC has a smaller area and power consumption, but still satisfies the resolution requirement.

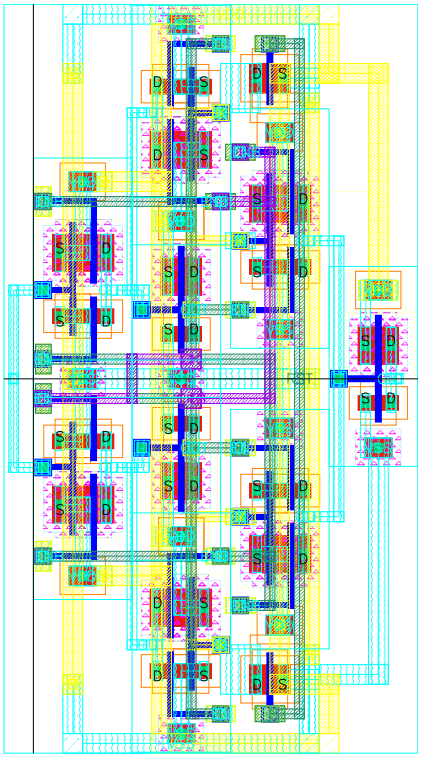

Figure 2.10: Layout of DEDFF

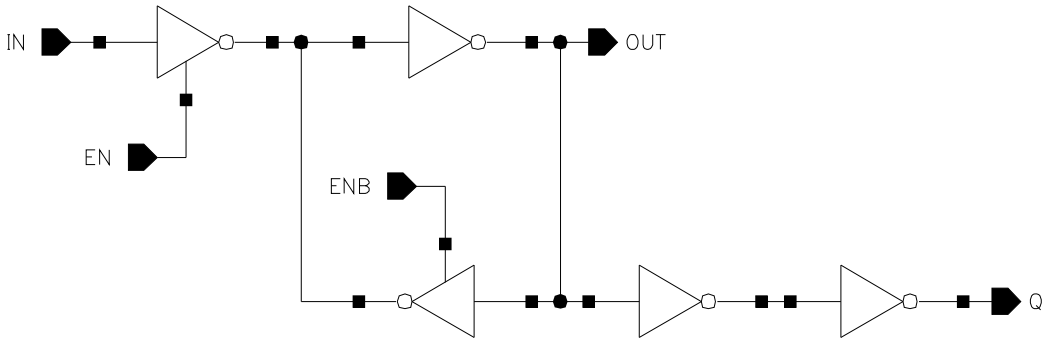

Figure 2.11: Circuit diagram of D-latch

The designed D-latch chain consists of 32 identical D-latches, with 31 data outputs and an overflow output. The circuit diagram of a single D-latch is shown in reffigDlatch. The resolution of the designed TDC can be written as

$$T_{TDC} = T_{tinv} + T_{inv} \tag{2.6}$$

The same as discussed in Section 2.2.1, the exact value of T_{inv} and T_{tinv} are only determined by their output load capacitance, which can be written as

$$C_{Linv} = 3C_i \tag{2.7}$$

$$C_{Linvt} = C_i + C_{otinv} \tag{2.8}$$

Detailed analysis will be discussed in later sections.

2.3.4 Single-complementary Conversion and Buffering

Since the input START and STOP signals are single phase signal, while the DEDFFs work on a complementary clock signal, a single-to-complementary signal converter is added before the DEDFFs. The converter is consist of two inverter chains, one with 2 inverters, and the other with 3 inverters, as shown in Figure 2.12. The transistor channel length of the first inverter in the upper chain is set to 150 nm to add more propagation delay, while the size of the reset inverts complies with the minimum design rules discussed in Section 2.2.2.

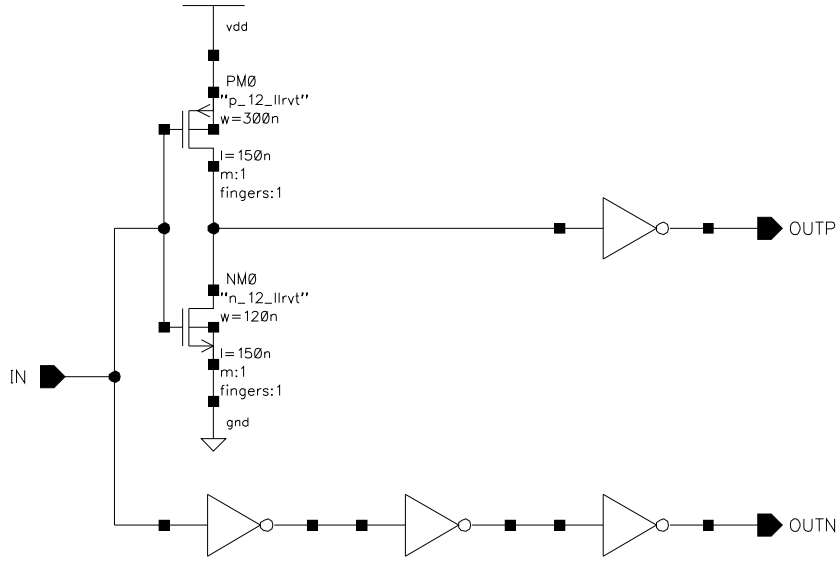

Figure 2.12: Circuit diagram of the single-to-complementary converter

The 32 EN and ENB inputs of the D-latch chain create a considerable amount of capacitive load to the "stop" signal, which must be buffered before connected to the D-latch chain, as the C2MOS structure requires a short clock transition time. A rough estimation shows the input capacitance of EN and ENB signal of the D-latch chain is comparable to 48 times of the input capacitance of an inverter, considering of the capacitance of the routing wires. Therefore, 16-time sized inverters are used to drive the EN and ENB signals of the D-latch chain to ensure a fast transition time. Besides, buffered "stop" and "stopbar" signals are connected to the the D-latch chain using a tree structure, so that the "clock skew" is minimized and a high TDC linearity is achieved. To resolve the offset resulted from the added buffer chain, the "start" signal is also delayed by an inverter chain. Figure 2.13 shows the buffer chains of the start" and stop" signal.

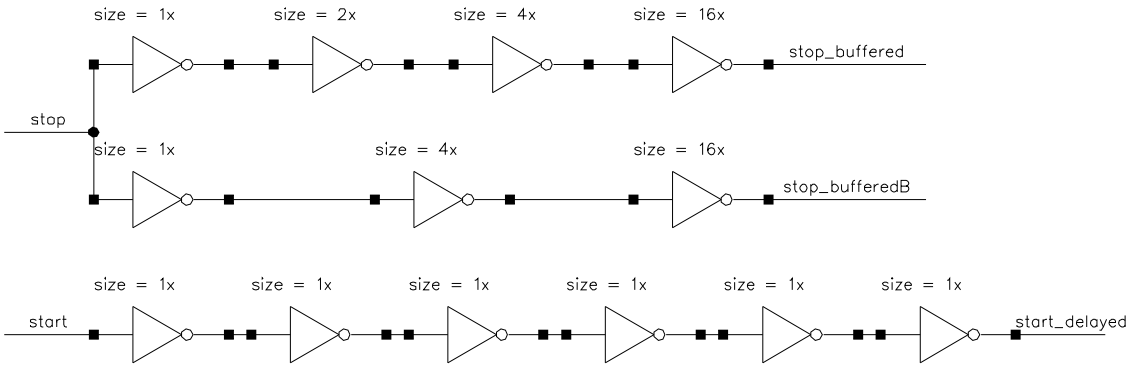

Figure 2.13: Buffering the "start" and "stop" signals

2.3.5 TDC Layout and Simulation

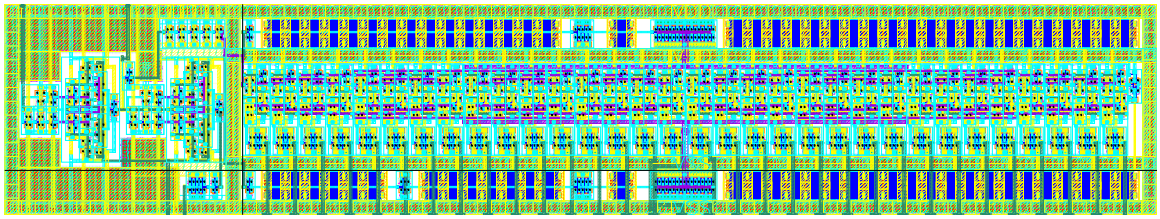

Figure 2.14: Layout of TDC

Figure 2.14 shows the layout of the TDC. Figure 2.15 shows the post-layout simulation result with a "START"-"STOP" transition interval of 4 ns in typical-typical corner. The resolution of the TDC is measured by dividing the delay between the first TDC code rising edge to the TDC overflow rising edge by 31. The calculated TDC resolution in all process corners are summarized in Table 2.2. It is obvious that the worst-case resolution is around 15% of 1ns period, which leads to a maximum phase quantization error of 75 ps as a rough estimation. As a conclusion, the resolution of the designed TDC satisfies the requirement of the proposed data recovery architecture operating at around 1Gbps.

Corner	TT	SS	FF	SNFP	FNSP
Resolution (ps)	108.7	149.2	81.69	117.5	101.3

Table 2.2: Summary of TDC resolution

2.3.6 TDC Measurement Range

A measurement of full TDC range occurs when all the bits of the TDC output thermometer code have risen to '1', while the TDC overflow output is still '0. This full range can be written as

$$T_{max} = 31T_{TDC} \tag{2.9}$$

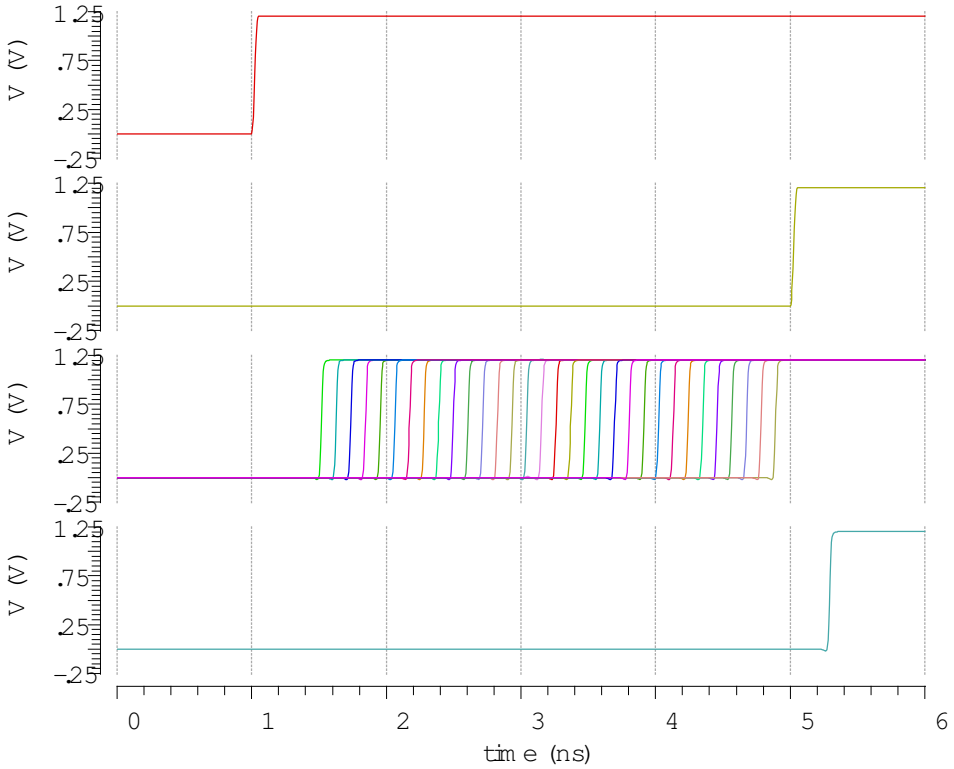

Figure 2.15: TDC post-layout simulation results (The waveforms from the top row to the bottom row are START, STOP, TDC output thermometer code, TDC finish)

Where T_{TDC} is the resolution of the TDC.

The minimum measurement range of the TDC is determined by the minimum "START"-"STOP" transition interval that can successfully generate the "stop" pulse from the edge detector. This means the set-up time of the second DEDFF must be satisfied, as shown in the timing diagram Figure 2.16. This seems to conclude that the minimum "START"-"STOP" transition interval is

$$T_{min} = T_{ck-q} + T_{su} \tag{2.10}$$

where T_{ck-q} is the clock-to-output delay, and T_{su} is the set-up time of the DEDFF. However, T_{ck-q} of the DEDFF is not constant, but varies with the data arrival time. In other words, T_{ck-q} becomes significantly larger as the D input transition approaches the clock transition due to the metastability problem, and eventually the DEDFF cannot read the D input properly. This makes it difficult to characterize a T_{ck-q} and a T_{su} to be used in Equation 2.10. Besides, if the "start" pulse is too close to the "STOP" transition, the T_{ck-q} of the second DEDFF becomes much larger than that of the first DEDFF, which will make the "start"-"stop" pulse interval larger than the "START"-"STOP" transition interval, and cause an error in the TDC measurement result. Therefore, real simulations are performed, in which the "START"-"STOP" transition interval is first sufficiently large, and gradually decreases until the TDC cannot measure the interval correctly.

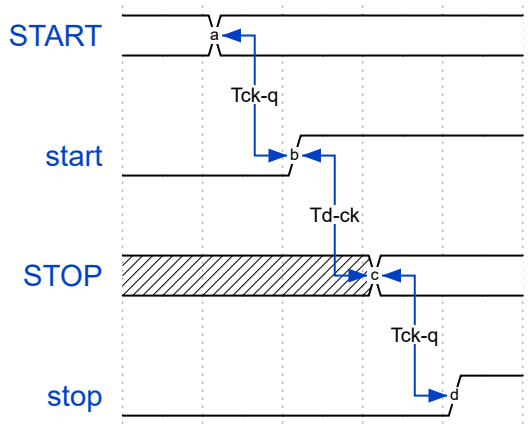

Figure 2.16: Timing diagram of the edge detector

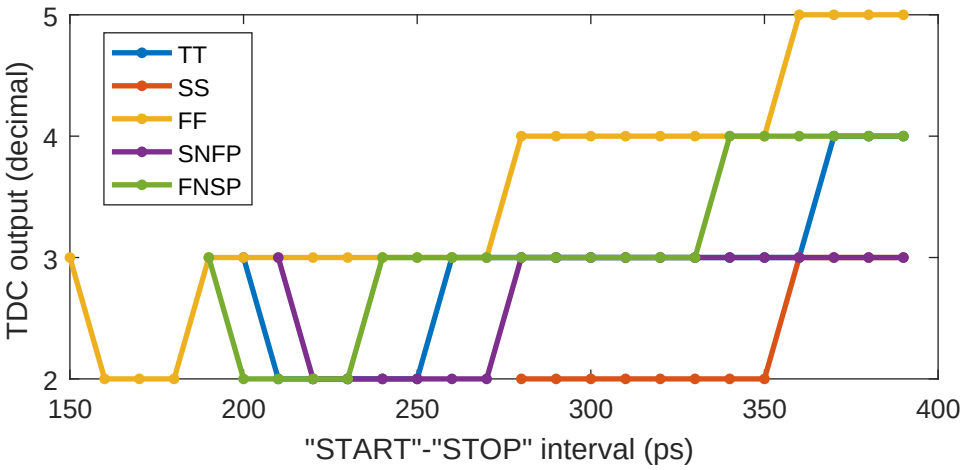

Figure 2.17: TDC output vs. input START"-STOP" transition interval

Figure 2.17 shows the TDC output with "START"-"STOP" transition interval in 10 ps incremental step. As shown in the results, some of the TDC outputs do not start from the minimum value of "2", but start at "3", and then decrease to "2", which can be explained by the discussion above. Therefore, these data points should be discarded as errors, and the following data points are chosen as the minimum "START"-"STOP" transition interval.

Table 2.3 summarizes the TDC measure range in all process corners. It is obvious that a bit time of 1 ns is sufficiently measurable using the designed TDC in all process corners. As a conclusion, the TDC measurement range satisfies the requirement of the

Corner	TT	SS	FF	SNFP	FNSP
T_{min} (ps)	210	280	160	220	200
T_{max} (ps)	3370	4625	2533	3641	3140

Table 2.3: Summary of TDC measurement range in all process corners

proposed data recovery architecture operating at around 1Gbps.

2.4 Finite-State Machine

In the proposed data recovery architecture, the FSM functions as the control circuit. The FSM needs to identify valid TDC measurement results, and generate a correct DCDL setting according. The FSM must also be able to do conversion between thermometer code and binary code, as both the TDC results and DCDL control are in thermometer code, while the internal arithmetic calculation is in binary.

2.4.1 From Algorithm to RTL

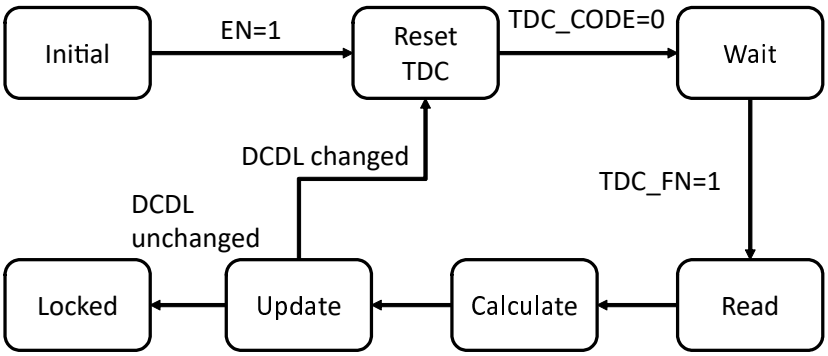

Figure 2.18: FSM state transition diagram

Figure 2.18 shows the state transition diagram of the FSM. A description of each state with its inputs and outputs is summarized as follows:

- Initial: DCDL (DLSET) is set to 0; if enable signal (EN) is "1", then go to next state; otherwise stay in current state;
- Reset TDC: TDC reset signal (TDC_RST) is set to "1"; if TDC outputs (TDC0_CODE, TDC1_CODE) are all "0", then go to next state; otherwise stay in current state;
- Wait: Wait for TDC measurement to finish; if TDC finish signal (TDC_FN) is "1", then go to next state; otherwise stay in current state;
- Read: Read in TDC measurement results (TDC0_CODE, TDC1_CODE);
- Calculate: Calculate the new DCDL setting value, based on previous DCDL setting value;
- Update: Update DCDL setting value (DLSET); if the new value is the same as the old value, then go to "Locked" state; otherwise, return to "Reset TDC" state;
- Locked: FSM has successfully set the correct DCDL setting value, and the phase difference between the data and clock is measured as 90°.

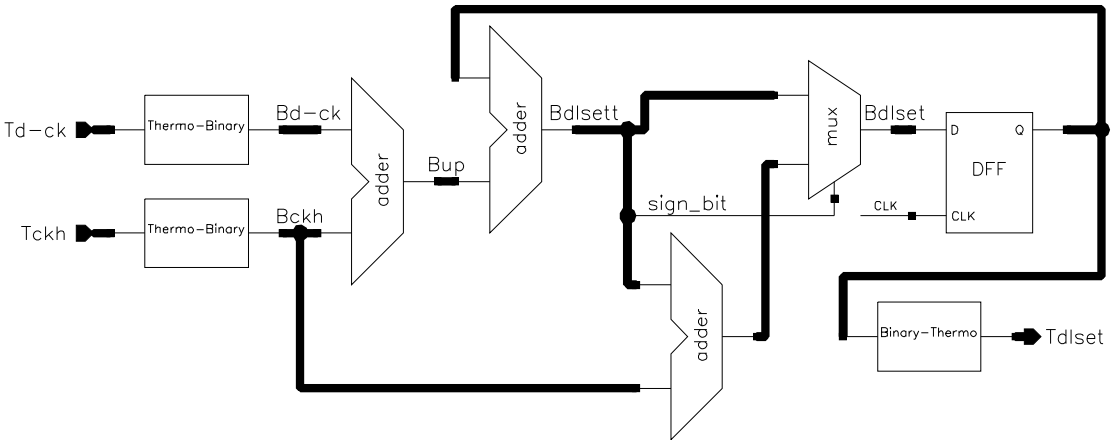

Figure 2.19: Block diagram of the DLSET calculation logic in the FSM

Figure 2.19 shows the block diagram of the DLSET calculation logic in the FSM. The DLSET calculation process is described as follows. The FSM converts the two TDC thermometer codes to binary form, B_{d-dk} and B_{ckh}, which represent the data-to-clock transition interval and clock-to-clock transition interval. Then, an update code is obtained using the following equation

$$B_{up} = B_{d-dk} - B_{ckh}/2 \tag{2.11}$$

B_{up} means how far the data transition is away from the 90° phase point of the clock. B_{up} is then added to the current DCDL setting value B_{DLSET}. In case B_{up} is negative, and $B_{DLSET} < B_{up}$, a B_{ckh} is also added to prevent underflow of the DCDL setting value. The new DCDL setting value can be written as

$$B_{DLSET} = B_{DLSET} + B_{up} \tag{2.12}$$

The new binary B_{DLSET} is then converted to the thermometer code T_{DLSET} and then transferred to the DCDL control input.

The FSM is written in Verilog HDL (Appendix B).

2.4.2 From RTL to Layout

Using the Verilog HDL RTL code, the FSM is synthesized in *Synopsys Design Compiler*, placed and routed in *Cadence Encounter Digital Implementation*. The FSM is then transferred to *Virtuoso* for physical verification and ready for top-level integration. The detailed steps are discussed in Appendix A. Figure 2.20 shows the placed & routed FSM in *Encounter*, and Figure 2.21 shows the final layout of the FSM.

2.5 Implementation of the Data Recovery Architecture

Figure 2.22 shows the circuit diagram of the implemented data recovery. There are two TDCs working in parallel: TDC0 measures the clock transition interval with its

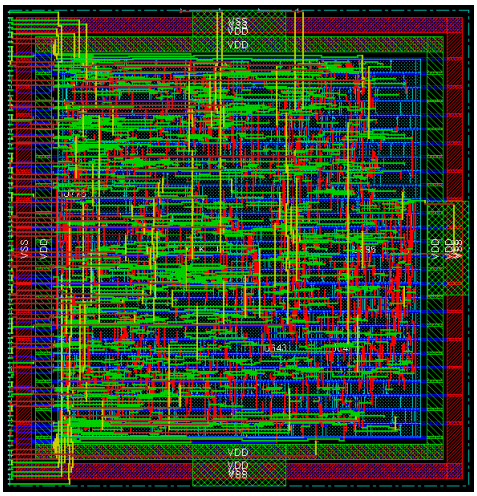

Figure 2.20: FSM placed & routed in *Encounter*

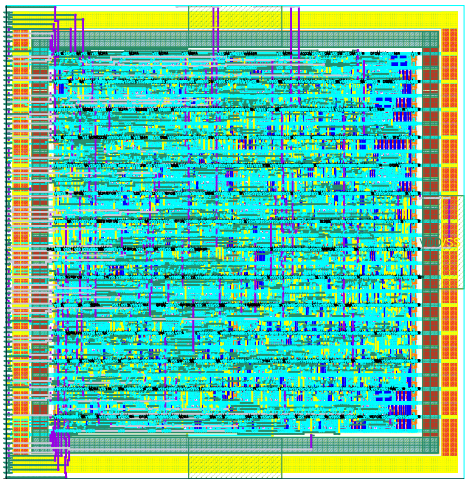

Figure 2.21: Final FSM layout in *Virtuoso*

"START" and "STOP" input both connected to the clock signal CK_SYN; TDC1 measures the transition interval between the delayed data signal and the clock signal, with its "START" input connected to the delayed data signal DOUT and "STOP" input connected to CK_SYN. Figure 2.23 shows the layout of the implemented data recovery. On the left side are the DCDL and TDCs with a decoupling capacitor, and on the right is the FSM.

