Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik
Fachgebiet Integrierte Elektronische Systeme

PHY Link Design and Optimization For High-Speed Low-Power Communication Systems

Yuan Fang

PhD Thesis, June 2014

Copyright © 2014
Institute of Computer Engineering
Technische Universität Darmstadt
Merckstr. 25, D-64283 Darmstadt, Germany

PHY Link Design and Optimization For High-Speed Low-Power Communication Systems

Vom Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades einer
Doktor-Ingenieurin (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Ing.
Yuan Fang
geboren am 16. October 1983
in Shanghai, China

Referent:	Prof. Dr.-Ing. Klaus Hofmann *Technische Universität Darmstadt*
Korreferent:	Prof. Dr.-Ing. Franko Küppers *Technische Universität Darmstadt*
Tag der Einreichung:	18.06.2014
Tag der mündlichen Prüfung:	26.09.2014

D17

Darmstädter Dissertationen 2015

Erklärung laut §9 der PromO

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Datum und Unterschrift

Dedicated to my lovely parents

Acknowledgment

This thesis is based on my work that I have started in July 2010 at *Fachgebiet Integrierte Elektronische Systeme, Institut für Datentechnik, Fachbereich Elektrotechnik und Informationstechnik, Technische Universität Darmstadt* as a three-year scholarship student and a one-year research assistant. Therefore, I would like to thank *graduate college "Tunable Integrated Components in Microwave Technology and Optics" (TICMO)* for providing me the scholarship to pursue my doctoral degree. Special thanks are due to my advisor Prof. Dr.-Ing. Klaus Hofmann for his advice, guidance, patience and providing me with excellent working environment for doing research. His warm and open personality, quality and curiosity in new things have made me benefited a lot.

As my co-advisor, I express my acknowledgment to Prof. Dr.-Ing. Franko Küppers for his critical reading of the thesis and advice.

I gratefully acknowledge Ashok Jaiswal for spending his time on deep technical discussion regarding my research, as well as on sharing his experience in both professional and personal life with me. I would like to thank all anonymous reviewers of my journal and conference papers for the positive critics and suggestions. Many thanks are due to the current staff members at *Fachgebiet Integrierte Elektronische Systeme*, Dr. Mohamed Salem, Harish Balasubramaniam, Mareiki Kaloumenos, Jing Ning, Muhammad Saif, Alex Schönberger, Lufei Shen, Eman Soliman, Boris Traskov, Botao Xiong and Haoyuan Ying for the friendship and cooperation as well as the former staff, Prof. Chunyue Huang.

I would also like to express my appreciation to the staff members at the former *Fachgebiet Mikroelektronische Systeme*, the head of the institute Prof. Dr.-Ing Prof. Dr. Dr. h.c. mult. Manfred Glesner and his research and teaching assistant staffs, Ramkumar Ganesan and Francois Philipp as well as the former staff members, Dr.-Ing Fizal Arya Samman and Dr.-Ing Ping Zhao. My acknowledgments are granted to our technical staff Andres Schmidt and Roland Brand for helping me in software and hardware matters, and to our secretary Silvia Hermann as well as our former secretary Iselona Klenk for helping me in many administrative matters.

I express my obligation to all my supervised students who have worked with me in pro-seminar/project seminar/master thesis. I would also like to give special thanks to Francois Philipp, Jan-Mark Metten, Damiano Gadler, Jonas Bargon and Ling Chen for proof-reading of my dissertation.

I appreciate all my friends from China, from Germany and also from all the other countries whom I could not mention all here. I am especially grateful to my best friends Lang Yu and Jan-Mark Metten in Munich for the emotional support and encouragement.

I deeply acknowledged all my teachers in my primary school, secondary school and my high school in Shanghai, China, as well as my lecturers at TU Dresden as well as TU Munich for providing me with basic and advanced knowledge.

Finally, I would like to thank my mother and my father for cheering me up and stood by me through the good and bad times from the depth of my heart. Without them this dissertation would not have been possible. Especially, I would give my deepest thank to my lovely father in heaven, who has motivated me the most to make him proud. I also would like to thank my family and friends in Shanghai who have taken good care of my mother during my stay in Germany.

Abstract

The ever-growing demands for high-bandwidth data transfer have been pushing towards advancing research efforts in the field of high-performing communication systems. Studies on the performance of single chip, e.g. faster multi-core processors and higher system memory capacity, have been explored. To further enhance the system performance, researches have been focused on the improvement of data-transfer bandwidth for chip-to-chip communication in the high-speed serial link. Many solutions have been addressed to overcome the bottleneck caused by the non-idealties such as bandwidth-limited electrical channel that connects two link devices and varieties of undesired noise in the communication systems. Nevertheless, with these solutions data have run into limitations of the timing margins for high-speed interfaces running at multiple gigabits per second data rates on low-cost Printed Circuit Board (PCB) material with constrained power budget. Therefore, the challenge in designing a physical layer (PHY) link for high-speed communication systems turns out to be power-efficient, reliable and cost-effective. In this context, this dissertation is intended to focus on architectural design, system-level and circuit-level verification of a PHY link as well as system performance optimization in respective of power, reliability and adaptability in high-speed communication systems.

The PHY is mainly composed of clock data recovery (CDR), equalizers (EQs) and high-speed I/O drivers. Symmetrical structure of the PHY link is usually duplicated in both link devices for bidirectional data transmission. By introducing training mechanisms into high-speed communication systems, the timing in one link device is adaptively aligned to the timing condition specified in the other link device despite of different skews or induced jitter resulting from process, voltage and temperature (PVT) variations in the individual link. With reliable timing relationships among the interface signals provided, the total system bandwidth is dramatically improved. On the other hand, interface training offers high flexibility for reuse without further investigation on high demanding components involved in high costs.

In the training mode, a CDR module is essential for reconstructing the transmitted bitstream to achieve the best data eye and to detect the edges of data stream in asynchronous systems or source-synchronous systems. Generally, the CDR works as a feedback control system that aligns its output clock to the center of the received data. In systems that contain multiple data links, the overall CDR power consumption increases linearly with the increase in number of links as one CDR is required for each link. Therefore, a power-

efficient CDR plays a significant role in such systems with parallel links. Furthermore, a high performance CDR requires low jitter generation in spite of high input jitter. To minimize the trade-off between power consumption and CDR jitter, a novel CDR architecture is proposed by utilizing the proportional-integral (PI) controller and three times sampling scheme.

Meanwhile, signal integrity (SI) becomes critical as the data rate exceeds several gigabits per second. Distorted data due to the non-idealties in systems are likely to reduce the signal quality aggressively and result in intolerable transmission errors in worst case scenarios, thus affect the system effective bandwidth. Hence, additional trainings such as transmitter (Tx) and receiver (Rx) EQ trainings for SI purpose are inserted into the interface training. Besides, a simplified system architecture with unsymmetrical placement of adaptive Rx and Tx EQs in a single link device is proposed and analyzed by using different coefficient adaptation algorithms. This architecture enables to reduce a large number of EQs through the training, especially in case of parallel links. Meanwhile, considerable power and chip area are saved.

Finally, high-speed I/O driver against PVT variations is discussed. Critical issues such as overshoot and undershoot interfering with the data are primarily accompanied by impedance mismatch between the I/O driver and its transmitting channel. By applying PVT compensation technique I/O driver impedances can be effectively calibrated close to the target value. Different digital impedance calibration algorithms against PVT variations are implemented and compared for achieving fast calibration and low power requirements.

Kurzfassung

Die ständig wachsenden Anforderungen an hohe Bandbreiten in Datenübertragungen haben die Forschungsanstrengungen auf dem Gebiet hochleistungsfähiger Kommunikationssysteme verstärkt. Schon immer wurde intensiv über die Steigerung der Leistungsfähigkeit einzelner integrierter Schaltkreise geforscht, z.B. durch schnellere Multi-core-Prozessoren oder durch höhere Speicherkapazitäten. Um die Systemleistung weiter zu verbessern, wurden Untersuchungen im Bereich der Optimierung der Datenübertragungsbandbreite in der Chip-zu-Chip Kommunikation mittels serieller Hochgeschwindigkeitsverbindung konzentriert. Viele Lösungen wurden vorgeschlagen, um durch Nichtidealitäten verursachte Engpässe in der Kommunikation zwischen zwei Bauteilen zu verhindern. Solche Nichtidealitäten sind z.B. die begrenzte Bandbreite des elektrischen Kanals oder verschiedenen Arten von unerwünschtem elektrischen Rauschen. Allerdings stellen diese Lösungen häufig hohe Anforderungen an die Umgebungsbedingungen wie Platinenqualität und erfordern hohen Energieaufwand. Auf preisgünstigen Platinen oder bei reduziertem Energieverbrauch führen diese Lösungen jedoch zu Einschränkungen der möglichen Übertragungsraten bei Hochgeschwindigkeits-Schnittstellen. Daher besteht die Herausforderung bei der Gestaltung eines PHY-Links für Hochgeschwindigkeitskommunikationssysteme darin, energieeffizient, zuverlässig und kostengünstig zu sein. In diesem Zusammenhang konzentriert sich diese Dissertation auf den Architekturentwurf, auf die Verifikation eines PHY-Links auf System- und Schaltungsebene, sowie auf die Optimierung der Systemleistung in Bezug auf Engerieverbrauch, Zuverlässigkeit und Anpassungsfähigkeit in Hochgeschwindigkeitskommunikationssystemen.

Der PHY besteht vor allem aus einer Einheit zur Taktwiederherstellung aus den Daten (CDR), aus Equalizern (EQ) und aus Hochgeschwindigkeits I/O Treibern. Die symmetrische Struktur des PHY-Links wird gewöhnlich in beiden Verbindungseinheiten zur bidirektionalen Datenübertragung dupliziert. Durch Einsatz von Trainingsmechanismen kann das Timing einer Kommunikationseinheit trotz verschiedener vorhandener Abweichungen des Timings oder durch Herstellungsprozess-, Spannungs- und Temperatur-Schwankungen (PVT) erzeugten Jitter adaptiv an die Timing-Bedingungen der anderen Kommunikationseinheit angepasst werden. Mit zuverlässigen Timingverhältnissen zwischen den Signalen der Schnittstellen wird die Gesamtsystembandbreite drastisch verbessert. Auf der anderen Seite bietet das Schnittstellen-Training eine hohe Flexibilität für die Wiederverwendung ohne weitere, kostenintensive Untersuchungen von ansonsten

notwendigen hochgenauen Komponenten.

Im Trainingsmodus ist ein CDR Modul notwendig um das beste Datenauge für die Rekonstruktion des übertragenen Bitstroms zu erreichen und um die Signalflanken des Datenstroms in asynchronen oder quellensynchronen Systemen zu ermitteln. Im Allgemeinen arbeitet der CDR als geschlossener Regelkreis, welcher sein ausgegebenes Taktsignal mittig zu den empfangenen Daten ausrichtet. In Systemen, die mehrere Datenverbindungen enthalten, steigt der Gesamtenergieverbrauch der CDR Einheiten linear mit der Anzahl von Verbindungen, da je ein eigenes CDR für jede einzelne Verbindung erforderlich ist. Deshalb spielt die Energieeffizienz der CDRs eine bedeutende Rolle in derartigen Systemen mit parallelen Verbindungen. Weiterhin darf ein Hochleistungs-CDR trotz hohen Eingangsjitters selber nur einen sehr geringen Ausgangsjitter erzeugen. Um den Kompromiss zwischen Energieverbrauch und CDR Jitter zu minimieren, wird eine neue CDR Architektur unter Verwendung eines Proportional-Integral (PI) Controllers und dreifach Sampling vorgeschlagen.

Bei Datenraten von mehreren Gigabit pro Sekunde wird ausserdem die Integrität der Daten sehr entscheidend. Verzerrte Daten aufgrund von Nichtidealitäten in Systemen können die Signalqualität massiv verringern und führen im schlimmsten Fall zu unbehebbaren Übertragungsfehlern, wodurch die effektive Bandbreite des Systems negativ beeinflusst wird. Daher werden für die Optimierung der Signalintegrität zusätzlich Sender (Tx) und Empfänger (Rx) EQ Trainings als Schnittstellen-Training eingesetzt. Weiterhin wird eine vereinfachte Systemarchitektur mit unsymmetrischer Anordnung der adaptiven Rx und Tx EQs in einem Single-Link-Bauelement durch die Verwendung verschiedener Koeffizienten-Adaptions-Algorithmen vorgeschlagen und analysiert. Diese Architektur ermöglicht es, durch das Training die insbesondere im Fall von parallelen Verbindungen große Anzahl benötigter EQs zu reduzieren. Gleichzeitig wird hierdurch erheblich Siliziumfläche eingespart und Energieverbrauch minimiert.

Zum Abschluss werden die Eigenschaften eines High-Speed I/O-Treibers in Abhängigkeit von PVT-Schwankungen analysiert. Kritische Probleme wie Beeinträchtigung der Daten in Form von Überschwingen und Unterschwingen werden in erster Linie durch Impedanzfehlanpassung zwischen dem I/O-Treiber und dem zugehörigen Sendekanal verursacht. Durch Anwendung einer PVT-Kompensationstechnik können die Impedanzen der I/O-Treiber effektiv an vorgegebene Zielwerte angepasst werden. Ausserdem werden unterschiedliche digitale Algorithmen zur Kalibrierung der Impedanzen gegen PVT-Schwankungen implementiert und bezüglich ihrer Geschwindigkeit und ihres Energieverbrauchs verglichen.

Table of Contents

1 Introduction and Overview 1

1.1 Background 1
1.2 Motivations 2
1.3 Research Objectives and Scope 4
1.4 Thesis Outline 5

2 Interconnect Standards in Computer Peripherals 7

2.1 PCI Express 8
2.2 HyperTransport 9
2.3 DDRx 10
2.4 GDDRx 12
2.5 Summary 14

3 Phase Interpolator Based Clock Data Recovery 15

3.1 State-of-the-art 17
3.2 PI^2 CDR Simulink Model 21
  3.2.1 Model Description 21
  3.2.2 Simulation Results 22
3.3 PI^2 CDR Components Design 25
  3.3.1 Digital Phase Interpolator 25
  3.3.2 Multiplexer 26
  3.3.3 Three Times Sampling CDR 28
3.4 Simulation Results 29
  3.4.1 Summary 31

4 Signal Integrity in Low-Power Adaptive Equalizer Trainings 33

4.1 State-of-the-Art 34
4.2 Signal Integrity Issues in High-Speed Communication System 35
4.3 System-Level Modeling of Equalizer Training 38
  4.3.1 System-Level Modeling of Receiver Equalizer Training 38

4.3.1.1 Model Description 38
4.3.1.2 Simulation Results 41
4.3.2 System-Level Modeling of Transmitter Equalizer Training 42
4.3.2.1 LMS Algorithm 42
4.3.2.2 Pilot Signal/Peak Detection 43
4.3.2.3 Direct Calculation 45
4.3.2.4 Simulation Results 46
4.4 Design and Implementation of Receiver Equalizer 49
4.4.1 Analog Receiver Equalizer 49
4.4.1.1 LMS Engine 49
4.4.1.2 Equalizer Summers 52
4.4.2 LMS Semi-Digital Receiver Equalizer 53
4.4.2.1 LMS Engine 53
4.4.2.2 Equalizer Summers 54
4.5 Design and Implementation of Transmitter Equalizer 54
4.5.1 Semi-Digital LMS Transmitter Equalizer 54
4.5.2 Semi-Digital Pilot Transmitter Equalizer 55
4.5.3 Full-Digital Direct Transmitter Equalizer 57
4.5.4 Simulation Results 59
4.5.4.1 Simulation Environment 59
4.5.4.2 Receiver Equalizer 62
4.5.4.3 Transmitter Equalizer 64
4.6 Summary 65

5 I/O Design 67
5.1 State-of-the-art 68
5.1.1 I/O Driver Structure 68
5.1.2 PVT Compensation Schemes 70
5.2 Proposed Hybrid Digital Impedance Calibration 74
5.2.1 Calibration Architecture 74
5.2.2 Hybrid Calibration Algorithm 76
5.2.2.1 Hybrid Linear Search 77
5.2.2.2 Hybrid Reduced Binary Search 77
5.3 Simulation Results 78
5.4 Summary 80

6 Summary and Outlook 81
6.1 Summary 81

6.2 Directions for Future Work 82

A Improvement of Signal Integrity by Adaptive Equalizer Trainings 85

A.1 Jitter and Bathtub 85

A.2 MATLAB Code for Signal Evaluation in Adaptive Trainings 86

A.2.1 Resample of prbs Data 88

A.2.1.1 Eye Shift Obtained from Impulse Response 88

A.2.2 Eye Shifts Addition 88

A.2.3 Eye Data Calculation 89

A.2.4 Eye Data Truncation at Start and Stop Points 89

A.2.5 Bathtub Data Calculation 89

A.2.6 Plot Bathtub 90

A.2.7 Jitter Calculation 90

A.2.8 Plot Jitter 91

A.3 Verilog Code for LMS Engine and Direct Tx EQ 91

A.3.1 Multiplier 91

A.3.2 Adder 93

A.3.3 Divider 93

References 99

Invention Disclosures 109

List of Own Publications 111

List of Unrelated Publications 113

Curriculum Vitae 117

List of Tables

3.1 Variable settings in modified, adaptive and PI^2 CDR systems 23
3.2 Phase errors in modified, adaptive and PI^2 CDR systems 23
3.3 Comparison among modified, adaptive and PI^2 CDRs ÆŠwith constant locking time 24
3.4 Load of the inputs of MUX_3_2 26
3.5 CDR jitter of modified and PI^2 CDR systems in corner simulation 30
3.6 Power of modified and PI^2 CDR systems in corner simulation 30
3.7 LSB varies with/without maximal random jitter in data 31

4.1 Coefficients of FFE and DFE after convergence 41
4.2 Comparison of eye diagrams 42
4.3 Coefficients of three algorithms after Tx EQ training 47
4.4 Eye opening among different EQ algorithms 48
4.5 Eye opening among different EQ algorithms 48
4.6 AC performance of OTA 52
4.7 Pilot signals and corresponding filter coefficients 56
4.8 Comparison of analog and semi-digital adaptive Rx EQ 64
4.9 Comparison of Tx EQs with three algorithms 65

5.1 Worst active power consumed for different algorithms 78
5.2 Statics for PFET section of driver 78
5.3 Statics for NFET section of driver 79

List of Figures

1.1 Peak memory bandwidth in CPUs and GPUs 2
1.2 An example of a PHY in a high-speed communication system 3

2.1 Intel® Z87 express chipset platform block diagram 7
2.2 PHY block diagram in PCI express 8
2.3 An example of HyperTransport links for chip-to-chip interconnect 9
2.4 A low-voltage differential signaling in HyperTransport links 10
2.5 Data rate per pin in generation of DDR memories 10
2.6 A simplified PHY link in DDR memories 11
2.7 Interconnect between north bridge and memory modules 11
2.8 Time of burst data for various memories 12
2.9 GDDR5 interface 13
2.10 Conventional interface training sequence 13

3.1 Read training in a GDDR5 memory system 16
3.2 Write training in a GDDR5 memory system 16
3.3 CDR trainings in high-speed communication systems 17
3.4 Conventional CDR block diagram 19
3.5 Modified CDR block diagram 19
3.6 Adaptive CDR block diagram 20
3.7 Evaluation of data transitions 20
3.8 Proposed PI^2 CDR block diagram 21
3.9 PI controller block diagram 22
3.10 Controlled variable versus time for different controllers 23
3.11 Phase error between ideal sampling point and actual phase of adaptive CDR with/without jitter 24
3.12 1-bit cell 25

3.13 A 3-bit phase interpolator 25
3.14 Glitch free techniques 26
3.15 MUX_3_2 27
3.16 MUX_4_2 27
3.17 3TS CDR block diagram 28
3.18 Alignment zone in a 3TS CDR 29
3.19 Phase alignment of modified CDR and PI^2 CDR 30

4.1 Adaptive EQs in the conventional architecture 34
4.2 Example of a channel between memory controller and GDDR5 memory . . 36
4.3 Proposed interface training sequence 36
4.4 Proposed system architecture using adaptive EQs 37
4.5 Working principle of receiver equalizer 38
4.6 FFE and DFE 38
4.7 Rx EQ training 39
4.8 Modeled channel in Matlab/Simulink 39
4.9 FFE in Rx EQ 40
4.10 DFE in Rx EQ 40
4.11 Tap in Rx EQ 40
4.12 Coefficients of FFE and DFE for Rx EQ training 41
4.13 Eye diagrams without/with Rx EQ 41
4.14 Tx EQ training using LMS in Matlab/Simulink 42
4.15 Coefficients during LMS training 43
4.16 The procedure of pilot signal/peak detection algorithm 43
4.17 Tx EQ training using pilot signal/peak detection in Matlab/Simulink 44
4.18 The signal generator 45
4.19 Tx EQ training using direct calculation in Matlab/Simulink 46
4.20 Coefficients after direct calculation in Simulink 46
4.21 Eye diagrams for Tx EQ 47
4.22 Eye diagrams for Tx EQ with a more lossy channel 48
4.23 DFE/FFE block diagrams in an analog receiver equalizer 49
4.24 Gilbert cell multiplier 50
4.25 Schematic of OTA 50
4.26 Simulation results of multiplier 51

4.27 The ac simulation of OTA 51
4.28 Schematic of FFE summer 52
4.29 Schematic of DFE summer 53
4.30 Digital LMS engine for FFE/DFE 53
4.31 Algorithm of pipelined multiplier 54
4.32 Proposed pilot algorithm for transmitter equalizer training 55
4.33 The block diagram of Tx EQ with pilot algorithm 55
4.34 Control unit in pilot Tx EQ 56
4.35 The state diagram of adaptation in pilot Tx EQ 57
4.36 Tx EQ training based on direct calculation 57
4.37 Coefficient calculation in direct algorithm 58
4.38 EQ summer of a full-digital adaptive Tx EQ 58
4.39 Procedure of piplined divider 59
4.40 Design flow for mixed-signal design 60
4.41 Co-simulation for the measurement 61
4.42 Adaptive bias voltages for receiver equalizer 62
4.43 Eye diagrams with/without adaptive Rx EQs 63
4.44 Jitter with/without adaptive Rx EQs 63
4.45 Bathtub curves with/without adaptive Rx EQs 63
4.46 Adaptive bias voltages using pilot and LMS algorithms 64
4.47 Eye diagrams using different algorithms in the adaptive Tx EQs 64
4.48 Jitters using different algorithms in the adaptive Tx EQs 65
4.49 Bathtub curves using different algorithms in the adaptive Tx EQs 65

5.1 PFET impedance variations over PVT variations 68
5.2 Schematic of simple programmable I/O driver 68
5.3 Schematic of poly resistance I/O driver 69
5.4 Schematic of current assist I/O driver 70
5.5 Conventional impedance calibration algorithm 70
5.6 Compensation schemes [66] 71
5.7 Analog Comparator block 72
5.8 Digital block for calibration 72
5.9 PVT variation detect circuit 73
5.10 Analog to digital converter 73

5.11 Illustration of binary search algorithm for 4 bits 74
5.12 Proposed calibration architecture 74
5.13 Mode and match block architecture 75
5.14 Proposed impedance calibration algorithm 76
5.15 PFET part in the I/O driver 77
5.16 Calibrated PFET impedance for PVT variations 79
5.17 Calibrated NFET impedance for PVT variations 79

A.1 The Dual-Dirac model for distribution of jitter 86
A.2 Bathtub cursor 86

List of Abbreviations

3TS	**T**hree **T**imes **S**ampling
ADC	**A**nalog **D**igital **C**onverter
ADDR	**A**ddress
ALPG	**A**lgorithmic **P**attern **G**enerator
BER	**B**it **E**rror **R**ate
CCO	**C**urrent **C**ontrolled **O**scillator
CDR	**C**lock **D**ata **R**ecovery
CMD	**C**ommand
DAC	**D**igital **A**nalog **C**onverter
DDR	**D**ouble **D**ata **R**ate
DES	**D**eserializer
DFE	**D**ecision **F**eedback **E**qualizer
DIMM	**D**ual **I**n-line **M**emory **M**odule
DJ	**D**eterministic **J**itter
DLL	**D**elay **L**ocked **L**oop
EDC	**E**rror **D**etection and **C**orrection
EL-TED	**E**arly-**L**ate **T**iming **E**rror **D**etector
EQ	**E**qualizer
FFE	**F**eed-**F**orward **E**qualizer
FIFO	**F**irst **I**n **F**irst **O**ut
FSB	**F**ront-**S**ide **B**us
FSM	**F**inite-**S**tate-**M**achine
GDDR	**G**raphic **DDR**
HT	**H**yper**T**ransport
IC	**I**ntegrated **C**ircuit
ISI	**I**nter-**s**ymbol **I**nterference
LDFF	**L**oad **FIFO**
LE	**L**inear **E**qualizer
LF	**L**ow **P**ass **F**ilter

LMS	**L**east **M**ean **S**quare
LS	**L**evel **S**hifter
LSB	**L**east **S**ignificant **B**it
MSB	**M**ost **S**ignificant **B**it
NFET	**N**-channel **F**ield-**E**ffect **T**ransistor
OCD	**O**ff- **C**hip **D**river
ODT	**O**n- **D**ie **T**erminator
OTA	**O**perational **T**ransconductance **A**mplifier
PCB	**P**rinted **C**ircuit **B**oard
PD	**P**hase **D**etector
PFET	**P**-channel **F**ield-**E**ffect **T**ransistor
PHY	**P**hysical **L**ayer
PI	**P**roprotional-**I**ntegral
PID	**P**roprotional-**I**ntegral-**D**erivative
PLL	**P**hase **L**ocked **L**oop
POD	**P**seudo **O**pen **D**rain
PVT	**P**rocess **V**oltage **T**emperature
PRBS	**P**seudo **R**andom **B**it **S**equences
QPI	**Q**uick**P**ath **I**nterconnect
RB	**R**educed **B**inary
RDTR	**R**ead **T**raining
RJ	**R**andom **J**itter
RLS	**R**ecursive **L**east **S**quare
Rx	**R**eceiver
SBR	**S**ingle **B**it **R**esponse
SDRAM	**S**ynchronous **D**ynamic **R**andom **A**ccess **M**emory
SDR	**S**ingle **D**ata **R**ate
SER	**Ser**ializer
SI	**S**ignal **I**ntegrity
SSO	**S**imultaneous **S**witching **O**utput
TSV	**T**hrough **S**ilicon **V**ia
Tx	**T**ransmitter
WRTR	**W**rite **Tr**aining

Chapter 1

Introduction and Overview

Contents

1.1 Background 1
1.2 Motivations 2
1.3 Research Objectives and Scope 4
1.4 Thesis Outline 5

1.1 Background

The trend of remarkable data traffic growth with the rise of multimedia-on-demand and cloud computing usage has led to incessant requests for high performance communication systems supporting considerable bandwidth data transmission. For example, fully integrated backplane transceivers that operated 9 years ago at speeds of 5 Gbps [1] runs at speeds up to 28 Gbps [2] now. To keep up with Moore's law, microprocessors in communication systems have shifted towards multi-core or even many-core as single-core processor has reached power wall by continuously increasing clock speeds.