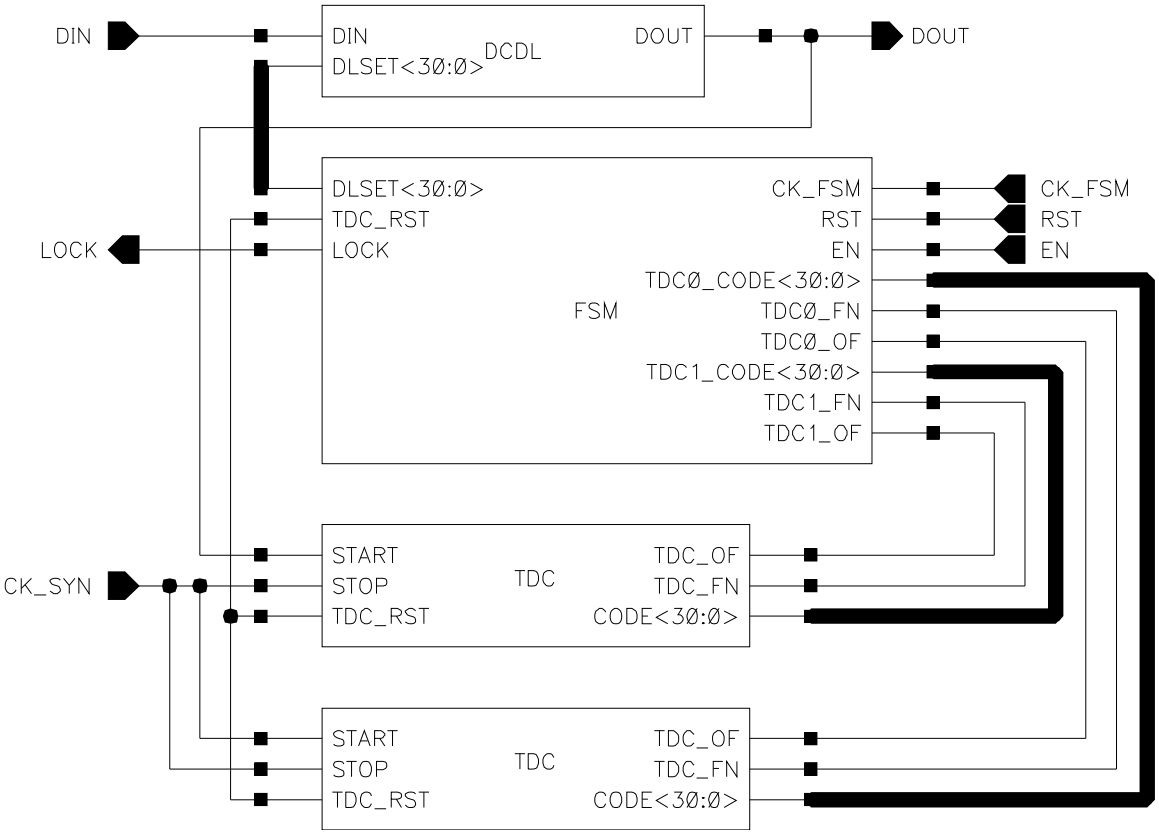

Figure 2.22: Circuit diagram of the data recovery

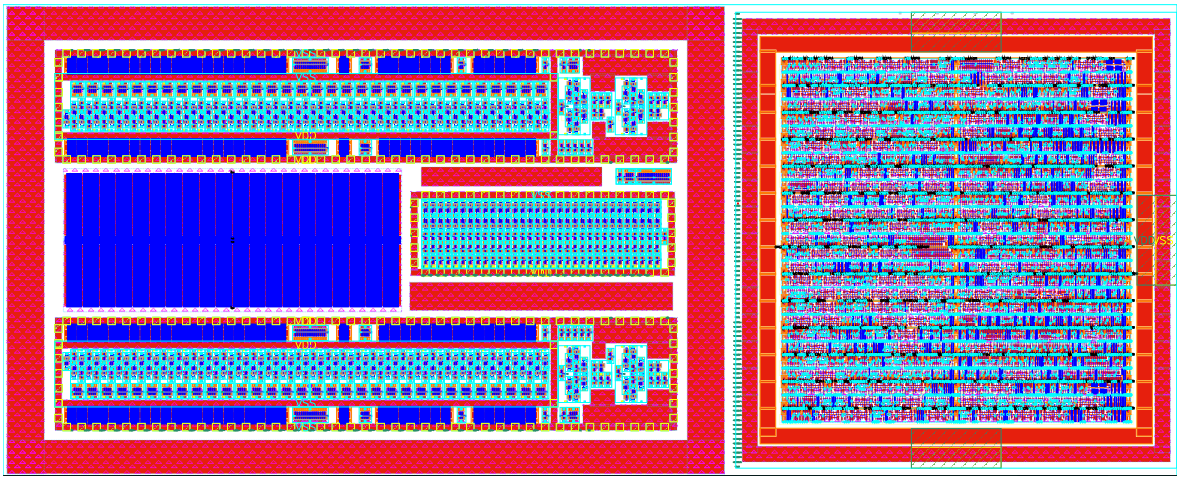

Figure 2.23: Layout of the data recovery (all metal layers are hidden, except vias and pins) (157 µm × 63 µm)

2.6 Mathematical Modeling of the Data Recovery Architecture

In this section, a mathematical model of the data recovery architecture in time domain is built to analyze its performance in detail.

2.6.1 Operating Range

From the discussion in Section 2.3.6, there are both upper and lower limits to the "START"-"STOP" transition interval for the TDC input. TDC0 can work correctly as long as the bit time T_{bit} is within the TDC measurement range.

$$T_{min} < T_{bit} < T_{max} \tag{2.13}$$

However, for TDC1 there is the possibility that the CK_SYN transition occurs below the T_{min} limit

$$0 < T_{d-ck} < T_{min} \tag{2.14}$$

which leads to the "stop" pulse being triggered at the next CK_SYN transition, as shown in the timing diagram in Figure 2.24.

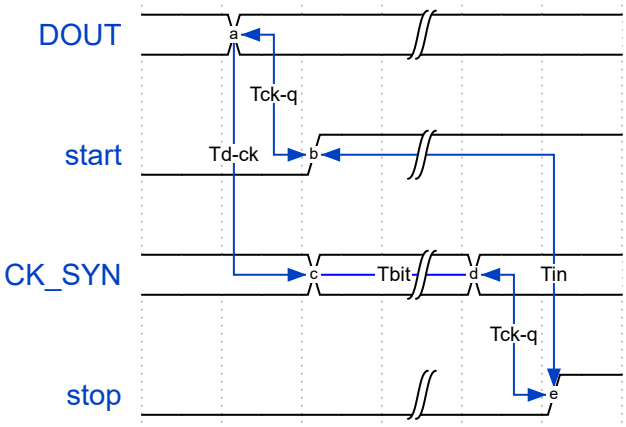

Figure 2.24: Timing diagram of the edge detector when $T_{d-ck} < T_{min}$

In this situation, the actual interval between the "start" and "stop" pulse can be derived from

$$T_{ck-q} + T_{in} = T_{d-ck} + T_{bit} + T_{ck-q} \tag{2.15}$$

$$T_{in} = Td - ck + T_{bit} \tag{2.16}$$

The above equation means the TDC measurement results is larger than the actual "DOUT"-CK_SYN" transition by one bit time. Therefore, T_{in} must be with the TDC measurement range

$$T_{min} < T_{in} < T_{max} \tag{2.17}$$

Corner	TT	SS	FF	SNFP	FNSP
T_{min} (ps)	210	280	160	220	200
T_{max} (ps)	3370	4625	2533	3641	3140
$T_{bit,min}$ (ps)	420	560	320	440	400
$T_{bit,max}$ (ps)	3160	4345	2373	3421	2940
$F_{bit,max}$ (Gbps)	2.38	1.79	3.13	2.27	2.50
$F_{bit,min}$ (Mbps)	316.5	230.1	421.4	292.3	340.1

Table 2.4: Summary of the operating range of the data recovery in all process corners

$$T_{min} - Td - ck < T_{bit} < T_{max} - T_{d-ck} \tag{2.18}$$

For the DCDL setting calculation in the FSM, T_{bit} is divided by 2 and compared with T_{d-ck}, and only when T_{d-ck} equals $T_{bit}/2$ will the FSM assert the "LOCK" signal. This requires that $T_{bit}/2$ must also be measurable by the TDC, and sets an inequality

$$T_{min} < T_{bit}/2 < T_{max} \tag{2.19}$$

$$2T_{min} < T_{bit} < 2T_{max} \tag{2.20}$$

We can solve the inequality group in (2.13), (2.14), (2.18), and (2.20) as

$$2T_{min} < T_{bit} < T_{max} - T_{min} \tag{2.21}$$

Equation 2.21 describes the maximum and minimum bit time required for the data recovery to work properly. Table 2.4 summarizes the operating range of the data recovery in all process corners. As a conclusion, a bit time between 560 ps to 2373 ps can ensure proper operation of the data recovery in all process corners, which give s a sufficient margin for bit rate at 1Gbps.

2.6.2 Convergence

It is preferred that the data recovery "locks" as fast as possible. The data recovery implemented in this work is based on a quantized phase acquisition and a quantized delay generation scheme, and the FSM assumes the LSBs, equal to the resolution, of the DCDL and TDC are equal for fast locking. However, even if the circuit design is perfect, there is always a mismatch between the LSB of the DCDL and TDC because of process, voltage and temperature (PVT) variation. This distracts the converging process from ideal, where the data recovery locks in 1 calibration iteration. The following mathematical analysis discusses the convergence of the data recovery when the resolution mismatch between the TDC and DCDL exists. The T_{bit} and T_{d-ck} are measured by the TDC, and can be expressed as

$$T_{bit} = N_0 T_{TDC} + \varepsilon_0 \tag{2.22}$$

$$T_{d-ck} = N_1 T_{TDC} + \varepsilon_1 \tag{2.23}$$

where N_0 and N_1 are the TDC measurement results, ε_0 and ε_1 are the TDC quantization errors, and we assume $|\varepsilon_0| \leq 0.5T_{TDC}$, $|\varepsilon_1| \leq 0.5T_{TDC}$. According to the FSM calculation process, the DCDL setting value can be expressed as

$$M = \left(N_1 - \frac{N_0}{2} \right) \tag{2.24}$$

Let τ be the resolution mismatch between the DCDL and the TDC, and we assume $|\tau| << T_{TDC}$, $|\tau| << T_{DCDL}$, we can write

$$T_{DCDL} = T_{TDC} + \tau \tag{2.25}$$

After one FSM iteration, the delay output of the DCDL is increased by

$$\Delta D_{DCDL} = D_{DCDL}(M) - D_{DCDL}(0) = M * T_{DCDL} = M(T_{TDC} + \tau) \tag{2.26}$$

The data is delayed by the above value, and the resulting new T_{d-ck}

$$T'_{d-ck} = T_{d-ck} - \Delta D_{DCDL} = N_1 T_{TDC} + \varepsilon_1 - M(T_{TDC} + \tau) = \frac{N_0 T_{TDC}}{2} + \varepsilon_1 - M\tau \tag{2.27}$$

In the ideal case, i.e. $\tau = 0$, the new T'_{d-ck} will be measured by the TDC as $\frac{N_0 T_{TDC}}{2}$, which is half of the bit time. This means the data recovery locks in 1 calibration iteration, with the second iteration to verify the "lock" condition. We rewrite (2.27) in a more general case where τ can be either positive, negative, or zero

$$T'_{d-ck} = \left(\frac{N_0}{2} + N_e \right) T_{TDC} + (\varepsilon_1 - M\tau - N_e T_{TDC}) \tag{2.28}$$

We define N_e as integer, representing the number of T_{TDC} contained in the error item $\varepsilon_1 - M\tau$. The new T'_{d-ck} will be measured by the TDC as $\frac{N_0}{2} + N_e$. To derive an expression of N_e, we define a "relative calibration error" as

$$\frac{\varepsilon}{T_{TDC}} = \frac{\varepsilon_1 - M\tau}{T_{TDC}} \tag{2.29}$$

According to its definition, N_e is the equivalent TDC measurement result if an input transition interval equal to the error item $\varepsilon_1 - M\tau$ were measured by the TDC. Therefore, we can write

$$N_e = int\left(\frac{\varepsilon_1 - M\tau}{T_{TDC}} \right) = int\left(\frac{\varepsilon}{T_{TDC}} \right) \tag{2.30}$$

where $int()$ is a function that rounds a value to its nearest integer. Figure 2.25 plots the value of N_e with the relative calibration error. If $N_e = 0$, the data recovery locks as discussed above. If $N_e \neq 0$, the data recovery needs another iteration to calibrate

the DCDL setting value, and N_e will be the number added to the DCDL setting value. Then the new T_{d-ck} is

$$T''_{d-ck} = T'_{d-ck} - N_e(T_{TDC} + \tau) = \frac{N_0 T_{TDC}}{2} + \varepsilon' \tag{2.31}$$

$$\left(\frac{\varepsilon}{T_{TDC}}\right)' = \frac{\varepsilon}{T_{TDC}} - N_e\left(1 + \frac{\tau}{T_{TDC}}\right) \tag{2.32}$$

The above equation gives a relationship between the new relative calibration error to the original relative calibration error, and is plotted in Figure 2.25 in different τ value situations.

When $\tau = 0$, the new relative calibration error is always within -0.5 to 0.5, and the data recovery can lock in one calibration iteration as discussed before. When $\tau > 0$, there are some regions that can cause the data recovery to "dither". For instance, at the point where $\frac{\varepsilon}{T_{TDC}}$ is slightly larger than but close to 0.5, the new relative calibration error, which is $\left(\frac{\varepsilon}{T_{TDC}}\right)'$, is at the point where $\frac{\varepsilon}{T_{TDC}}$ is slightly smaller than but close to -0.5; then, the next $\left(\frac{\varepsilon}{T_{TDC}}\right)'$ will return to the first $\frac{\varepsilon}{T_{TDC}}$ value. In this situation, the data recovery "dithers" around the two points, and can never lock.

Therefore, $\tau \leq 0$ is a requirement for the data recovery to converge to the locking condition, which means the resolution of the DCDL must be no greater than that of the TDC. However, a too small DCDL resolution will increase the locking time as discussed above, and has no contribution to improve the phase error. As a result, the optimum DCDL resolution should be slightly smaller than the TDC resolution.

2.6.3 Fine Tuning the DCDL resolution

Recalling the discussion in Section 2.2.1 and Section 2.3.3, the resolution of the DCDL and TDC are only determined by the load capacitance of their inverters and tri-state inverters. The the tri-state inverters, the load capacitance in the TDC and DCDL are equal, while for the inverter, the load capacitance in the TDC is $3C_i$, but $2C_i$ in the DCDL. This means the DCDL resolution is smaller than the TDC resolution, and there is some space to reduce the locking time as discussed in the previous section.

To realize this, a dummy inverter is added to the output of all the inverters in the DCDL, so that the propagation delay of the inverters in the DCDL are mathematically equal to that in the TDC. The modified DCDL circuit diagram is shown in Figure 2.26. The resolution of the TDC and DCDL, with their resolution mismatch are summarized in Table 2.5. As a conclusion, a resolution mismatch less than zero is realized in all the process corners to avoid non-convergence, and are designed to be as close to zero as possible to reduce locking time. Applying the relative resolution mismatch τ/T_{TDC} in Table 2.5 into the discussion in the previous section, the designed data recovery takes a maximum of 2 iterations to calibrate the DCDL setting value.

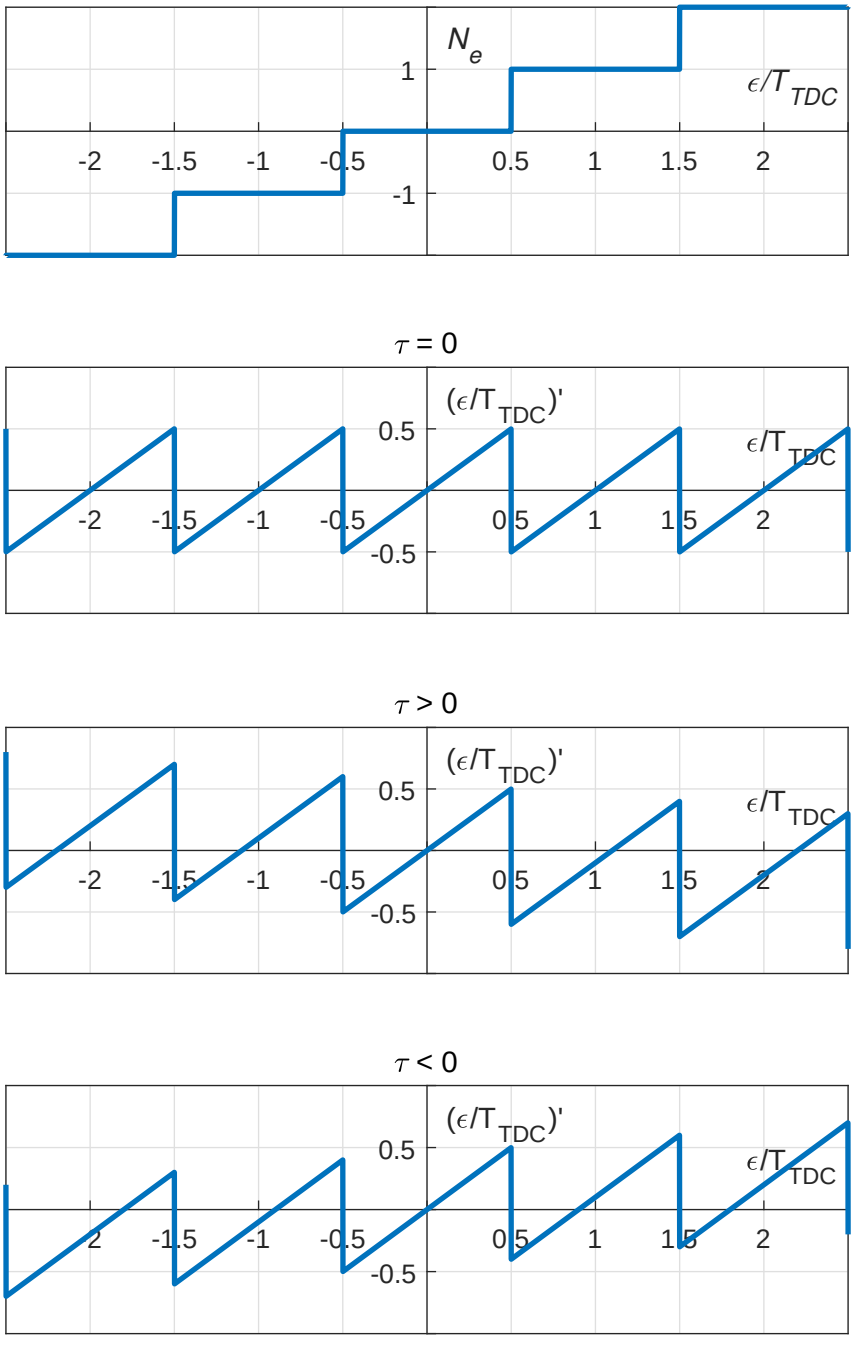

Figure 2.25: N_e and new relative calibration error vs. relative calibration error

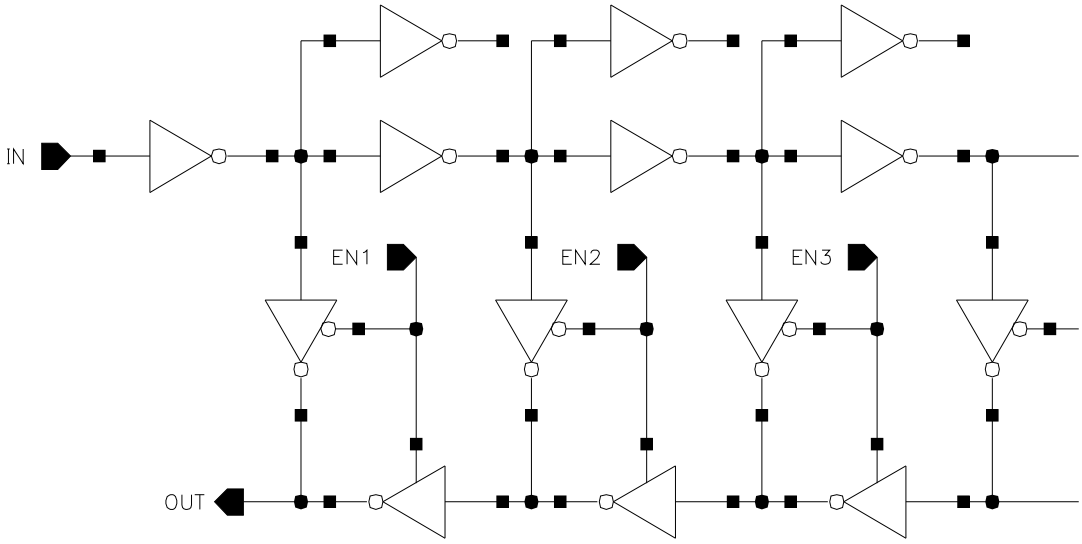

Figure 2.26: Circuit diagram of the DCDL with dummy inverters

Corner	TT	SS	FF	SNFP	FNSP
T_{TDC} (ps)	108.7	149.2	81.69	117.5	101.3
T_{DCDL} (ps)	105.4	141.5	81.12	110.7	101.2
τ (ps)	-3.3	-7.7	-0.57	-6.8	-0.1
τ/T_{TDC}	-0.030	-0.052	-0.007	-0.058	-0.001

Table 2.5: Summary of the resolution of TDC and DCDL, resolution mismatch and relative resolution mismatch

2.6.4 Phase Error

Another performance metric of the data recovery of interest is the phase error after locking, which is defined as the difference between the actual data phase to the ideal 90° phase relative to the clock. Two factors contribute to the overall phase error: one is the TDC quantization error, and the other is the error in the half bit time calculation in the FSM, as any odd integer divided by 2 results a 0.5 error.

Using (2.22) and (2.23), we can write the phase error after locking as

$$T_{error} = T_{d-ck} - \frac{T_{bit}}{2} = \left(\left[\frac{N_0}{2} \right] - \frac{N_0}{2} \right) T_{TDC} + \varepsilon_1 - \varepsilon_0 \tag{2.33}$$

where $\left[\frac{N_0}{2} \right]$ represents the integer part of $\frac{N_0}{2}$. By assuming the two quantization error ε_1 and ε_0 are randomly distributed between $-0.5T_{TDC}$ and $0.5T_{TDC}$, the worst-case phase error occurs when N_0 is an odd integer, and $|\varepsilon_1 - \varepsilon_0| = -T_{TDC}$, therefore

$$T_{error,worst} = -0.5T_{TDC} - T_{TDC} = -1.5T_{TDC} \tag{2.34}$$

As a conclusion, the data recovery in this work has a worst-case phase error of $-1.5T_{TDC}$. For 1ns bit time, the worst-case value in SS corner makes the data transition

224 ps away from the ideal phase point, and leaves 276 ps to the nearest clock transition, which can be considered as a sufficient timing margin in most cases.

2.7 Simulation of the Data Recovery

According to the discussion in Section 2.6.1, 560 ps and 2373 ps are the worst-case theoretical minimum and maximum bit time for the data recovery to work properly in all process corners. Thus, the following three test conditions are set to test two extreme cases as well as a typical case:

- A bit time of 1000 ps at TT corner;
- A bit time of 560 ps at SS corner;
- A bit time of 2373 ps at FF corner.

An initial 0° phase shift between the data and clock is set such that the DCDL setting change is maximum. Building the test bench of the data recovery is as straightforward as adding bit sources to the input of the data recovery, and is omitted here. Figure 2.27, Figure 2.28 and Figure 2.29 shows the simulation results of the three test conditions.

Another simulation test bench is set up where different input data phases with 100 ps step are generated. Figure 2.30, Figure 2.31 and Figure 2.32 shows the input data (DIN), and the phase-calibrated output data (DOUT), with the reference clock (CK_SYN), operating at TT corner. It shows that the data recovery is able to calibrate any input data phase to be aligned to the 90° of the reference clock.

The simulation results prove the correctness of the designed data recovery. A maximum of 2 calibration iterations are observed for the data recovery to calibrate the DCDL setting value, which complies with the discussion in Section 2.6.2. The third iteration is used to verify the locking condition, thus the total lock time is around 24 cycles when operating at 1Gbps. The measured phase errors are all within 224 ps as expected in Section 2.6.4.