Meanwhile, the issue of chip-to-chip communication has received considerable critical attention for high-bandwidth oriented applications, for example, multi-socket connectivity in server systems. As the number of processors increases, the improvement of data-transfer bandwidth is saturated by conventional highly capacitive shard front-side bus (FSB) architecture. Thereby, a new system concept utilizing point-to-point architecture such as AMD HT [3] or Intel QuickPath Interconnect (QPI) [4] technology has been introduced in the recent years, which enables each processor to be connected directly through a fast bus. Similarly, a direct access between processor and memory is also provided by embedding the north bridge into the individual core. Figure 1.1 illustrates the peak memory bandwidth in both CPUs and GPUs over the last seven years. Trends show that the memory bandwidth in GPUs increases about four-fold while that in CPUs increases about two-fold.

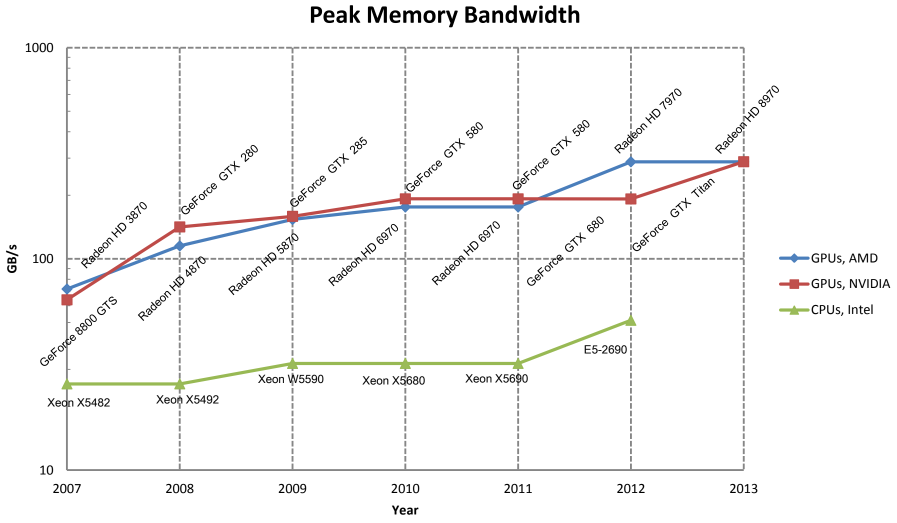


Figure 1.1: Peak memory bandwidth in CPUs and GPUs

Although aggregate throughput has been tremendously improved by shortening the inter-chip latency, the system performance is severely constrained by non-idealities of channels that connect two chips. Various advanced chip-to-chip electrical interconnect techniques have been employed in a physical layer (PHY) interfacing a digital core to a physical medium (channel) to compensate the non-idealities. For example, some authors are focused on equalizers (EQs) [5, 6] to eliminate inter-symbol interference (ISI) caused by lossy channels. Some authors compared single-ended with differential links for graphic memory interfaces [7] and others proposed data encoding technique [8] to reduce simultaneous switching output (SSO) noise due to simultaneous I/O switching.

More recently, some researches have moved toward 3D Integrated Circuit (IC) technology to further enhance the system performance in chip-to-chip communication, where multiple planar device layers stack with short interconnect latencies. Authors in [9, 10] have addressed their focus on Through Silicon Via (TSV), a vertical interconnection between two link chips, since the yield of 3D chips strongly relies on the yield and the number of TSV while other authors concerned about the wide-I/O in mobile SDRAM [11] using TSV-based stacking.

1.2 Motivations

Figure 1.2 illustrates an example of a PHY in a high-speed communication system which is composed of two level shifters (LS), a serializer (SER), a transmitter equalizer (Tx EQ), I/O, a receiver equalizer (Rx EQ), a control unit, a deserializer (DES), a clock data recovery (CDR) and a phase locked loop (PLL). In communication systems, a PHY plays an important role to ensure error free bit-level data transmission between two devices and also it supports electrical interfaces connecting to the channel. In general, tasks of a PHY

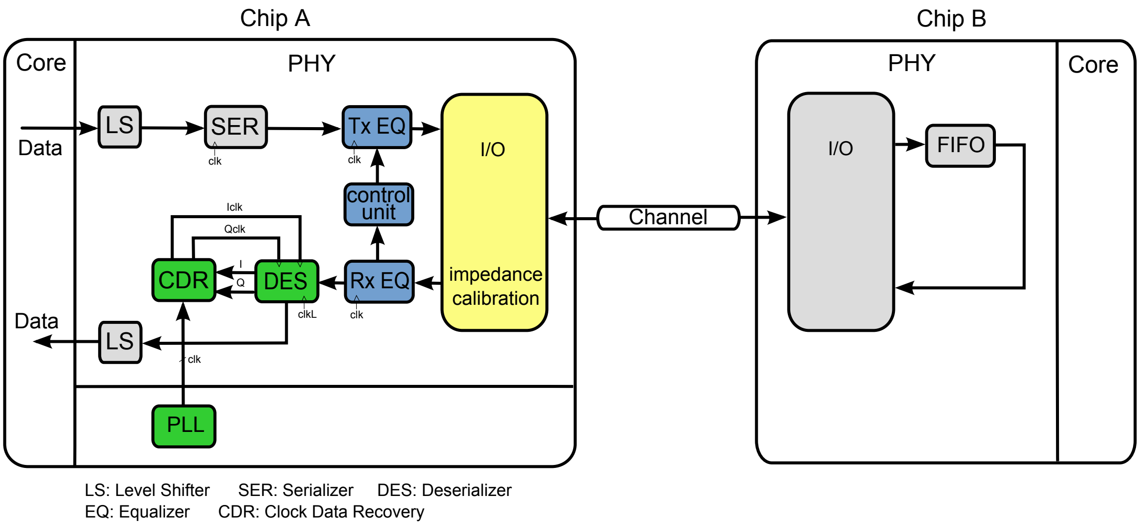


Figure 1.2: An example of a PHY in a high-speed communication system

include but not limited to the following:

- Voltage level translation
- Data encoding and signal modulation in a physically transmittable form
- Data synchronization for synchronous serial links
- Signal equalization to ensure reliable data transmission
- Clock and data recovery
- Forward error correction coding

A PHY link, as its name suggests, is established from a PHY in one link device to a PHY in the other link device. So far, most studies in the PHY link design targeting at high data bandwidth have only been carried out by following standard protocols for their specific applications, e.g. PCI Express for expansion cards, RapidIO Serial in wireless base stations. Although many improvements have been made by using advanced interconnect techniques to minimize data errors and to meet the requirements of the protocols, data have run into limitations of the timing margins on low-cost Printed Circuit Board (PCB) material. The limitations prevent higher speed data transmission in modern communication systems.

In the industry, customized design is usually performed for different demands, which ensures robust and reliable signal transmission at high data rate. However, redesign becomes problematic in terms of cost and time-to-market. In worst case scenarios, undesired issues such as process, voltage and temperature (PVT) variations and random noise contributing to random timing jitter may perturb the system.

Last but not least, the complexity of additional advanced interconnect circuits has made PHY link reach its limit in the total system power consumption, thus reduces the power efficiency. Power dissipated in the PHY link is proportional to the number of the links in both Tx and Rx directions, which implies that to increase the system bandwidth, the number of link is however, constrained due to limited power budget.

Therefore, the challenge in the PHY link design for high-speed communication systems turns out to be adaptive, reliable, power-efficient and cost-effective. The idea and motivation applied for the PHY link design and optimization are as follows.

- **Low cost and adaptability of a PHY link design for high-speed communication systems**: Components such as Tx EQ and Rx EQ comprising a PHY link are frequently employed in a wide variety of high-speed communication systems due to lossy channels on PCB board. For the reason of cost saving, an optimization or a new design of PCB boards is not considered. Hence, a sophisticated adaptive mechanism to allow higher-speed data transmission despite of various low-lost PCB boards is required for future high-speed communication systems.
- **Reliability of a PHY link design for high-speed communication systems**: As data rate increases to several gigabits per second, timing jitter in data mainly caused by jitter in the sampling clock can significantly affect signal quality and data transmission rate. Therefore, a reliable high-speed data transfer is conditioned on low-jitter sampling clock generation.
- **Power-aware design of a PHY link for high-speed communication systems**: The design and architecture of the PHY link mainly focuses on low power dissipation. Reduction in power consumption on a single PHY link offers possibility to increase the number of links for a given power budget, and thus increase the entire system bandwidth. Therefore, an idea with regards to the optimization and structural development of the PHY link based on low power is proposed.

1.3 Research Objectives and Scope

The general objective of the thesis is to present improved algorithms, system architecture and sophisticated design concepts of a power-aware adaptive PHY link for high-speed communication systems, especially components such as CDR, Tx EQ, Rx EQ and I/O circuits. The specific objectives are:

- To optimize a CDR system in the PHY link on the trade-off between jitter generation and power consumption for high-speed data transmission [12]. A three times sampling (3TS) scheme in phase detector (PD) is proposed to decrease the dither in the recovered sampling clock.

- To introduce two adaptive EQ trainings [13] [14] [15] [16] in the PHY link for high-speed communication systems, where different algorithms applied in EQ trainings are either optimized or proposed at both system-level using Matlab/Simulink and gate-level using Matlab/Simulink-Cadence co-simulation [17]. Evaluations of the signal quality and hardware usages are analyzed for different algorithms.
- To reduce total power dissipated in the PHY link for EQ trainings by simplifying system architecture in high-speed communication systems, where Tx EQ and Rx EQ are placed asymmetrically in a single link device.
- To present low-power I/O driver employing hybrid digital impedance calibration technique for PVT compensations with short calibration time [18].

As this dissertation proposes the design and optimization of a power-efficient adaptive PHY link by means of trainings to support high-speed communication systems, the research scope is architectural design, system-level and circuit-level verification for implementation of the PHY link. The main scope of this thesis is:

- Optimization on low-jitter power-efficient CDR system for read training and write training at both system-level and gate-level
- Introduction to Tx and Rx EQ trainings for signal integrity (SI) improvement combined with other interface trainings
- Implementation of different algorithms applied in the EQ trainings at system-level and gate-level
- Analysis of signal quality in the EQ trainings for evaluation
- Efficient mixed-signal design methodology
- Optimization on low-power I/O driver using hybrid digital impedance calibration technique at the gate-level

1.4 Thesis Outline

The remaining chapters are briefly described in the following.

- *Chapter 2*: This chapter gives an overview of various interconnect standards in computer peripherals such as PCI Express and HT. To support these interconnect standards, different PHY links architectures have been proposed to fulfill the growing bandwidth requirements.

- *Chapter 3*: This chapter discusses the design of phase interpolator-based CDR system which targets at low power and low jitter for high-speed communication systems. A CDR based on digital phase interpolators using proportional-integral (PI) controller, PI^2 CDR, is proposed for graphic double data rate 5 (GDDR5) interface trainings, i.e. read training and write training. The proposed CDR is composed of a DES, a PD, a programmable accumulator, a PI controller and four digital phase interpolators instead of two for the purpose to be glitch free. Issues such as glitches during the phase switching and phase variation caused by different loads are discussed. Simulation results show that the proposed PI^2 CDR provides trade-off between low CDR jitter and low power consumption compared to other CDRs such as modified CDR and adaptive CDR. As the proposed CDR has the drawback of high dither, a 3TS CDR is suggested to keep the phase of the recovered clock constant and to stop the current training.

- *Chapter 4*: This chapter describes a simplified system architecture, where the EQs are only applied in one single link device when integrating Tx and Rx EQ trainings into other interface trainings, e.g. read training and write training. This novel system architecture is verified at the gate-level by implementing the Rx equalizer training with both analog and semi-digital circuits while the Tx equalizer training uses different algorithms 1) Least Mean Square (LMS) algorithm 2) pilot signal/-peak detection 3) direct calculation implemented with semi-digital and full-digital circuits. Among the investigated algorithms results show that LMS in semi-digital Rx and Tx equalizer is the best performing algorithm in the high-speed communication systems with medium consumed die areas. Compared to other algorithms, it can reduce the ISI caused by both pre- and post-cursors of the channel.

- *Chapter 5*: This chapter introduces a power-aware hybrid digital impedance calibration technique with short calibration time as I/O interfaces require precision impedance matching to maintain the SI and avoid signal reflections due to PVT variations in advanced high-speed communication systems. In comparison to the conventional binary search algorithm the proposed hybrid algorithms can reduce the calibration time by more than 12% and power by 8.91mW for the corners close to reference Vdd/2. The impedances are calibrated within +/-2% of target impedance for PFET and +3.3%/-2.8% for NFET.

- *Chapter 6*: The new contributions of this thesis are summarized in this chapter. Any directions for future works will also be briefly described in this chapter.

Chapter 2

Interconnect Standards in Computer Peripherals

Contents

2.1 PCI Express 8
2.2 HyperTransport 9
2.3 DDRx 10
2.4 GDDRx 12
2.5 Summary 14

Many standards that use physical layers are predefined to interconnect devices from different vendors. The layer that interfaces to the physical world determines both the physical and electrical characteristics of the protocol. In general, interconnect standards

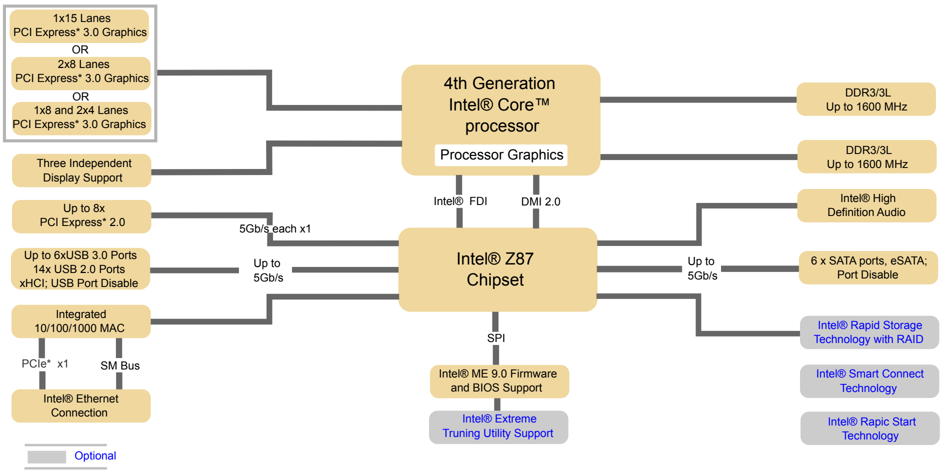

1x15 Lanes
PCI Express* 3.0 Graphics
OR
2x8 Lanes
PCI Express* 3.0 Graphics
OR
1x8 and 2x4 Lanes
PCI Express* 3.0 Graphics
Three Independent Display Support
Up to 8x PCI Express* 2.0
5Gb/s each x1
Up to 6xUSB 3.0 Ports
14x USB 2.0 Ports
xHCI; USB Port Disable
Up to 5Gb/s
Integrated 10/100/1000 MAC
PCIe* x1
SM Bus
Intel® Ethernet Connection
Optional
4th Generation Intel® Core™ processor
Processor Graphics
Intel® FDI
DMI 2.0
Intel® Z87 Chipset
SPI
Intel® ME 9.0 Firmware and BIOS Support
Intel® Extreme Truning Utility Support
DDR3/3L Up to 1600 MHz
DDR3/3L Up to 1600 MHz
Intel® High Definition Audio
Up to 5Gb/s
6 x SATA ports, eSATA; Port Disable
Intel® Rapid Storage Technology with RAID
Intel® Smart Connect Technology
Intel® Rapic Start Technology

Figure 2.1: Intel® Z87 express chipset platform block diagram

can be applied in either network (e.g. Ethernet) or computer peripherals (e.g. PCI express). As an example, Figure 2.1 illustrates the 4th generation Intel® Core™ processor that features 64-bit, multi-core processors with Intel® Z87 express chip platform block diagram [19], which supports 16 lanes PCI express 3.0, 8 PCI express 2.0 ports, 6 SATA ports, 14 USB ports, etc. Trend shows that the number of interconnect standards is increasing and various interconnect standards are under continuous development and improvement in fast transfer rate. In this chapter, an overview of various interconnect standards in computer peripherals is given.

2.1 PCI Express

PCI express [20] is an extension of PCI bus developed for high-speed backplane applications as parallel architecture in PCI bus can hardly provide feasible transfer data for fast peripherals and graphics interfaces. Figure 2.2 demonstrates an example of PHY block diagram in a serial interface PCI express, where parallel data are first serialized and encoded using 8b/10b with embedded clock information at the Tx side. Compared to the source-synchronous clocking scheme where clock is forwarded along with the data from Tx chip to Rx chip, this architecture requires no clock pin. Instead, CDR at the Rx side is employed to reduce the overhead of deskew circuits that can remove the skew between data and clock signals or the skew between parallel lanes.

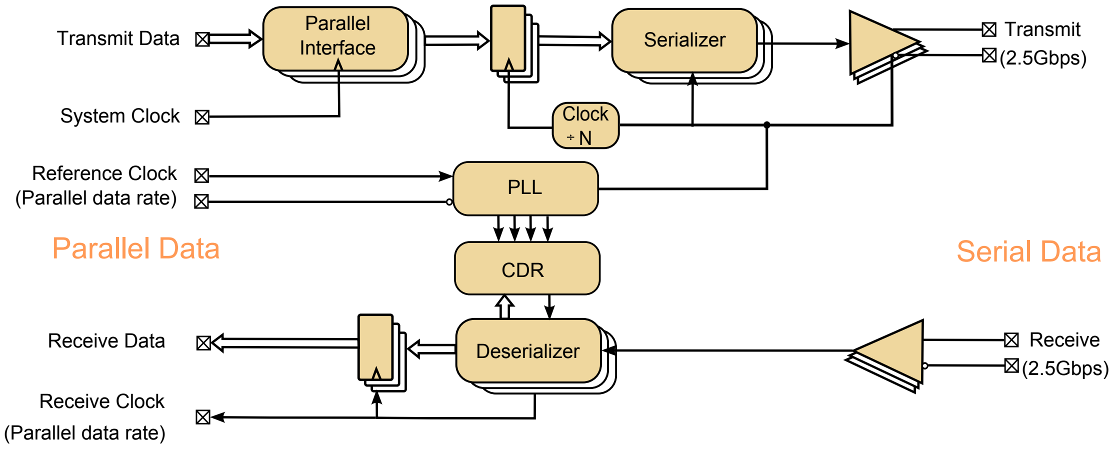


Figure 2.2: PHY block diagram in PCI express

Furthermore, the PCI express features point-to-point architecture between the master and the peripheral using wired interfaces, where one pair of wires is for transmitting and one pair for receiving. Differential signaling makes PCI express superior to single-ended in noise margins and provides flexibility in varying supply voltages for both Tx and Rx chips. When data is transmitted, only one bit is allowed every cycle. On a single connection (either to transmit or to receive) maximum data transfer rate of 2.5 Gbps can be achieved. For various usages, e.g. graphics adapters, PCI express link can be configured up to 32 lanes, which deliver a throughput of 16 GBps at the maximum (20% 8b/10b conversion penalty).

2.2 HyperTransport

HT [21] features point-to-point source-synchronous interconnects with one clock signal per byte. Figure 2.3 illustrates an example of HT links that utilize bus width up to 32 lanes for chip-to-chip interconnects. To minimize pin counts, no separate command or address lanes are required in the HT interface where a signal CTL indicates whether a signal CAD represents control or data packets. Moreover, power consumption is saved with no overhead of serializer, deserializer as well as 8b/10b conversion.

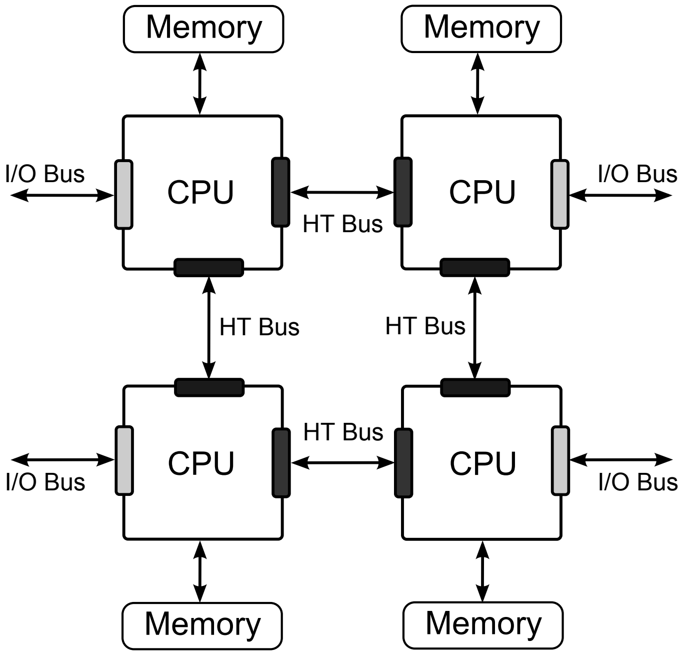


Figure 2.3: An example of HyperTransport links for chip-to-chip interconnect

Requirements for various compliant channels in the time domain for HT are well defined in [22]. Meanwhile, a number of fixed Tx equalization settings and optional Rx equalization settings are provided as well to compensate the non-idealties in those channels. Similar to PCI express, HT also adopts differential signaling. Figure 2.4 demonstrates a low-voltage differential signaling with differential impedance of 100Ω and characteristic line impedance of 60Ω. HT also uses training and equalization schemes for high performance and signal reliability.

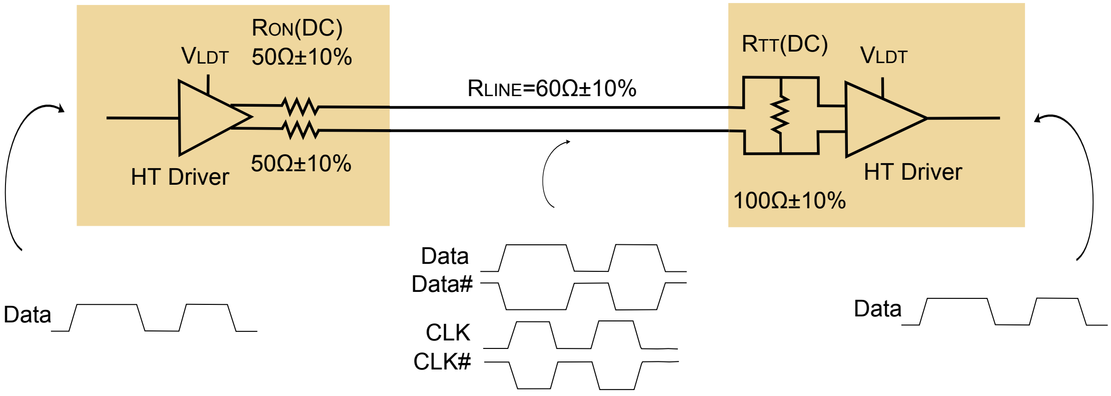


Figure 2.4: A low-voltage differential signaling in HyperTransport links

2.3 DDRx

Double data rate synchronous Dynamic Random Access Memories (DDR SDRAMs) [23] such as DDR2 and DDR3 are memory integrated circuits. As high performance PC memories, generations of DDR memories have significantly been advanced in the data rate. Figure 2.5 shows the growth of data rate per pin for DDRx memories from single dara rate (SDR) memories. The evolution in data bandwidth is mainly contributed by fast clock frequencies.

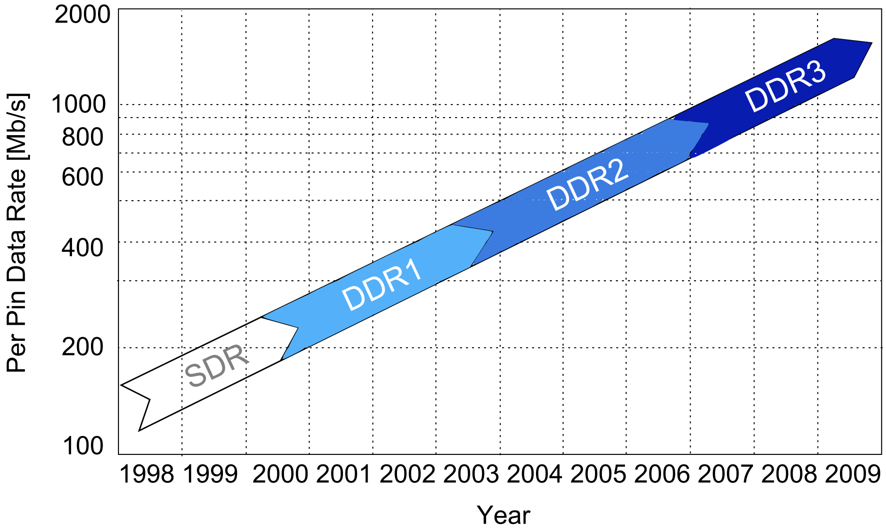


Figure 2.5: Data rate per pin in generation of DDR memories

Figure 2.6 shows a simplified PHY link in DDRx memories using source-synchronous clocking scheme. In this architecture no CDR circuits are applied in both Tx and Rx chips by using bidirectional DQs data strobe signal that indicates the validness of the data.

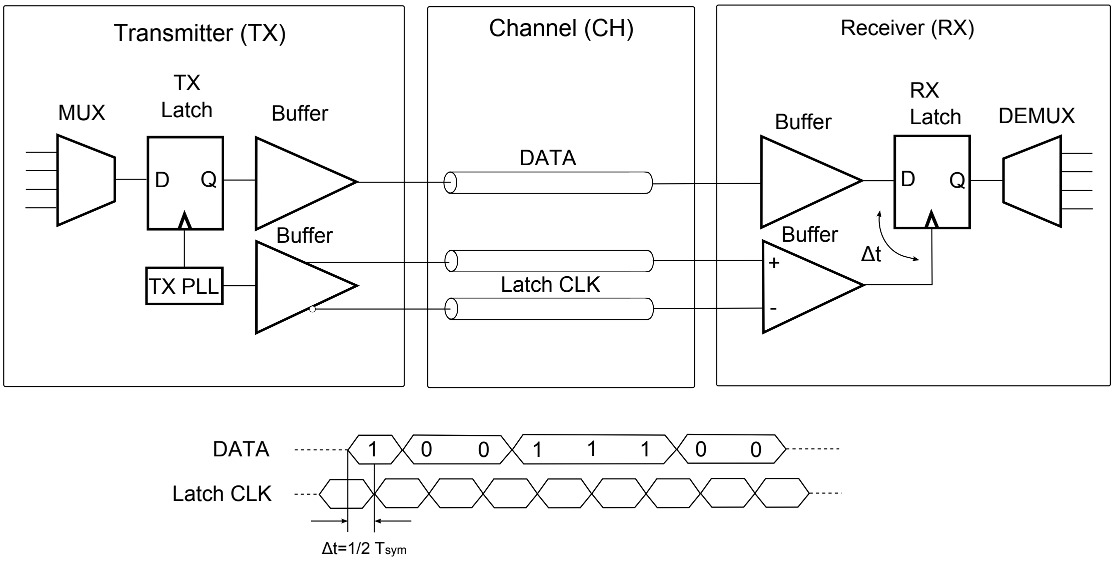
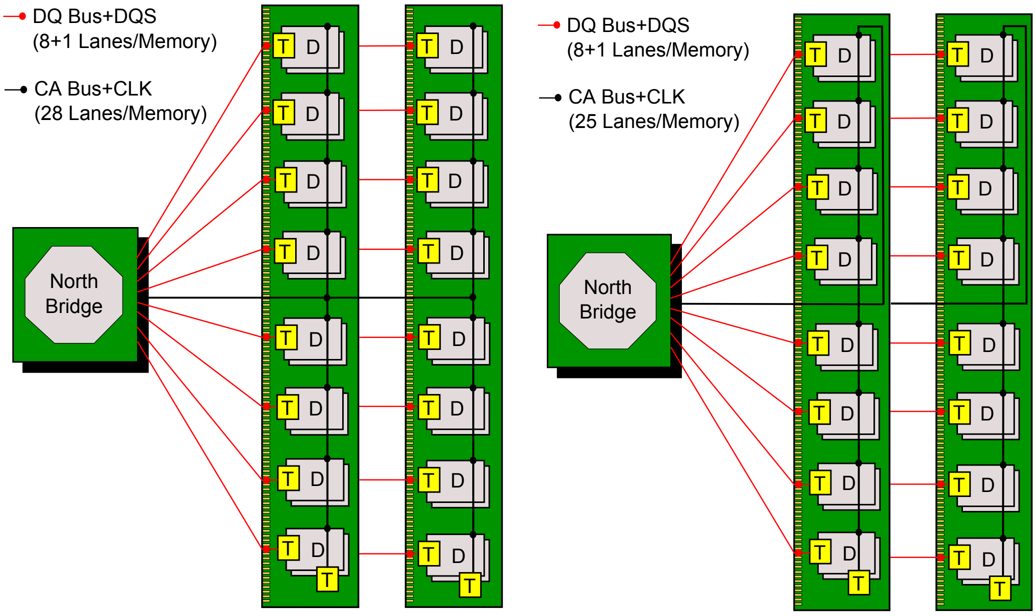


Figure 2.6: A simplified PHY link in DDR memories

Figure 2.7 demonstrates interconnects between north bridge and different memory modules (DDR2 and DDR3). In DDR2 dual in-line memory module (DIMM) the control/address/clock signals are routed to the memory devices in a tree-shaped topology while in DDR3 DIMM in a fly-by topology to improve the SI. The change in the architecture, however, introduces time skew due to the different delays required when signals



(a) DDR2 memory modules

(b) DDR3 memory modules

Figure 2.7: Interconnect between north bridge and memory modules

are passed through each memory sequentially. To compensate the skew, PHY in memory controller needs to support write and read leveling, which enables DQs to move towards the rising edge of the clock.

2.4 GDDRx

As high bandwidth memories, graphic DDRx (GDDRx) [23] in graphic cards is commonly used in game consoles and desktop personal computers. Figure 2.8 shows the time of burst data for various memories. It is clearly to see that GDDR5, currently the highest bandwidth memory commercially available in the market, has the fastest burst data speed compared to the other memories.

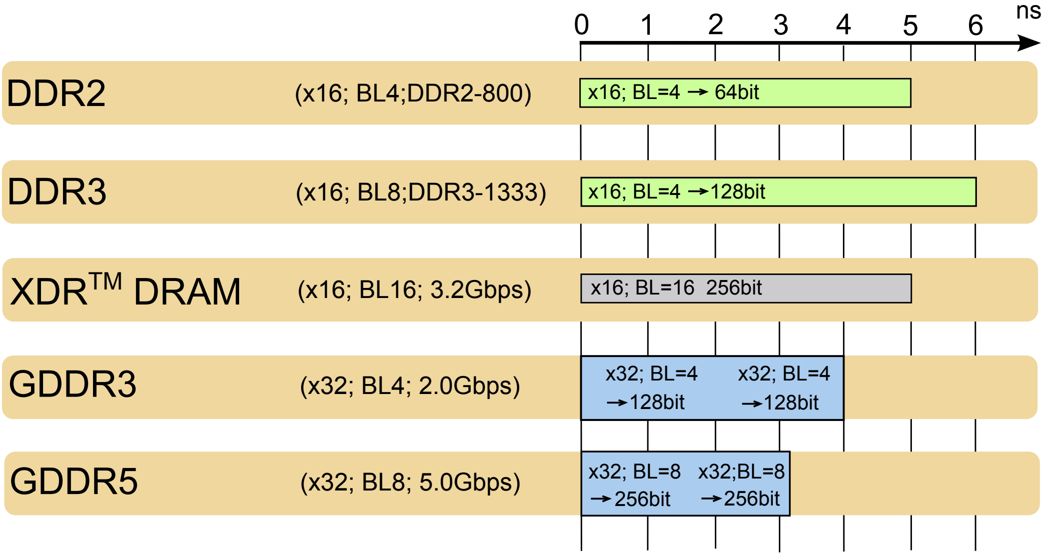


Figure 2.8: Time of burst data for various memories

Different from DDR3, GDDR5 inherited no data strobe signal. Figure 2.9 shows a GDDR5 interface, where a memory controller (master IC) and a GDDR5 memory (slave IC) are connected via channels. Source-synchronous clocking scheme is adopted in GDDR5 memory systems. Unidirectional signals such as address/command (ADDR/CMD) run at the system clock (CK) while bidirectional signals such as data (DQ) work at the data clock (WCK), twice as high as CK. Error detection and correction signal (EDC) which carries different feedback information for different modes is also unidirectional and runs asynchronously. For the sake of simplicity many other control signals e.g. scan enable are not drawn in the figure.

In GDDR5 memory systems, the timing of memory controller and memory is adapted by introducing training mechanisms, which benefits both flexibility and low system cost. Several interface trainings i.e. address training, wck2ck training, read training, write

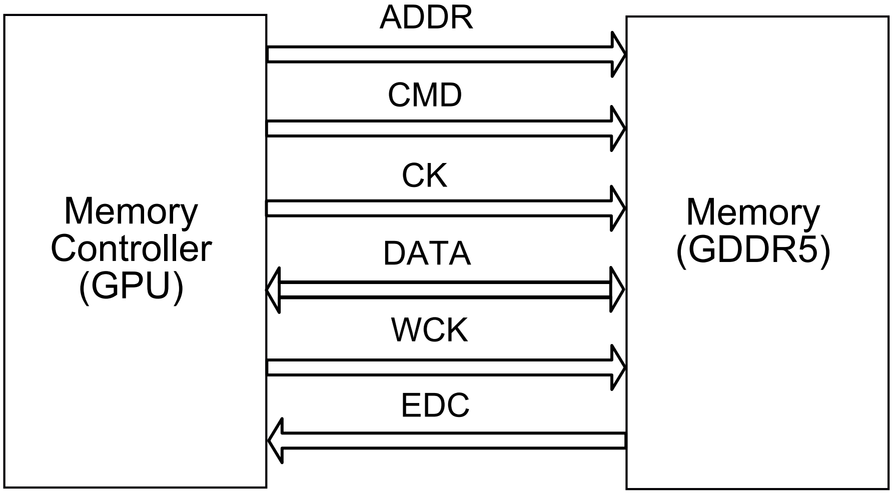


Figure 2.9: GDDR5 interface

training are predefined after power up in JEDEC [8]. The training sequence has to be operated to guarantee a defined interface timing relationship for high-speed data transmission in GDDR5 as illustrated in Figure 2.10. Address training is aimed to adjust the

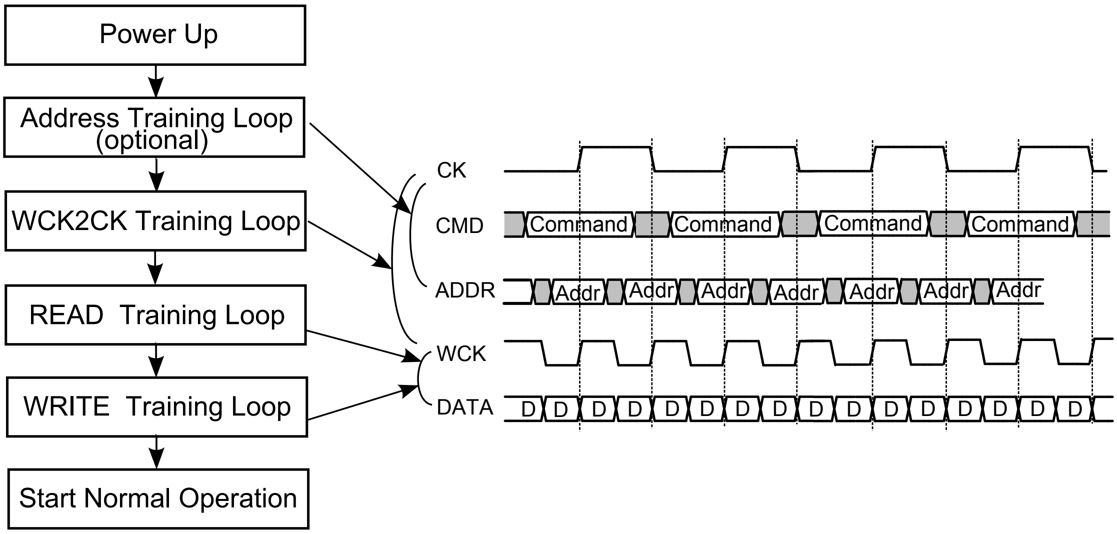


Figure 2.10: Conventional interface training sequence

phase of address signals relative to the CK to meet the timing at the DRAM. It is optimal as long as address input setup time (tAS) and address input hold time (tAH) fulfill the system specification. WCK training is to adjust the WCK phase relative to the CK to meet the timing at the DRAM. As next, read training is operated in order to adjust the receive phase for the read data on DQs to meet the timing at the sampler in the memory controller. Once read training is successful, write training is followed to adjust the transmit phase for the write data on DQs to meet timing at the sampler in DRAM.

2.5 Summary

To summarize, many efforts have been made to explore different PHY links architectures supporting various interconnect standards in computer peripherals. Some PHYs require CDR circuits under the consideration of limited pin counts and others employ deskew circuits to compensate the different arrival time of the signals. With regards to EQs, some PHYs provide no EQs as signal quality is not significantly affected at low transmission speed with all predefined system requirements and others have fixed settings of EQs. Furthermore, different PHYs also specify various signaling requirements in I/O interfaces. The challenge involved in PHY link design is to provide reliable high-speed data transmission with considerably low power and cost. Issues such as interface timings have to be guaranteed when low-cost PCB boards are used. Therefore, a smart system which can adapt different system environments in terms of various channel characteristics and PVT variations turns out to be trendy for future high-speed communication systems. In the following chapters, CDR, EQ trainings and I/O driver are discussed in details to fulfill the requirements for the power-efficient and reliable data transmission.

Chapter 3

Phase Interpolator Based Clock Data Recovery

Contents

3.1 State-of-the-art **17**

3.2 PI^2 CDR Simulink Model **21**

3.2.1 Model Description 21

3.2.2 Simulation Results 22

3.3 PI^2 CDR Components Design **25**

3.3.1 Digital Phase Interpolator 25

3.3.2 Multiplexer 26

3.3.3 Three Times Sampling CDR 28

3.4 Simulation Results **29**

3.4.1 Summary 31

In most high-speed asynchronous or even plesiochronous communication systems, a CDR is an indispensable part at the receiving end to extract both the sampling clock and transmitted data. In case of PVT variations in high-speed systems, clocks of samplers in both transmit and reverse directions with the same frequency, but variations in the phase may lead to signal incorrectness. By applying CDR the sampling time margin is maximized and bit error rate (BER) is improved.

In general, CDR architectures provide higher data-rates at low power efficiency. For applications using parallel links, the total power consumption and complexity of CDR architectures will increase linearly when the number of links grows.

Figure 3.1 illustrates read training [8] in a GDDR5 memory system. A memory controller sends training patterns to first in first out (FIFO) in GDDR5 through a trained

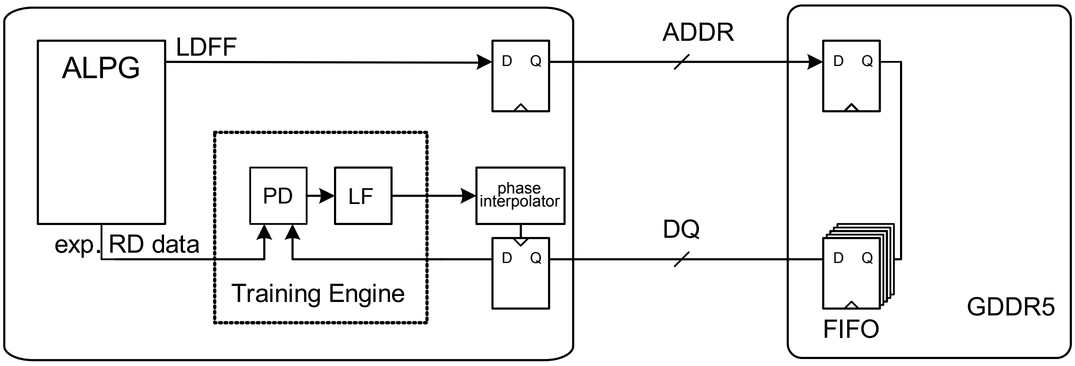


Figure 3.1: Read training in a GDDR5 memory system

address channel that usually has a better SI than high-speed data channel with the command load FIFO (LDFF). After successfully writing the data to the FIFO, the command read training (RDTR) is started. By comparing the read data and the expected read data, the receive phase is adjusted in the memory controller. To achieve the right clock phase, a CDR circuit composed of PD, low pass filter (LF) and phase interpolator is usually applied.

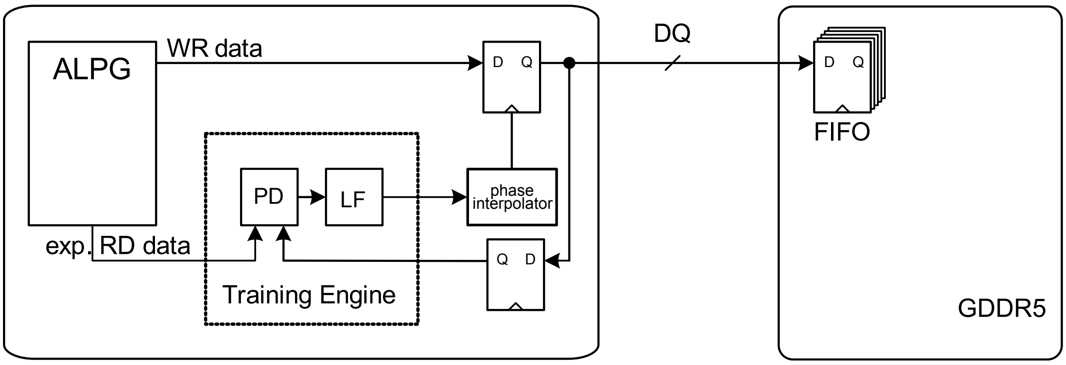


Figure 3.2: Write training in a GDDR5 memory system

Once read training is finished, write training [8] depicted in Figure 3.2 begins. Data are first written into the FIFO through a DQ channel with the write training (WRTR) command and data are transferred error free due to the read training back into the memory controller with RDTR commands. By comparing the write data and the expected read data, the transmit phase is adjusted in the memory controller.

As a result, data eyes are sent at different time by changing the sampling clock phase in the controller so that data arrive simultaneously at the Rx. Figure 3.3 illustrates the timing situation before and after write training for GDDR5.

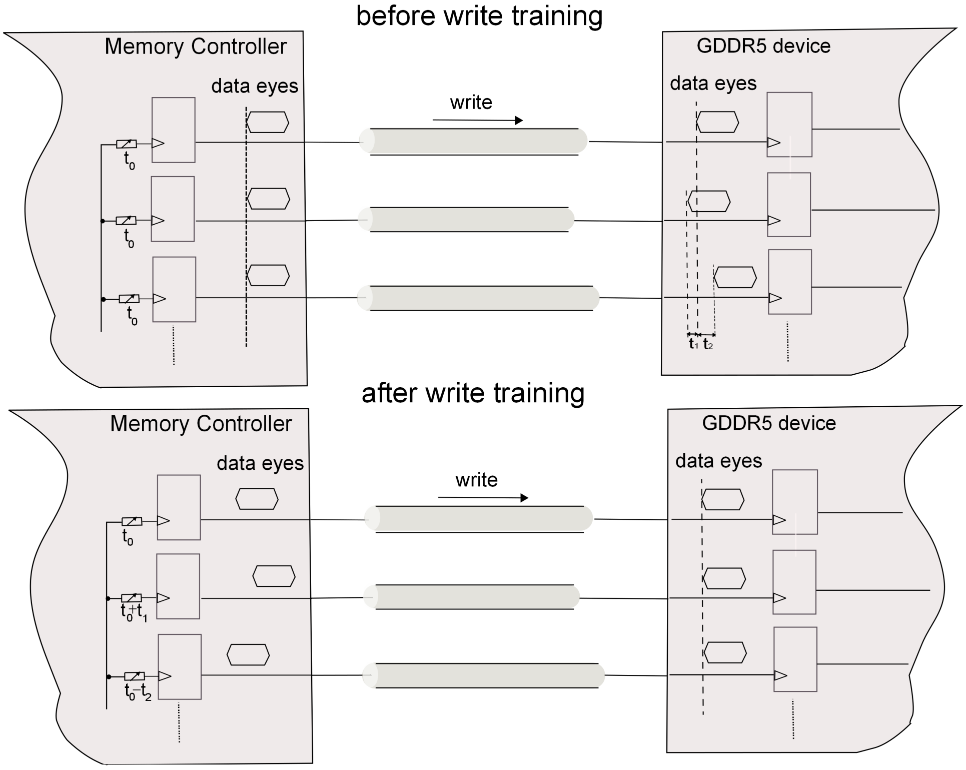


Figure 3.3: CDR trainings in high-speed communication systems

However, one major problem in both read training and write training is when to compare the expected data and received data due to different component delays (channel delays and other component delays) for various vendors. To address the problem, the proposed CDR architecture is mainly inherited from the conventional CDR, but focused on the trade-off between power and CDR jitter.

3.1 State-of-the-art

A wide variety of CDRs are proposed in the papers. Authors in [24] suggested a digital CDR using analog digital converter (ADC)-based front-end. The received analog signals are first equalized through a continuous-time analog EQ and its output is then converted into 5-bit digital data by flash ADCs. In this architecture, the number of valid data is adjusted instead of the phase of the sampling clock by applying zero cross PD. When data is faster, one additional data is inserted. Otherwise one duplicated data is removed. Authors in [25] proposed a 2-threshold eye-tracking CDR when loop-unrolling decision feedback EQ (DFE) is applied. Due to the fact that the time difference between the edge and the eye center of the post-DFE signal is not equal to 0.5UI, two different thresholds operated for edge and data signals are required. The drawback of this architecture is its significant circuit complexity. In [26], authors suggested a phase-tracking CDR with dynamic control of jitter transfer bandwidth to increase the jitter tolerance. In [27], the phase tracking CDR is optimized with a pre-detector to reduce the close-loop latency. The pre-detector is provided to improve intrinsic jitter caused by the update periods needed in the frequency detector and the accumulator of finite-state-machine (FSM). In the pre-detector the signs of input and output in FSM are compared. Authors in [28] proposed sub-rate phase tracking CDR using fully-differential approach in the tail current sources in phase interpolator design which avoids switching clocks and thus resulting better clock jitter.

In general, CDRs are classified into several categories [29] such as PLL based CDR [30], delay locked loop (DLL) based CDR [31] and phase interpolator based CDR [32]. Due to the dual tracking loop in both phase and frequency, PLL-based CDR provides good input jitter rejection. But in the mean time, it suffers from jitter accumulation and stability issues by high-order PLL. Instead, a DLL-based CDR using voltage-controlled delay line has shown its benefits in the stability analysis. However, the DLL-based CDR has limited phase capturing range, which can be solved by replacing the voltage-controlled delay line with a phase interpolator. Other CDRs such as dual-edge injection-locking-based CDR [33], baud-rate timing recovery [34], phase tracking CDR with sleeping mode [35] and CDR using symbol-rate PD [36] targeting at fast lock or low power and area consumption are also proposed.

Among different types of CDR structures, phase interporlator-based CDR has been gaining popularity due to many advantages. The primary advantage of the phase interpolator based CDR is that a single clock source can be shared by multiple phase interpolator based CDR circuits. This feature reduces the system power consumption, die size overhead as well as cross-talk between adjacent channels [37]. Furthermore, the phase interpolator based CDR offers short re-locking time, better noise suppression during phase tracking and simple compensation scheme for offset due to device mismatching [38].

Figure 3.4 shows the basic building block of a conventional CDR [39]. After sampling the data, PD generates early-late information, which is accumulated in the controller. The phase interpolator adjusts the phase of its output clock according to the output signal of the controller. The updated clocks are then applied to the sample block.

A modified phase interpolator based CDR [40] is shown in Figure 3.5. Compared to the conventional CDR, this architecture includes the deserializer in the synchronization loop that enables most digital components to work with low sampling clocks, and thus reduces the complexity and the power of the whole system. The system is composed of a PLL, a deserializer, a PD, a programmable accumulator, a binary phase accumulator and two analog phase interpolators. The 1 to 8 deserializer first samples the incoming data with half-rate in-phase (ICLK) and quadrature (QCLK) clocks and synchronizes the outputs to the low system clock (CLKL) which is one quarter of the ICLK. A modified Alexander PD [41] takes two parallel bytes from the deserializer and outputs a moving average of phase error to obtain a high possibility of phase error detection for the present jitter. A programmable accumulator Pre-filter provides an updated carry signal when a

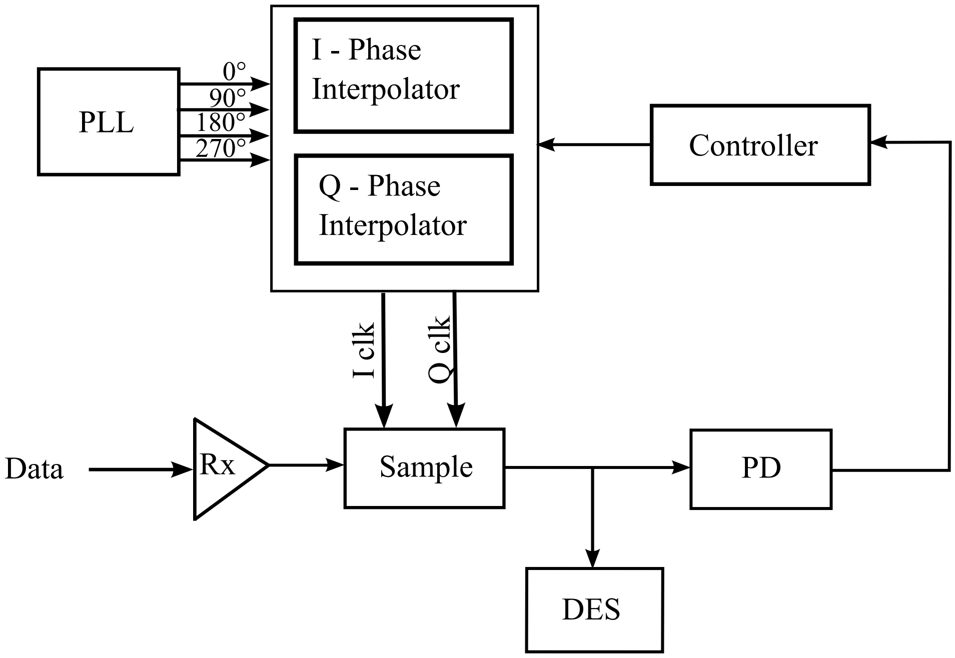


Figure 3.4: Conventional CDR block diagram

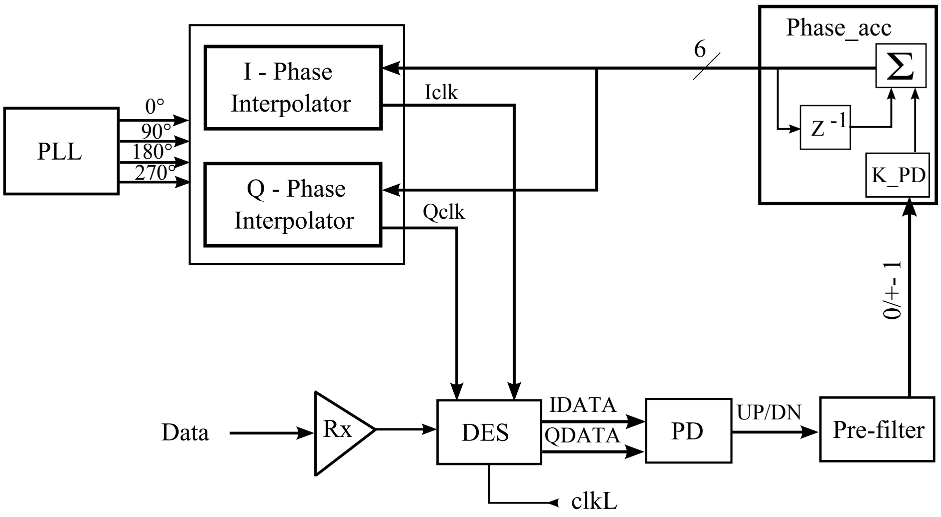


Figure 3.5: Modified CDR block diagram

certain threshold Pre_filt is reached. A phase accumulator accumulates the carry signal with the PD gain k_PD and converts the result into digital bits. By feeding the digital information and 4-phase clocks generated by PLL into two phase interpolators, ICLK and QCLK are recovered. This CDR generates a low dither on the recovered clock due to the linear accumulators. However, high CDR jitter is a major drawback of this CDR.

To reduce the CDR jitter, an early-late PD with adaptive algorithm and PI controller are utilized in an alternative CDR [42] shown in Figure 3.6. The sample and DEMUX

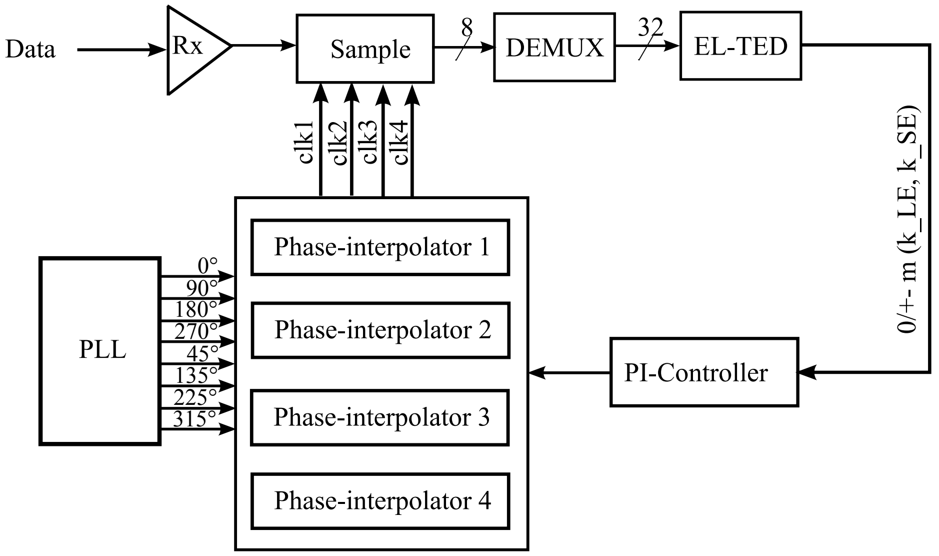


Figure 3.6: Adaptive CDR block diagram

blocks generate 4 oversampling points for each symbol. For eight symbols 32 parallel binary data are provided at the input of the early-late timing error detector (EL-TED).