2.8 Summary of the Data Recovery

This chapter has discussed the proposed data recovery architecture for source-synchronous interface, as well as its implementation and simulation. The designed data recovery can calibrate the phase of the received data to be aligned to 90° phase of the received half-rate reference clock with a maximum phase error of 224 ps. The data recovery locks in around 24 cycles when operating at 1Gbps, and consumes 1.27 mW power during calibration process. After locking, the FSM shuts down to save power, leading to a total power consumption of 143.2 µW. The total layout area is 9891 μm^2, including a decoupling capacitor. Table 2.6 and Table 2.7 summarize the area and power consumption of the designed data recovery.

Conventional CDRs generate or delay the clock signal which use the received data as a phase reference to sample the received data. Instead, the data recovery proposed

in this thesis delays the received data and uses the received reference clock as a phase reference to sample the received data, which eliminates the need of any global or local PLL or DLL and saves a considerable amount of silicon area and power consumption. As a conclusion, the proposed data recovery provides a low-cost efficient solution to error free data sampling for source-synchronous inter-chip communication, which is specially suitable for cost-constraint applications.

Part	Area
DCDL	$35.4\,\mu m \times 11.3\,\mu m$
TDC	$83.3\,\mu m \times 15.2\,\mu m$ x 2
FSM	$59\,\mu m \times 61\,\mu m$
Total active area	$6531.34\,\mu m^2$
Total layout area	$9891\,\mu m^2$

Table 2.6: Area of the designed data recovery

Part	Power before locking	Power after locking
DCDL and TDC	474.9 µW	142.9 µW
FSM	798.8 µW	255.0 nW
Total	1.27 mW	143.2 µW

Table 2.7: Power consumption of the data recovery @ 1Gbps

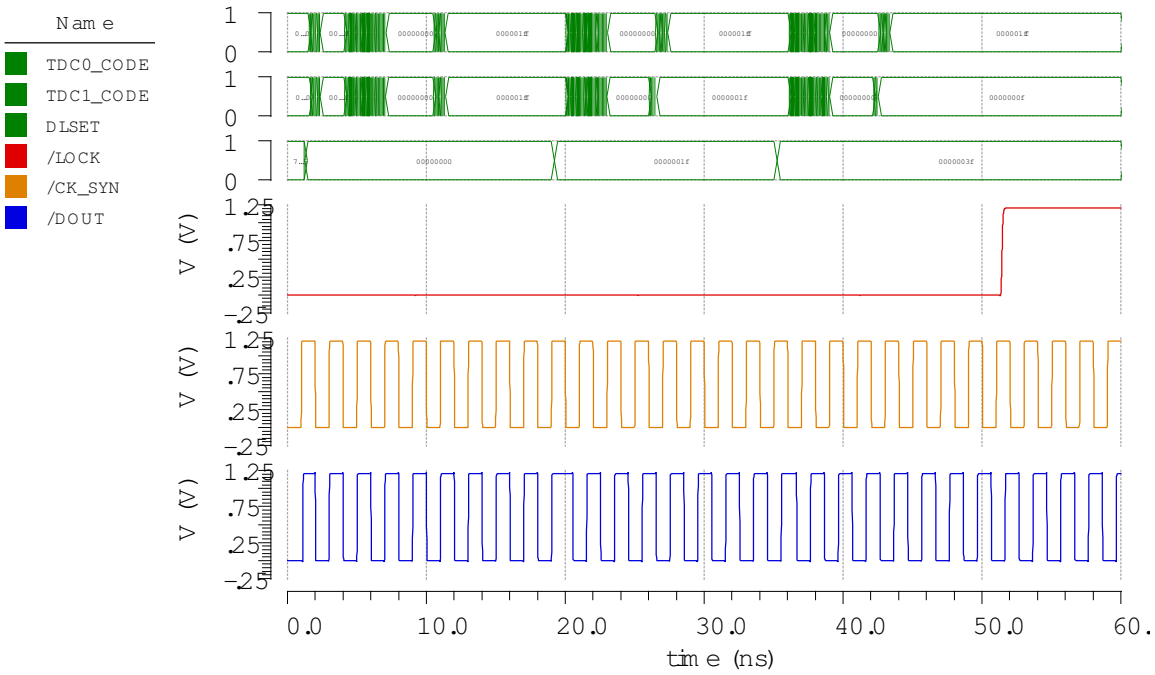

Figure 2.27: Simulation results of the data recovery @ T_{bit}=1000 ps, TT corner

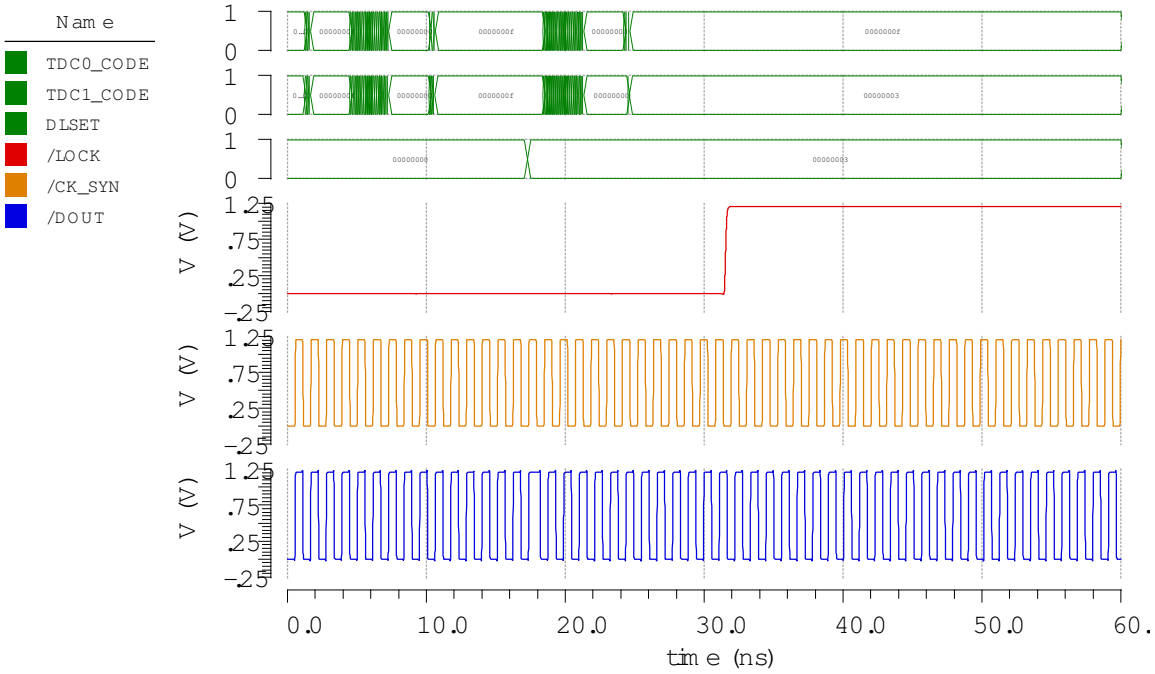

Figure 2.28: Simulation results of the data recovery @ T_{bit}=560 , SS corner

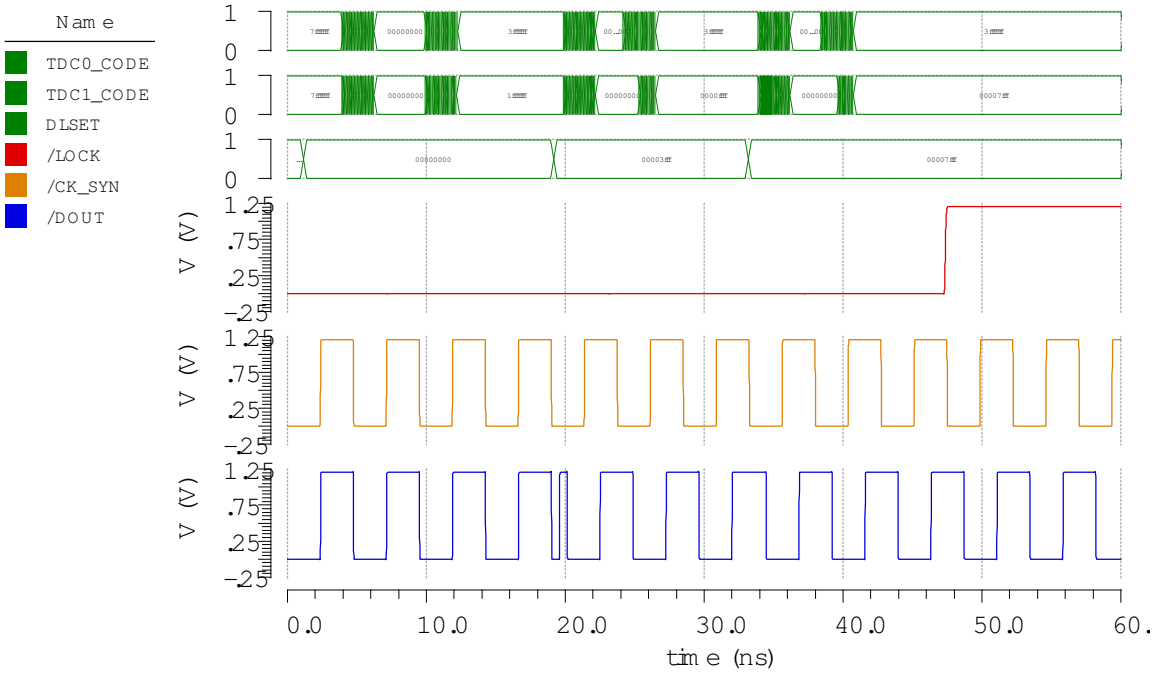

Figure 2.29: Simulation results of the data recovery @ T_{bit}=2373 ps, FF corner

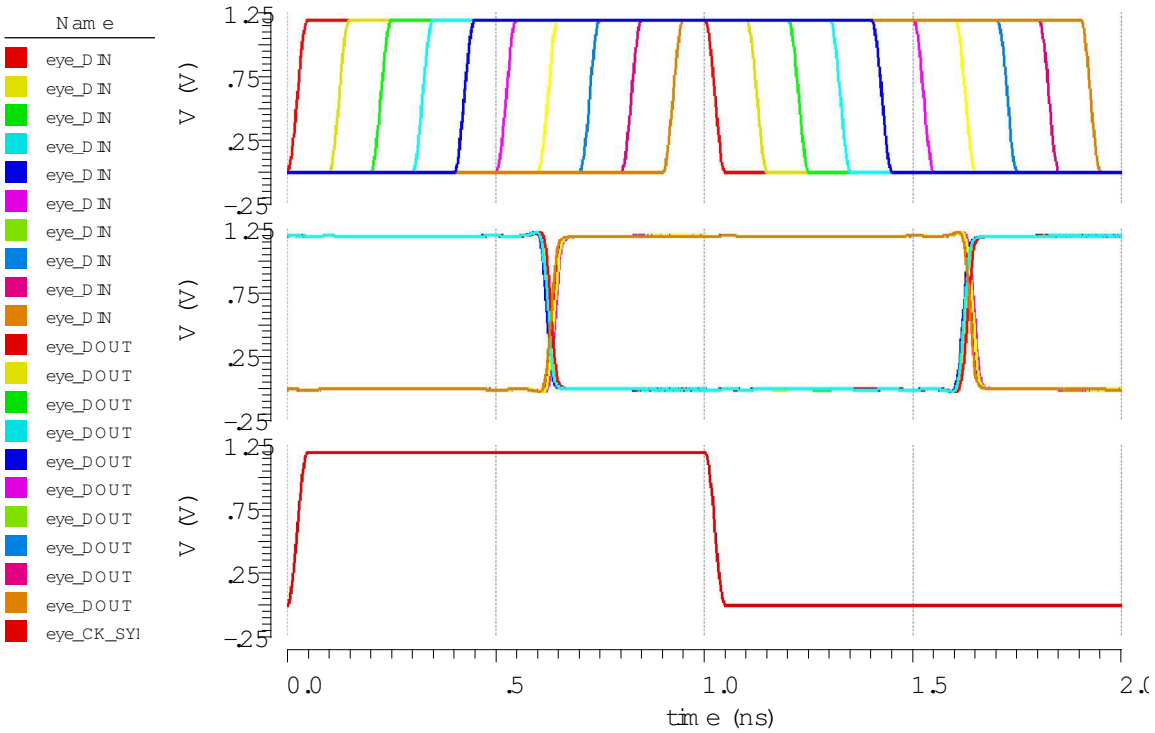

Figure 2.30: Moving the data phase to 90° of the reference clock @ T_{bit}=1000 ps, TT corner

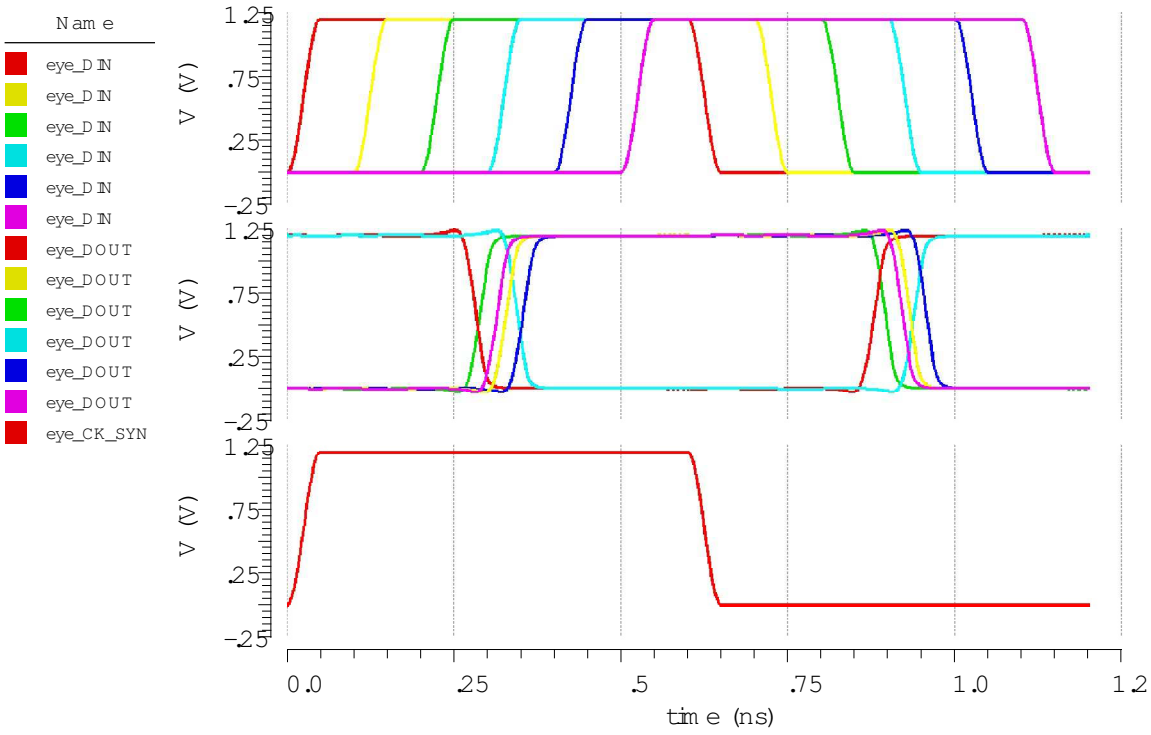

Figure 2.31: Moving the data phase to 90° of the reference clock @ T_{bit}=600 ps, SS corner

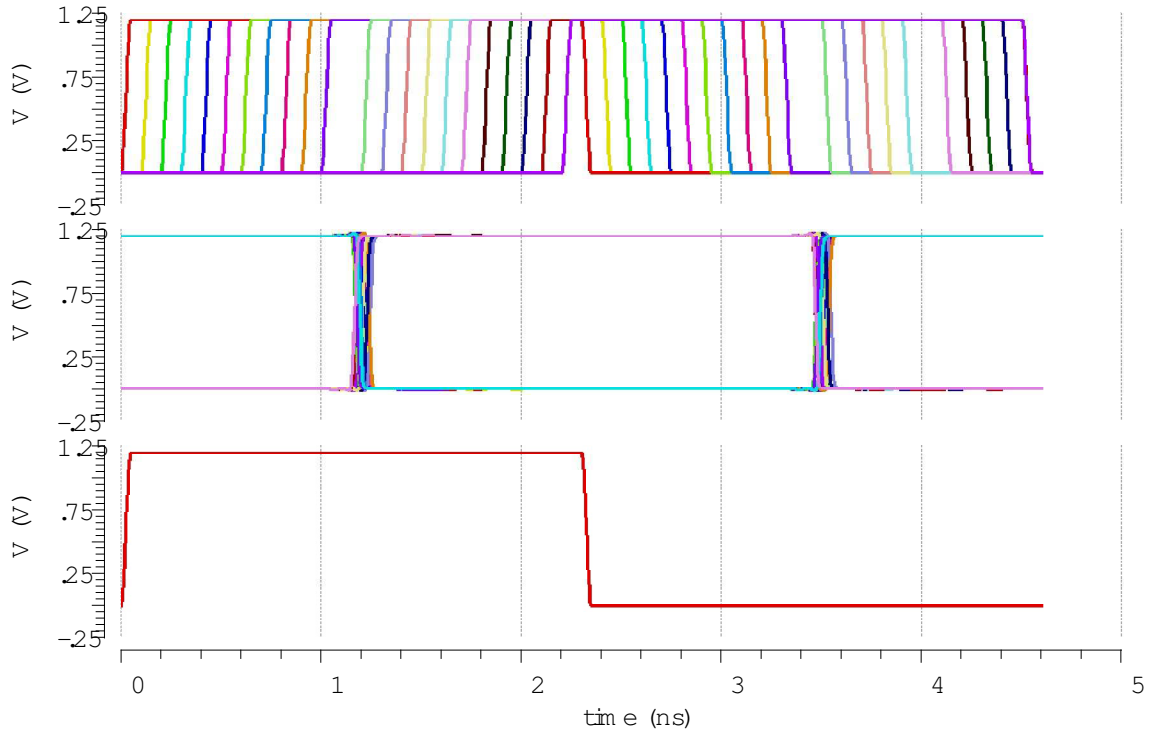

Figure 2.32: Moving the data phase to 90° of the reference clock @ T_{bit}=2300 ps, FF corner

3 LVDS Transceiver

The LVDS transceiver in this work functions as the high-speed external I/O buffer to communicate with Xilinx FPGAs. Therefore, the primary design requirement is to comply with the Xilinx FPGA I/O signaling specifications, which is shown in Table 3.1. Since this work is intended to be used on a small prototyping chip, small area and low power consumption are highly preferred.

In this chapter, the designed LVDS driver and receiver with their layout and simulation results will be discussed in detail. All the circuits are implemented in UMC 65 nm low-leakage technology, using both 2.5 V thick-gate-oxide devices and 1.2 V thin-gate-oxide devices. The entire design process is carried out in *Cadence Virtuoso Design System*, with DRC, LVS and parasitic extraction in *Mentor Graphics Calibre*. The simulations are performed in *Virtuoso Analog Design Environment* with *Spectre* simulator, and the results are plotted and analyzed in *Virtuoso Visualization & Analysis*.

	Virtex-6	Virtex-7 (HR Bank)	Virtex-7 (HP Bank)
Input differential voltage (mV)	350 (100 ∼ 600)	350 (100 ∼ 600)	350 (100 ∼ 600)
Input common-mode voltage (V)	1.2 (0.3 ∼ 2.2)	1.2 (0.3 ∼ 1.5)	1.2 (0.3 ∼ 1.425)
Output differential voltage (mV)	350 (247 ∼ 600)	350 (247 ∼ 600)	350 (247 ∼ 600)
Output common-mode voltage (V)	1.25 (1.075 ∼ 1.425)	1.25 (1 ∼ 1.425)	1.25 (1 ∼ 1.425)

Table 3.1: LVDS signaling specification of Xilinx Virtex-6 and Virtex-7 FPGAs [28] [29]

3.1 LVDS Driver

Topologies of LVDS driver can be mainly classified into bridge driver, double current source driver and switchable current source driver [9]. The characteristics of these types of LVDS driver are summarized in Table 3.2. According to the design requirements, the bridge driver topology is adopted.

3.1.1 H-bridge Based Driver

Figure 3.1 shows the circuit diagram of the designed LVDS driver. The current sources NM2 and PM2 provide a nominal constant current of 3.5 mA, flowing through the

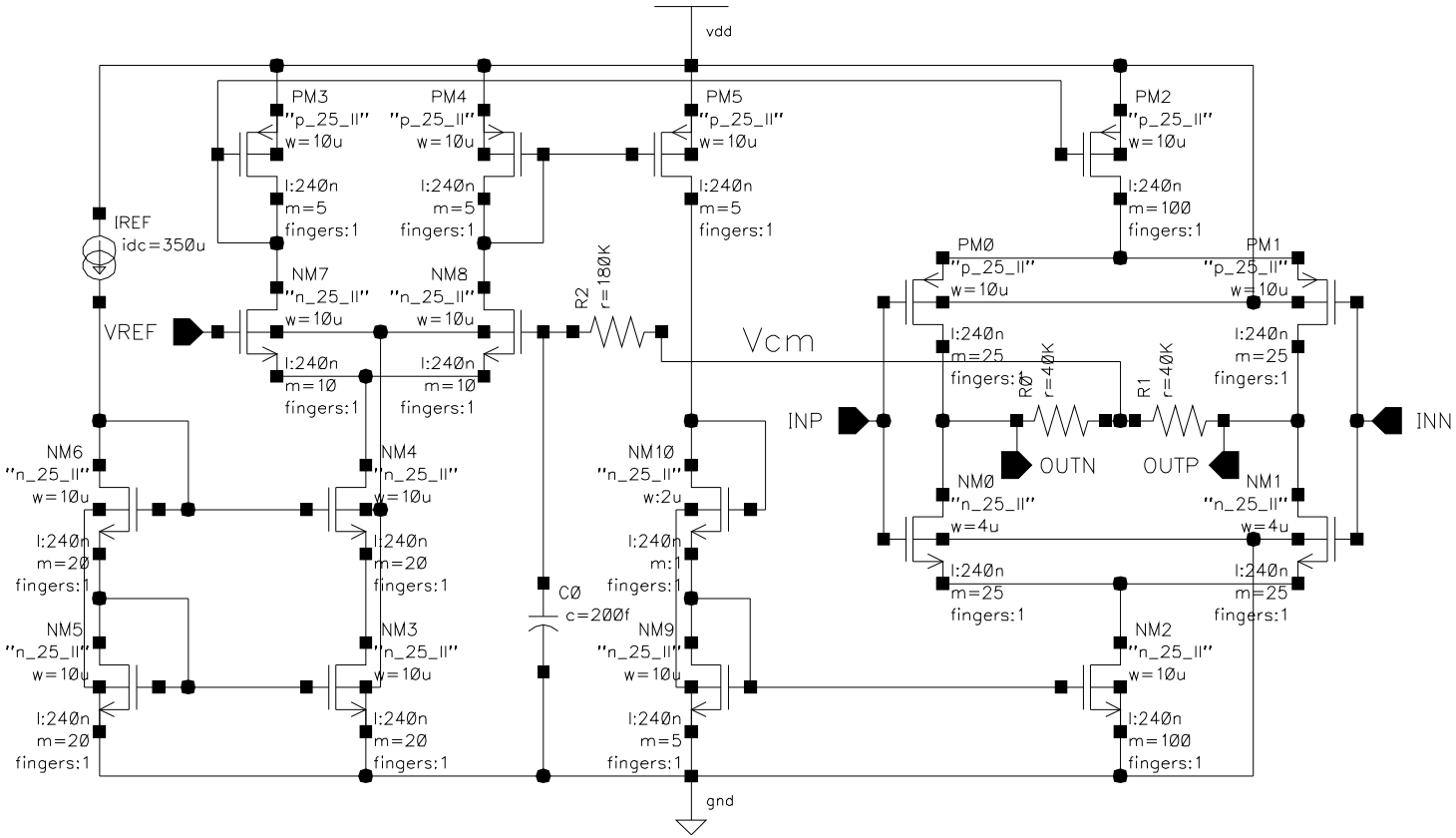

Figure 3.1: Circuit diagram of the LVDS driver

	Bridge	Double current source	Switchable current source
Static power	Low	High	Low
Area	Small	Large	Small
Circuit compelxity	Low	Low	High
Speed	High	Low	Low

Table 3.2: Comparison of different LVDS driver topologies

polarity-switching H-bridge driver formed by NM0, NM1, PM0 and PM1. With a nominal 100 Ω termination resistor between OUTP and OUTN, an output differential voltage of ±350 mV is produced. The input signals INP and INN have 0 V to 2.5 V signal swing, so that NM0, NM1, PM0 and PM1 are either cut-off or fully conducting.