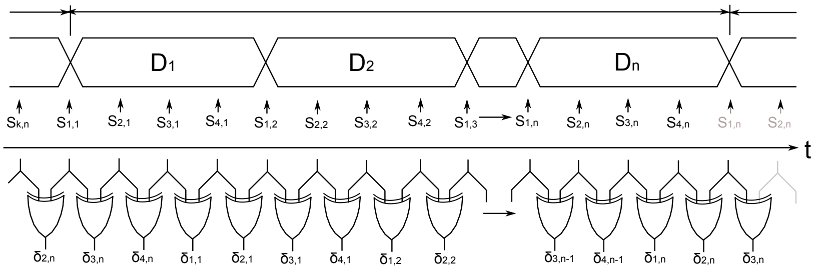


Figure 3.7: Evaluation of data transitions

Figure 3.7 shows the evaluation of data transitions by oversampling. Depending on the average phase error δ at each oversampling point, an early or late information $\epsilon_{b,w}$ which is a function of k_LE and k_SE is adaptively generated. This can be expressed as

$$f_1(k\_LE) = k\_LE * (\sum_{n=1}^{L} |\delta_{1,n}| - \sum_{n=1}^{L} |\delta_{4,n-1}|) \tag{3.1}$$

$$f_2(k\_SE) = k\_SE * (\sum_{n=1}^{L} |\delta_{2,n}| - \sum_{n=1}^{L} |\delta_{3,n-1}|) \tag{3.2}$$

$$\epsilon_{b,w} = [f_1(k\_LE) + f_2(k\_SE)] * f(tr) \tag{3.3}$$

where k_LE, k_SE and $f(tr)$ represent the gain of PD for large errors, the gain of PD for small errors and a function of input data transitions, respectively. L is the total length of data symbols, where the early-late information is averaged. In our case, L is set to 8. The data transitions in the middle of one symbol $\delta_{1,n}$ and $\delta_{4,n-1}$ are weighted with a large value k_LE, which indicates large phase errors taking place at these positions. However, transitions at both beginning and end of the symbol are weighted with k_SE which indicates small phase errors. The PI controller takes the resulted early-late error information and provides the digital controller bits for four analog phase interpolators which can be expressed as

$$phase_{b,w} = (\epsilon_{b,w} * \boldsymbol{\beta} * \boldsymbol{E}^T + \sum_{n=1}^{L} \epsilon_{b,w} * \boldsymbol{\alpha} * \boldsymbol{E}^T) * k\_PD \tag{3.4}$$

where vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the coefficients of the PI controller, $\boldsymbol{E}$ represents identity matrix and k_PD is the fundamental gain of the PD.

The adaptive CDR is suitable for very low CDR jitter requirement and usually it provides a fast locking time due to its fast adaptivity of the input jitter. However, it has the drawback of high dither in the output clock and consumes considerable power by using four phase interpolators. By combining the advantage of the first architecture that has low dither and the advantage of the second architecture that outputs low CDR jitter, a novel system architecture, PI^2 CDR, is proposed and described in details in the following section.

3.2 PI^2 CDR Simulink Model

3.2.1 Model Description

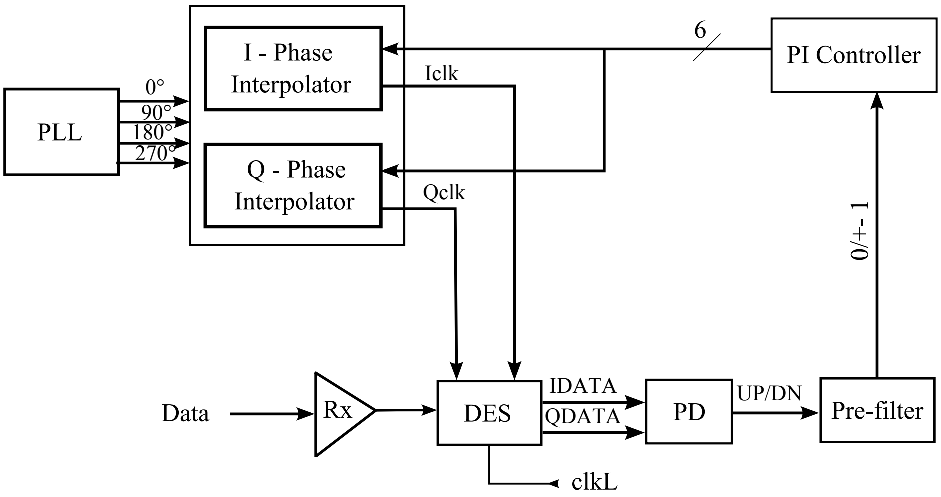


Figure 3.8: Proposed PI^2 CDR block diagram

Figure 3.8 shows the proposed PI^2 CDR architecture using PI controller, where the phase accumulator in the modified CDR is replaced. The internal block of the PI controller

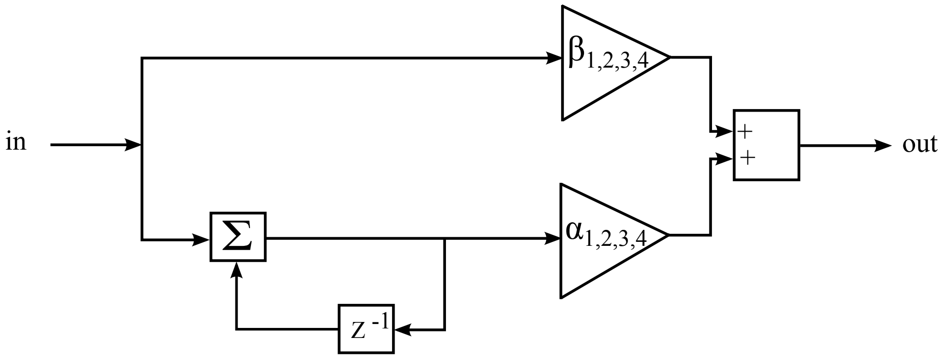


Figure 3.9: PI controller block diagram

is given in Figure 3.9. Instead of using $\epsilon_{b,w}$ in the adaptive CDR, which is dependent on the k_LE and k_SE and δ, the output of Pre_filter is connected to the input of the PI controller in the proposed CDR. Compared to the output of the phase accumulator in the modified CDR

$$phase_{b,w} = (\sum_{n=1}^{L} \epsilon_b) * k\_PD \tag{3.5}$$

the output of the PI controller can be expressed as

$$phase_{b,w} = (\epsilon_b * \boldsymbol{\beta} * \boldsymbol{E}^T + \sum_{n=1}^{L} \epsilon_b * \boldsymbol{\alpha} * \boldsymbol{E}^T) * k\_PD \tag{3.6}$$

where ϵ_b is the output of the programmable accumulator. In the implementation $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are chosen as $(2^{-10}, 2^{-9}, 2^{-8}, 2^{-7})$ and $(2^{-8}, 2^{-7}, 2^{-6}, 2^{-5})$, respectively, realized by shift registers. The major advantage of the proposed CDR is that the circuit complexity is kept as simple as the modified CDR.

Figure 3.10 shows the variation of the controlled variable for varieties of controllers such as proportional (P) controller and integral (I) controller. By adding the P controller into the phase accumulator, which works similar to an I controller, the system outputs a low overshoot due to the P controller with no steady-state error because of the I controller. Furthermore, it can be observed that using PI controller the system responses fast to the controlled variable. In the figure, the proportional-integral-derivative (PID) controller shows even better result in terms of the response time than the PI controller. However, additional hardware is needed in the PID controller that increases the system complexity.

3.2.2 Simulation Results

The total simulation in Matlab/Simulink takes 280ns where 1400 data symbols are sent (data rate= 5Gbps). The sent data is a repeating sequence [1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0] with random jitter that changes every 200ps. One of the reference clock with 0 degree generated by PLL is assumed to have a 90 degree phase shift

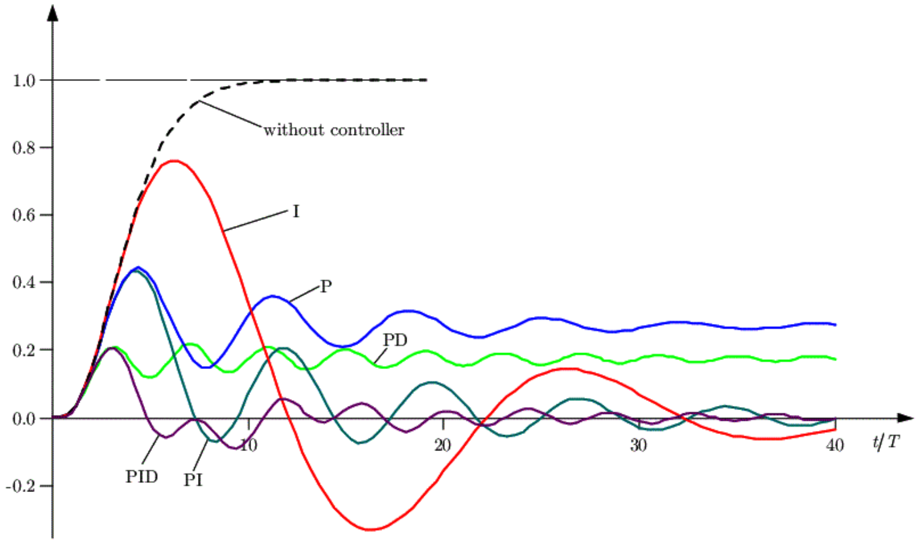


Figure 3.10: Controlled variable versus time for different controllers

to the ideal sampling clock (middle of the data). Table 3.1 shows the variable settings in modified, adaptive and PI^2 CDRs at system-level simulation.

Table 3.1: Variable settings in modified, adaptive and PI^2 CDR systems

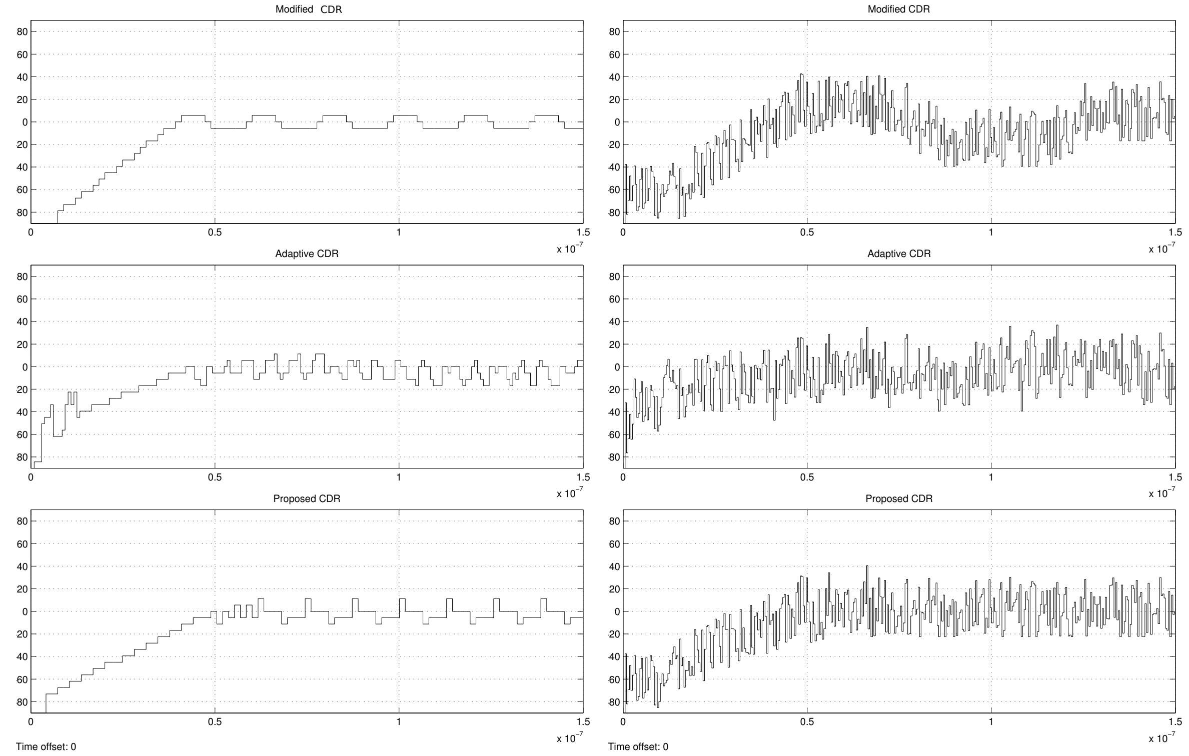
	Variables
Modified CDR	k_PD=725; Pre_filt=2 or 8
Adaptive CDR	k_PD=26; k_LE=4; k_SE=1
PI^2 CDR	k_PD=32; Pre_filt=2 or 8

For the simulation the loop gain of the CDR is chosen so that the locking time of all three models is approximately the same. Figure 3.11a and Figure 3.11b depict the phase error between the ideal sampling point and the actual phase of the adaptive CDR without random jitter and with maximal random jitter 60e-12s, respectively. In Figure 3.11a, the locking process is complete when the CDR begins to dither around the ideal sampling point that happens in the setup between time point 0.4e-7s and 0.5e-7s.

Table 3.2: Phase errors in modified, adaptive and PI^2 CDR systems

Maximal RJ in second	0 Pre_filt=2	6e-12 Pre_filt=2	18e-12 Pre_filt=2	60e-12 Pre_filt=2	6e-12 Pre_filt=8	18e-12 Pre_filt=8	60e-12 Pre_filt=8
Modified CDR	11.0711	11.0224	12.3257	17.4357	14.2508	16.7797	19.5531
Adaptive CDR	10.6471	10.3966	10.8724	16.3235	10.2693	10.3928	16.1897
Proposed CDR	11.5353	11.3381	11.8110	16.5613	12.9823	12.8890	17.3397

Table 3.2 shows the phase errors in degree for modified, adaptive and proposed CDR systems when the maximal random jitter and threshold of the programmable accumulator vary. The CDR jitter is calculated by the sum of the absolute jitter at each simulation point divided by the number of simulation points. The simulation results show that the adaptive CDR has the lowest CDR jitter among all three models due to its capability of following fast input data phase changes, but with the trade-off of large dither. The modified



(a) Phase error (random_jitter=0)

(b) Phase error (maximal random_jitter=60e-12s and Pre_filt=8)

Figure 3.11: Phase error between ideal sampling point and actual phase of adaptive CDR with-/without jitter

CDR and proposed PI^2 CDR do not differentiate too much in the phase error, especially in the case of very small jitter (jitter=0s and 6e-12s). But by higher jitter (e.g. jitter=60e-12), the proposed CDR shows better performance. The usage of a high value in the threshold of the programmable accumulator is not recommended in both modified and proposed CDR when the target is to achieve a low CDR phase error. But it is an option to reduce the frequency at which the dither changes. For Pre_filt=8 the phase error in the proposed CDR is 1.3 degree up to 3.9 degree less than the modified CDR. When large CDR jitter is expected or small phase error with less clock changes (high value on Pre_filt) is required, the proposed model is usually preferred.

Table 3.3: Comparison among modified, adaptive and PI^2 CDRs ÆŠwith constant locking time

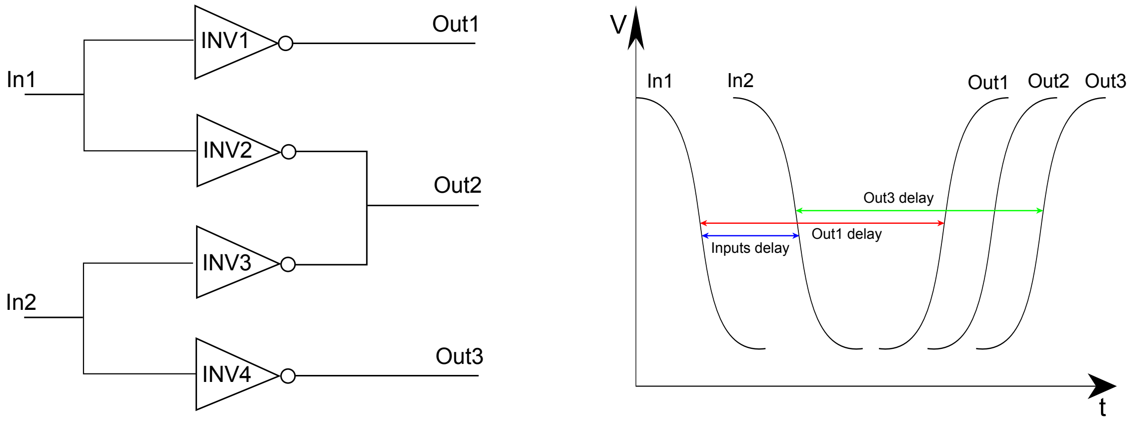
	Advantages	Disadvantages
Modified CDR	Low dither	High CDR jitter
Adaptive CDR	Lowest CDR jitter	High power consumption; Highest dither
Proposed CDR	Low CDR jitter with lower power consumption	High dither

Table 3.3 depicts the comparison among modified, adaptive and proposed PI^2 CDR systems in terms of the CDR jitter, dither and power consumption under the condition that locking time is kept the same for all models.

3.3 PI^2 CDR Components Design

3.3.1 Digital Phase Interpolator

Due to mismatch, high current leakage, especially in modern technologies, and high power consumption [43], analog phase interpolator becomes less competitive than digital phase interpolator. Based on [43], digital phase interpolator composed of four inverters shown in Figure 3.12 is applied in the proposed PI^2 CDR, where inverter 1 and inverter 4 produce delayed and inverted outputs from their inputs *In1* and *In2*. *Out2* is the interpolated output.



(a) An 1-bit cell schematic

(b) Inputs and outputs waves of an 1-bit cell

Figure 3.12: 1-bit cell

Figure 3.13 illustrates an example of a 3-bit phase interpolator. Out of two inputs (phase 0° and phase 45°) a 2nd phase step $(2 * 5.625^\circ = 11.25^\circ)$ represented by binary code "010" is generated. Due to the inverter delay, even same phases, for example, four 0° phases, have different phase positions.

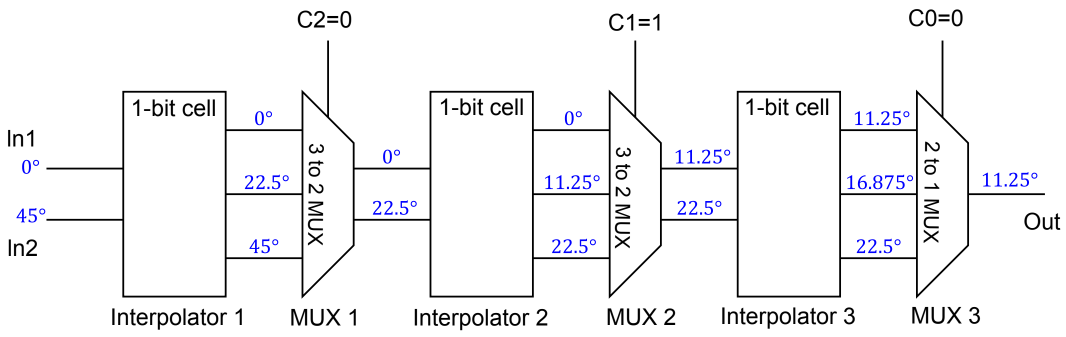
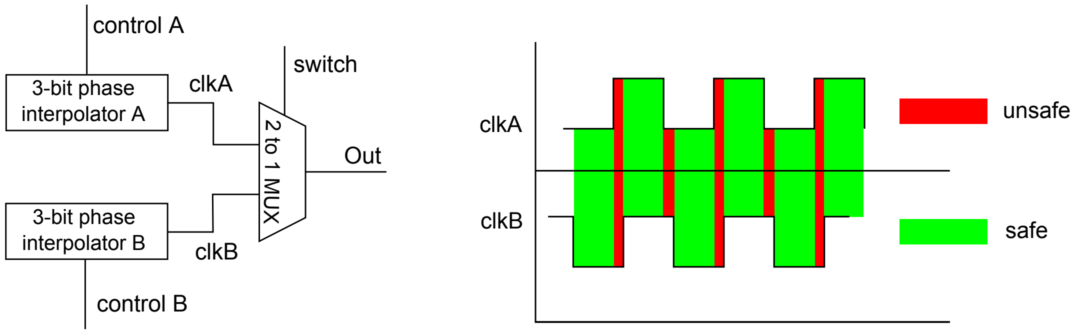


Figure 3.13: A 3-bit phase interpolator

Different from [43], the implemented phase interpolator works in the gigahertz range (2.5GHz), not megahertz. The rise/fall time of input signals has to be about 0.4 to 2.6 times longer than the phase delay between two input signals in TSMC 65nm LP technology depending on the stage where the cell lies. Additionally, multiplexers are required at the input of the phase interpolator to switch between reference clocks with small phase differences as well as between two stages. However, unwanted signal spikes take place during the switching process. To avoid this, two 3-bit digital phase interpolators are applied illustrated in Figure 3.14a.



(a) Glitch free phase interpolator (b) Clock switching schema

Figure 3.14: Glitch free techniques

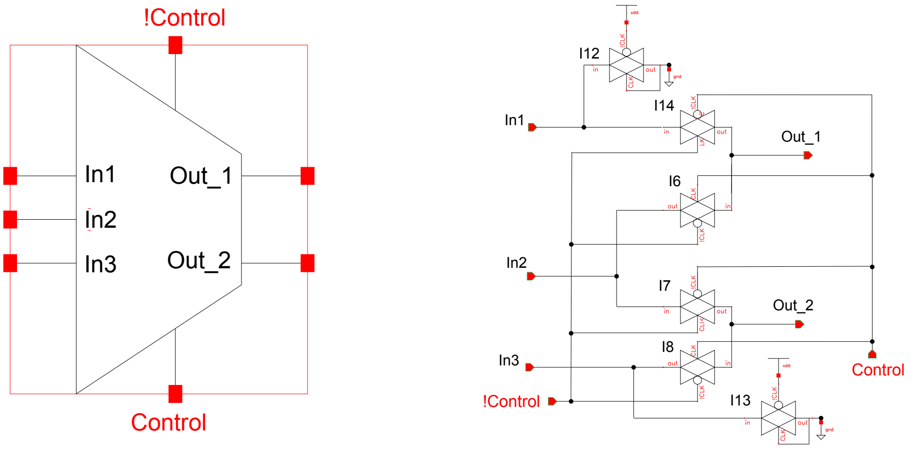
When the phase of the input clock changes, a switching protocol will run. Assuming that phase interpolator A is passed to the output (switch="0") at the beginning, once the control signal changes, signal "control B" changes accordingly to the wanted phase. At least one clock cycle is then waited to give phase interpolator B enough time to settle. At a safe time point, phase interpolator B is passed to the output (switch="1"). To prevent glitches during the switching, the switching is only allowed in the safe area (green) shown in Figure 3.14b, where clkA and clkB (outputs of phase interpolator A and phase interpolator B, respectively) have the same value, either "0" or "1".

3.3.2 Multiplexer

As shown in Figure 3.13, multiplexer is required in the phase interpolator to forward 2 phases out of 3 phases to the next stage. Figure 3.15a and Figure 3.15b depict the symbol and schematic of a MUX_3_2, respectively, where 3 represents the number of inputs and 2 the number of outputs. All the multiplexers are built by transmission gates. In Figure 3.15b, two dummy transmission gates I12 and I13 are inserted.

Table 3.4: Load of the inputs of MUX_3_2

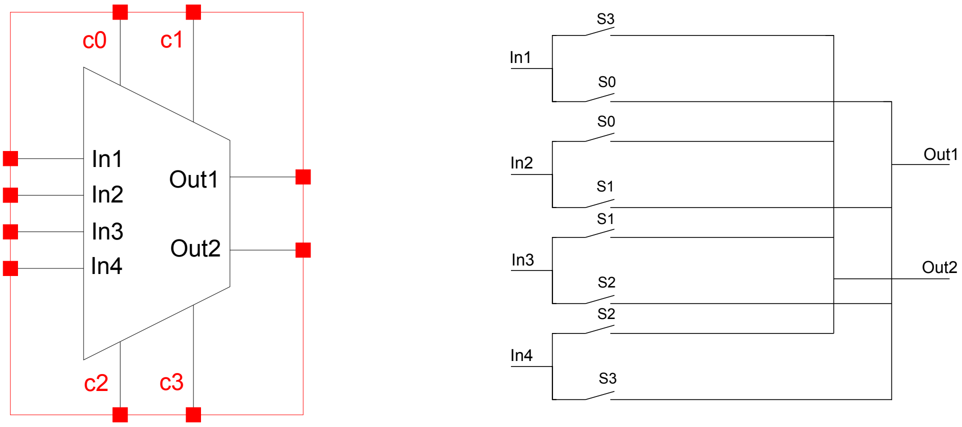
Input	With Dummy	Without Dummy
In1	I14 (on) & I12 (off)	I14 (on)
In2	I6 (on) & I7 (off) | I6 (off) & I7 (on)	I6 (on) & I7 (off) | I6 (off) & I7 (on)
In3	I8 (on) & I13 (off)	I8 (on)



(a) MUX_3_2 symbol (b) MUX_3_2 schematic

Figure 3.15: MUX_3_2

Table 3.4 shows the load of the inputs with and without the dummies. By inserting dummies the number of load for all input signals are equal to ensure the same phase difference at outputs as at inputs.



(a) MUX_4_2 symbol (b) MUX_4_2 schematic

Figure 3.16: MUX_4_2

MUX_4_2 shown in Figure 3.16a with 4 inputs and 2 outputs is required to assign reference phases from PLL to phase interpolators. To guarantee the same number of loads for all inputs, a different structure from MUX_3_2 is used to have a small amount of dummy elements. Figure 3.16b shows 4 inputs and each input is connected with 2 switches, where the upper switch is connected to *Out1* and the lower switch is connected to *Out2*. To pass the signal pairs (*In1 and In2, or In2 and In3, or In3 and In4, or In4 and In1*) to the outputs *Out1* and *Out2*, two switches always have to be closed. The switches $S1..S4$ realized by a transmission gate must be controlled in the way that every time only one switch is closed, which is guaranteed by a control logic. Through this symmetrical building it is guaranteed that at every switching point both active inputs always have one open and one closed transmission gates as loads. For the unused inputs loads are not cared.

3.3.3 Three Times Sampling CDR

To reduce the high dither in the proposed PI^2 CDR, a 3TS CDR is proposed shown in Figure 3.17. In the figure, an additional DES and a phase interpolator are employed to stop the phase shifting in the sampling clocks when Iclk is aligned to the center of the data.

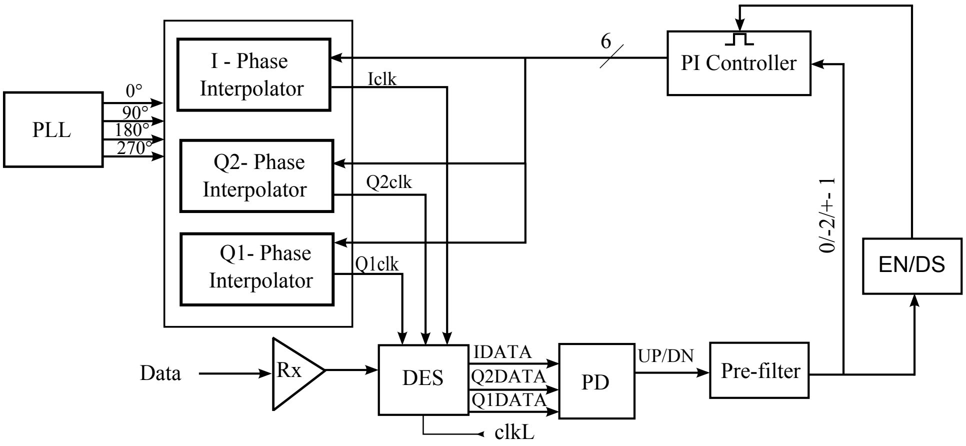


Figure 3.17: 3TS CDR block diagram

In the conventional CDR systems, Iclk is expected to be aligned to center of the data while Qclk to the edge of the data. However, both clocks move away or towards the expected positions when CDR is in the alignment state due to the drawback of conventional Alexander PD. In the 3TS CDR, instead of having one clock Qclk, two clocks named as Q1clk and Q2clk are generated with certain phase differences, which create a zone to check for the alignment of the CDR. If the data edge falls in that zone, then CDR is said to be aligned and the phase of the sampling clocks will remain constant. Figure 3.18 illustrates the 3TS method applied in the 3TS CDR.