At the output ports OUTN and OUTP, the common-mode voltage V_{cm} is sensed by a potential divider which consists of two 40 kΩ resistors, and is connected to the common-mode feed back (CMFB) circuit through a large RC filter to improve stability. The CMFB circuit is a differential amplifier biased with a 350 µA cascode current mirror formed by NM3∼NM6. The input to the differential amplifier are 1.25 V voltage reference and the V_{cm}. If V_{cm} equals 1.25 V, the current flowing through each differential pair will be equal to 175 µA, and are mirrored to NM2 and PM2 with a current gain of 20, which generate a nominal current of 3.5 mA. If V_{cm} deviates from 1.25 V, the differential amplifier will change the gate voltage of NM2 and PM2 until V_{cm} is close to 1.25 V.

Due to channel length modulation effect, the actual current flowing through NM2 and PM2 is larger than 3.5 mA. Simulation shows a reference current of around 300 µA leads to an output current of around 3.5 mA.

3.1.2 Pre-driver

The pre-driver shown in Figure 3.2 converts the single-phase input signal to a complementary signal, and shifts the signal swing from 1.25 V to 2.5 V. Through proper transistor sizing, the skew between the output signals OUTP and OUTN, as well as their transition time, are small enough for the H-bridge driver to work properly.

3.1.3 LVDS Driver Layout

Figure 3.3 shows the layout of the LVDS driver. Although the LVDS driver is constant current driver in theory, the pre-driver is still push-pull structure, and can inject switching noise into the power supply. To alleviate this problem, four large decoupling capacitors are inserted in the area near the H-bridge driver. Besides, large double guard rings are drawn around all the NMOS and PMOS transistors, to absorb any possible noise injected into the substrate and prevent latch-up.

3.1.4 LVDS Driver Simulation

Figure 3.4 shows the circuit diagram of the test bench of the LVDS driver. On the left side is the LVDS driver designed in this work, which includes parasitic resistance and

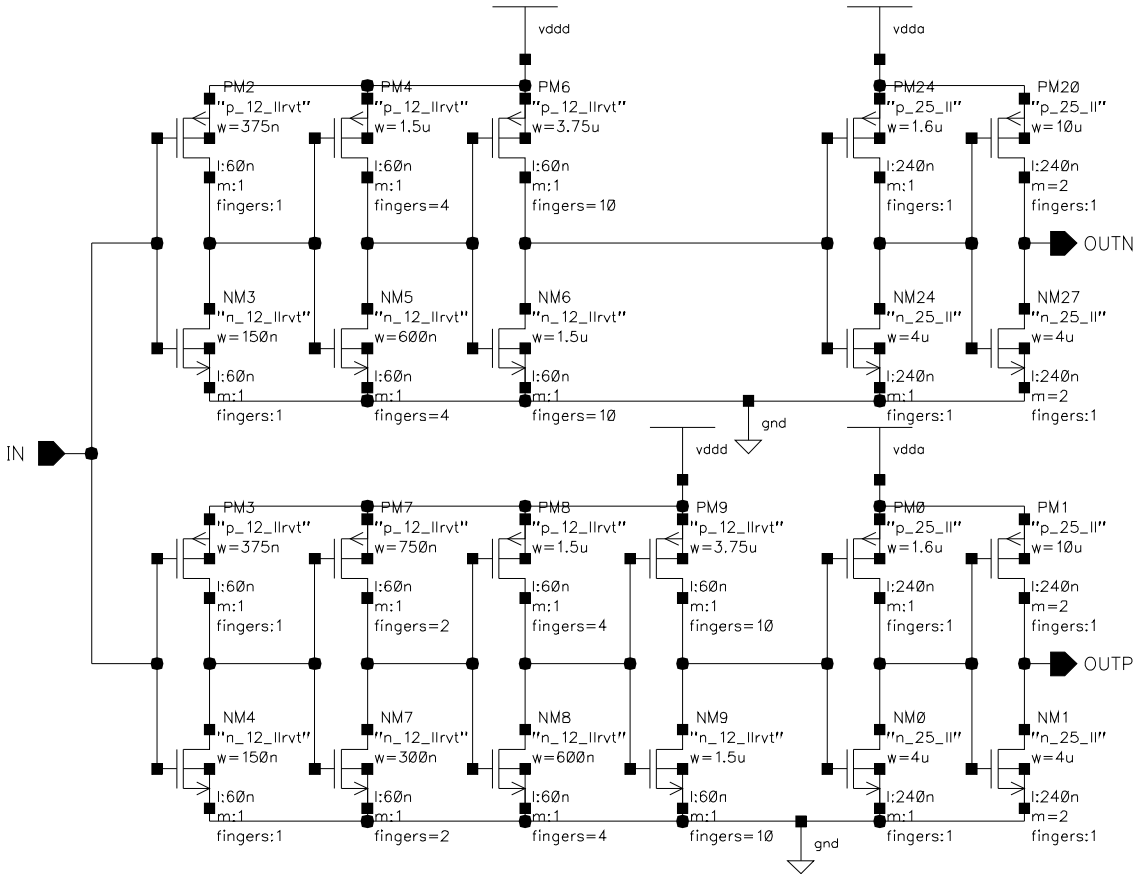

Figure 3.2: Circuit diagram of the LVDS pre-driver

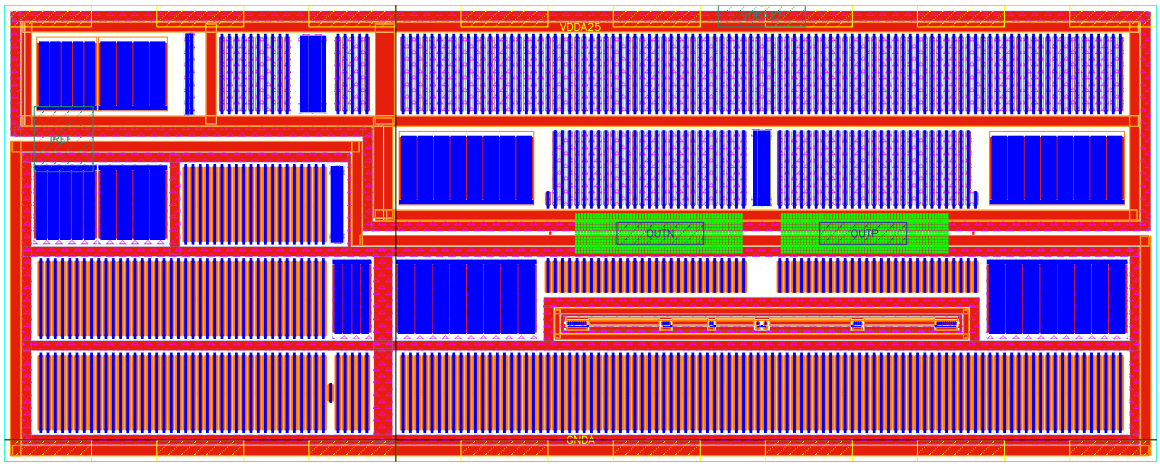

Figure 3.3: Layout of the LVDS driver

capacitance extracted from the layout. The following four diodes represent the ESD circuits in the standard I/O cells, and the parameters are chosen properly according to the data book of the UMC 65nm Low-Leakage standard I/O cell library. The FPGA

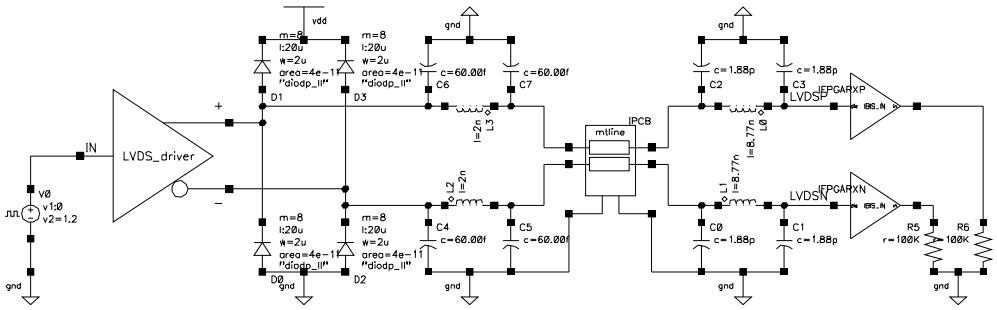

Figure 3.4: Test bench for the LVDS driver

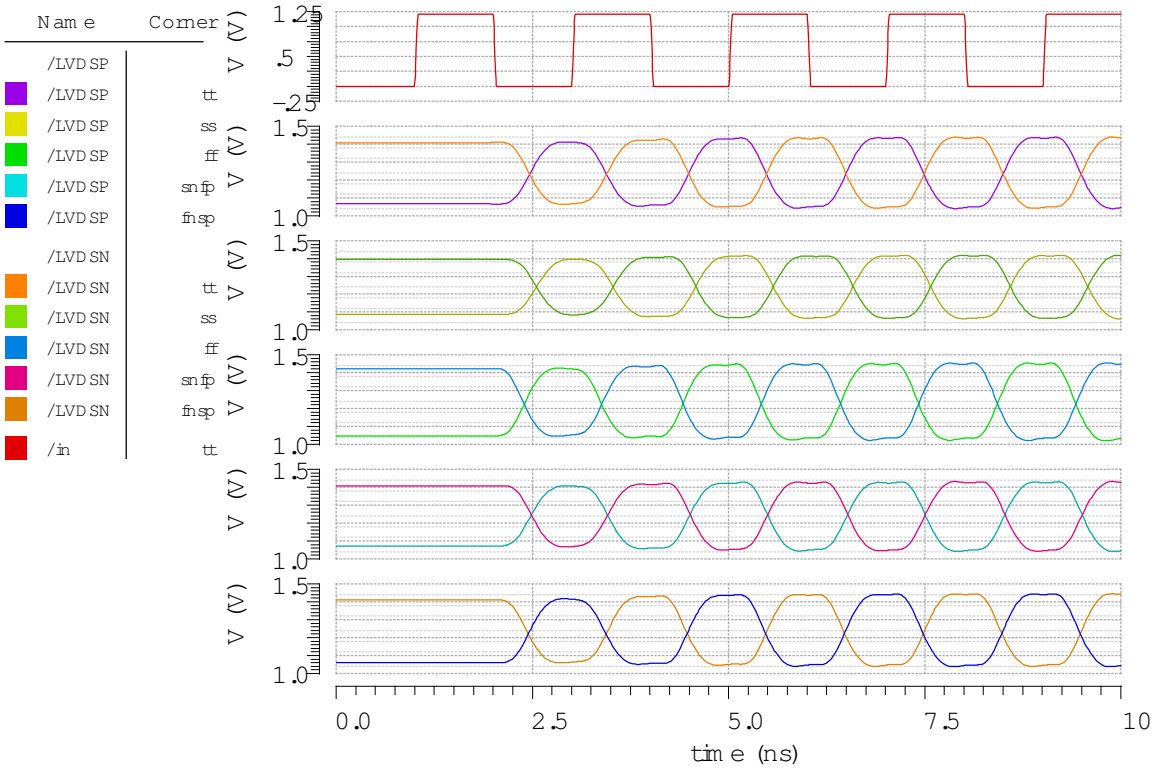

Figure 3.5: Transient simulation results of the LVDS driver @1Gbps

LVDS receiver is modeled using "Xilinx IBIS Models for VIRTEX7 IO" on the right side. The package parasitics are included using the π-model. The parameters of the FPGA package parasitics are obtained from the "Xilinx IBIS Models for VIRTEX7 IO", where the RLC value is 1.16 Ω, 8.77 nH, 3.77 pF. For the ASIC side where the designed LVDS driver is located, 2 mm-length 25 µm-diameter gold package bonding wires are assumed, which gives an empirical RLC value as 0.1 Ω, 2 nH, 0.12 pF [1]. The transmission line is modeled as a 100 Ω-impedance, 100 mm, copper PCB strip line.

Figure 3.5 shows the transient simulation results of the LVDS driver operating at 1Gbps. Figure 3.6 and Figure 3.7 plots the eye diagram at the FPGA LVDS receiver input and output respectively, with 200 PRBS bits at the input of the designed LVDS

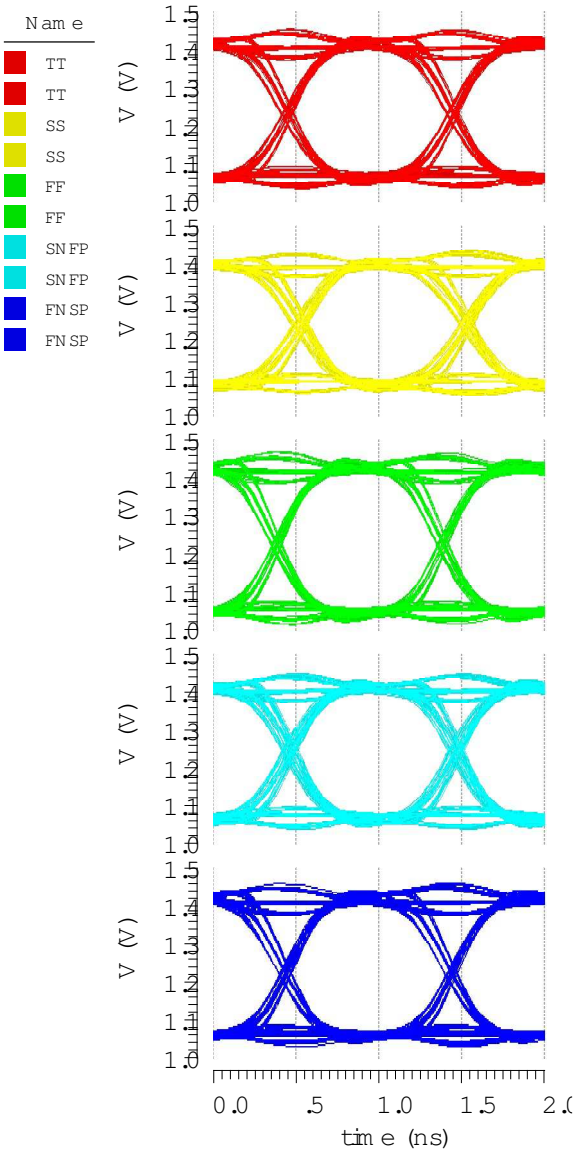

Figure 3.6: Eye diagram at the input of the FPGA receiver

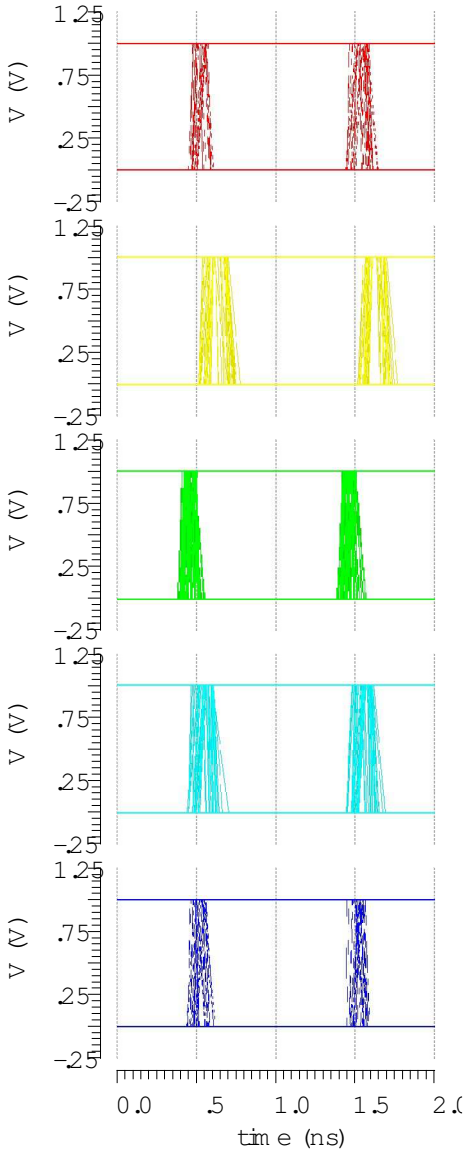

Figure 3.7: Eye diagram at the output of the FPGA receiver

driver at 1Gbps. It shows a sufficiently opened eye and exhibits small jitter, which allows the following stages to sample the data correctly.

Table 3.3 shows the output common mode voltage and differential voltage of the designed LVDS driver, as well as its output current and power consumption in all process corners. The common mode voltage and differential voltage are measured at the input of the FPGA receiver. As a conclusion, the designed LVDS driver complies with the Xilinx FPGA signaling specifications, and is able to transmit data properly at 1Gbps.

Corner	TT	SS	FF	SNFP	FNSP
V_{CM} (V)	1.238	1.24	1.234	1.239	1.235
V_{DF} (mV)	340.0	305.5	369.3	332.0	348.0
I_{OUT} (mA)	3.56	3.25	3.81	3.52	3.60
P_{total} (mW)	13.73	12.47	14.85	13.65	13.80

Table 3.3: LVDS driver output specifications and power consumption

3.2 LVDS Receiver

There are mainly two types of LVDS receivers: one is the conventional hysteresis comparator based receiver, and the other is the conventional hysteresis comparator with a folded cascode-input stage. The advantage of the second type of receiver is that it supports full LVDS common-mode input range, which is 0.1 V to 2.4 V [26], [6], [27], [14]. Since the LVDS receiver in this work is intended to interface with the Xilinx FPGA, a support of full-range common-mode input range is not necessary as shown in Table 3.1. Therefore, the conventional hysteresis comparator based LVDS receiver is adopted in this work to reduce area and power consumption.

3.2.1 Hysteresis Comparator Based Receiver

Figure 3.8 shows the circuit of the LVDS receiver, which is composed of a hysteresis stage and a post driver stage. An on-chip 100 Ω termination resistor is connected between the LVDS input port INP and INN, converting the input current into a differential voltage. The output of the hysteresis stage has a signal swing from around 0 V to 1.2 V, thus, the conventional level-down shifter is not necessary, and a buffer is followed instead. The buffer consists of two inverters, with the input one built with thick-gate-oxide transistors to avoid possible break down, and the output one built with thin-gate-oxide transistors to improve driving strength. Through proper transistor sizing, the duty cycle distortion and transition time of the output signal at OUTP and OUTN is sufficiently small. The layout of the LVDS receiver and biasing part in shown in Figure 3.9 and Figure 3.10.

3.2.2 LVDS Receiver Simulation

Figure 3.11 shows the test bench of the LVDS receiver. On the right side is the designed LVDS receiver in this work, which includes parasitic resistance and capacitance extracted from the layout. The FPGA LVDS driver is modeled using "Xilinx IBIS Models for VIRTEX7 IO" on the left side. The package parasitics and transmission line are the same as in the LVDS driver test bench.

Figure 3.12 shows the transient simulation result of the LVDS receiver in all process corners, with real LVDS input signals working at 1Gbps. The waveforms show a decent output signals which have 0 V to 1.2 V signal swing, and sufficiently fast signal transition time with load capacitance equal to a 4-time sized inverter.

Figure 3.13 shows the duty cycle of the LVDS receiver output waveform, with an ideal 50%-duty-cycle, 500MHz clock signal at the input of the FPGA LVDS driver .

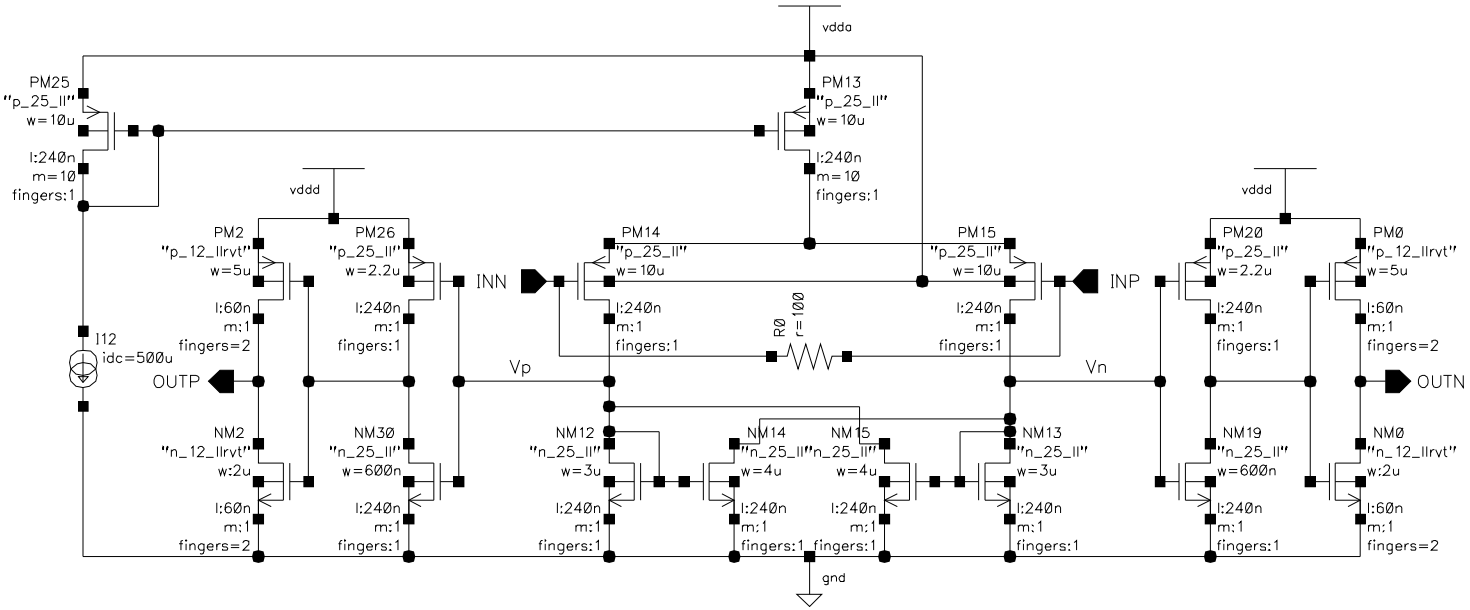

Figure 3.8: Circuit diagram of the LVDS receiver

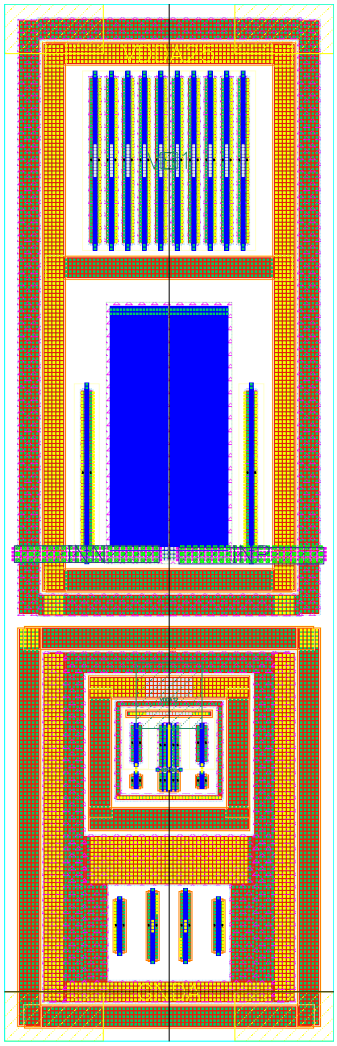

Figure 3.9: Layout of the LVDS receiver

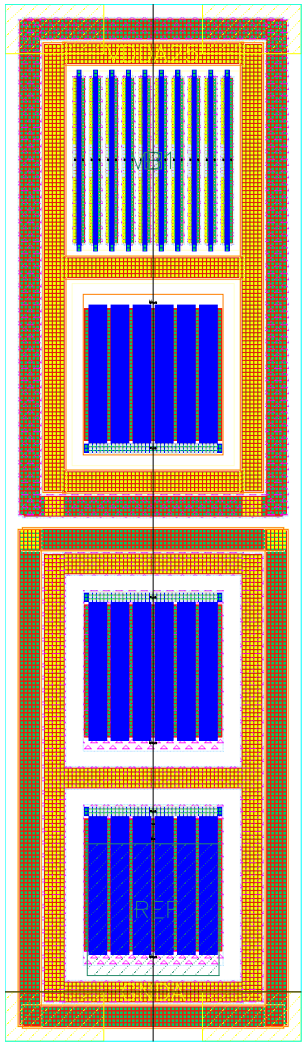

Figure 3.10: Layout of the LVDS receiver biasing

Therefore, the duty cycle performance reported includes the effect of external transmission channels, and shows a result between 49% to 53% in all process corners, which is sufficient to receive data properly.

Figure 3.14 plots the eye diagram at the LVDS receiver output in all process corners with 200 PRBS bits at the input of the FPGA LVDS driver at 1Gbps. It shows a sufficiently opened eye and exhibits small jitter, which allows the following stages to sample the data correctly.

Table 3.4 summaries the power consumption of the LVDS receiver working at 1Gbps in all process corners. The biasing part is included. A DC sweep shows the designed LVDS receiver has a hysteresis voltage of around 70 mV, which can resist differential input noise below 70 mV.

As a conclusion, the designed LVDS receiver complies with the Xilinx FPGA signaling specifications, and is able to receive data properly at 1Gbps.

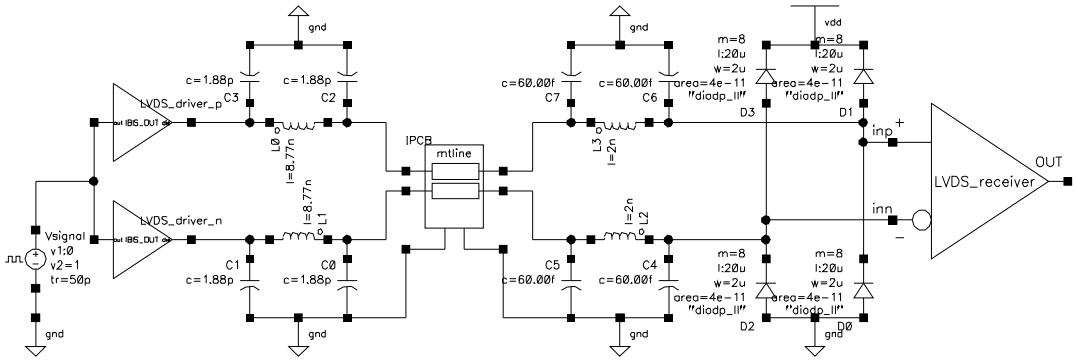

Figure 3.11: Circuit diagram of the LVDS receiver test bench

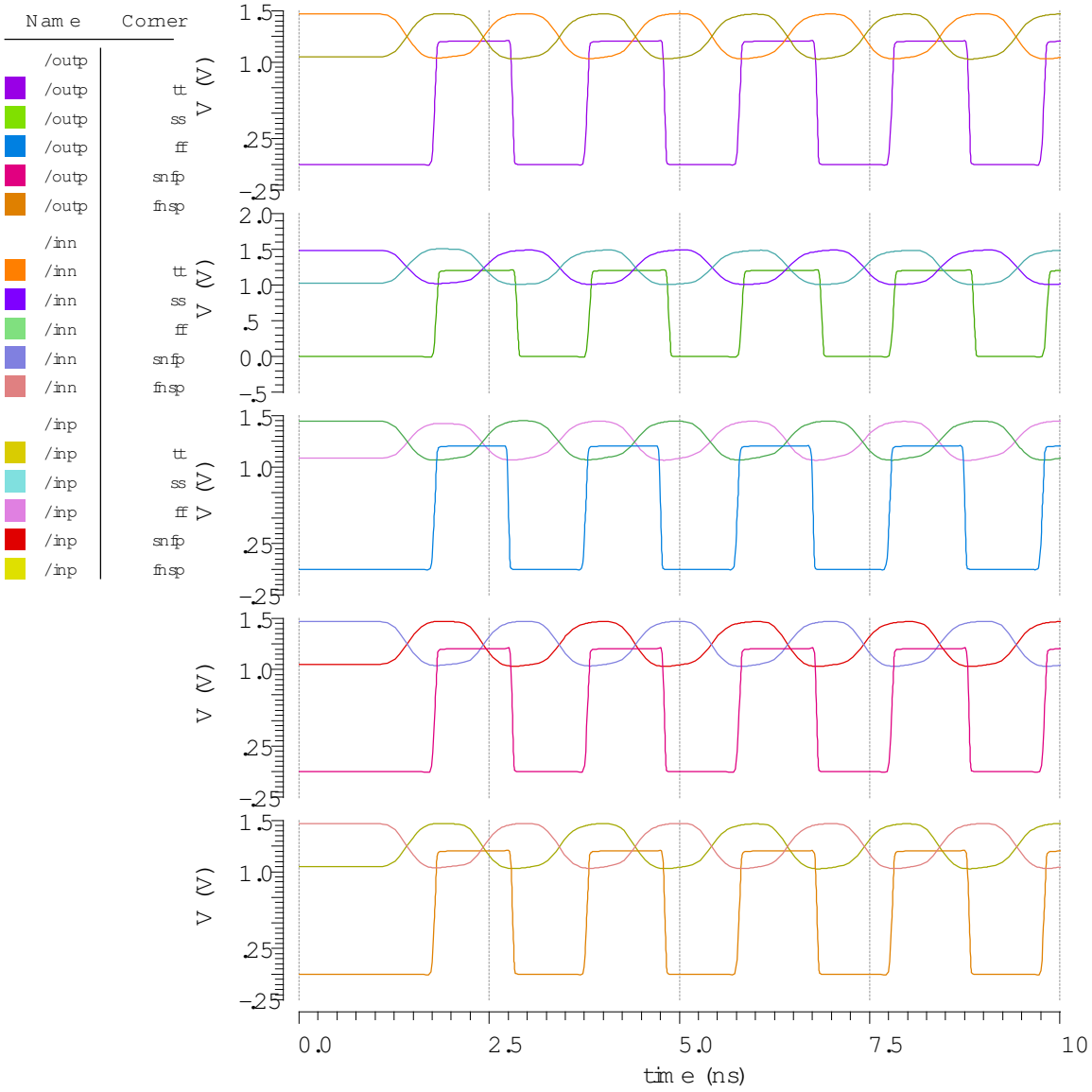

Figure 3.12: Transient simulation results of the LVDS receiver in all process corners @1Gbps

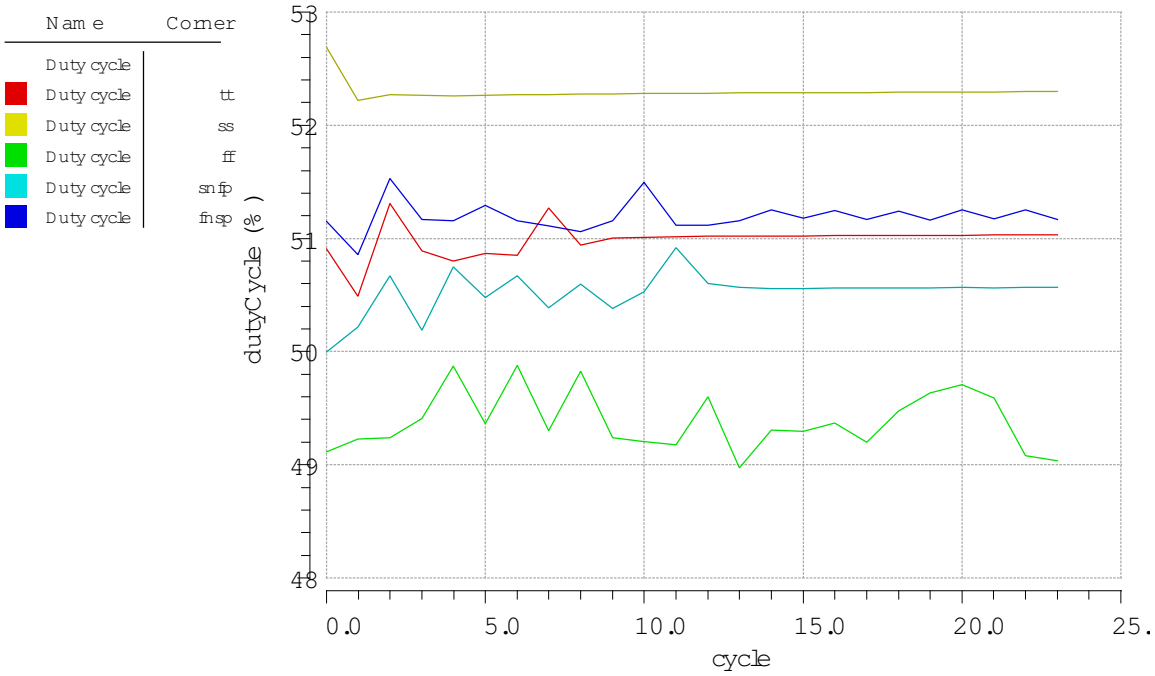

Figure 3.13: Duty cycle of the receiver output waveform in all process corners @1Gbps

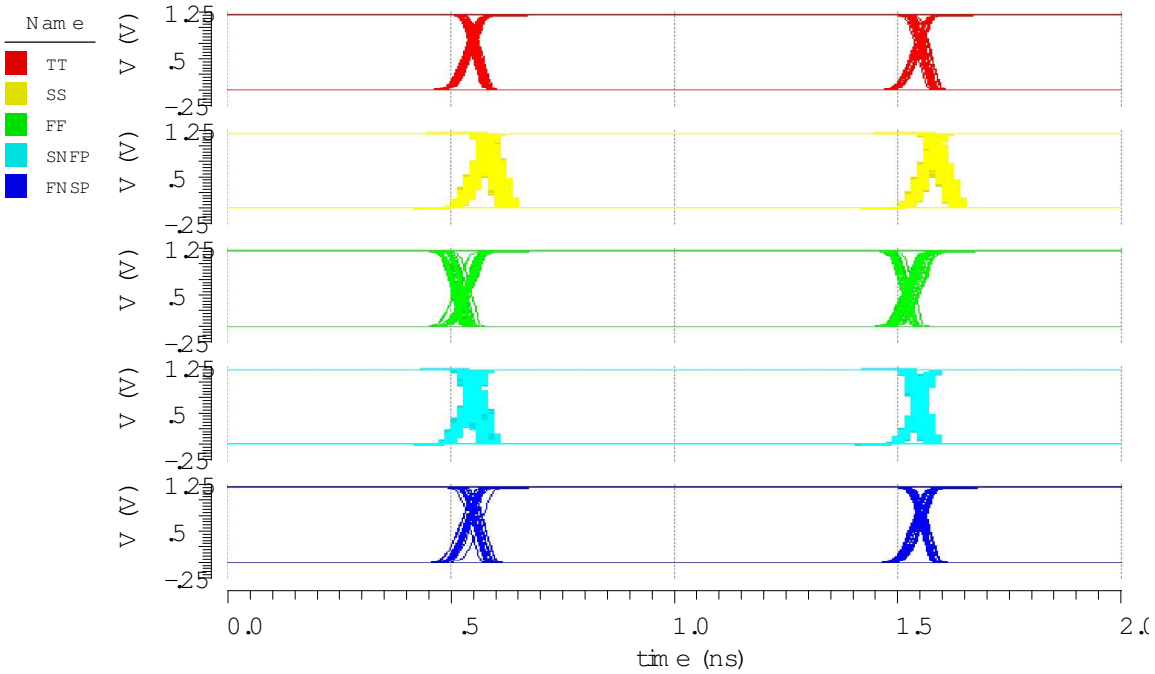

Figure 3.14: Eye diagram of the LVDS receiver output in all process corners @1Gbps

Corner	TT	SS	FF	SNFP	FNSP
P_{total} (mW)	2.37	2.29	2.44	2.40	2.34

Table 3.4: Power consumption of the LVDS receiver in all process corners

Top-level Design and Simulation

4

This chapter discusses the top-level design and simulation of the source-synchrnous interface.

4.1 Top-level Design

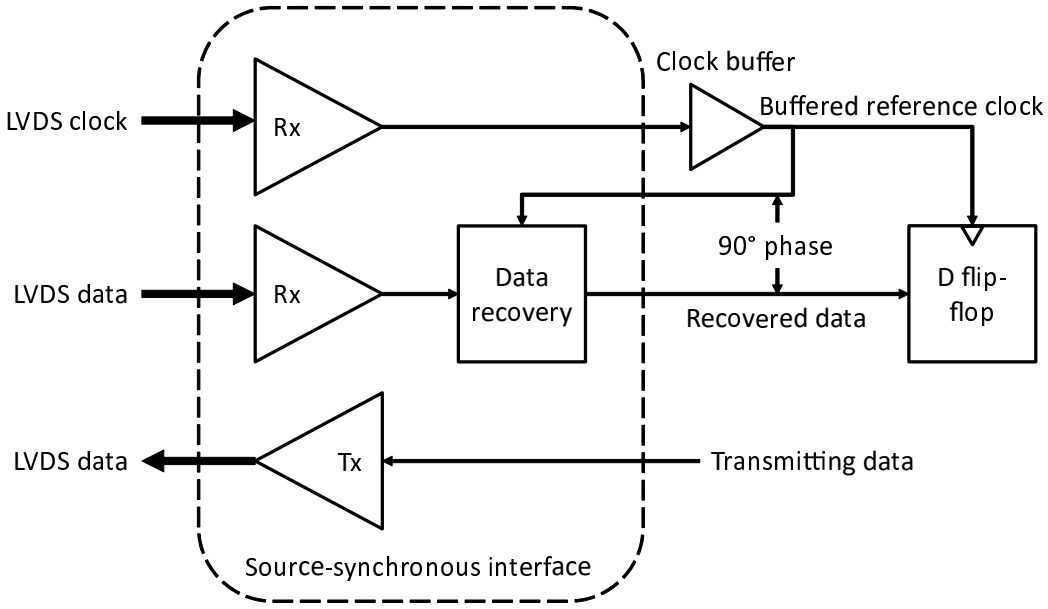

Figure 4.1: Block diagram of the top-level interface

Figure 4.1 shows the block diagram of the top-level interface, which contains an LVDS driver, two LVDS receivers and a data recovery block. One of the LVDS receiver is used for half-rate reference clock reception, and the other used for data reception. The LVDS receivers convert LVDS signals to standard digital signals which have 0 V to 1.2 V signal swing. The received clock signal first goes through the external clock buffers, and then return to the data recovery as a half-rate reference clock. The received data signal is phase-calibrated in the data recovery, aligning the 90° phase of the reference clock. In other words, the edges of the reference clock are located at the center of the recovered data eye, which can ensure a correct data sampling by the following D flip-flop using the same reference clock. The LVDS driver converts the standard digital signal to LVDS signal without any timing manipulation, as the data will be recovered by the receiver at the other side of the communication link.

Like most other source-synchronous interfaces, the LVDS half-rate reference clock must be generated from the same clock source as for the LVDS data, as any frequency

shift in the data is also reflected in the reference clock. This ensures the jitter and skew of the received data is finite relative to the reference clock, such that the data recovery is able find a correct phase for the recovered data.

Figure 4.2 shows the layout of the top-level interface. The top-level layout is designed to fit the top-level chip floor plan, resulting blank spaces between active components, which are filled by decoupling capacitors. The active components from left to the right are LVDS data receiver, LVDS receiver biasing block, data recovery block, LVDS clock receiver, and LVDS driver. A large double guarding (5 µm for both N+ and P+ diffusion) is made around the whole block to absorb any possible substrate noise and prevent latch-up. The total area is 688 µm × 87 µm. Table 4.1 summarizes the area of the designed source-synchronous interface.

Part	Area
Data recovery	9.891 μm^2
LVDS receiver	1.260 μm^2 x2
LVDS receiver biasing	1.134 μm^2
LVDS driver	10.017 μm^2
Total active area	23.562 μm^2
Total layout area	59.856 μm^2

Table 4.1: Area summary of the source-synchronous interface

4.2 Top-level Simulation

The test bench for the top-level interface is made by combining the test bench of the data recovery, the LVDS driver (Figure 3.4) and LVDS receiver (Figure 3.11), and is omitted here. The test conditions for typical case and worst maximum bit time case are the same as in the test bench for data recovery. While for worst minimum bit time case, a bit time of 800 ps at SS corner is used, as simulations have shown that the Xilinx FPGA cannot provide a decent waveform at data rates higher than that, which causes difficulty for the data recovery to lock. The simulation results are all plotted as eye diagram after the data recovery has locked.

The total power consumption of the designed source-synchronous interface is 16.74 mW during calibration process of the data recovery, and 16.06 mW after locking and transmitting PRBS at 1Gbps, measured at TT corner. Simulation results prove that the designed source-synchronous interface is able to transmit and receive data without error within data rate range from 421.4Mbps to 1.25Gbps in all process corners.

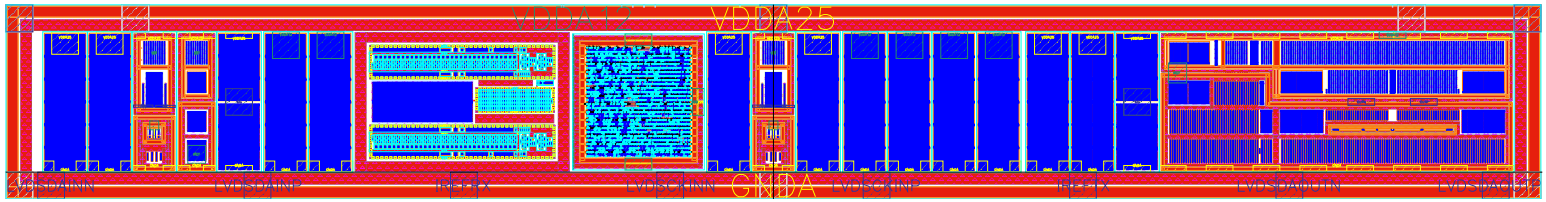

Figure 4.2: Top-level layout (688 µm × 87 µm) (all metal layers are hidden, execept vias and pins)

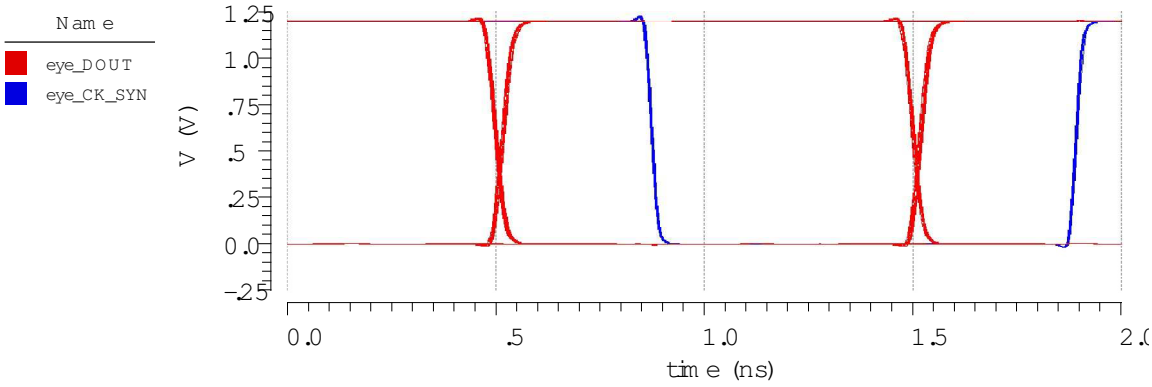

Figure 4.3: Eye diagram of recovered data with reference clock @1Gbps, TT corner

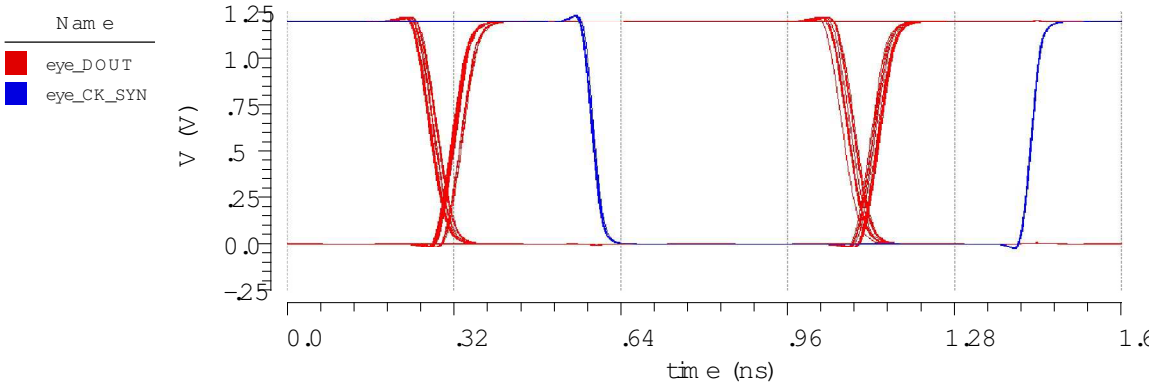

Figure 4.4: Eye diagram of recovered data with reference clock @1.25Gbps, SS corner

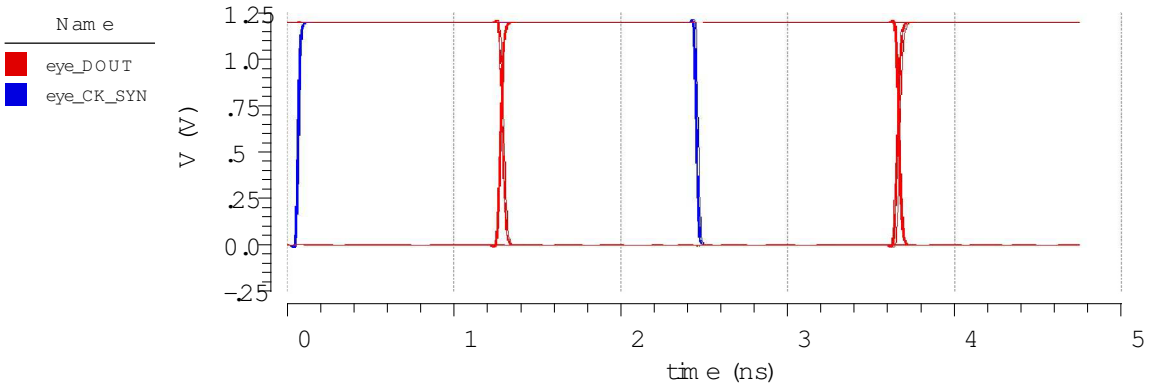

Figure 4.5: Eye diagram of recovered data with reference clock @421.4Mbps, FF corner