Accordingly, a modified Alexander PD is proposed to contain four possible states, i.e., early (+1), late (-1), aligned (-2) and no change (0). Inherited by the conventional Alexander PD, the phase error at time point k for 3TS CDR is calculated by:

$$e(k) = (I(k+1) \oplus Q2(k)) - (I(k+1) \oplus Q2(k+1)) \tag{3.7}$$

where I(k) and Q2(k) represent the data at the time point k sampled by Iclk and Q2clk. To build an average phase error E(k) over a time period, e(k) is summed up as follows:

$$\text{E(k)} = \sum_{k=0}^{n} (I(k+1) \oplus Q2(k)) - (I(k+1) \oplus Q2(k+1)) \tag{3.8}$$

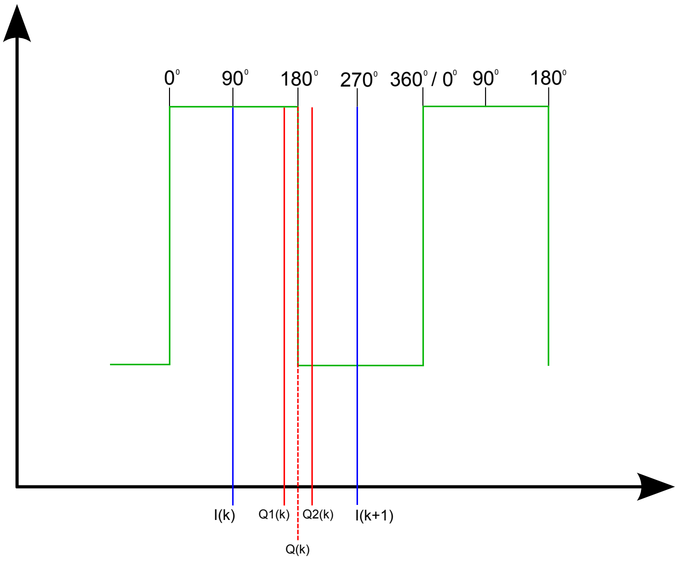


Figure 3.18: Alignment zone in a 3TS CDR

By comparing E(k) with zero, either early (+1) or late (-1) or no change (0) is provided at the output of the PD. The aligned state in the modified Alexander PD is conditioned by:

$$\text{E(k)} = \sum_{k=0}^{n} (Q1(k) \oplus Q2(k)) \tag{3.9}$$

Once the state of the 3TS CDR is determined, EN/DS system in Figure 3.17 is employed to either enable or disable the digital controller. After the read training is finished, PD and DES are further reused in the write training.

3.4 Simulation Results

Modified CDR and PI^2 CDR systems using TSMC 65nm LP technology are simulated in Cadence. The controller part including PI controller, Pre-filter, and PD is written in Verilog. Figure 3.19 shows the phase alignment of modified CDR and proposed CDR systems in the transistor-level simulation. Both systems have approximately the same locking time ($120ps$). The maximal jitter of modified CDR is $18ps$ while that of PI^2 CDR is $15.4ps$. In the nominal case (VDD=$1.2V$) the proposed CDR consumes $4mW$ power.

Table 3.5 and Table 3.6 illustrate the jitter and power of modified CDR and PI^2 CDR in corner simulation. The total number of corners adds up to 16. Following are the specifications on which CDRs are subjected to:
Transistor corners: FF, FS, SF, SS
Voltage corners: 1.08V, 1.32V
Temperature corners: -25°, 75°

Although the jitter of PI^2 CDR varies from $11.57ps$ to $42.87ps$ over all process corners,

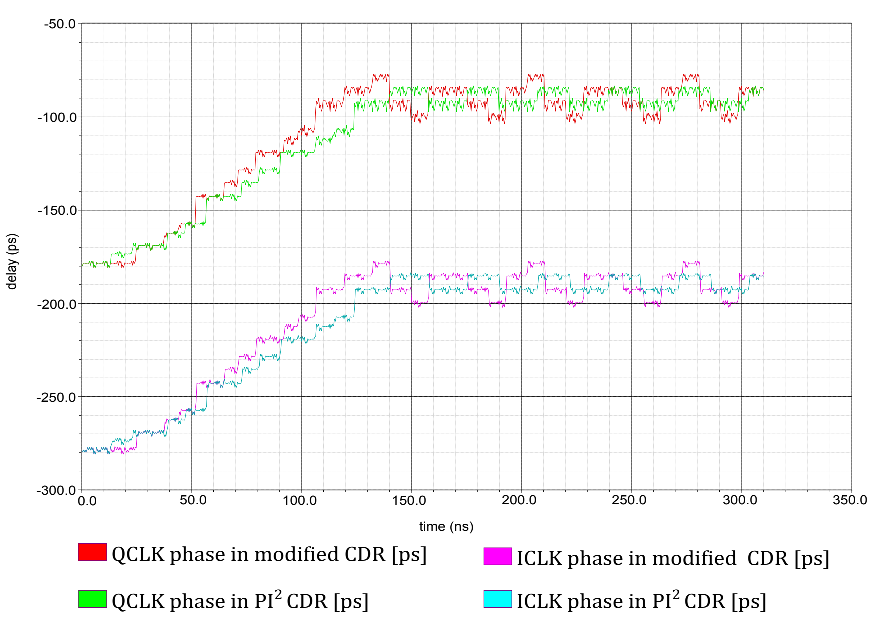


Figure 3.19: Phase alignment of modified CDR and PI^2 CDR

Table 3.5: CDR jitter of modified and PI^2 CDR systems in corner simulation

	min. jitter [ps]	max. jitter [ps]
Modified CDR	8.573	44.909
PI^2 CDR	11.57	42.87

Table 3.6: Power of modified and PI^2 CDR systems in corner simulation

	min. power (VDD=1.08V)	max. power (VDD=1.32V)
Modified CDR	2.788	5.611
PI^2 CDR	2.781	5.626

it ($< 80ps$) fulfilled the required valid data length of the eye diagram ($120ps$) for GDDR5 application. The power consumption lies between $2.8mW$ and $5.6mW$ for both models (only analog part), since the analog parts of both models are identical.

In the 3TS CDR, Q1clk is selected as the reference clock while others clocks Iclk and Q2clk are generated so that they lag the reference clock. As the implemented phase interpolator has 64 steps, 1LSB refers to 5.625° (6.25ps). When the jitter in the data increases the zone of Q1clk and Q2clk are enhanced to ensure the alignment of the CDR. Table 3.7 shows the variation of the LSB with/without maximal random jitter in data.

Table 3.7: LSB varies with/without maximal random jitter in data

LSB	max. jitter [ps]
2	0
6	50

3.4.1 Summary

In this section, we present a novel phase interpolator based CDR using PI controller for read/write trainings. By combining the advantages of the modified and the adaptive model a good compromise between power consumption and phase jitter is achieved in the proposed CDR. The adaptive model has the overall good performance in all cases except the doubled power consumption compared to the other models due to the need of four interpolators instead of two. Compared to the adaptive model, the modified model has the disadvantage that it is not possible to follow fast phase changing cause by the jitter. The PI^2 CDR combines the benefits of both adaptive and modified models, which can follow fast phase changes by using nearly the same power as the modified model. But the disadvantage of the proposed CDR is that it is a bit slower in the phase tracking than the adaptive CDR and has overall high dither. To address the high dither problem in the proposed CDR, a 3TS CDR with a modified Alexander PD is suggested. With the help of the 3TS CDR, the system is able to detect whether the recovered clock is aligned to the middle of the data as well as the read/write training state.

Chapter 4

Signal Integrity in Low-Power Adaptive Equalizer Trainings

Contents

4.1 State-of-the-Art 34

4.2 Signal Integrity Issues in High-Speed Communication System 35

4.3 System-Level Modeling of Equalizer Training 38

4.3.1 System-Level Modeling of Receiver Equalizer Training 38

4.3.2 System-Level Modeling of Transmitter Equalizer Training 42

4.4 Design and Implementation of Receiver Equalizer 49

4.4.1 Analog Receiver Equalizer 49

4.4.2 LMS Semi-Digital Receiver Equalizer 53

4.5 Design and Implementation of Transmitter Equalizer 54

4.5.1 Semi-Digital LMS Transmitter Equalizer 54

4.5.2 Semi-Digital Pilot Transmitter Equalizer 55

4.5.3 Full-Digital Direct Transmitter Equalizer 57

4.5.4 Simulation Results 59

4.6 Summary 65

In high-speed communication systems, adaptive equalizers are widely applied to improve SI in both master chip and slave chip. In this chapter, a novel architecture with adaptive receiver and transmitter equalizers applied only in the master chip is proposed for the low-power design through adaptive equalizer trainings. The system architecture is verified at the gate-level by implementing the receiver equalizers training with both analog and semi-digital circuits while the transmitter equalizer training uses different algorithms: 1) LMS algorithm 2) pilot signal/peak detection 3) direct calculation implemented with semi-digital and full-digital circuits. Among the investigated algorithms results show that LMS in semi-digital receiver and transmitter equalizers is the best performing algorithm in the high-speed source-synchronous memory systems with medium consumed areas. Compared to other algorithms, it can reduce the inter-symbol interference caused by both pre- and post-cursors of the channel.

4.1 State-of-the-Art

In high-speed communication systems, non-idealities of channels, such as losses and cross talks, are the main reasons for causing the severe SI problems, for example, ISI. The impact of ISI degrades both the timing margin and the voltage margin of the inter-chip signaling link, which results in a small or in a closed eye opening in the eye diagram. To compensate for the non-ideal channels, EQs [44] are incorporated in high-speed communication system. In some practical applications for unknown or time-varying channels, adaptive equalization circuits [45] [46] are required. Several adaptive equalization algorithms, such as LMS algorithm and recursive least square (RLS) algorithm, have been developed for adaptively adjusting filter coefficients.

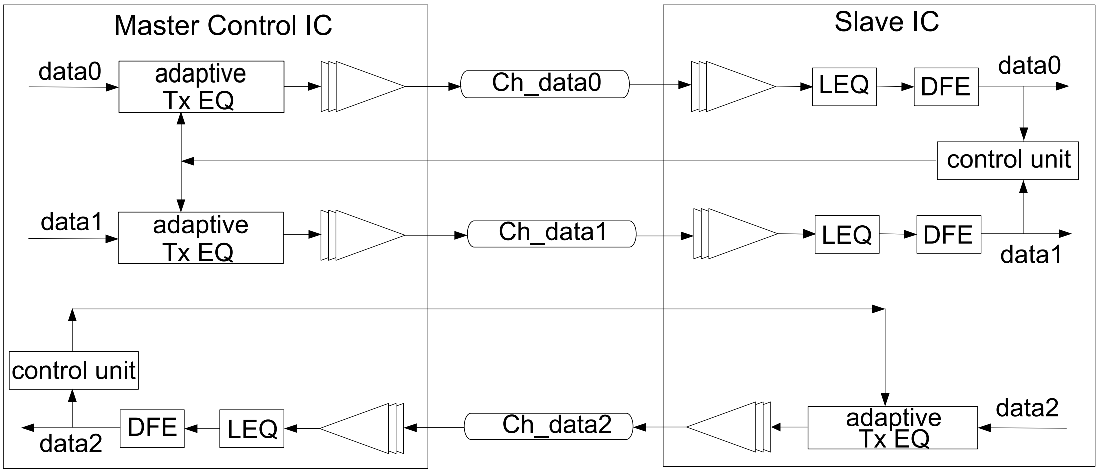


Figure 4.1: Adaptive EQs in the conventional architecture

Zerbe et al. [47] proposed an architecture to improve the performance of interconnections between IC devices. They suggested providing a signaling system with an adaptive pre-emphasizing transmitter and an adaptive equalizing receiver, coupled to each other via a high-speed signal path. In [48], digital adaptive folded multitap Tx and Rx EQs in the backplane environment are presented with the unidirectional data flow from the transmitter to the receiver. For applications where bidirectional data flow is required, Figure 4.1 illustrates the conventional system architecture with two ICs, a master control IC and a slave IC, where both can send and receive data. When data are transmitted to the slave IC through a channel, the quality of data such as BER is measured in the control unit at the receiver. This information is then used to adaptively establish appropriate transmit pre-emphasis and receive equalization settings. For example, the lowest power for which the signaling system provides minimum communication bandwidth without exceeding a desired BER, can be configured. Similarly, Tx and Rx EQs are adjusted when data are transmitted to the master control IC.

However, this architecture requires EQs and control units on both sides of the channels, which significantly increases the overall cost of the system. Particularly, when more than one slave IC are available in the high-speed communication systems, a large amount of Tx and Rx EQs are required in the slave ICs. In the case of existing slave ICs, redesign is usually avoided. Besides, the feedback signals generated by the control units need additional back channels and pins. The back channels have a certain impact on the SI so that the feedback signals can be distorted when adjusting the adaptive Tx EQ.

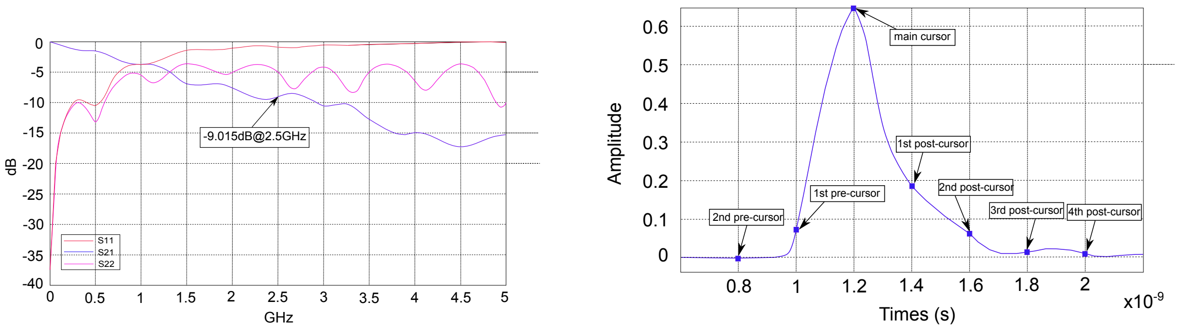
Amirkhany et al. [49] proposed a tri-modal asymmetric memory controller interface consisting of a voltage-mode one-tap transmitter equalization in the write direction and a linear EQ (LEQ)/1-tap DFE in the read direction to compensate for channel ISI. On the DRAM side, only LEQ is provided in the write direction. This architecture enables to shift most of the complexity to the controller and keep the DRAM design as simple as possible. However, it is a non-adaptive architecture, where the Tx EQ setting is customized to the known channel characteristics of 3-inch stripline FR4 trace. With this architecture the PCB designs for high-speed communication system are considerably stressed.

4.2 Signal Integrity Issues in High-Speed Communication System

Figure 4.2a shows the simulated characteristics of a chip-to-chip channel between GPU to Memory GDDR5. The channel mainly consists of a FBGA package (GPU) with package subtract traces, a FR4 PCB trace (including short traces on layers and board vias) and a wirebond FBGA package (GDDR5). The insertion loss in dB (S21) is almost a linear function of the frequency, which implies the dielectric loss dominant, as dielectric loss is proportional to the frequency. However, the ripples curve on the insertion loss curve is contributed by the discontinuity due to the PCB/package interface. As the channel is a linear time invariant system, a single bit response (SBR), as shown in Figure 4.2b, is applied to describe the channel and convolution calculation is used to verify the adaptation algorithms of the equalizer in the system level design. The channel mainly consists of one pre-cursor, one main cursor and 4 post-cursors.

Due to the fact that with the predefined interface training as depicted in Figure 2.10, data error caused by the SI have not been taken into account. Figure 4.3 illustrates proposed interface training sequence, which can handle more noisy channels, and thus relax PCB design for high-speed communication system.

After read training is performed, Rx EQ training is interspersed to improve the SI and



(a) S parameters (b) Impulse response

Figure 4.2: Example of a channel between memory controller and GDDR5 memory

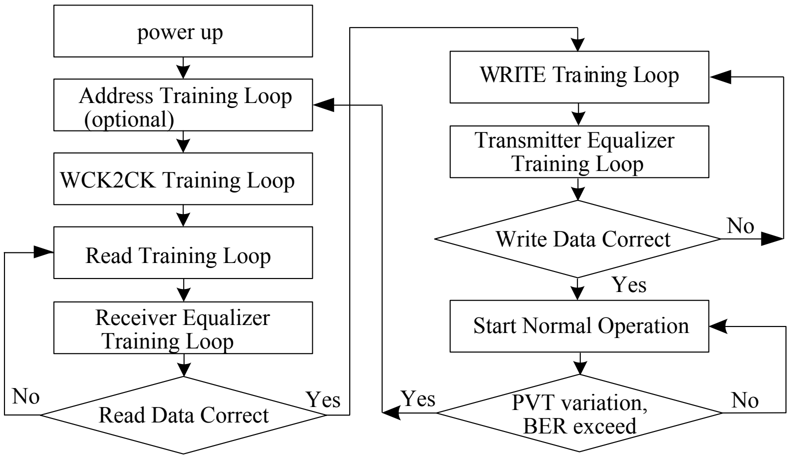


Figure 4.3: Proposed interface training sequence

to make the eye opening larger. The setting of the Rx EQ will be performed in a way so that errors between the training data and the received data will be below a certain threshold value. Whether this condition has already been achieved will be checked in a consecutive interrogation. If the read data are correct, then write training loop is followed. Otherwise when there were any errors in data due to SI problems, data could not be read correctly. After solving the SI problems by performing Rx EQ training, the training sequence is jump back to redo the read training in order to adjust the right receive phase for corrected read data. The similar process is executed with Tx EQ training loop. After the interface training, normal operation can be started. Once the BER that is monitored in the memory controller exceeds to a certain threshold, e.g. 10^{-12}, due to PVT variations, the interface needs to be retrained.

Figure 4.4 depicts the proposed architecture [13] by combining the advantages of the system architecture [47] that uses adaptive EQs and the system architecture [49] that has a simple slave IC. In order to adjust DFE, LEQ and Tx EQ, two steps are required. The first step is Rx EQ training which is to adjust the coefficients of Rx EQs by sending data

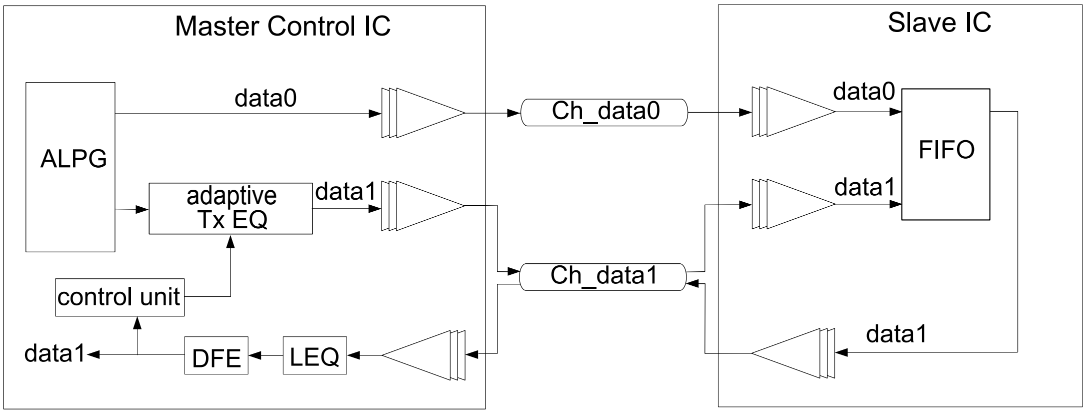


Figure 4.4: Proposed system architecture using adaptive EQs

patterns provided by algorithmic pattern generator (ALPG) into FIFO in the slave IC through a trained channel e.g. Ch_data0. Alternatively, the trained channel can be a data line that is provided for other purposes e.g. transmitting address signals with a lower data transfer speed and a better signal quality. This loop helps to determine the channel characteristic of Ch_data1. Once channel characteristic is known, the next step is Tx EQ training by sending data patterns through Ch_data1. This loop is built up by the same data channel. The control unit uses the known channel characteristic to set the Tx EQ coefficients.

Compared to the conventional architecture in Figure 4.1, the proposed configuration enables the reduction of the complexity in the slave IC dramatically in terms of area and power. All the adaptive EQs and control units are available only in the master control IC. Instead, a FIFO is required in the slave IC to store the test data patterns for the training purpose. Furthermore, the number of the control units can be reduced if two or more Tx EQs can share the same control unit for those nearby data lines which may have the similar channel characteristics.

The remainder of this chapter deals with 1) System-level modeling of EQ Trainings in Matlab/Simulink, 2) Design and implementation of analog Rx EQ and semi-digital Rx EQ , 3) Implementation of Tx EQ as well as investigation of different algorithms (i.e. LMS, pilot signal/peak detection, direct).

4.3 System-Level Modeling of Equalizer Training

4.3.1 System-Level Modeling of Receiver Equalizer Training

4.3.1.1 Model Description

In general, a Rx EQ is composed of a feed-forward equalizer (FFE) and a DFE. FFE can be applied to cancel both pre- and post-cursors. As DFE gives a more promising performance than FFE to cancel the post-cursors [50], FFE is usually used for canceling the pre-cursors only. Figure 4.5 illustrates the working principle of Rx EQ, when a SBR is given at the input. Due to the non-ideality of the channel, the unequalized signal is shaped with the peak in the center and its ISI components (three pre-cursors and four post-cursors) at both sides.

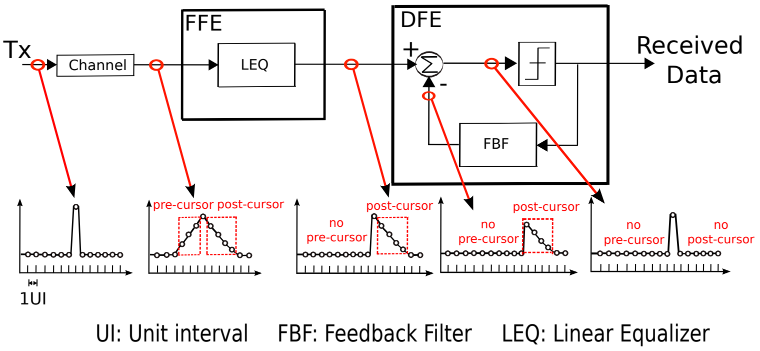
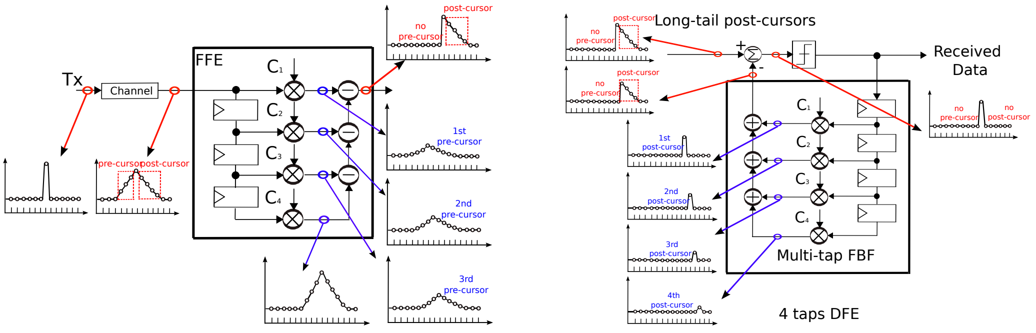


Figure 4.5: Working principle of receiver equalizer

Figure 4.6a and Figure 4.6b depict how FFE cancels the pre-cursors and DFE cancels the post-cursors internally. In both EQs, each coefficient C_i is multiplied with either the input signal or delayed versions of the input signal, which results ISI components. By subtracting the ISI components caused by pre- and post-cursors, the impulse signal is recovered with delays.



(a) Working principle of a FFE (b) Working principle of a DFE

Figure 4.6: FFE and DFE

The adaptive control for the coefficients are usually adaptively generated using LMS algorithm [51] [52], which can be expressed as:

$$C_i(k+1) = C_i(k) + \mu \cdot e(k) \cdot \nu(k-i+1) \tag{4.1}$$

$$C_j(k+1) = C_j(k) + \mu \cdot e(k) \cdot I(k-j) \tag{4.2}$$

$$Y(k) = \sum_{i=1}^{N} C_i \nu(k-i+1) + \sum_{j=1}^{M} C_j I(k-j) \tag{4.3}$$

$$e(k) = I(k) - Y(k) \tag{4.4}$$

where C_i, C_j, μ, e, ν, $Y(k)$, $I(k)$ indicate the coefficients in FFE and DFE, step size, error, the data distorted by channel, output of the equalizer and ideal data, respectively. Figure 4.7 illustrates Rx EQ training in Matlab/Simulink composed of a channel, *prbs*, a FFE and a DFE.

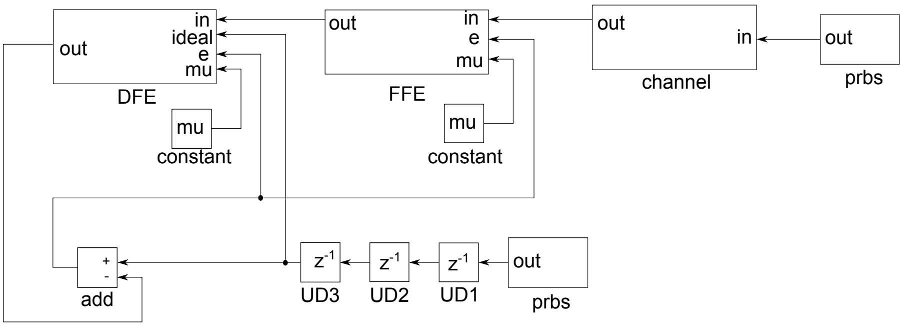


Figure 4.7: Rx EQ training in Matlab/Simulink

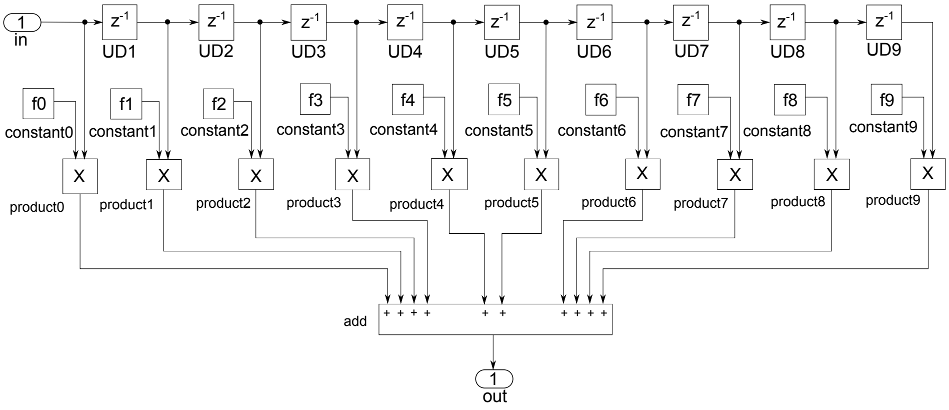


Figure 4.8: Modeled channel in Matlab/Simulink

As the channel has low-pass filter characteristic, it is modeled as a low pass filter in Matlab/Simulink shown in Figure 4.8. Constants $f0...f9$ represent the coefficients of the low-pass filter derived from the impulse response of the channel.