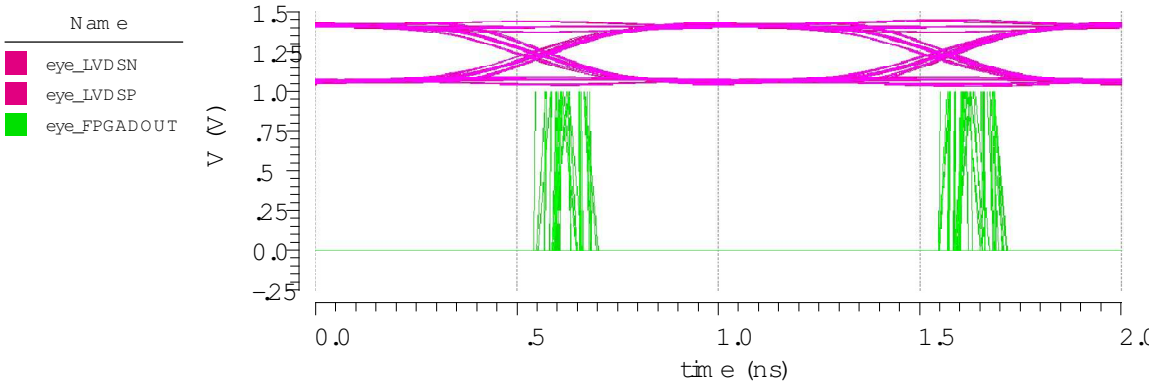

Figure 4.6: Eye diagram at FPGA LVDS receiver input and output @1Gbps, TT corner

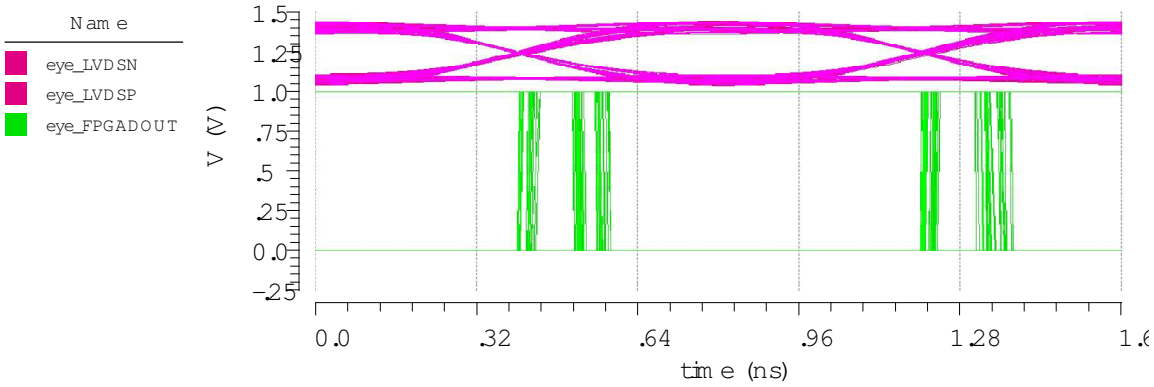

Figure 4.7: Eye diagram at FPGA LVDS receiver input and output @1.25Gbps, SS corner

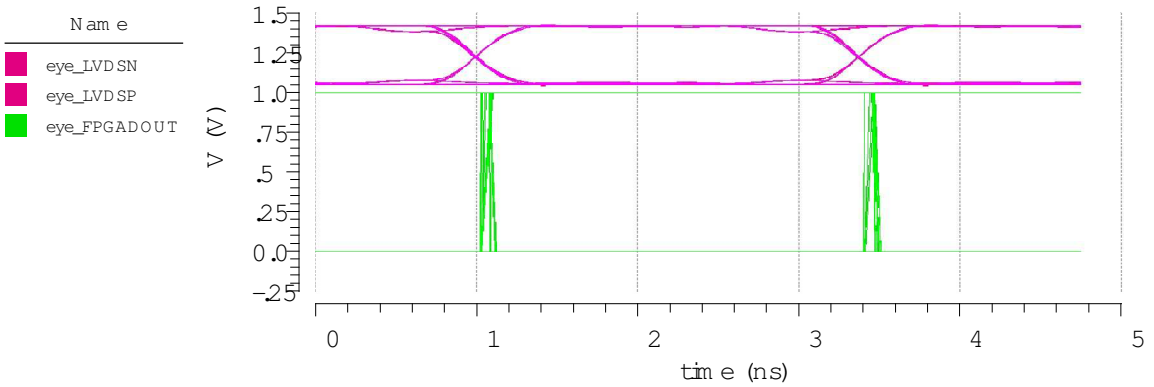

Figure 4.8: Eye diagram at FPGA LVDS receiver input and output @421.4Mbps, FF corner

5 Conclusion and Future Work

5.1 Conclusion

A source-synchronous interface with all-digital data recovery for inter-chip communication has been designed and verified to work properly in this thesis project. The interface is composed of LVDS transceivers and an all-digital data recovery.

The LVDS transceivers function as external I/O buffers for the interface, and is designed to comply with the Xilinx FPGA LVDS specifications. Post-layout simulations are performed with FPGA IBIS I/O model, package parasitics and lossy transmission line included. The LVDS driver is able to generate a sufficiently opened eye at the FPGA receiver input, and the LVDS receiver is able to receive the transmitted data without errors at 1Gbps in all process corners.

The all-digital data recovery is used to calibrate the received data phase to be aligned to the 90° phase of the received half-rate reference clock for error free data sampling. It uses TDC to measure the phase relationship between the data and clock, then converts the phase information into delay control in the FSM, and delays the data using a DCDL. Post-layout simulation results show that the proposed data recovery is able to work with bit rate from 421Mbps to 1.79Gbps in all process corners. It locks in around 24 clock cycles with a maximum phase error of 224 ps. The data recovery occupies 9891 μm^2 silicon area, and consumes 1.27 mW power during calibration process.

This thesis has proposed a novel data recovery scheme which in principle can be used in any source-synchronous interfaces. Conventional CDRs generate or delay the clock signal which use the received data as a phase reference to sample the received data. Instead, the data recovery proposed in this thesis delays the received data and uses the received reference clock as a phase reference to sample the received data, which eliminates the need of any global or local PLL or DLL and saves a considerable amount of silicon area and power consumption. The recovered data exhibits a small jitter because of the fixed delay line which is controlled digitally, and has a phase error well within acceptable range. As a conclusion, the proposed data recovery provides a low-cost efficient solution to error free data sampling for source-synchronous inter-chip communication, which is specially suitable for cost-constraint applications.

Integrating the data recovery with LVDS transceivers, the complete source-synchronous interface in this work occupies total area of 688 µm × 87 µm, including decoupling capacitors. Active components take 39.4% of the total layout area. The total power consumption is 16.74 mW during calibration process of the data recovery, and 16.06 mW after locking and transmitting PRBS at 1Gbps, measured at TT corner. The LVDS transceivers take 92.4% of the total power consumption, and the data recovery takes 7.6% of that.

Due to the limitation of the Xilinx FPGA, the maximum operating speed of the

entire interface is measured to be at around 1.25Gbps. Given an FPGA with higher performance, the theoretical maximum operating speed, which is defined by the data recovery, can be achieved as high as 1.79Gbps.

5.2 Future Work

Two improvement aspects can be investigated as future work: increasing the operating speed and extending the operating range. For the data recovery part, the following improvements can be carried out:

- Improve the resolution of the TDC and DCDL to increase the operating speed;
- Use coarse and fine structure for the DCDL and TDC to extend the operating range of the data recovery.
- Improve the edge detector of the data recovery to minimize the "blind region.

Besides, as the bit rate goes up to several gigabits per second, the I/O buffer of the interface, i.e. LVDS transceiver in this work, also needs improvements to keep up with the speed of the data recovery, for example:

- Use signal conditioning techniques, such as pre-emphasis and equalization;
- The LVDS has a theoretical maximum bit rate of 3.125Gbps [25]; for higher bit rates, other signaling standards can be considered, for example current-mode logic (CML).
- Add a duty cycle corrector in the receiver, as duty cycle distortion also causes bottleneck in DDR communication.

Multiple receiver and transmitter channels can be easily built by duplicating relevant blocks. Figure 5.1 shows the block diagram of the interface with multiple receiver channels included. The data recovery in each channel can share a single reference clock as long as the data are all transmitted with the same clock.

5.3 Comparison with State-of-the-art

This section compares the data recovery in this work with similar state-of-the-art works.

* Off-chip components needed.

Table 5.1: Comparison with state-of-the-art

Reference	**This work**	**Bae, CAS 2016** [5]	**Chung, CAS 2016** [8]	**Chen, VLSI 2016** [7]	**Park, IS-CAS 2015** [21]	**Ma, IMS 2015** [18]	**Kim, CAS 2015** [15]	**Lee, VLSI 2013** [16]	**Huang, MWS-CAS 2012** [13]
Technology	65 nm	65 nm	65 nm	28 nm	65 nm	65 nm	65 nm	130 nm	65 nm
Architecture	Data phase calibration	DLL	ILO	Multi-phase DLL	PLL	ILO+PI	ILO	DLL, input data delay	Phase interpolator
Data rate	0.4 $\sim$ 1.67 Gbps	12.5 Gbps	10 Gbps	3.2 $\sim$ 6.4 Gbps	10 Gbps	9.8 Gbps	9.6 Gbps	3.2 Gbps	10 Gbps
Jitter-tracking bandwidth	All frequency	-	All frequency	500 kHz	14.4 MHz	-	120 $\sim$ 600 MHz	-	2 MHz
Jitter tolerance	-	1.4 UIpp 300 MHz	0.7 UIpp 1.5 GHz	-	0.2 UIpp 14.4 MHz	-	0.22 UIpp 100$\sim$500 MHz	-	-
Area per receiver	0.0099 mm^2	0.025 mm^2	0.014 mm^2	0.02 mm^2	0.014 mm^2	0.16 mm^2 *	0.0162 mm^2	0.039 mm^2	-
Energy efficiency	0.14 pJ/bit	0.36 pJ/bit	0.71 pJ/bit	1.2 pJ/bit	2.21 pJ/bit	0.67 pJ/bit *	0.96 pJ/bit	4.6 pJ/bit	2.5 pJ/bit
Locking time	24 cycles @1Gbps (48ns)	-	-	-	-	-	-	32 cycles @400MHz	-

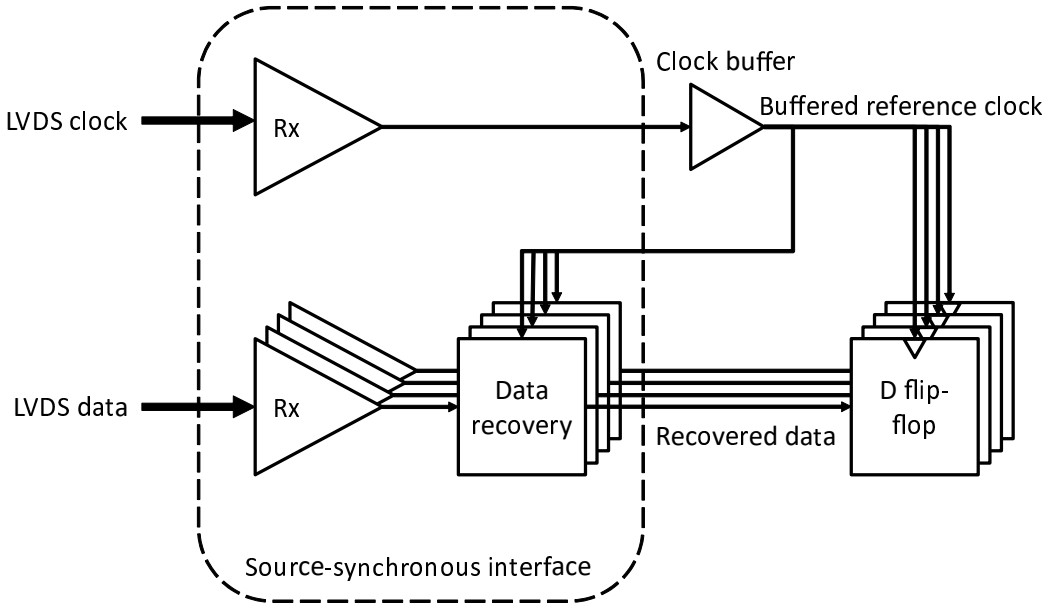

Figure 5.1: Block diagram of the interface with multiple receiver channels

Bibliography

[1] Csp leadless qfn. http://www.asetwn.com.tw/content/2-4-4-1.html. Accessed: 2017-05-28.

[2] International technology roadmap for semiconductors. http://www.itrs2.net/itrs-reports.html.

[3] Bilal I. Abdulrazzaq, Izhal Abdul Halin, Shoji Kawahito, Roslina M. Sidek, Suhaidi Shafie, and Nurul Amziah Md. Yunus. A review on high-resolution cmos delay lines: towards sub-picosecond jitter performance. *SpringerPlus*, 5(1):434, 2016.

[4] Abhijit Athavale and Carl Christensen. High-speed serial i/o made simple, a designers guide with fpga, April 2005.

[5] W. Bae, G. S. Jeong, K. Park, S. Y. Cho, Y. Kim, and D. K. Jeong. A 0.36 pj/bit, 0.025 mm^2, 12.5 gb/s forwarded-clock receiver with a stuck-free delay-locked loop and a half-bit delay line in 65-nm cmos technology. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(9):1393–1403, Sept 2016.

[6] A. Boni, A. Pierazzi, and D. Vecchi. Lvds i/o interface for gb/s-per-pin operation in 0.35- mu;m cmos. *IEEE Journal of Solid-State Circuits*, 36(4):706–711, Apr 2001.

[7] S. Chen, H. Li, and P. Y. Chiang. A robust energy/area-efficient forwarded-clock receiver with all-digital clock and data recovery in 28-nm cmos for high-density interconnects. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(2):578–586, Feb 2016.

[8] S. H. Chung, Y. J. Kim, Y. H. Kim, and L. S. Kim. A 10-gb/s 0.71-pj/bit forwarded-clock receiver tolerant to high-frequency jitter in 65-nm cmos. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(3):264–268, March 2016.

[9] H. S. Gupta, R. Parmar, and R. Dave. High speed lvds driver for serdes. In *2009 International Conference on Emerging Trends in Electronic and Photonic Devices Systems*, pages 92–95, Dec 2009.

[10] Stephan Henzler. *Time-to-Digital Converter Basics*, pages 5–18. Springer Netherlands, Dordrecht, 2010.

[11] S. Hu, C. Jia, K. Huang, C. Zhang, X. Zheng, and Z. Wang. A 10gbps cdr based on phase interpolator for source synchronous receiver in 65nm cmos. In *2012 IEEE International Symposium on Circuits and Systems*, pages 309–312, May 2012.

[12] K. Huang, Z. Wang, X. Zheng, X. Ma, K. Yu, C. Zhang, and Z. Wang. A novel clock and data recovery scheme for 10gbps source synchronous receiver in 65nm cmos. In *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 932–935, Aug 2012.

[13] K. Huang, Z. Wang, X. Zheng, X. Ma, K. Yu, C. Zhang, and Z. Wang. A novel clock and data recovery scheme for 10gbps source synchronous receiver in 65nm cmos. In *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 932–935, Aug 2012.

[14] Xu Jian, Wang Zhigong, and Niu Xiaokang. Design of high speed lvds transceiver ics. *Journal of Semiconductors*, 31(7):075014, 2010.

[15] Y. J. Kim, S. H. Chung, K. S. Ha, S. J. Bae, and L. S. Kim. A 9.6 gb/s 0.96 mw/gb/s forwarded clock receiver with high jitter tolerance using mixing cell integrated injection-locked oscillator. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(10):2495–2503, Oct 2015.

[16] J. W. Lee, H. J. Kim, C. S. Jeong, J. J. Lee, and C. Yoo. Skew compensation technique for source-synchronous parallel dram interface. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(11):2155–2159, Nov 2013.

[17] Y. Liu, Y. Wang, S. Jia, and X. Zhang. 180.5mbps-8gbps dll-based clock and data recovery circuit with low jitter performance. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1394–1397, May 2015.

[18] S. Ma, S. Manoj, H. Yu, J. Ren, and R. Weerasekera. A 9.8 gbps, 6.5 mw forwarded-clock receiver with phase interpolator and equalized current sampler in 65 nm cmos. In *2015 IEEE MTT-S International Microwave Symposium*, pages 1–4, May 2015.

[19] Peter McCrorie and Inc. Mladen Nizic, Cadence Design Systems. Solutions for mixed-signal ip, ic, and soc implementation. 2012.

[20] T. Nakao, Y. Hidaka, S. Sakabayashi, T. Hashida, Y. Tomita, Y. Koyanagi, and H. Tamura. An equalizer-adaptation logic for a 25-gb/s wireline receiver in 28-nm cmos. In *2013 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 217–220, Nov 2013.

[21] K. Park, W. Bae, H. Ju, J. Lee, G. S. Jeong, Y. Kim, and D. K. Jeong. A 10 gb/s hybrid pll-based forwarded clock receiver in 65-nm cmos. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2389–2392, May 2015.

[22] B. Razavi. Challenges in the design high-speed clock and data recovery circuits. *IEEE Communications Magazine*, 40(8):94–101, Aug 2002.

[23] L. Smith. Simultaneous switch noise and power plane bounce for cmos technology. In *IEEE 8th Topical Meeting on Electrical Performance of Electronic Packaging (Cat. No.99TH8412)*, pages 163–166, 1999.

[24] M. t. Hsieh and G. E. Sobelman. Architectures for multi-gigabit wire-linked clock and data recovery. *IEEE Circuits and Systems Magazine*, 8(4):45–57, Fourth 2008.

[25] T. Instruments. *LVDS Owners Manual*, fourth edition edition, 2008.

[26] TIA/EIA-644-A. *ELECTRICAL CHARACTERISTICS OF LOW VOLTAGE DIFFERENTIAL SIGNALING (LVDS) INTERFACE CIRCUITS*, 1.2 edition, May 2000.

[27] K. w. Yoo and J. B. Kim. Design of a high-speed lvds i/o interface using telescopic amplifier. In *2006 8th International Conference on Solid-State and Integrated Circuit Technology Proceedings*, pages 1987–1989, Oct 2006.

[28] Xilinx. *Virtex-6 FPGA Data Sheet: DC and Switching Characteristics*, v3.6 edition, March 18 2014.

[29] Xilinx. *Virtex-7 T and XT FPGAs Data Sheet: DC and AC Switching Characteristics*, v1.26 edition, March 28 2016.

Mixed-signal Design Flow

A

This work is carried out using both "schematic-driven" design flow and 'netlist-driven" design flow for mixed-signal designs [19], as shown in Figure A.1.

In the "top-level floorplaning" step, the design ideas are translated to an initial plan of the top-level layout, which defines the location and size of each block, as well as the size of the entire design. In the "custom block implementation" (using "schematic-driven" design flow) and "digital block implementation" (using "netlist-driven" design flow) step, the detailed design of each block takes place. In the "top-level implementation" step, the top-level layout of the entire design is carried out by instantiating the sub-blocks implemented in the previous step. The "custom block implementation" and "digital block implementation" step can also be hierarchical, which means each block, as well as their sub-blocks, can be designed using the same flow mentioned above.

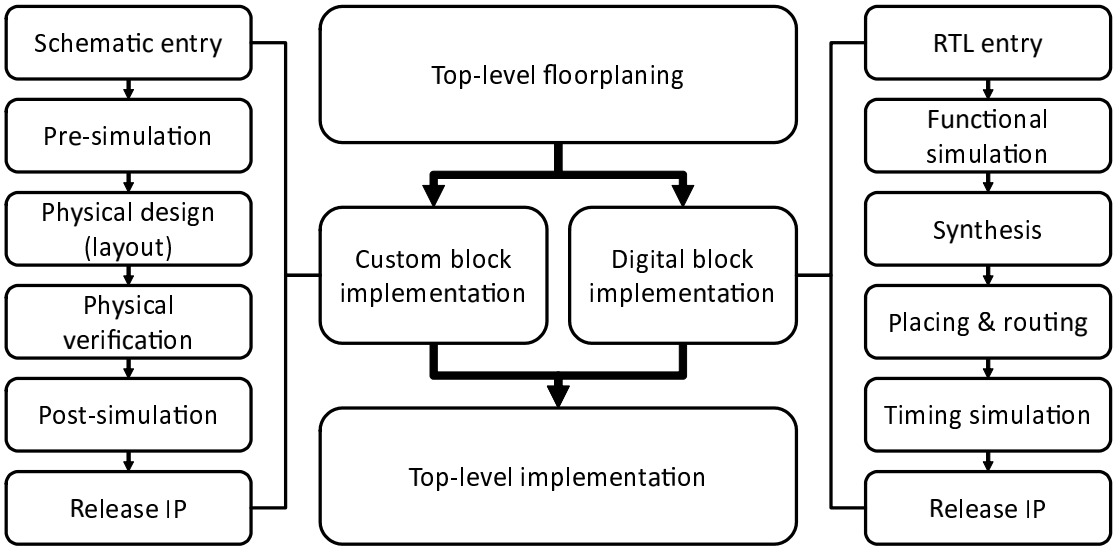

Figure A.1: Design flow of this work

A.1 Schematic-driven Design Flow

The "schematic-driven" design flow is carried out by hand, thus no script or data files are involved. In this work, all the custom designed blocks, i.e. LVDS transceiver, data recovery with its sub-blocks are implemented in *Cadence Virtuoso Design System* (IC 6.15) at transistor-level. The technology library used is "UMC 65 nm Low K Logic/Mixed Mode Low Leakage Processes for OpenAccess". 2.5 V thick-gate-oxide devices and 1P8M1T0F1U metalization option are used. The schematic of each circuit with its

testbench is designed using *Virtuoso Schematic Editor*. After completing the schematic of each circuit and its testbench, the simulation can be performed in *Virtuoso Analog Design Environment*. Before starting the simulation, the simulation environment needs to be set up, including setting the simulator to "Spectre", setting technology libraries and process corners, adding tests (DC, AC, transient, etc.), and adding signals to be observed, etc. After that, the layout of the circuit can be designed in *Virtuoso Layout Suit*. Then physical verification, which includes "design rules check (DRC)", "layout versus schematic (LVS)", and "parasitic extraction (PEX)", can be performed in *Calibre*. The outcome of the PEX is a schematic with all parasitic components included in "Calibre" view, which can replace the original circuit in the testbench and be used in post-layout simulation. The post-layout simulation is the same as pre-layout simulation except that the original circuit is replaced by the layout-extracted circuit. After the correct results are observed in post-layout simulation, the design can be released as an IP by using the export tools in *Virtuoso*. Typical output files include a netlist file (for example, .cdl format) for LVS, a LEF file which includes the abstract of the design, and a GDS file which includes the layout of the design.

A.2 Netlist-driven Design Flow

The "netlist-driven" design flow is highly automated by using scripts. The FSM in this work is implemented using the "netlist-driven" design flow. The RTL design is written in a text file and simulated in *Modelsim*. After having the correct function of the RTL, it is used as a design input to *Design Compiler* for synthesis, as well as a script for running *Design Compiler*. The script includes all the information for setting up the synthesis environment, synthesis commands, constraints to the design, and exporting reports. The technology library used is "UMK65LSCLLMVBBR_B UMC 65nm Low-K Multi-Voltage Low Leakage RVT Tapless Standard Cell Library". The operating conditions are SS corner, 1.08 V power supply, and 125 °C. The outcome of the synthesis is a gate-level netlist file mapped to the standard cells in the technology library. Then, the netlist file is used as a design input to *Encounter* for placing and routing (P&R), as well as a script for running *Encounter*. The script includes all the information for setting up the P&R environment, P&R commands, constraints to the design, and exporting reports as well as relevant files, including a complete gate-level netlist, and a design file in DEF format. The netlist can be converted into SPICE format for LVS by using "v2lvs". Then, by importing the DEF file into *Virtuoso* in "OpenAccess" format, and switching the cell view from "abstract" to "layout", the "physical verification", "post-layout simulation" and "release IP" steps can be performed the same as in the "schematic-driven" design flow. Since the DEF file only contains the cell placing information, the technology library must be imported into *Virtuoso* in "OpenAccess" format as "abstract" view and "layout" view beforehand.