The block *prbs* produces pseudo random bit sequences (PRBS) as training test patterns transmitted to the memory controller through a mathematically modeled channel. For simplicity, FIFO in the memory is replaced with the *prbs* block which is identical with the *prbs* in the memory controller side. The delays caused by FFE shown in Figure 4.6a are modeled by three unit delays to generate an error signal between the ideal test patterns and the output of the Rx EQ.

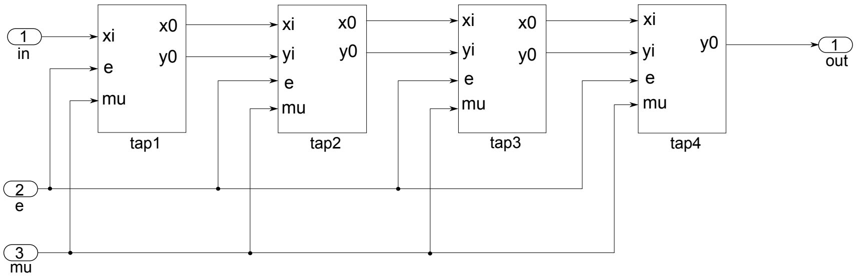


Figure 4.9: FFE in Rx EQ

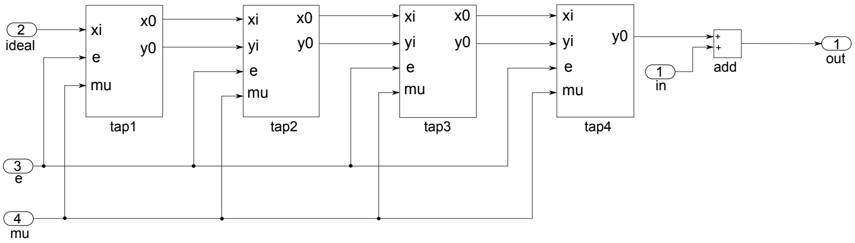


Figure 4.10: DFE in Rx EQ

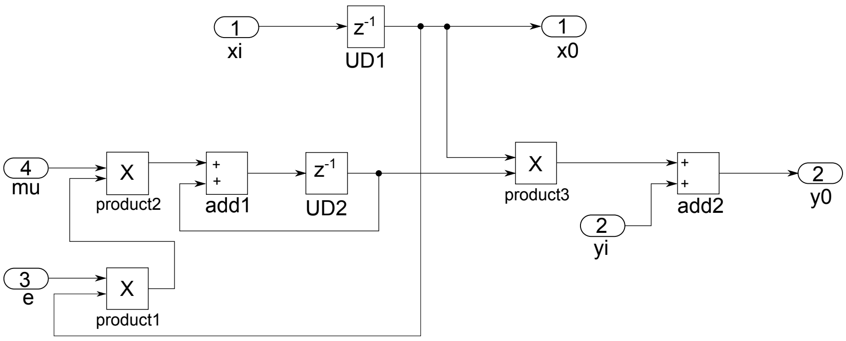
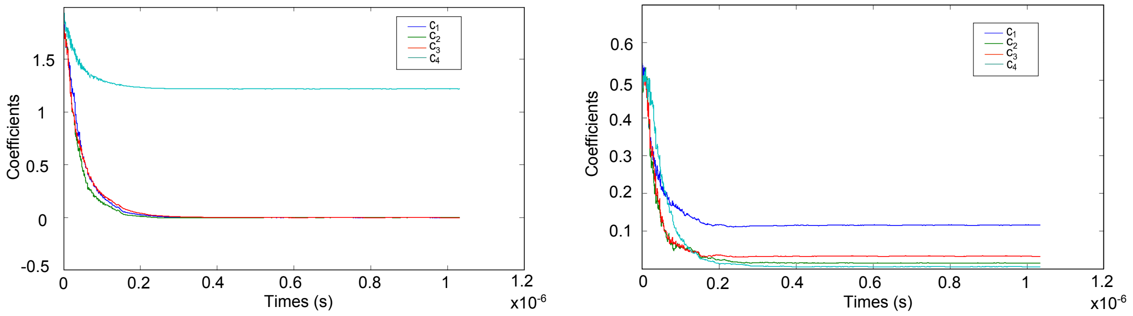


Figure 4.11: Tap in Rx EQ

Figure 4.9 and Figure 4.10 depict FFE and DFE, respectively. Both EQs consist of 4 *tap* blocks. Figure 4.11 illustrates the *tap* block, where the adaptation of coefficients in Equations (4.1) and (4.2) takes place.

4.3.1.2 Simulation Results

Figure 4.12 illustrates the coefficients in FFE and DFE during the Rx EQ training. The total simulation time adds up to 1.05e-6s. The coefficients converge after 0.4e-6s. Table 4.1 lists the coefficients in FFE and DFE after convergence.



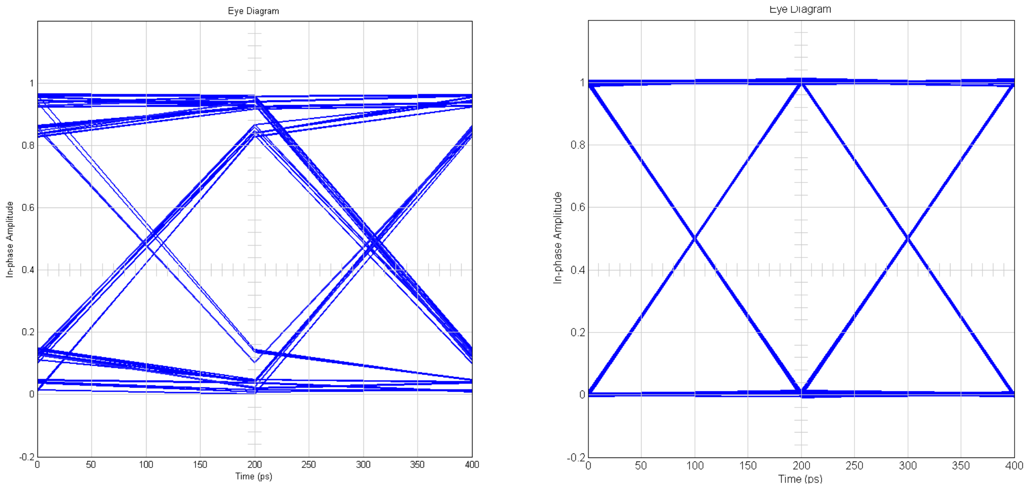
(a) Coefficients in FFE

(b) Coefficients in DFE

Figure 4.12: Coefficients of FFE and DFE for Rx EQ training

Table 4.1: Coefficients of FFE and DFE after convergence

	C_1	C_2	C_3	C_4
FFE	0.026	0.026	0.026	1.17
DFE	0.15	0.068	0.064	0.064



(a) Distorted data after channel

(b) Equalized data

Figure 4.13: Eye diagrams without/with Rx EQ

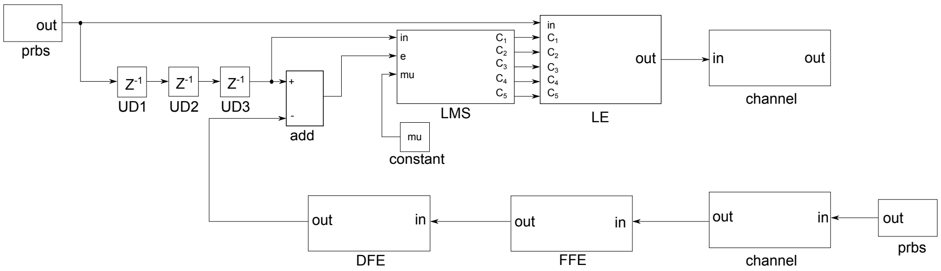


Figure 4.14: Tx EQ training using LMS in Matlab/Simulink

Figure 4.13 depicts the eye diagrams of distorted data and equalized data. Table 4.2 shows their vertical and horizontal eye openings. With Rx EQ, the vertical eye opening is improved by 42.8% while the horizontal eye opening by 8.1%.

Table 4.2: Comparison of eye diagrams

Eye diagram	Eye Opening(vertical)	Eye Opening(horizontal)
Without Rx EQ	0.7	185ps
With Rx EQ	0.99	200ps
Improvement	42.8%	8.1%

4.3.2 System-Level Modeling of Transmitter Equalizer Training

Tx EQ is also known as pre-emphasis EQ. It is usually implemented as LE such as FFE. Three algorithms for coefficient adaptation are applied for Tx EQ training, i.e. LMS algorithm, pilot signal/peak detection and direct calculation. In the following sections these algorithms are described in details.

4.3.2.1 LMS Algorithm

The principle of LMS algorithm [14] is to adjust the Tx EQ coefficients depending on the error at the output of EQ compared to the expected data as shown in Equation (4.1). The EQ output EQ(k) is given as:

$$EQ(k) = \sum_{j=1}^{N} C_j \nu(k - i + 1) \tag{4.5}$$

Figure 4.14 illustrates the adaptive Tx EQ using LMS algorithm after Rx EQ training in Matlab/Simulink. The block *LMS* and *LE* work similar to the block *FFE* in Figure 4.9. The block *prbs* is reused at the input of the block *LMS*. Through the trained Rx EQ data are transmitted from memory to Tx EQ without large errors. As shown in Figure 4.15, the coefficients converge after 300ns.

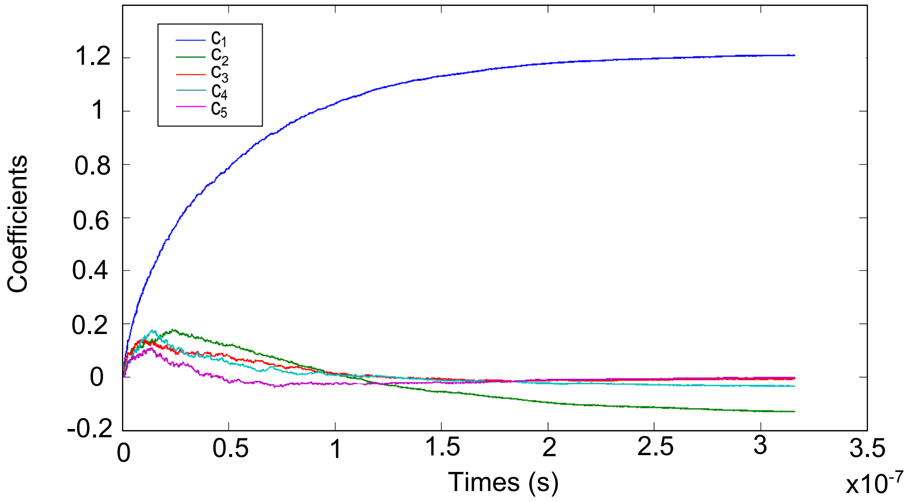


Figure 4.15: Coefficients using LMS algorithm for Tx EQ training

4.3.2.2 Pilot Signal/Peak Detection

Pilot signaling and peak detection [15] is another alternative implementation for the control unit in the Tx EQ training. This method provides the flexibility of adjusting coefficient values sequentially in adaptive Tx EQ.

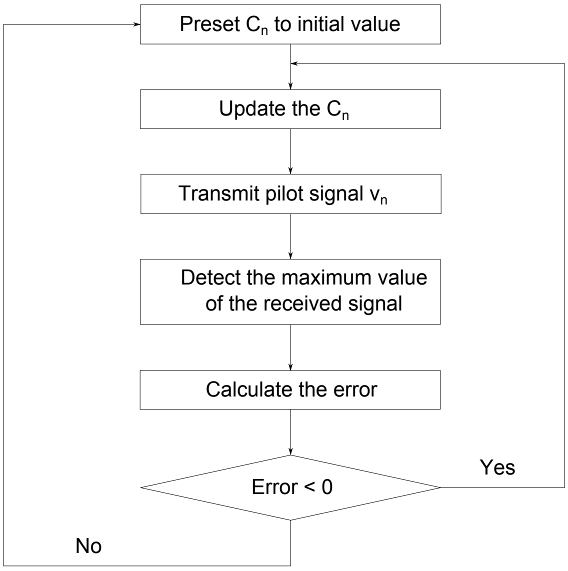

Preset Cn to initial value
Update the Cn
Transmit pilot signal vn
Detect the maximum value of the received signal
Calculate the error
Error < 0
Yes
No

Figure 4.16: The procedure of pilot signal/peak detection algorithm

Figure 4.16 illustrates the procedure of pilot signal/peak detection algorithm. The current coefficient of the Tx EQ is initialized with some large value so that the data peak after channel can be detected in the receiver while those undetermined coefficients are set to zero. During the Tx EQ training, several known pilot signals are sent from the transmitter. The peak is detected and compared with an expected value. When the peak is larger than the expected value, the current coefficient will be reduced by one LSB of a DAC that is used to convert the digital coefficient into analog value, until the peak is smaller than that value. The next pilot signal will be then sent. This procedure continues until all the coefficients are determined. Impulse responses for an EQ and a channel are given by:

$$F_{EQ}(n) = \sum_{k=1}^{n} C_k \delta(n-k+1) \tag{4.6}$$

$$F_{Ch}(n) = \sum_{k=1}^{n} f_k \delta(n-k+1) \tag{4.7}$$

where C_k represents Tx EQ coefficient and f_k is channel coefficient. Therefore, the output after channel is given by:

$$Y(m) = v \otimes C \otimes f = \sum_{j=1}^{m} f_j [\sum_{k=1}^{m+1-j} C_k v(m+2-j-k)] \tag{4.8}$$

where v denotes pilot signals. Pilot signals, i.e. "10000", "11000", "10100", "10010" and "10001" are sent to determine the coefficients. By knowing channel coefficients, pilot signals and DFE outputs, Tx EQ coefficients are derived.

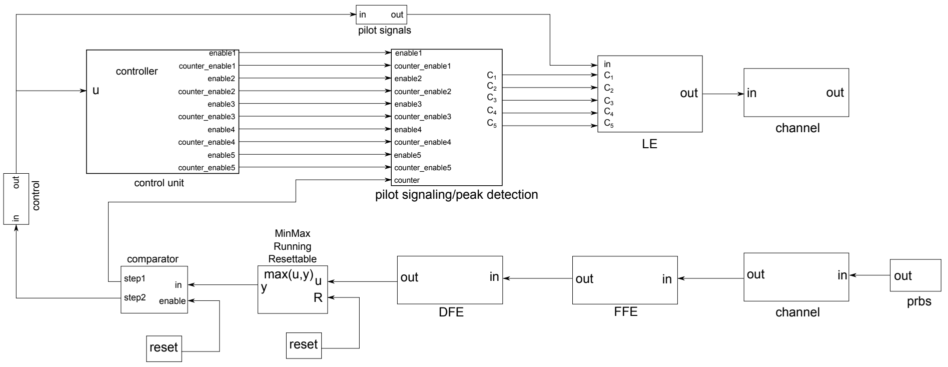


Figure 4.17: Tx EQ training using pilot signal/peak detection in Matlab/Simulink

Figure 4.17 illustrates adaptive Tx EQ using pilot signal/peak detection algorithm after Rx EQ training in Matlab/Simulink. The peak detection is realized by the block *MinMax Running Resetttable*, which outputs the maximal value of all the past inputs. In the "reset" mode, the past inputs are set to zero and replaced by the current inputs. The *comparator* compares the peak with the ideal value and decides whether the adaptation of the next coefficient starts or the current coefficient has to be updated. Depending on the output of the comparator, the block *control* enables the next pilot signal to be sent. The block *control unit* decides which coefficient is currently enabled for adaptation. When all the coefficients are available, the block *pilot signals* as shown in Figure 4.18 starts to transmit PRBS through LEQ.

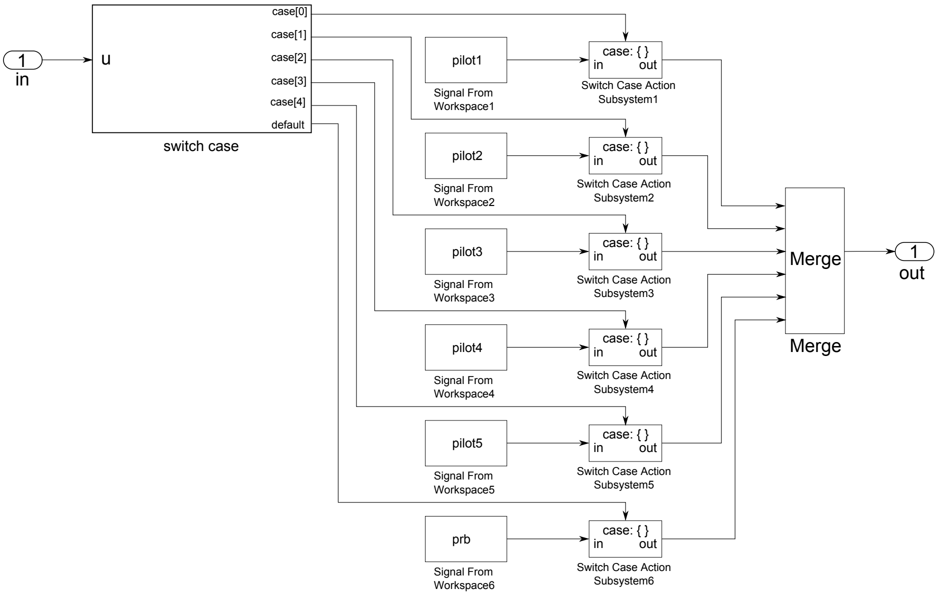


Figure 4.18: The signal generator

4.3.2.3 Direct Calculation

A direct way to calculate the coefficients of LEQ requires single test pattern, for example, an impulse signal. The initial coefficients are set to zero. After Rx EQ training, the test pattern is transmitted from memory to memory controller. Let's assume y represents the data at the input of FFE, which can be expressed as:

$$y(k) = v \otimes f = \sum_{j=1}^{k} f_j v(k-j+1) \tag{4.9}$$

and Y denotes the output after channel through LEQ based on Equation (4.8). After Rx EQ training, the data at the output of DFE can be ensured as good as the ideal data. With the assumption that channels in both directions have exactly the same characteristics, Y has approximate value as the output of DFE. By comparing y and Y in Equations (4.9) and (4.8) respectively, coefficients of Tx EQ can be derived as:

$$C_k = \begin{cases} Y(1)/y(1) & (k=1) \\ \left[Y(k) - \sum\limits_{j=1}^{k-1} C_j y(k+1-j) \right] C_1/Y(1) & (k >= 2) \end{cases} \tag{4.10}$$

where $k \in \mathbb{N}$.

Figure 4.19 illustrates the adaptive Tx EQ using direct calculation after Rx EQ training in Matlab/Simulink. Delays due to FFE shown in Figure 4.6a are under the consideration for y at the input of the block *direct calculation*. Figure 4.20 illustrates the block *direct calculation* internally. The synchronization of the coefficients is taken into account by adding various integer delays.

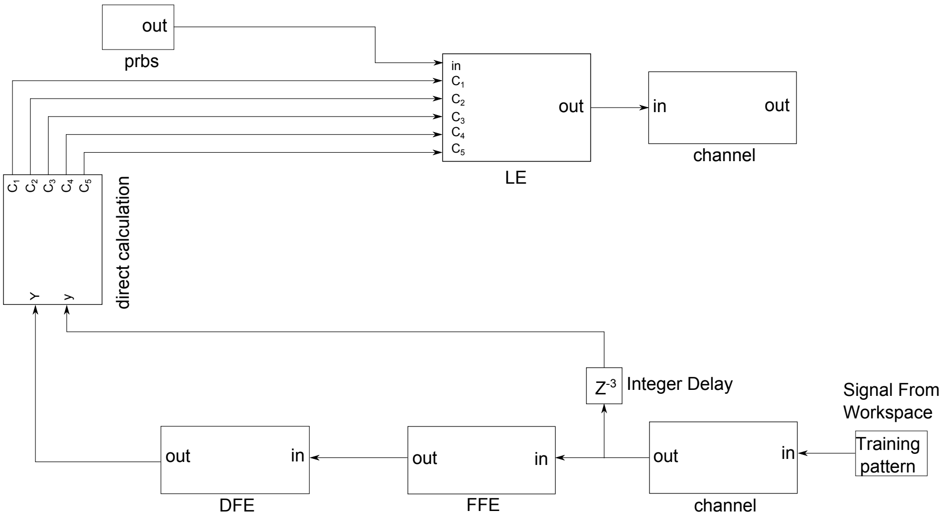


Figure 4.19: Tx EQ training using direct calculation in Matlab/Simulink

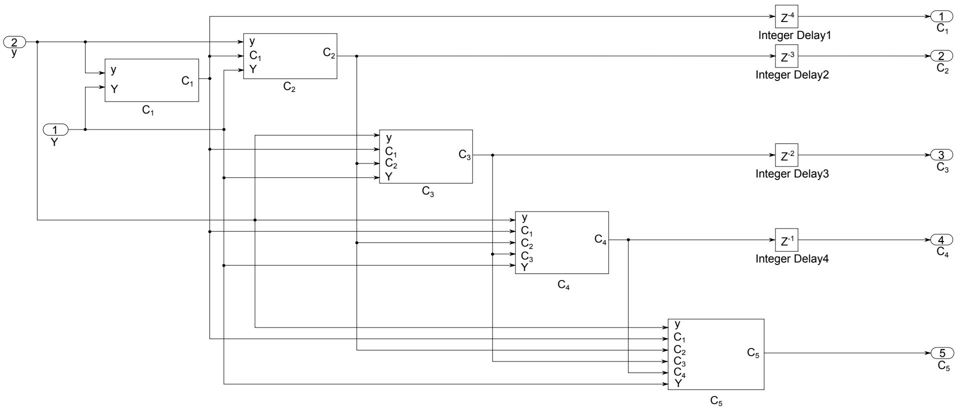


Figure 4.20: Coefficients after direct calculation in Simulink

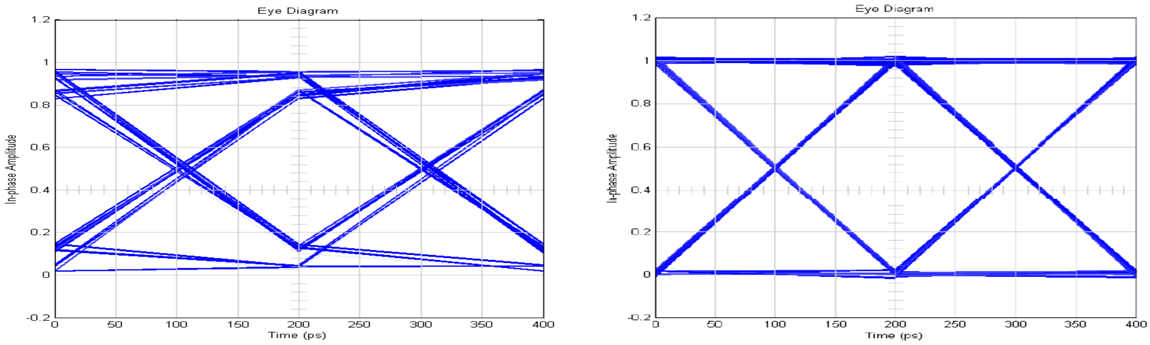
4.3.2.4 Simulation Results

Table 4.3 compares different Tx EQs based on LMS algorithm, pilot signal/peak detection and direct calculation. The main parameters considered are the coefficients of Tx EQ and the numbers of test data patterns required for the training. Results show that direct calculation has similar coefficients as LMS algorithm, but only 5 data bits are required for the training. However, direct calculation has the constraint that it assumes the same channel characteristic in both the write and the read direction, which may not be the case in the reality since channel characteristics depend on many factors e.g. impedance matching. There is usually some difference in the channel characteristics for the same channel where data are transmitted in different directions. In comparison to direct calculation, adaptive EQs using LMS algorithm and pilot signal/peak detection need much more time to adjust the coefficients. The numbers of test data bits depend on the individual channel and the numbers of the taps. Moreover, pilot signal/peak detection can only improve the post-cursor in the impulse response of the channel since peak detection only searches for the maximal output.

Table 4.3: Coefficients of three algorithms after Tx EQ training

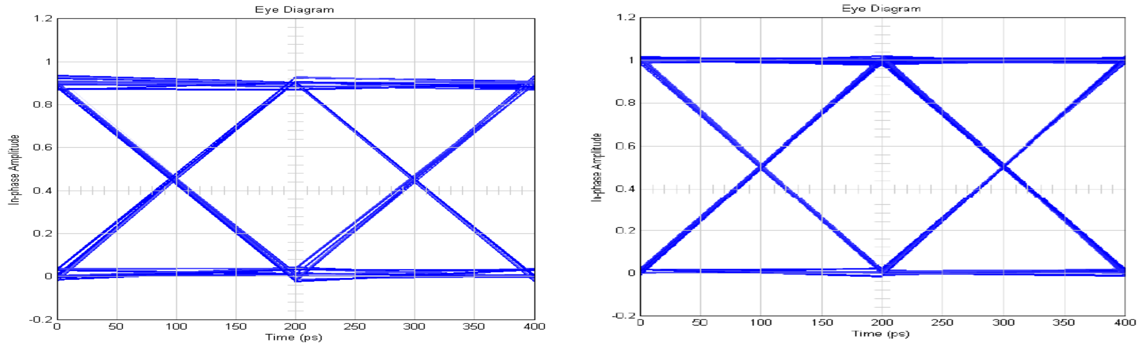
Algorithms	Tap1	Tap2	Tap3	Tap4	Tap5	Test data bits
LMS	1.213	-0.1421	-0.0033929	-0.03777	0.002635	2000
Pilot signal/peak detection	1.094	-0.15625	0	0	0	1256
Direct calculation	1.218	-0.143	-0.002264	-0.04078	-0.001556	5

Figure 4.21 shows discrete-time eye diagrams available Simulink communications blockset tool for Tx EQ based on LMS algorithm, pilot signal/peak detection and direct calculation, respectively. The upper bound of Figure 4.21c is slightly less than 1V due to the fact that 6-bit DAC is used. The resolution of DAC directly affects the coefficients of Tx EQ in pilot signal/peak detection method. Table 4.4 compares eye openings in both vertical and horizontal for various algorithms. Without EQ the vertical and horizontal eye opening are 68% and 88.75% with respect to ideal eye opening.