Verilog RTL Code of the FSM

B

```
1
module FSM (RST, CLK, EN, TDC0_CODE, TDC1_CODE, TDC0_FN, TDC1_FN, TDC0_OF
    , TDC1_OF, TDC_RST, DLSET, LOCK); //module decleration

//define input/output ports
input RST, CLK, EN, TDC0_FN, TDC1_FN, TDC0_OF, TDC1_OF;
6 input [30:0] TDC0_CODE, TDC1_CODE;
output reg [30:0] DLSET;
output TDC_RST, LOCK;

//define internal signals
11 reg [6:0] state, next_state; //state and next state signal
reg [30:0] code0_reg; //bit time thermometer code
reg [4:0] code0_bin; //bit time binary code
reg [30:0] code1_reg; //data-to-clock time thermometer code
reg [4:0] code1_bin; //data-to-clock time binary code
16 reg [5:0] update_code;
reg [5:0] DLSET_bin_trial;
reg [4:0] DLSET_bin; //DCDL setting binary code
reg [30:0] DLSET_next; //DCDL setting thermometer code

21 parameter [6:0] INIT    = 7'b0000001; //initial state
parameter [6:0] RSTTDC  = 7'b0000010; //reset TDC
parameter [6:0] WAIT    = 7'b0000100; //wait for TDC results
parameter [6:0] READ    = 7'b0001000; //read in TDC results and calculate
parameter [6:0] CALC    = 7'b0010000; //caculation for 1 cycle
26 parameter [6:0] SETDL   = 7'b0100000; //set DLSET value
parameter [6:0] LOCKED  = 7'b1000000; //locked state

integer i, j, k;

31 assign LOCK = (state == LOCKED) ? 1'b1 : 1'b0;
assign TDC_RST = (state == RSTTDC);

always @ (state, EN, TDC0_FN, TDC1_FN, TDC0_OF, TDC1_OF, TDC0_CODE,
   TDC1_CODE, update_code) //next state logic
begin
36     next_state = INIT;
    case (state)
        INIT: begin
            if (EN == 1'b1)
                next_state = RSTTDC;
41          else
                next_state = INIT;
```

```
            end
            RSTTDC: begin
                if (TDC0_CODE == 31'b0 && TDC1_CODE == 31'b0)
                    next_state = WAIT;
                else
                    next_state = RSTTDC;
            end
            WAIT: begin
                if (TDC0_OF == 1'b1 || TDC1_OF == 1'b1)
                    next_state = RSTTDC;
                else if (TDC0_FN == 1'b1 && TDC1_FN == 1'b1)
                    next_state = READ;
                else
                    next_state = WAIT;
            end
            READ: next_state = CALC;
            CALC: next_state = SETDL;
            SETDL: begin
                if (update_code == 6'b0)
                    next_state = LOCKED;
                else
                    next_state = RSTTDC;
            end
            LOCKED: next_state = LOCKED;
            default: next_state = INIT;
        endcase
    end

    always @ (posedge CLK) //state trnasition with rising edge of CLK
    begin
        if (RST == 1'b1)
            state <= INIT;
        else
            state <= next_state;
    end

    always @ (posedge CLK)//registered inputs and outputs
    begin
        if (RST == 1'b1)
        begin
            code0_reg <= 31'b0;
            code1_reg <= 31'b0;
            DLSET <= 31'b0;
        end
        else
            case (state)
                INIT: begin

                end
                RSTTDC: begin

                end
                WAIT: begin
```

```
96
                end
                READ: begin
                    code0_reg <= TDC0_CODE;
                    code1_reg <= TDC1_CODE;
101             end
                CALC: begin

                end
                SETDL: begin
                    DLSET <= DLSET_next;
106             end
                LOCKED: begin

                end
111             default:;
            endcase
    end

    always @ (code0_reg) //thermometer to binary conversion
116 begin
        j = 30;
        code0_bin = 5'b11111;
        while (code0_reg[j] == 1'b0 && j >= 0)
        begin
121         code0_bin = code0_bin - 1'b1;
            j = j - 1;
        end
    end

126 always @ (code1_reg) //thermometer to binary conversion
    begin
        k = 30;
        code1_bin = 5'b11111;
        while (code1_reg[k] == 1'b0 && k >= 0)
131     begin
            code1_bin = code1_bin - 1'b1;
            k = k - 1;
        end
    end
136
    always @ (code0_bin, code1_bin, DLSET_bin) //calculate next trial DCDL
       binary code
    begin
        update_code = code1_bin - (code0_bin >> 1);
        DLSET_bin_trial = DLSET_bin + update_code;
141 end

    always @ (posedge CLK) //determine next DCDL binary code
    begin
        if (RST == 1'b1)
146         DLSET_bin <= 5'b0;
        else if (state == CALC)
```

```
        begin
            if (DLSET_bin_trial[5] == 1'b1)
                DLSET_bin <= DLSET_bin_trial + code0_bin;
151         else
                DLSET_bin <= DLSET_bin_trial;
        end
    end

156 always @ (DLSET_bin) //binary to thermometer conversion
    begin
        i = 0;
        DLSET_next = 31'b0;
        for (i = 0; i < 31; i = i + 1)
161     begin
            if (i + 1 <= DLSET_bin)
                DLSET_next[i] = 1'b1;
        end
    end
166
    endmodule
```

Script for Synthesizing the FSM

C

```
#start GUI
gui_start

#read in RTL code
analyze -library WORK -format verilog {/users/shizhaozhang/FSM/RTL/FSM.v}

# eleborate design
elaborate FSM -architecture verilog -library WORK

#define input clock port "CLK" with 2ns period
create_clock -name "CLK" -period 2 -waveform { 1 2  }  { CLK  }
set_fix_hold  CLK

#define constrains

#define the driving strengh at the input ports as a reference cell in the
    technology library
set_driving_cell  -lib_cell CKINVM4R  -library uk65lscllmvbbr_108c125_wc
    [all_inputs]

#define load capacitane at the output ports
set_load 0.005 [all_outputs]

#define the maximum signal transition time at input and output ports
set_max_transition 0.5 [all_inputs]
set_max_transition 0.5 [all_outputs]

#define special load capacitance and output transition time for port
   TDC_RST
set_load 0.01 {TDC_RST}
set_max_transition 0.05 {TDC_RST}

#prevent reset signal being buffered
set_ideal_network -no_propagate RST

#define the maximum reset path delay as 1ns
set_input_delay 1 -clock CLK {RST}

#compile design
compile_ultra
check_design

#repot design
report_area  > /users/shizhaozhang/FSM/SYN/FSM_area.txt
```

```
report_power > /users/shizhaozhang/FSM/SYN/FSM_power.txt
report_timing > /users/shizhaozhang/FSM/SYN/FSM_timing.txt
report_resources > /users/shizhaozhang/FSM/SYN/FSM_resources.txt
report_reference > /users/shizhaozhang/FSM/SYN/FSM_reference.txt
report_cell > /users/shizhaozhang/FSM/SYN/FSM_cell.txt
report_net_fanout > /users/shizhaozhang/FSM/SYN/FSM_net_fanout.txt

#exclude capital letter A-Z from the generated netlist
define_name_rules verilog -restricted {A-Z} -type net
change_names -rules verilog -hierarchy

#export gate-level Verilog netlist and timing constraint files
write -hierarchy -format ddc -output /users/shizhaozhang/FSM/SYN/FSM.ddc
write -hierarchy -format verilog -output /users/shizhaozhang/FSM/SYN/
    FSM_mapped.v
write_sdf -version 2.1 /users/shizhaozhang/FSM/SYN/FSM.sdf
write_sdc -nosplit /users/shizhaozhang/FSM/SYN/FSM.sdc
```

Script for Placing & Routing the FSM

D

```
#init design
set ::DelayCal::esigUseVDC 1
set ::TimeLib::tsgMarkCellLatchConstructFlag 1
set defHierChar /
set gpsPrivate::dpgNewAddBufsDBUpdate 1
set gpsPrivate::lsgEnableNewDbApiInRestruct 1
set init_design_settop 0
set init_gnd_net VSS
set init_lef_file {Database/u065giol125mvir_8m1t0f1u.lef Database/
    uk65lscllmvbbr.lef}
set init_mmmc_file PAR/umc65.viewDefinition.tcl
set init_oa_search_lib {}
set init_pwr_net VDD
set init_verilog SYN/FSM_mapped.v
set lsgOCPGainMult 1.000000
set pegDefaultResScaleFactor 1.000000
set pegDetailResScaleFactor 1.000000
set timing_library_float_precision_tol 0.000010
set timing_library_load_pin_cap_indices {}
init_design

#floorplan: set block size as 59*61um2
getIoFlowFlag
setIoFlowFlag 0
floorPlan -coreMarginsBy die -site CORE -s 46.6 48.6 6.2 6.2 6.2 6.2
uiSetTool select
getIoFlowFlag

#set power nets
clearGlobalNets
globalNetConnect VDD -type pgpin -pin VDD -inst *
globalNetConnect VSS -type pgpin -pin VSS -inst *

#add power rings: VDD: ME3, with extra ME1 at top and bottom; VSS: ME2,
    with extra ME1 at top and bottom
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell
    -stacked_via_top_layer ME8 -type core_rings -jog_distance 2.0
    -threshold 2.0 -nets VDD -follow core -stacked_via_bottom_layer ME1
    -layer {bottom ME1 top ME1 right ME3 left ME3} -width 2 -spacing 0.5
    -offset 0.85
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell
    -stacked_via_top_layer ME8 -type core_rings -jog_distance 2.0
    -threshold 2.0 -nets VDD -follow core -stacked_via_bottom_layer ME1
```

```
    -layer {bottom ME3 top ME3 right ME3 left ME3} -width 2 -spacing 0.5
    -offset 0.85
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell
    -stacked_via_top_layer ME8 -type core_rings -jog_distance 2.0
    -threshold 2.0 -nets VSS -follow core -stacked_via_bottom_layer ME1
    -layer {bottom ME1 top ME1 right ME2 left ME2} -width 2 -spacing 0.5
    -offset 3.35
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell
    -stacked_via_top_layer ME8 -type core_rings -jog_distance 2.0
    -threshold 2.0 -nets VSS -follow core -stacked_via_bottom_layer ME1
    -layer {bottom ME2 top ME2 right ME2 left ME2} -width 2 -spacing 0.5
    -offset 3.35
editPowerVia -bottom_layer ME1 -add_vias 1 -orthogonal_only 0 -top_layer
    ME8

#add power pins: VDD: ME3, at the top, right and bottom side of the block
    ; VSS: ME2, at the top, right and bottom side of the block
createPGPin VDD -net VDD -geom ME3 23.5 0 35.5 5.3
createPGPin VSS -net VSS -geom ME2 23.5 0 35.5 2.8
createPGPin VDD -net VDD -geom ME3 23.5 55.7 35.5 61
createPGPin VSS -net VSS -geom ME2 23.5 58.2 35.5 61
createPGPin VDD -net VDD -geom ME3 53.7 24.5 59 36.5
createPGPin VSS -net VSS -geom ME2 56.2 24.5 59 36.5


#add power rails
sroute -connect { blockPin padPin padRing corePin floatingStripe }
    -layerChangeRange { ME1 ME3 } -blockPinTarget { nearestTarget }
    -padPinPortConnect { allPort oneGeom } -padPinTarget { nearestTarget }
     -corePinTarget { firstAfterRowEnd } -floatingStripeTarget { blockring
     padring ring stripe ringpin blockpin followpin } -allowJogging 1
    -crossoverViaLayerRange { ME1 ME8 } -allowLayerChange 1 -nets { VDD
    VSS } -blockPin useLef -targetViaLayerRange { ME1 ME8 }

#place pins: signal pins connecting to TDC and DCDL: ME3, at the left
    side of the block; external signal pins: ME4, at the top side of the
    block;
setPinAssignMode -pinEditInBatch false
editPin -pinWidth 0.1 -fixedPin 1 -snap MGRID -fixOverlap 1
    -spreadDirection clockwise -side Left -layer 3 -spreadType side -pin {
    TDC0_OF {TDC0_CODE[30]} {TDC0_CODE[29]} {TDC0_CODE[28]} {TDC0_CODE
    [27]} {TDC0_CODE[26]} {TDC0_CODE[25]} {TDC0_CODE[24]} {TDC0_CODE[23]}
    {TDC0_CODE[22]} {TDC0_CODE[21]} {TDC0_CODE[20]} {TDC0_CODE[19]} {
    TDC0_CODE[18]} {TDC0_CODE[17]} {TDC0_CODE[16]} {TDC0_CODE[15]} {
    TDC0_CODE[14]} {TDC0_CODE[13]} {TDC0_CODE[12]} {TDC0_CODE[11]} {
    TDC0_CODE[10]} {TDC0_CODE[9]} {TDC0_CODE[8]} {TDC0_CODE[7]} {TDC0_CODE
    [6]} {TDC0_CODE[5]} {TDC0_CODE[4]} {TDC0_CODE[3]} {TDC0_CODE[2]} {
    TDC0_CODE[1]} {TDC0_CODE[0]} TDC0_FN TDC_RST {DLSET[0]} {DLSET[1]} {
    DLSET[2]} {DLSET[3]} {DLSET[4]} {DLSET[5]} {DLSET[6]} {DLSET[7]} {
    DLSET[8]} {DLSET[9]} {DLSET[10]} {DLSET[11]} {DLSET[12]} {DLSET[13]} {
    DLSET[14]} {DLSET[15]} {DLSET[16]} {DLSET[17]} {DLSET[18]} {DLSET[19]}
     {DLSET[20]} {DLSET[21]} {DLSET[22]} {DLSET[23]} {DLSET[24]} {DLSET
    [25]} {DLSET[26]} {DLSET[27]} {DLSET[28]} {DLSET[29]} {DLSET[30]}
```

```
    TDC1_FN {TDC1_CODE[0]} {TDC1_CODE[1]} {TDC1_CODE[2]} {TDC1_CODE[3]} {
    TDC1_CODE[4]} {TDC1_CODE[5]} {TDC1_CODE[6]} {TDC1_CODE[7]} {TDC1_CODE
    [8]} {TDC1_CODE[9]} {TDC1_CODE[10]} {TDC1_CODE[11]} {TDC1_CODE[12]} {
    TDC1_CODE[13]} {TDC1_CODE[14]} {TDC1_CODE[15]} {TDC1_CODE[16]} {
    TDC1_CODE[17]} {TDC1_CODE[18]} {TDC1_CODE[19]} {TDC1_CODE[20]} {
    TDC1_CODE[21]} {TDC1_CODE[22]} {TDC1_CODE[23]} {TDC1_CODE[24]} {
    TDC1_CODE[25]} {TDC1_CODE[26]} {TDC1_CODE[27]} {TDC1_CODE[28]} {
    TDC1_CODE[29]} {TDC1_CODE[30]} TDC1_OF}
editPin -pinWidth 0.2 -fixedPin 1 -snap MGRID -fixOverlap 1 -unit MICRON
    -spreadDirection clockwise -side Top -layer 5 -spreadType center
    -spacing 5.0 -pin {RST CLK EN LOCK}

#add tap: well-tap pitch: 23um
addWellTap -cell WT3R -cellInterval 23 -prefix WELLTAP

#
setDesignMode -process 65

#place standard cells
setPlaceMode -fp false
placeDesign -inPlaceOpt

#cloc tree synthesis
createClockTreeSpec -bufferList {CKBUFM12R CKBUFM16R CKBUFM1R CKBUFM20R
    CKBUFM22RA CKBUFM24R CKBUFM26RA CKBUFM2R CKBUFM32R CKBUFM3R CKBUFM40R
    CKBUFM48R CKBUFM4R CKBUFM6R CKBUFM8R CKINVM12R CKINVM16R CKINVM1R
    CKINVM20R CKINVM22RA CKINVM24R CKINVM26RA CKINVM2R CKINVM32R CKINVM3R
    CKINVM40R CKINVM48R CKINVM4R CKINVM6R CKINVM8R} -file PAR/Clock.ctstch
clockDesign -specFile PAR/Clock.ctstch -outDir clock_report
    -unfixedInstBeforeCTS

#route
setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithTimingDriven 0
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -routeTopRoutingLayer default
setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail

#optimize
setAnalysisMode -analysisType onChipVariation
optDesign -postRoute -incr

#add filler
addFiller -cell FIL16R FIL1R FIL2R FIL32R FIL4R FIL64R FIL8R FILE16R
    -prefix FILLER

#save design and netlist
saveDesign PAR/FSM.enc
saveNetlist -excludeLeafCell -excludeCellInst {WT3R FIL16R FIL1R FIL2R
```

```
    FIL32R FIL4R FIL64R FIL8R} -phys PAR/FSM_par.v
93
#export DEF
global dbgLefDefOutVersion
set dbgLefDefOutVersion 5.7
defOut -floorplan -netlist -routing PAR/FSM.def
```

Bit Sequence Comparison

E

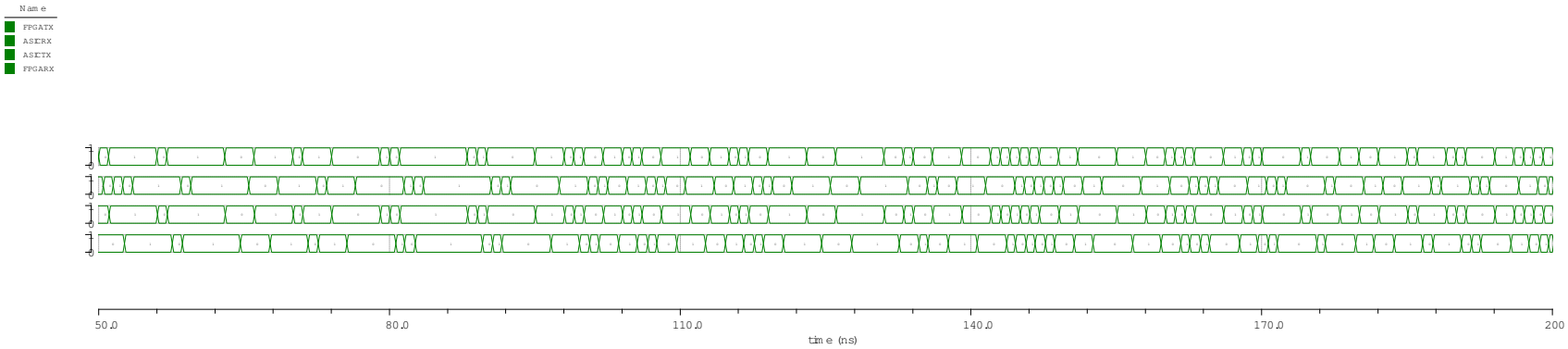

Figure E.1: Comparison of transmitted and received bit sequence

Source-Synchronous Interface Application Note

F

F.1 Overview

This source-synchronous interface (SSI) provides a low-cost solution to the inter-chip communication. It supports data rate from 421.4Mbps to 1.25Gbps. The interface provides a data transmission channel, a data reception channel, and a half-rate clock reception channel. It uses LVDS as external signaling standards. The embedded data recovery can calibrate the received data phase to be aligned to the 90° phase of the half-rate reference clock. The interface is designed for UMC 65 nm Low-leakage technology.

F.2 Physical Information

Figure F.1 shows the physical dimension information of the SSI. The location of each port and available accessing metal layers are also shown. The distance between the center of adjacent external signal ports is 92.5 µm. A full metal width of 12 µm is recommended for the connections between SSI external signal ports to I/O cells, and the connections should also be as short as possible to reduce parasitic resistance. As many power connections as possible is also preferred to improve power supply.

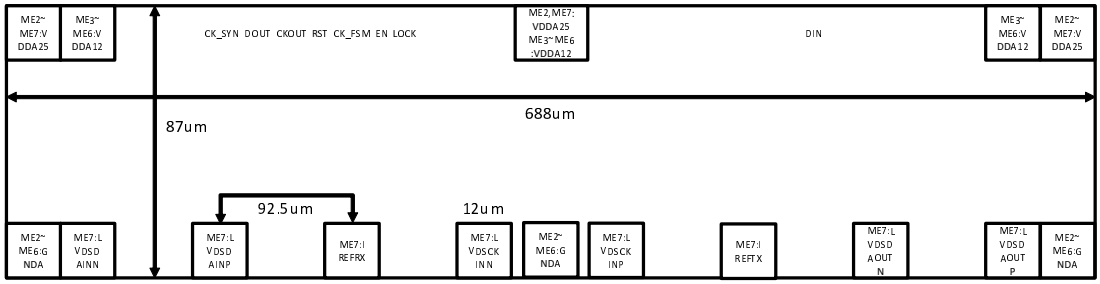

Figure F.1: Physical dimension information of SSI

F.3 Port Description

All digital signals are active high.

Table F.1: Port description

Port name	Direction	Description
CK_FSM	Input	Clock input for the internal finite state machine
CK_SYN	Input	Half-rate reference clock input for the data recovery
DIN	Input	Input data to be transmitted
EN	Input	Enable signal for the data recovery
LVDSCKINN/P	Input	LVDS half-rate reference clock input
LVDSDAINN/P	Input	LVDS data input
RST	Input	Synchronous reset for the data recovery
CKOUT	Output	Direct clock output from the LVDS clock receiver
DOUT	Output	Recovered data output
LOCK	Output	Lock indication for the data recovery
LVDSDAOUTN/P	Output	LVDS data output
IREFTX	N/A	Reference current for the LVDS transmitter
IREFRX	N/A	Reference current for the LVDS,receiver
VDDA25	N/A	2.5V power supply
VDDA12	N/A	1.2V power supply
GNDA	N/A	0V power supply

F.4 Configuration

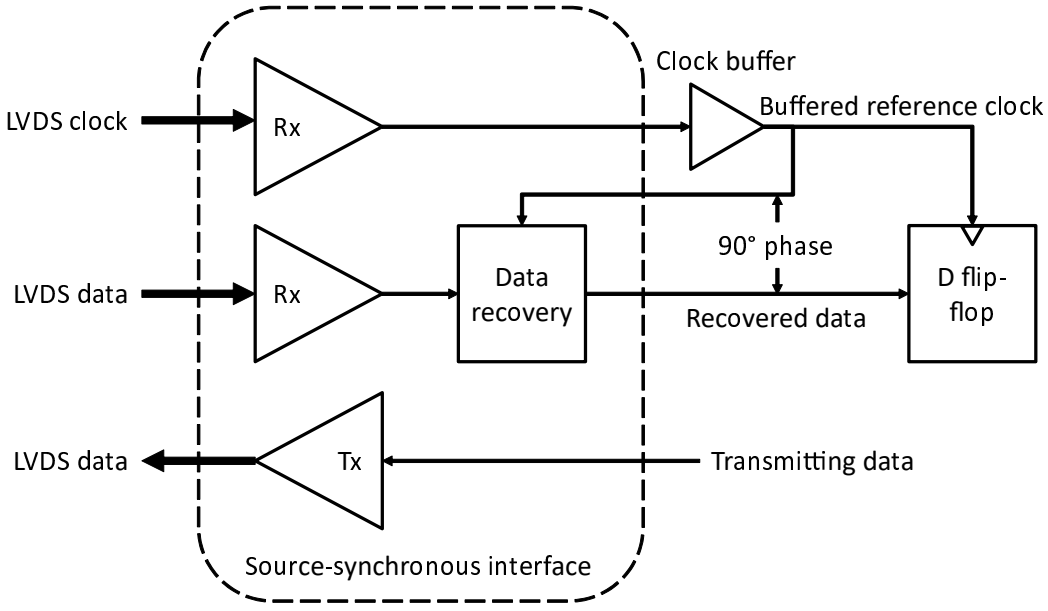

Figure F.2: Block diagram of the SSI configuration

Figure F.2 shows the block diagram of the top-level interface, which contains an LVDS driver, two LVDS receivers and a data recovery block. One of the LVDS receiver is used for half-rate reference clock reception, and the other used for data reception. The LVDS receivers convert LVDS signals to standard digital signals which have 0 V to 1.2 V signal swing. The received clock signal first goes through the external clock buffers, and then return to the data recovery as a half-rate reference clock. The received data signal is phase-calibrated in the data recovery, aligning the 90° phase of the reference clock. In other words, the edges of the reference clock are located at the center of the recovered data eye, which can ensure a correct data sampling by the following D flip-flop using the same reference clock. The LVDS driver converts the standard digital signal to LVDS signal without any timing manipulation, as the data will be recovered by the receiver at the other side of the communication link.

Like most other source-synchronous interfaces, the LVDS half-rate reference clock must be generated from the same clock source as for the LVDS data, as any frequency shift in the data is also reflected in the reference clock. This ensures the jitter and skew of the received data is finite relative to the reference clock, such that the data recovery is able find a correct phase for the recovered data.

The configuration shown in Figure F.2 can be duplicated to realize a multiple instance configuration, where each instance has its own reference clock. It is also possible for multiple instances to share a common buffered reference clock, provided that all instances operate at the same data rate. For unused LVDS clock receivers, their input must be connected to power supply (VDDA25).