(a) without EQ

(b) Using LMS algorithm



(c) Using pilot signal/peak detection

(d) Using direct calculation

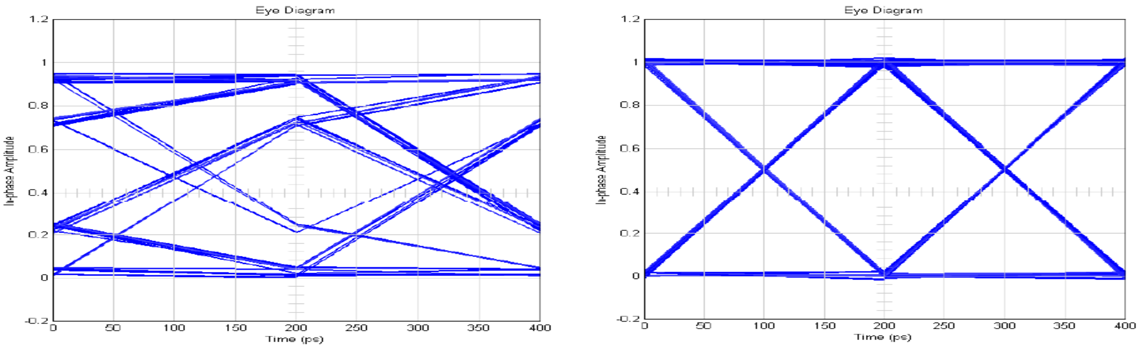
Figure 4.21: Eye diagrams for Tx EQ

The test channel delivers good performance in terms of small loss as shown in Figure 4.21a. To further explore the adaptivity of the system, a highly lossy channel with smaller

Table 4.4: Eye opening among different EQ algorithms

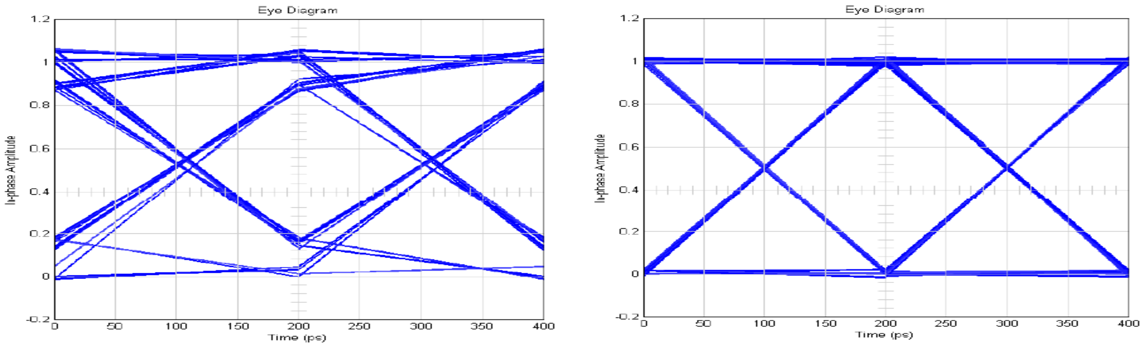
Tx EQ	Eye Opening (vertical)	Eye Opening (horizontal)
No Tx EQ	68%	88.75%
LMS	99%	99%
Pilot signal/peak detection	91.3%	96%
Direct calculation	99%	99%

eye openings is investigated. Figure 4.22 shows eye diagrams for the lossy channel using the same algorithms. Table 4.5 compares eye openings in both vertical and horizontal. Without EQ the vertical and horizontal eye opening are 45% and 82.5% with respect to ideal eye opening. All of three implementations give better performance than without EQ regarding to the SI. Direct calculation and LMS algorithm show better eye opening than pilot signal/peak detection.



(a) without EQ

(b) Using LMS algorithm



(c) Using pilot signal/peak detection

(d) Using direct calculation

Figure 4.22: Eye diagrams for Tx EQ with a more lossy channel

Table 4.5: Eye opening among different EQ algorithms

Tx EQ	Eye Opening (vertical)	Eye Opening (horizontal)
No Tx EQ	45%	82.5%
LMS	99%	99%
Pilot signal/peak detection	67%	95%
Direct calculation	99%	99%

4.4 Design and Implementation of Receiver Equalizer

4.4.1 Analog Receiver Equalizer

An adaptive Rx EQ is mainly composed of LMS engines for each coefficient and EQ summers in both FFE and DFE as shown in Figure 4.23.

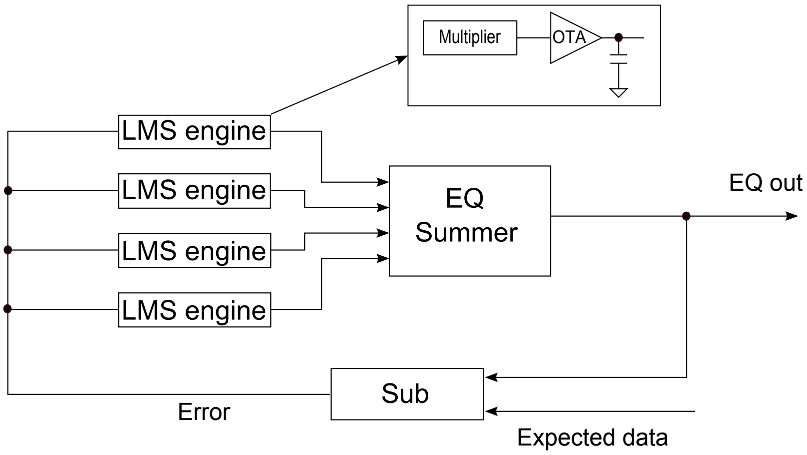


Figure 4.23: DFE/FFE block diagrams in an analog receiver equalizer

4.4.1.1 LMS Engine

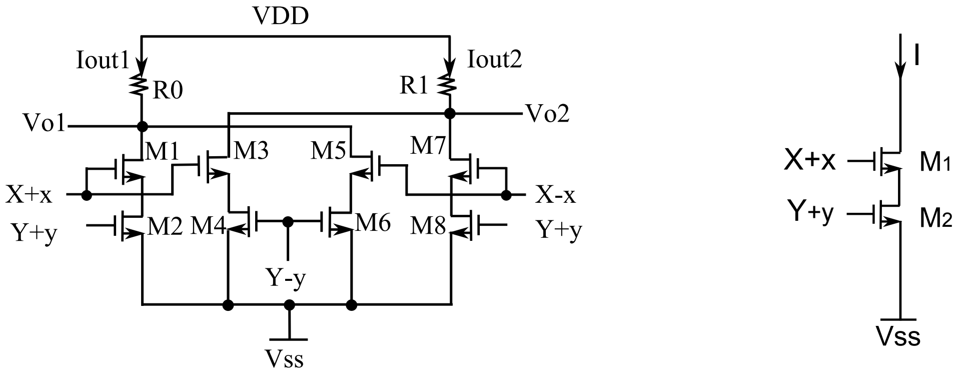
According to Equations (4.1) and (4.2), LMS engine works similar to the behavior of an integrator. These two equations, however, differ in their inputs where expected data is replaced with the distorted data in case of a FFE. Based on papers [53] [54], the LMS engine is implemented by a multiplier and an integrator.

As introduced in [55], eight types of transconductance multipliers are classified. Compared to the other transconductance multipliers, the multiplier with Gilbert cell structure shows good linearity and low noise. Figure 4.24a shows the structure of a Gilbert cell four quadrant multiplier. It consists of four one quadrant multipliers as seen in Figure 4.24b. The output current of one quadrant structure can be expressed as:

$$I \propto (X + x)(Y + y) \tag{4.11}$$

where X, Y and x, y indicate dc input voltages and ac input voltages, respectively. Therefore,

$$\begin{aligned} I_{out} \propto (&[(X + x)(Y + y) + (X - x)(Y - y)] \\ &-[(X - x)(Y - y) + (X + x)(Y - y)]) \end{aligned} \tag{4.12}$$



(a) Four quadrant

(b) One quadrant

Figure 4.24: Gilbert cell multiplier

As a result, the output voltage of the Gilbert cell multiplier V_{out} is proportional to 4xyR_L, where R_L is the load resistor. By selecting a proper R_L value, the V_{out} can be equal to xy.

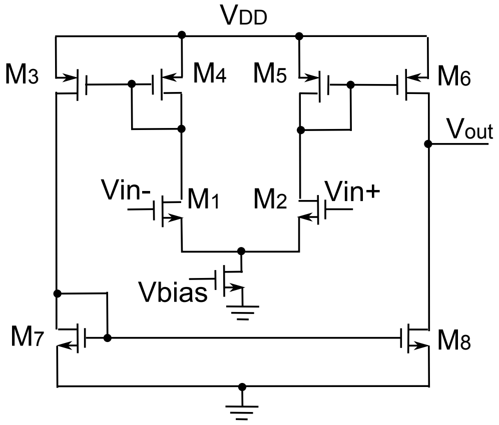


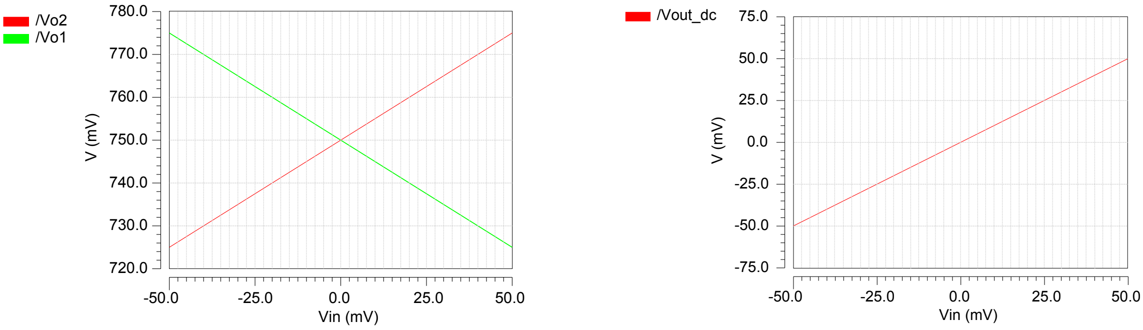
Figure 4.25: Schematic of OTA

Figure 4.26 shows the simulation result when Vin is swept from -50mV to 50mV. The output of the multiplier is set to be 0.75V in dc state.

One method to realize an integrator as introduced in [56] is to use G_m-C technology. The transconductor G_m is implemented by a simple operational transconductance amplifier (OTA) shown in Figure 4.25. Three current mirrors M3-M4, M5-M6 and M7-M8 copy the currents converted from the input voltages by the input differential pair M1-M2. The output voltage is derived by:

$$V_{out} = (g_{m1}g_{m3}/g_{m2}C)\int(V_{in+} - V_{in-})dt \tag{4.13}$$

where g_{m1}, g_{m2}, g_{m3} indicate the transconductance of differential pair $M1 - M2$, current mirror $M3 - M4$ or $M5 - M6$ and current mirror $M7 - M8$. Comparing the Equation



(a) Two differentials outputs (b) The difference between two outputs

Figure 4.26: Simulation results of multiplier

(4.13) with the Equation (4.1), $g_{m1}g_{m3}/g_{m2}C$ is corresponding to the step size μ, which depends on the LMS convergence condition. If the step size is set too large, the system becomes unstable and the coefficients may not be adapted to the ideal value. Otherwise, it will takes long training time to reach convergence. Normally the step size is set any value between $0<\mu<<1$.

Figure 4.27 shows the ac simulation of OTA. Table 4.6 shows the ac performance of OTA in detail.

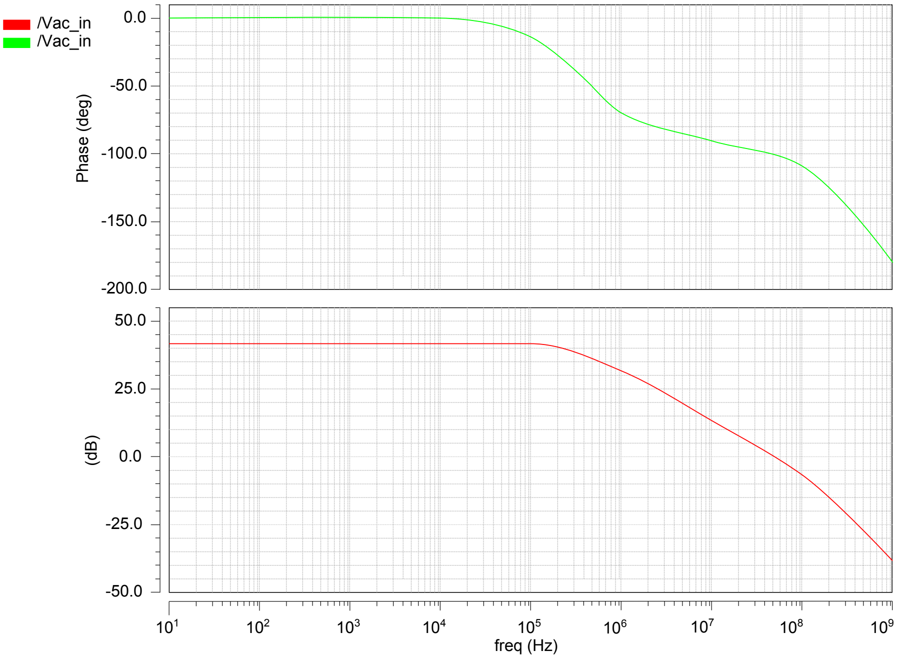


Figure 4.27: The ac simulation of OTA

Table 4.6: AC performance of OTA

Parameters	Values
Unit Gain Frequency	44MHz
Open loop gain(G)	> 42dB
Phase margin	82°

4.4.1.2 Equalizer Summers

Based on [56], the summers in DFE and FFE are realized by using series-gated differential pairs with common load resistors, as shown in Figure 4.28 and Figure 4.29. The dc voltages $V_1 - V_4$ and $V_5 - V_8$ are connected to the OTA outputs. According to Figure 4.6a, the main tap of receiver equalizer is positioned at the fourth differential pair in the FFE summer. To subtract the ISI components caused by the pre-cursors, the previous three differential pairs are connected with the fourth one in an opposite way. Similarly, in the DFE summer the first differential pair is connected with the following four differential pairs differently in order to cancel the post-cursors. The FFE output is given by:

$$V_{ffeout} = R_L(g_{m4}\nu(4) - \sum_{i=1}^{3} g_{mi}\nu(i-1)) \tag{4.14}$$

where g_{m1}, g_{m2}, g_{m3}, g_{m4} represent the transconductances of four input differential pairs in FFE summer. Similarly,

$$V_{dfeout} = R'_L(g_m V_{ffeout} - \sum_{i=1}^{4} g'_{mi} I(i-1)) \tag{4.15}$$

where g_m, g'_{m1}, g'_{m2}, g'_{m3}, g'_{m4} represent the transconductances of five differential pairs in DFE summer.

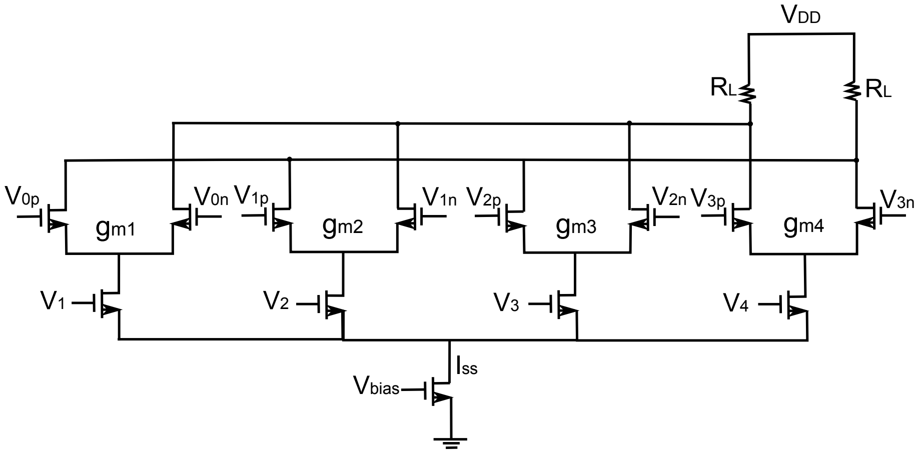


Figure 4.28: Schematic of FFE summer

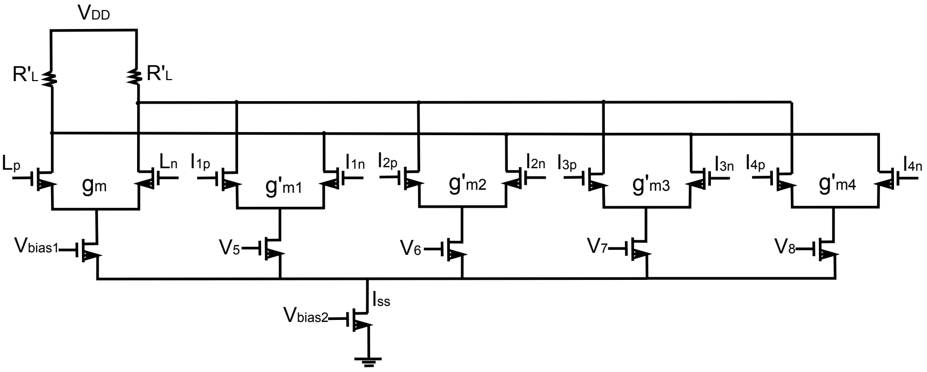


Figure 4.29: Schematic of DFE summer

4.4.2 LMS Semi-Digital Receiver Equalizer

An alternative implementation of LMS engine is realized using digital components, while equalizer summer is implemented in analog.

4.4.2.1 LMS Engine

Figure 4.30 shows the diagram of a digital LMS engine, which consists of two roms for encoding and decoding, an adder and two pipelined multipliers. The Verilog code for the adder and the multiplier is given in Appendix A.3. Based on Equations (4.1) and (4.2) coefficients are updated. The rom for encoding converts the ADC codes of ideal or distorted data into the floating data while the rom for decoding converts the absolute floating number into digital analog converter (DAC) codes. In the implementation ADC and DAC are modeled in Verilog codes. In order to increase the speed of coefficient adaptation, pipelined structures are used in adders and multipliers.

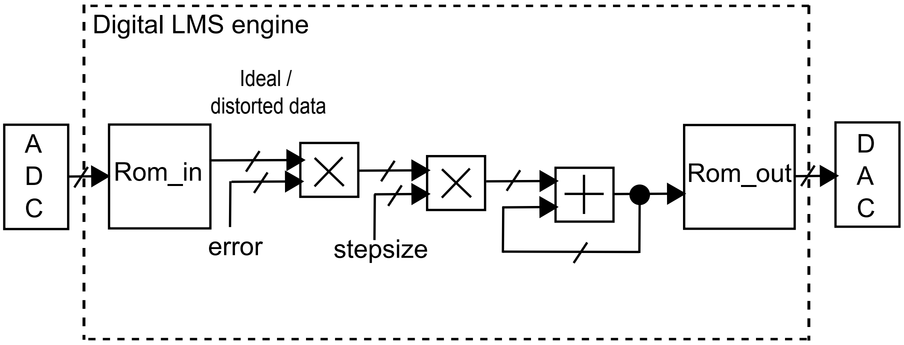


Figure 4.30: Digital LMS engine for FFE/DFE

Figure 4.31 shows the implementation of a pipelined multiplier. The ten bits data consists of one sign bit, one integer bit and eight floating bits. If the input is larger than 1, it is right shifted by 1 bit first. According to the sign bit, complement codes of each input are calculated. After multiplication the result is left shifted, if the input was right shifted before. The highest 9 bits are the final result and the complement codes are calculated again depending on the sign bit of the result.

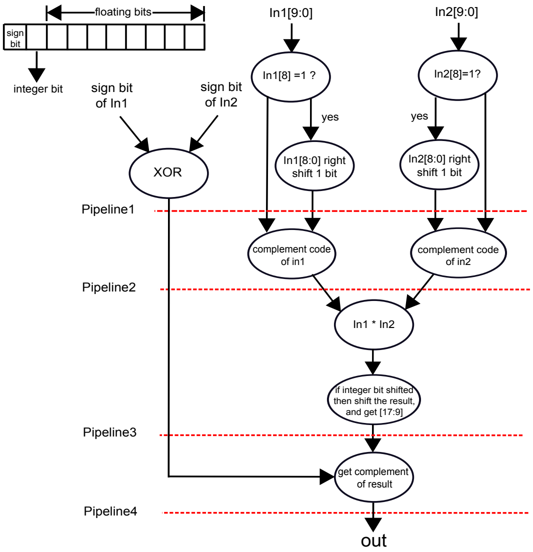


Figure 4.31: Algorithm of pipelined multiplier

4.4.2.2 Equalizer Summers

In semi-digital equalizer, different from the fixed inputs at each differential pair shown in Figure 4.28 and Figure 4.29 the signs of the coefficients in the semi-digital Rx EQ are indicated at the output of the adder. By using switches the two differential data inputs (positive/negative) are exchanged based on the sign bit.

4.5 Design and Implementation of Transmitter Equalizer

4.5.1 Semi-Digital LMS Transmitter Equalizer

The implementation of semi-digital LMS Tx EQ is similar with the semi-digital Rx EQ. Instead of the combination of FFE and DFE in Rx EQ, an LEQ summer with eight coefficients aimed of canceling both the pre-cursors and post-cursors is applied.

4.5.2 Semi-Digital Pilot Transmitter Equalizer

Conventional pilot signal/peak detection method shown in Figure 4.16 requires analog peak detection circuit for the peak detection. In order not to change the memory architecture dramatically, instead of reducing the coefficients by one LSB, the coefficients in the proposed algorithm in Figure 4.32 are increased by one LSB equalizer [15].

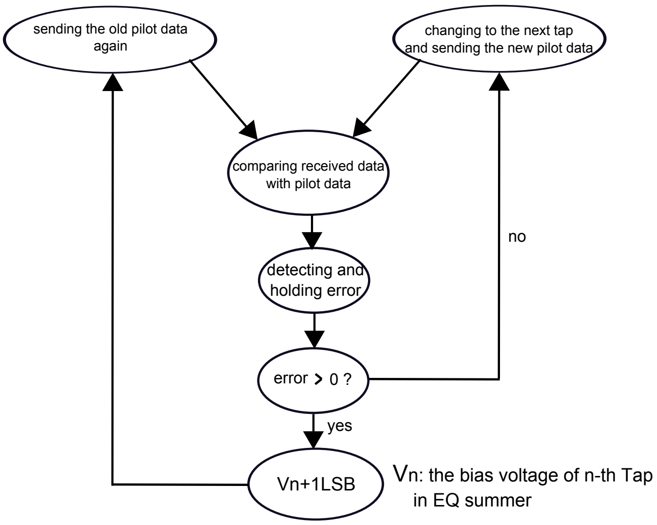

sending the old pilot data again
changing to the next tap and sending the new pilot data
comparing received data with pilot data
detecting and holding error
error > 0 ?
no
yes
Vn+1LSB
Vn: the bias voltage of n-th Tap in EQ summer

Figure 4.32: Proposed pilot algorithm for transmitter equalizer training

At the memory side, the same sampler, but with different preset reference voltage (usually set larger than half of the idea data amplitude) is reused to detect the entire pilot signals, as shown in Figure 4.33. During the Tx EQ training the output of sampler is first

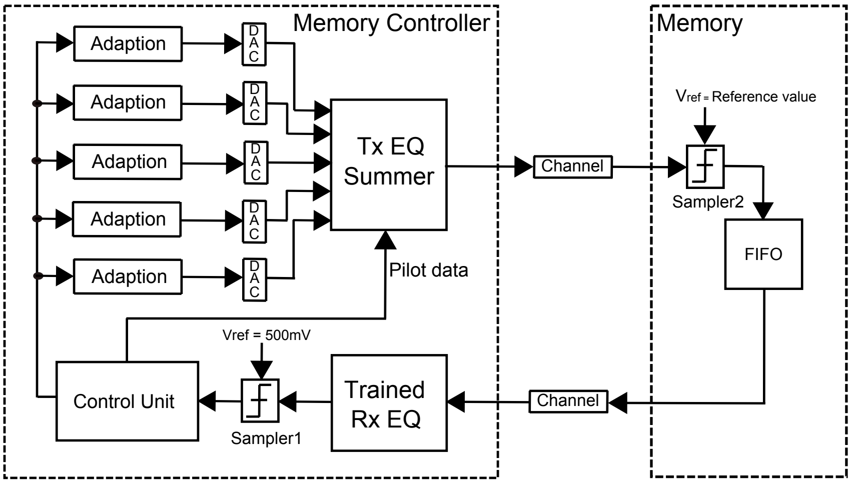


Figure 4.33: The block diagram of Tx EQ with pilot algorithm

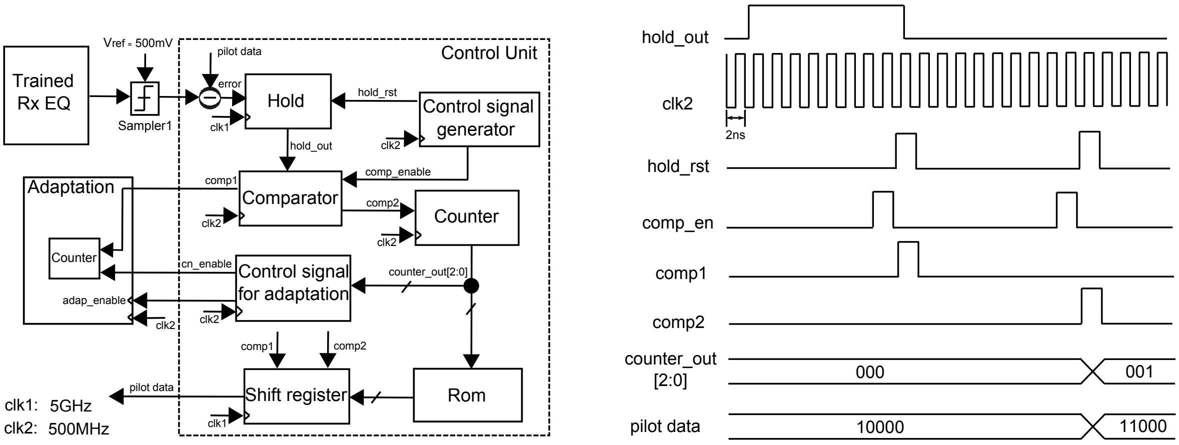
sent to the FIFO in the memory. After that the distorted pilot data in the FIFO are sent through the trained Rx EQ to the control unit, where they are compared with the ideal pilot data in the memory controller. If there is an error, the corresponding bias voltage in equalizer summer is added by one LSB of DAC. And the same pilot data will be sent again. Otherwise, the adaptation for this tap is finished and the new pilot data for next tap will be sent. In this way, coefficients are adjusted one by one.

Table 4.7: Pilot signals and corresponding filter coefficients

Pilot signals	Filter coefficients	Corresponding peaks
10000	C_1	Y(2)
11000	C_2	Y(3)
10100	C_3	Y(4)
10010	C_4	Y(5)
10001	C_5	Y(6)

Assume f_1 is the pre-cursor and f_2 is the main cursor, the training pilot signals and its peak corresponding to the coefficient after Tx EQ and channel is given in Table 4.7. When adjusting the current coefficient e.g. C_2, due to the pre-cursor Y(1) will slightly change its value based on Equation (4.8). However, this change is considerably small as coefficient is increased from a minimal value (negative). The drawback of pilot algorithm is that it can not cancel the ISI components caused by the pre-cursors of the channel since pilot algorithm works based on the peak detection.

Figure 4.34a shows the block diagram how the comparison works in the control unit. The hold component detects the error and keeps the error until the reset signal comes.



(a) The block diagram of control unit

(b) Time sequence of pilot control unit

Figure 4.34: Control unit in pilot Tx EQ

Based on the signal hold_out, the comparator decides whether to update the current tap coefficient (comp1='1') or to move to the next tap coefficient (comp2='1') as shown in Figure 4.34b. Meanwhile signals comp1 and comp2 also control the shift register, when it loads and sends the pilot data. The control signal generator produces the control signal for hold and comparison periodically. In order to ensure the pilot data are sent in the center of the hold_rst and the comp_en, the frequency of these two control signals is set to ten times of the working frequency of generator. Therefore, the adaptation of the coefficient is every 20ns. The address corresponding to all the pilot data in the rom is connected to the output of the counter in the control unit.