F.5 Data Recovery

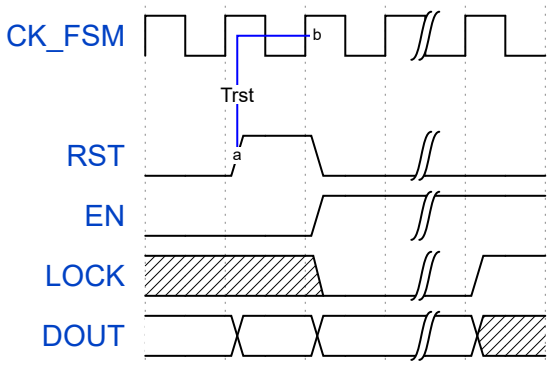

Figure F.3: Timing diagram of data recovery

The built-in data recovery can calibrate the received data phase to be aligned to the 90° phase of the half-rate reference clock. Figure F.3 shows the timing diagram of the data recovery. The reset signal “RST” must become valid-high for at least 1 ns before the clock signal “CK_FSM ” to ensure proper reset for the data recovery. If the enable signal “EN” is low, the data recovery is not working, and there will be a minimum delay from LVDS data input to the data output at “DOUT”. After ”EN” becomes high, the data recovery starts to calibrate the data phase according to the reference

clock at "CK_SYN ". It usually takes 24 cycles before the lock signal "LOCK" becomes valid-high, depending on the operating data speed and clock frequency at "CK_FSM ". As soon as "LOCK" becomes valid-high, the phase of the recovered data at "DOUT" is calibrated to the 90° of the reference clock with a finite phase error.

In order for the data recovery to work properly, a square-wave data sequence, i.e."101010...", is required to start appearing before the data recovery starts to work, until the lock signal "LOCK" becomes valid-high. This is to ensure that the data recovery skips the possible initial unstable operation of the LVDS transceiver, when much higher jitter could be present.

F.6 Port Input Capacitance

Table F.2: Port input capacitance

Symbol	**Description**	**Min.**	**Typ.**	**Max.**
C_{CK_FSM}	Input capacitance of port CK_FSM		24.8 fF	
C_{CK_CYN}	Input capacitance of port CK_SYN		17.7 fF	
C_{DIN}	Input capacitance of port DIN		7.9 fF	
C_{EN}	Input capacitance of port EN		5.8 fF	
C_{RST}	Input capacitance of port RST		25.8 fF	

F.7 Constraints

Table F.3: Constraints

Symbol	**Description**	**Min.**	**Typ.**	**Max.**
C_{CKOUT}	Load capacitance of port CKOUT			5 fF
C_{DOUT}	Load capacitance of port DOUT			20 fF
T_{r,CK_SYN}	Input transition time of port CK_SYN			50 ps (1)
DC_{CKREF}	Duty cycle of the reference clock for data recovery		50% (2)	

(1) A transition time of 50 ps or less is required for the data recovery to work properly. The transition time is defined as the time interval between 10% to 90% of the full signal swing.

(2) A duty cycle as close to 50% as possible is preferred for the data recovery to work properly.

F.8 AC Characteristics

Table F.4: AC Characteristics

Symbol	Description	Min.	Typ.	Max.
F_{CK_FSM}	Clock frequency for the internal FSM			500 MHz
T_{bit}	Data bit time, half clock period	800 ps		2373 ps
F_{bit}	Data bit rate	421.4Mbps		1.25Gbps
F_{CK_SYN}	Reference clock frequency	210.7 MHz		562.5 MHz

F.9 DC Characteristics

All digital signals are active high and (should) have 0 V to 1.2 V signal swing. Digital signals include CK_FSM, CK_SYN, DIN, EN, RST, CKOUT, DOUT and LOCK.

All LVDS signals (should) have nominal 1.25 V common-mode voltage, and nominal 350 mV differential voltage.

In the following graphs, V_{out} means the differential output voltage measured at the LVDS driver output ports; R_{out} means the total load resistance measured at the LVDS driver output ports. Therefore, designers should take care of the actual resistance of the top-level chip routing wire, package parasitic resistance, and the resistance of the transmission line, as well as the resulting voltage drop.

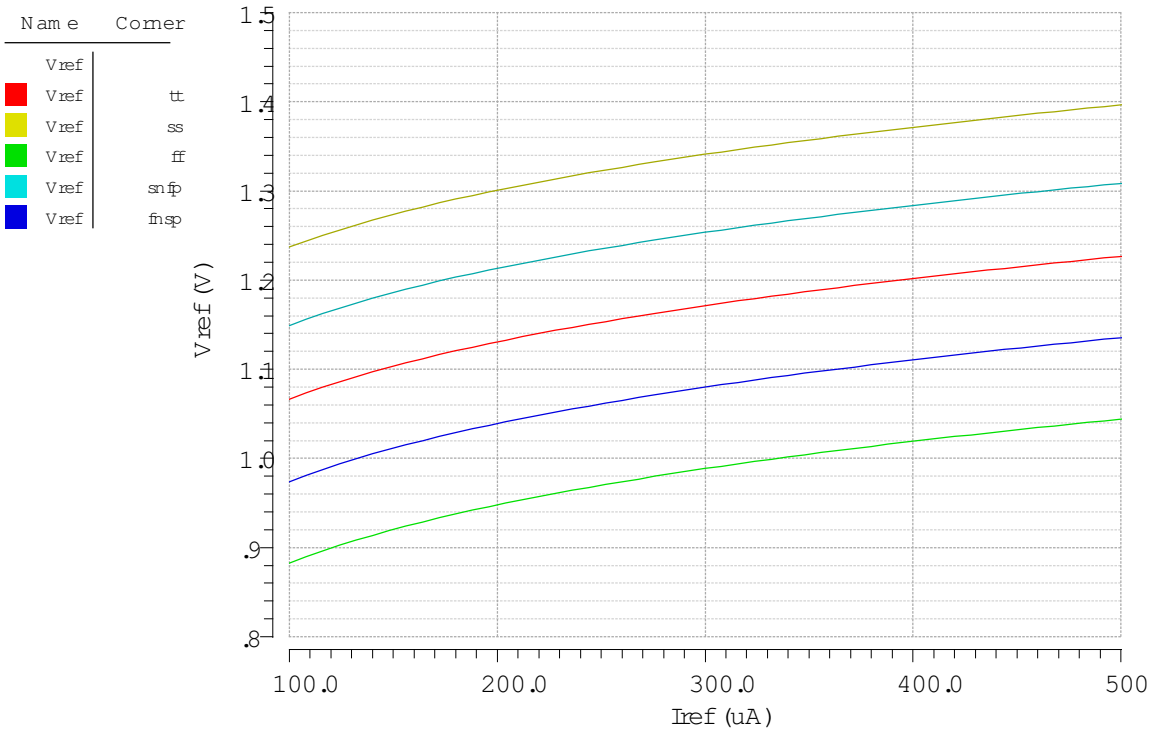

Figure F.4: Reference voltage vs. reference current of LVDS driver

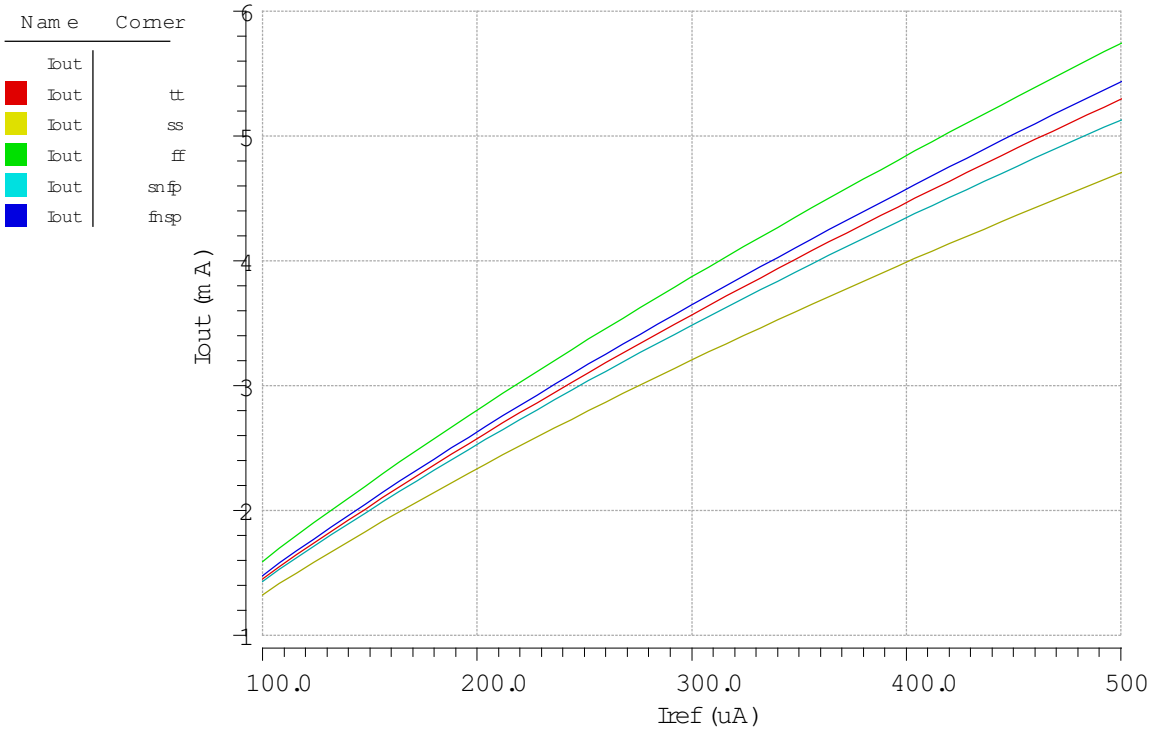

Figure F.5: Output current vs. reference current of the LVDS driver @R_{out}=110 Ω

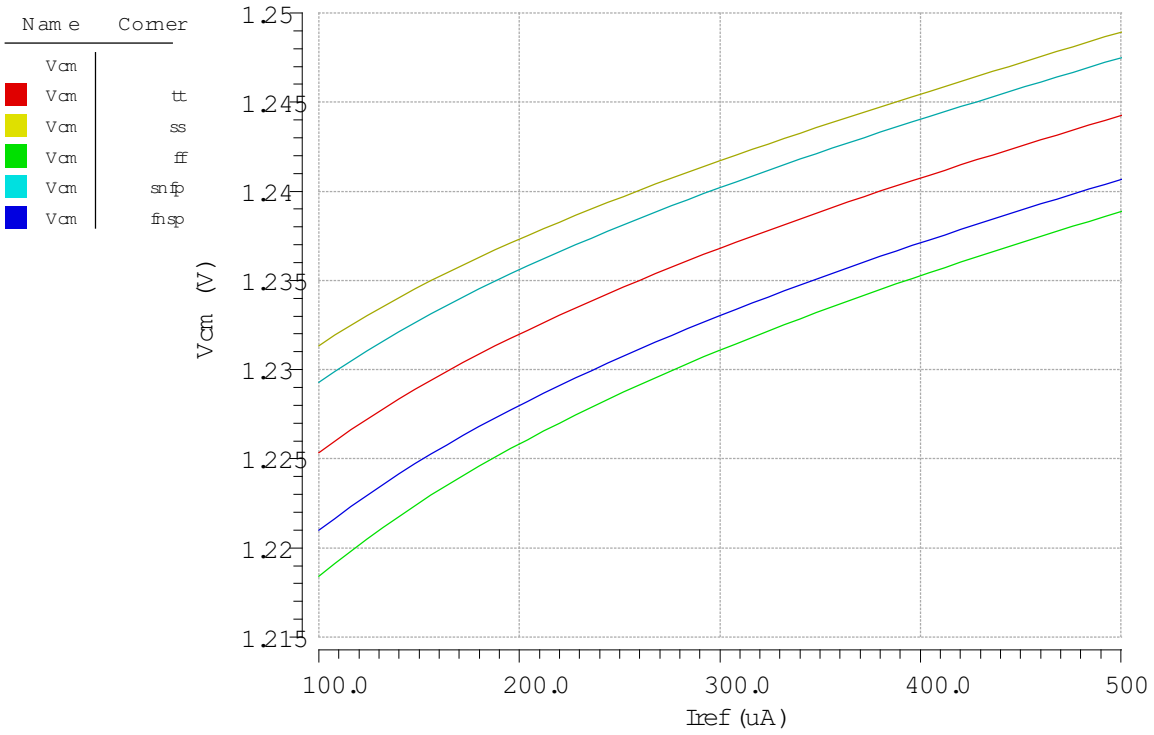

Figure F.6: Output common-mode voltage vs. reference current of the LVDS driver @R_{out}=110 Ω

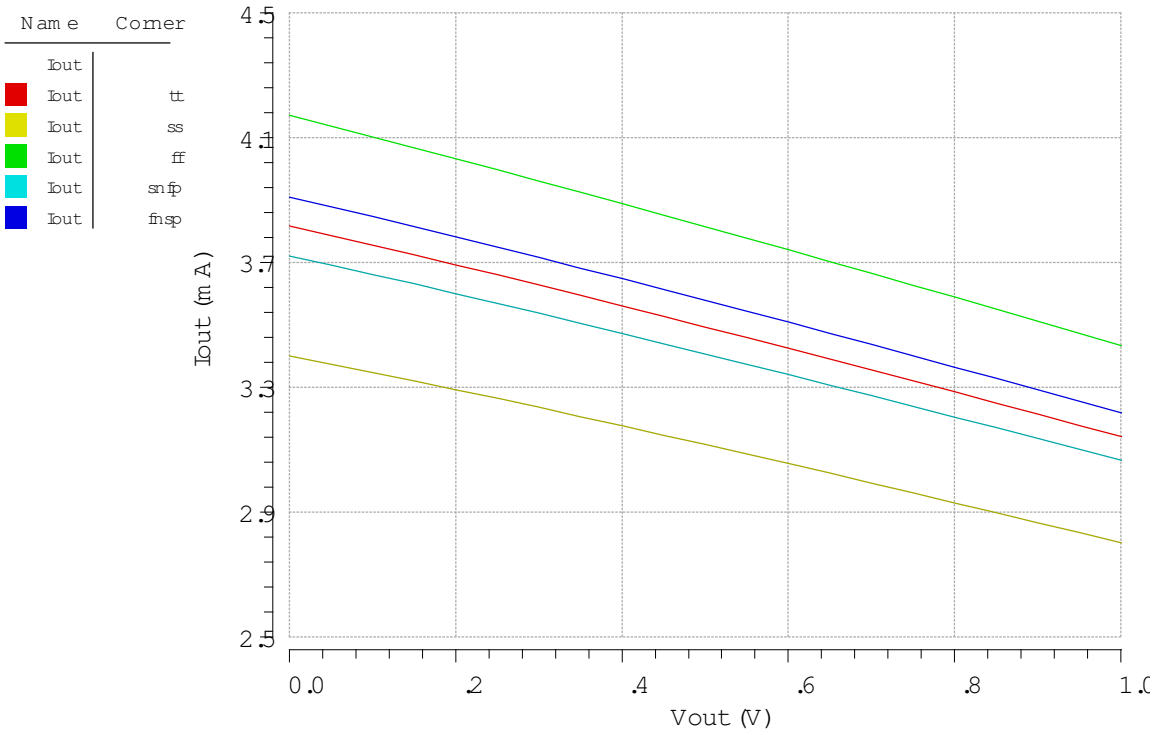

Figure F.7: Output current vs. output differential voltage of the LVDS driver @Iref=300 µA

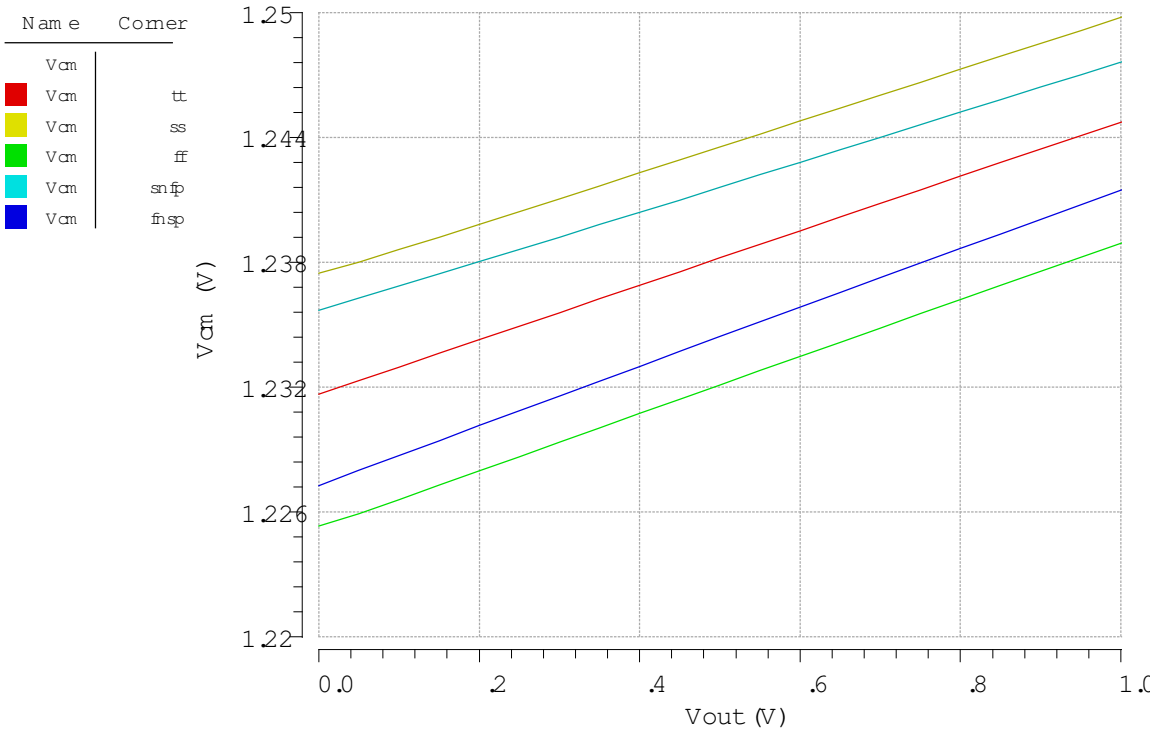

Figure F.8: Output common-mode mode voltage vs. output differential voltage of the LVDS driver @Iref=300 µA

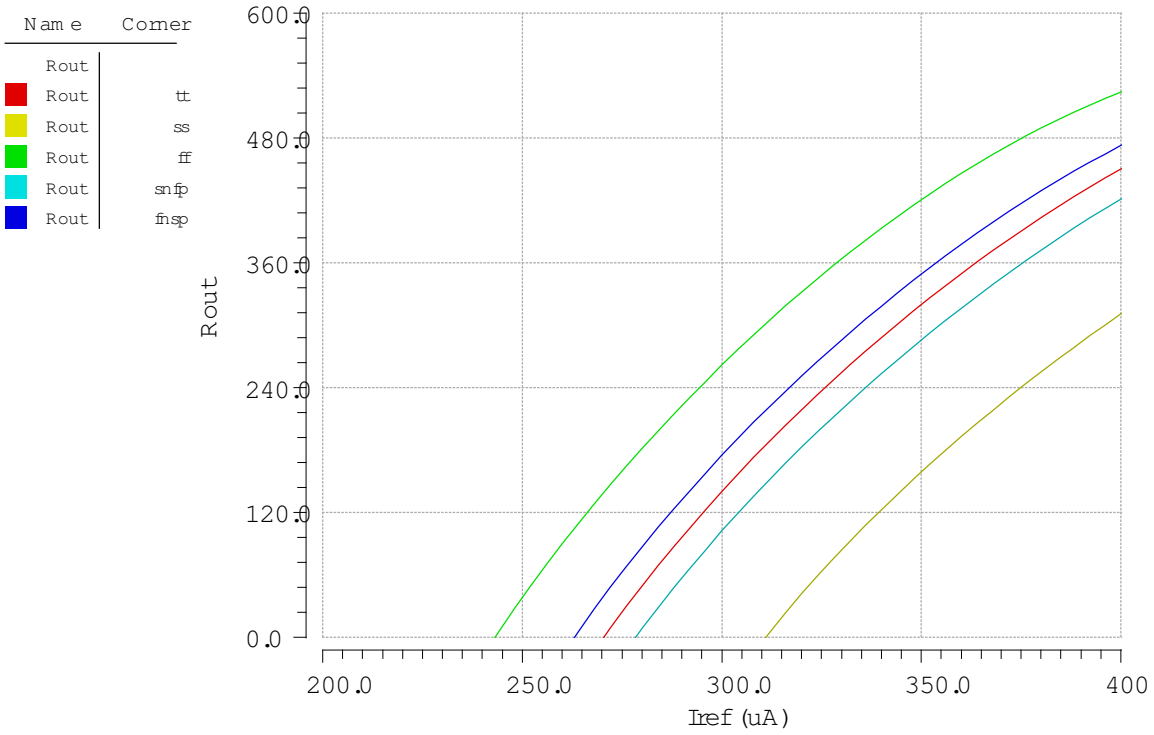

Figure F.9: Total output resistance vs. reference current of the LVDS driver @Iout=3.5 mA

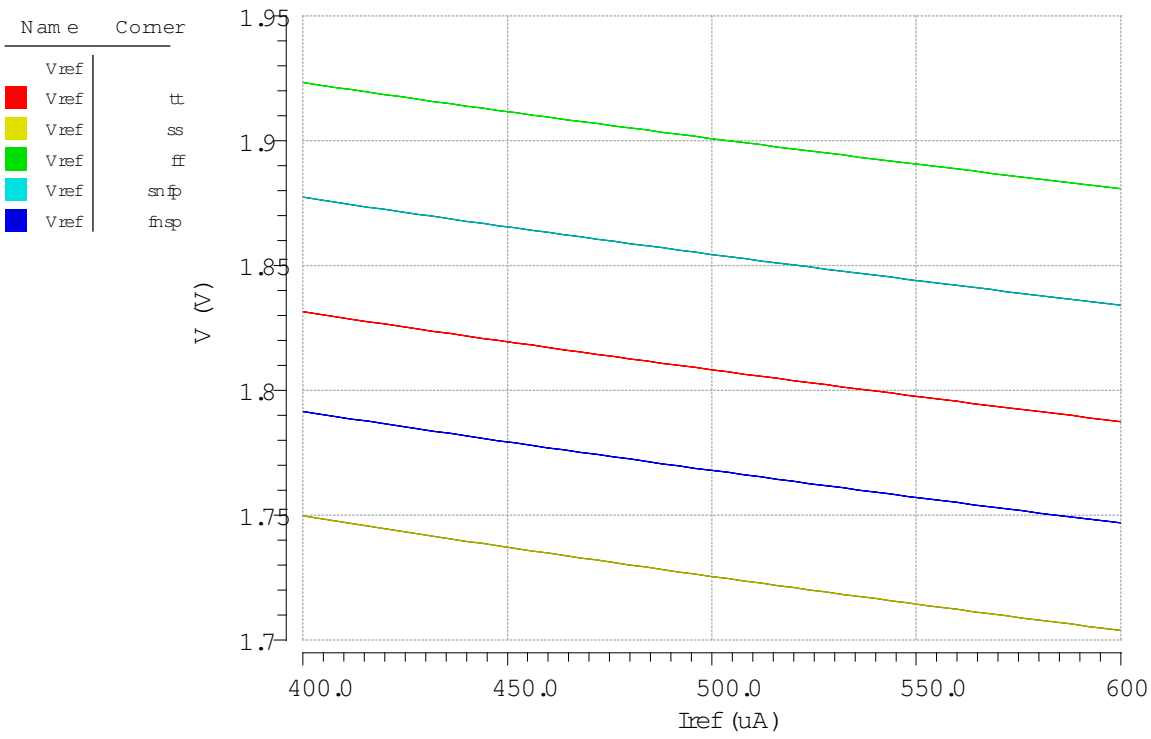

Figure F.10: Reference voltage vs. reference current of LVDS receiver