To ensure the data rate of equalizer at 5Gbps, two clock frequencies are used: 5GHz for hold and shift register and 500MHz for other components. Each adaptation block is controlled by two signals, cn_enable and adap_enable shown in Figure 4.35, which made adaptation work in either idle or adapting coefficient or keeping coefficient.

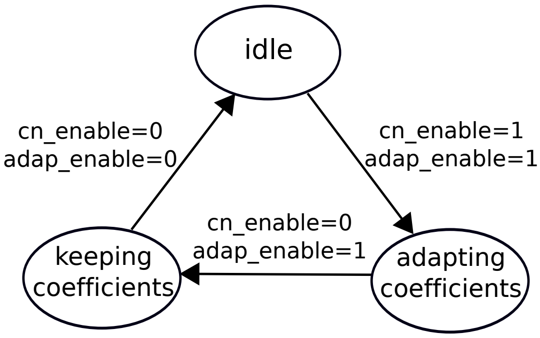


Figure 4.35: The state diagram of adaptation in pilot algorithm

4.5.3 Full-Digital Direct Transmitter Equalizer

Other than refreshing the coefficients of Tx and Rx EQs by adapting the bias voltages of equalizer summers in LMS and pilot algorithms, a full digital Tx EQ is implemented. Figure 4.36 illustrates the loop training for direct calculation.

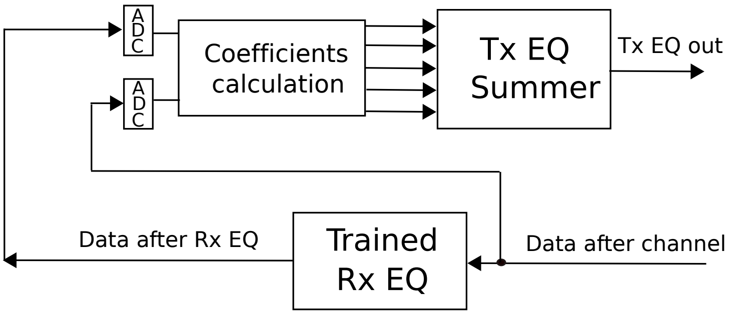


Figure 4.36: Tx EQ training based on direct calculation

Figure 4.37 and Figure 4.38 show the implementation of a full-digital adaptive Tx EQ, which contains multipliers, subtractors, adders and dividers based on Equation (4.10).

All of these digital components are designed using pipeline structure. Each component has a different delay. The delay of each coefficient adds up on to the delay of its previous coefficients. The black value indicates the output delay in the behavior model, while the red one gives the delay values after synthesis.

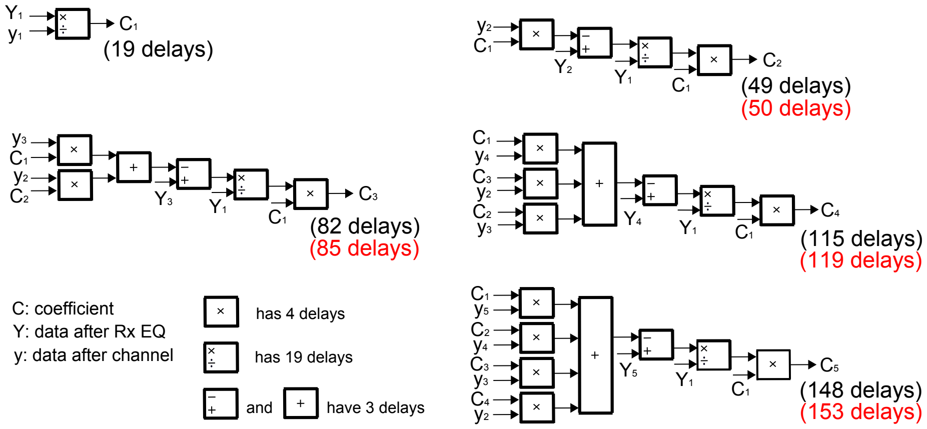


Figure 4.37: Coefficient calculation in direct algorithm

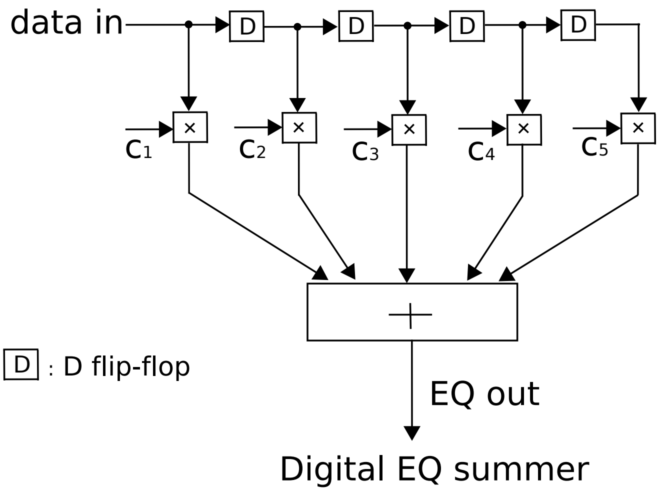


Figure 4.38: Digital EQ summer in a full-digital adaptive Tx EQ

Figure 4.39 shows the implementation of a pipelined divider. The 10 bits data consist of a sign bit, an integer bit, and eight floating bits. The dividend is first extended to 18 bits and the integer bit of divisor is checked whether divisor is smaller than dividend. The calculation of division begins at the highest bit of dividend. If remainder is larger than divisor, remainder subtracts divider. Then the result of subtraction is shifted to the higher bit of remainder and next bit in dividend is shifted to the low bit of remainder. Meanwhile, one is set in quotient and shifted to the higher bit. Otherwise if remainder is smaller than divisor, the next bit in dividend is directly shifted to the low bit of remainder.

Zero is then set in quotient and shifted to the higher bit. The loop continues until the lowest bit in dividend is shifted. The final result (9 bits + 1 sign bit) is given depending on the dividend and divisor values. The Verilog code for the pipelined divider is given in Appendix A.3.

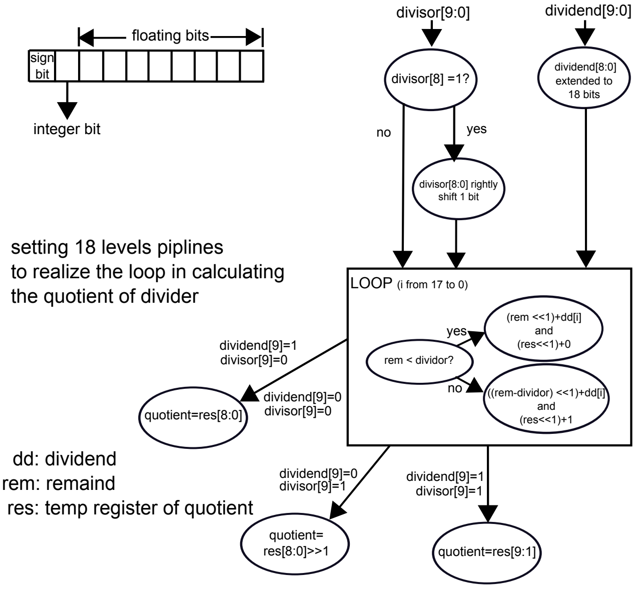

floating bits
sign bit
integer bit
divisor[9:0]
dividend[9:0]
divisor[8] =1?
dividend[8:0] extended to 18 bits
no
yes
divisor[8:0] rightly shift 1 bit
setting 18 levels piplines to realize the loop in calculating the quotient of divider
LOOP (i from 17 to 0)
rem < dividor?
yes
(rem <<1)+dd[i] and (res<<1)+0
no
((rem-dividor) <<1)+dd[i] and (res<<1)+1
dividend[9]=1 divisor[9]=0
dividend[9]=0 divisor[9]=0
quotient=res[8:0]
dd: dividend
rem: remaind
res: temp register of quotient
dividend[9]=0 divisor[9]=1
quotient= res[8:0]>>1
dividend[9]=1 divisor[9]=1
quotient=res[9:1]

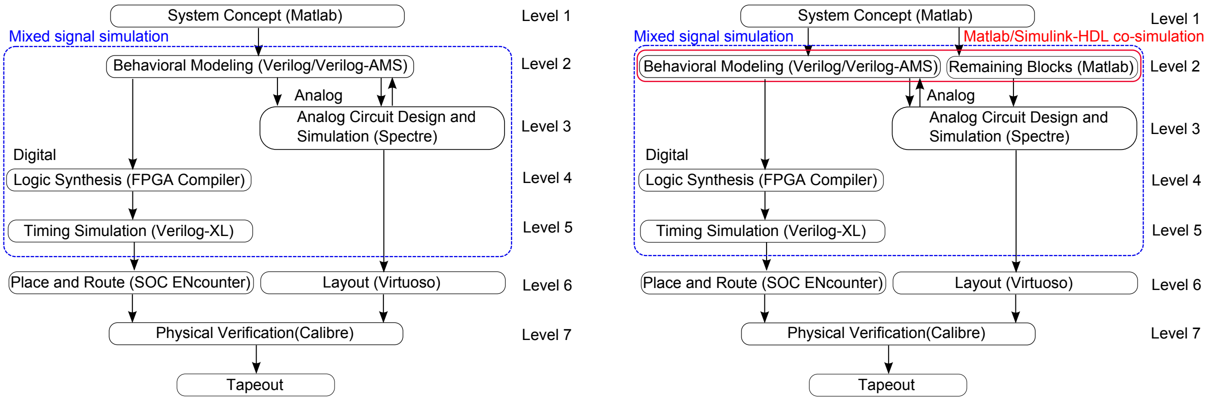
Figure 4.39: Procedure of piplined divider

4.5.4 Simulation Results

4.5.4.1 Simulation Environment

Figure 4.40a illustrates a conventional mixed-signal top-down design flow. The flow mainly consists of seven design levels. It begins with the mathematical verification of system concept. Once system mathematical modeling is verified, block-level simulation based on behavioral models will be performed. This level enables designers to verify whether block-level subsystems can meet the system-level requirements. After successful block-level simulation, system is divided into digital and analog portions. Digital sub-flow comprises several levels such as synthesis, timing simulation and place and route while analog sub-flow is followed by schematic design and layout. Finally, the designed layout for mixed-signal is integrated and implemented in ASIC.

For system-level simulation, the Matlab/Simulink framework has been widely adopted for many years because of its available toolset and ease of use for algorithmic implementation. Many efforts have been made to integrate Matlab into popular EDA tool flows for different fields of applications, e.g. RFID [57], Power Electronics [58]. B. Gestner et al. [59] developed Modelsim-Matlab interface for RTL debugging and verification.


System Concept (Matlab)
Mixed signal simulation
Matlab/Simulink-HDL co-simulation
Behavioral Modeling (Verilog/Verilog-AMS)
Remaining Blocks (Matlab)
Analog
Analog Circuit Design and Simulation (Spectre)
Digital
Logic Synthesis (FPGA Compiler)
Timing Simulation (Verilog-XL)
Place and Route (SOC ENcounter)
Layout (Virtuoso)
Physical Verification(Calibre)
Tapeout
Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7

(a) Conventional design flow (b) Proposed design flow

Figure 4.40: Design flow for mixed-signal design

For behavioral-level simulation, Verilog-AMS has been applied for mixed-signal design environments. As an extension of Verilog-HDL and Verilog-A, Verilog-AMS provides an accurate behavioral modeling with a single simulator. P. Frey et al. [60] explained the semantics of Verilog-AMS connect modules between analog and digital contexts.

However, a major problem of the conventional design flow is that it requires a higher level of refinement from Matlab/Simulink to Verilog-AMS. Although some existing tools, like Simulink HDL Coder by Mathworks or Synphony HLS by Synopsys, allow synthesizable HDL code generation direct from Matlab/Simulink environment, generated RTL are often unsuitable for efficient FPGA or ASIC implementation due to their limitation in supported blocks. Therefore, a "full-custom" Verilog-AMS for behavior modeling often remains competitive. To benefit from both Matlab/Simulink and Verilog-AMS and to minimize the design time, Matlab/Simulink-HDL co-simulation is proposed in this paper.

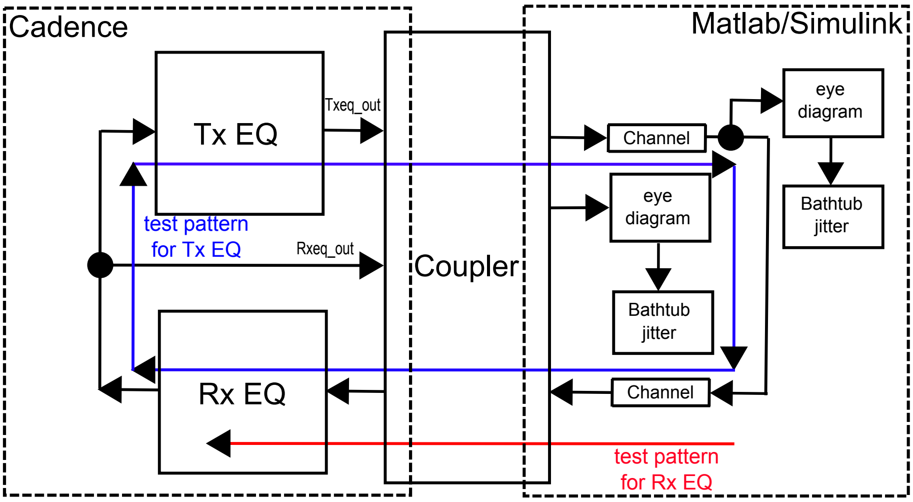
The need for co-simulation is indispensable in mixed-signal design. It allows individual components to exchange information in a collaborative manner by different simulation tools running simultaneously. Moreover, different components at different abstraction levels can be simulated using co-simulation platform. In this way, both design acceleration and verification of low-level designs with system-level models can be achieved.

P. Daglio et al. [61] proposed a VHDL-AMS method to explore the mixed-signal domain at different abstraction levels by integrating different EDA tools. Based on the design flow, an example of an embedded flash macro-cell is illustrated as a test case. Verilog-AMS which is an alternative HDL for mixed-signal design is compared with VHDL-AMS by F. Pecheux et al. [62] such as language aspects, expression of structure, signal-flow semantics, etc., where an airbag system application was demonstrated as a case study. However, both of them are close to implementation level. Their relatively slow simulator for system validation makes them inefficient for complex applications such as system-on-chips.

To fill the gap between algorithmic domain and hardware design, many methods [63] [50] are proposed. Matlab-VHDL [63] shows its potential in validation of different system architectures, but it is unsuitable for mixed-signal design due to its lack of analog/mixed-signal support. SystemC-AMS [64] which is an extension of SystemC, combines the powerful means to describe the systems from abstract specifications to RTL models and to model continuous-time systems/discrete-event systems.

Figure 4.40b illustrates the proposed design flow. Compared to the conventional design flow, Matlab/Simulink-HDL co-simulation is added in level 2. After system modeling, some design blocks are refined in Verilog-AMS. Co-simulation between Matlab and Verilog-AMS enables verification acceleration of block level implementation. At level 3, Verilog-AMS models of analog circuits can be replaced by Spectre netlist, if available. Multiple iterations are needed between level 2 and level 3 to make Verilog-AMS model fine-tuned with real circuits. Thus, Matlab/Simulink Cadence co-simulation is required.

To verify different algorithms, Matlab/Simulink Cadence co-simulation environment [17] is set up. The test patterns i.e. PRBS for LMS Tx EQ, pilot patterns for pilot Tx EQ and an impulse pattern for direct Tx EQ are generated in Matlab and the communication between Matlab and Cadence is through the coupler as shown in Figure 4.41. The Tx EQ and Rx EQ are implemented in TSMC LP 65nm technology in Cadence while eye diagrams, jitter measurements and bathtubs are evaluated in Matlab (see Appendix A.2). The eye diagram is generated by overlaying the impulse responses of a data sequence once all the coefficients in the Tx and Rx EQs are determined. Jitter and bathtub are evaluated according to Appendix A.1.

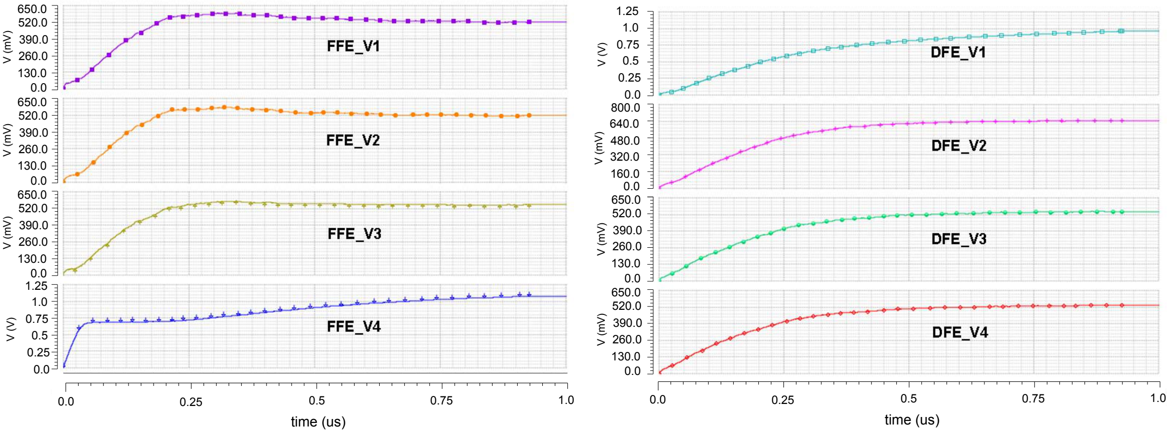


Co-simulation of Tx and Rx EQs

Figure 4.41: Co-simulation for the measurement

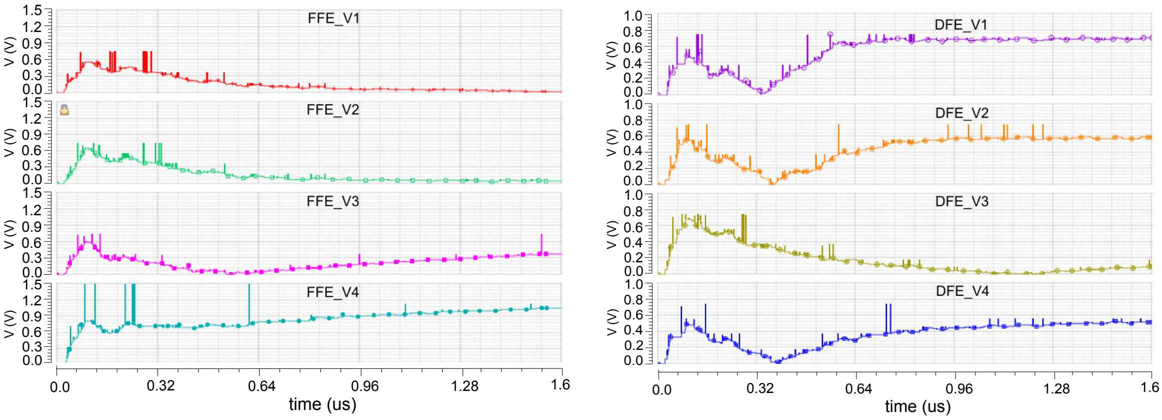
4.5.4.2 Receiver Equalizer

Figure 4.42 shows bias voltages for the coefficient adaption at the gate-level simulation in adaptive receiver EQs. The peaks on the voltage curves in Figure 4.42c and Figure 4.42d are due to the different delays of the digital components after synthesis. The voltage of FFE_V4 has the highest value among four coefficients, which implies the fourth tap in FFE is the main tap. Due to the slight difference in the equalizer summer circuit between analog and semi-digital Rx EQ, the voltages such as FFE_V3 in Figure 4.42c decreased to zero volt and then stayed stable at some negative voltages (sign bit is not shown) while the voltages in the analog Rx EQ are always positive. In the implementation of semi-digital adaptive Rx EQ, 8-bit ADC and DAC are used, which implies the resolution of approximately 3.9mV (Vdd=1V). With a fixed step size μ=0.032, the equalizer is not able to detect the error lower than 120mV (=3.9mV/0.032), which in return leads to the coefficients of FFE_3, DFE_3,and DFE_4 to be zero, although their voltages are negative.



(a) Analog Rx EQ: FFE

(b) Analog Rx EQ: DFE

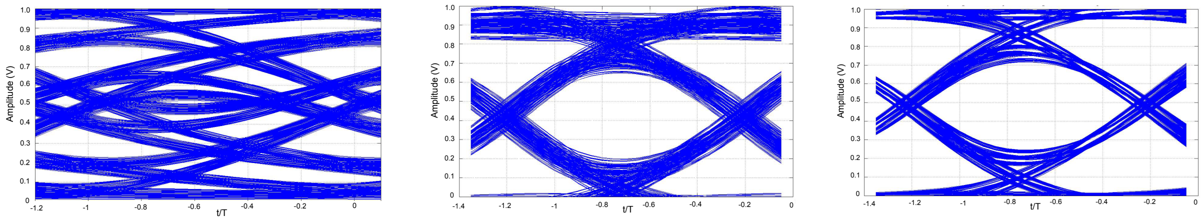


(c) Semi-digital Rx EQ: FFE

(d) Semi-digital Rx EQ: DFE

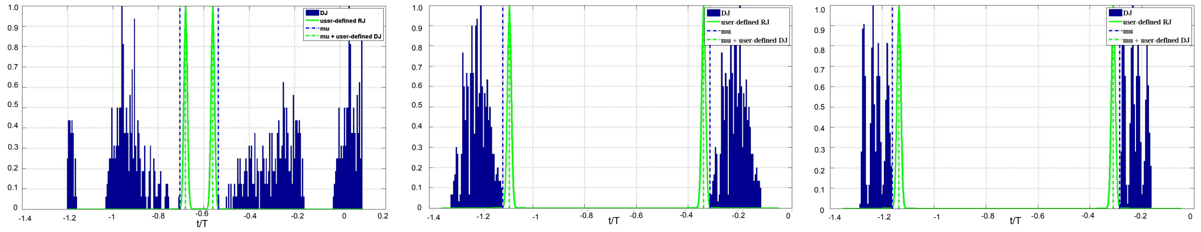
Figure 4.42: Adaptive bias voltages for receiver equalizer

Figure 4.43, Figure 4.44 and Figure 4.45 illustrate the eye diagram, jitter and bathtub with/without adaptive Rx EQs, respectively.



(a) Without EQ (b) Analog LMS Rx EQ (c) Semi-digital LMS Rx EQ

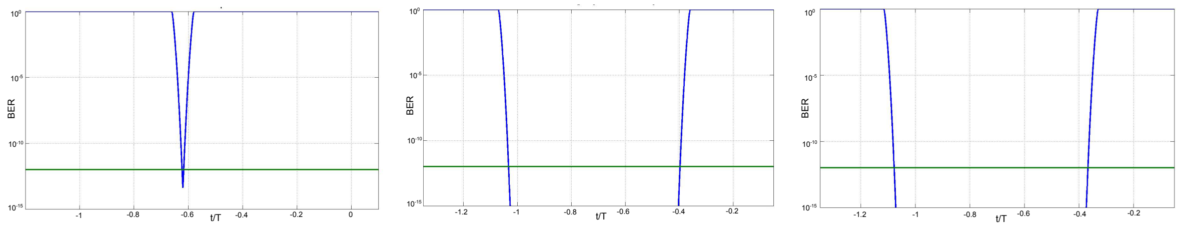
Figure 4.43: Eye diagrams with/without adaptive Rx EQs



(a) Without EQ (b) Analog LMS Rx EQ (c) Semi-digital LMS Ex EQ

Figure 4.44: Jitter with/without adaptive Rx EQs

Due to the ISI, the eye is almost closed in Figure 4.43a. As a result, the deterministic jitter (DJ) and random jitter (RJ) are distributed in the entire region, shown in Figure 4.44a. The green solid lines indicate additional user-defined RJ distribution, where its center is marked with green dotted lines. The two green dotted lines approach to the eye center when additional user-defined DJ is present. Figure 4.45a demonstrates its bathtub curves, where they intersect at each other at BER=10^{-12}. In contrast to Figure 4.43a, the eye openings for both analog and semi-digital LMS Rx EQs in Figure 4.43b and Figure 4.43c are improved dramatically.



(a) Without EQ (b) Analog LMS Rx EQ (c) Semi-digital LMS Rx EQ

Figure 4.45: Bathtub curves with/without adaptive Rx EQs

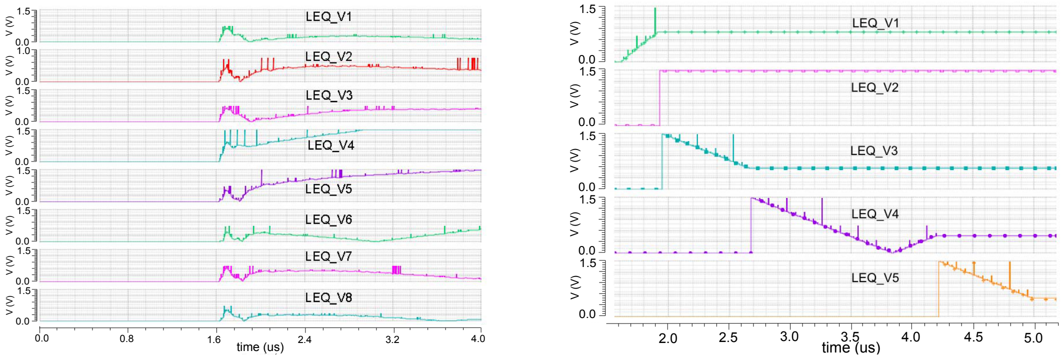
Figure 4.44b, Figure 4.44c, Figure 4.45b and Figure 4.45c illustrate that the Rx EQs can reduce the jitter and make bathtub curves wider. Compared with the analog Rx EQ, the semi-digital Rx EQ has better eye diagram, fewer jitter and wider bathtub curves. However, it needs more time to reach the convergence shown in Table 4.8.

Table 4.8: Comparison of analog and semi-digital adaptive Rx EQ

	Analog Rx EQ	Semi-digital Rx EQ
Training time (μs)	1	1.6
Data rate (Gbps)	5	5
FFE coefficients	-0.04, -0.0398, -0.0436, 1.6747	0, 0, 0, 1.6357
DFE coefficients	-0.4294, -0.1616, -0.0385, -0.0385	-0.4275, -0.1962, 0, 0

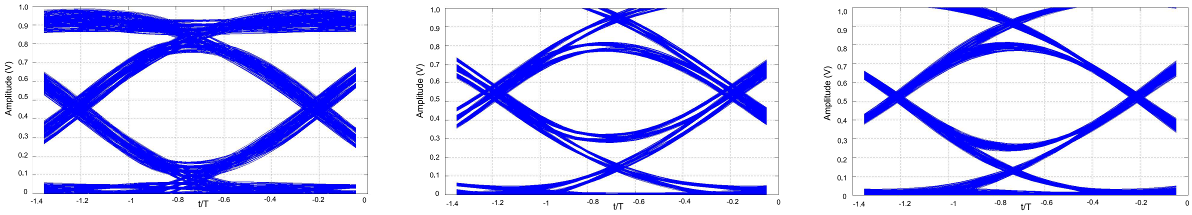
4.5.4.3 Transmitter Equalizer

Figure 4.46 shows the bias voltages for the coefficient adaption at the gate-level simulation in LMS and pilot adaptive Tx EQs. The main difference between these two algorithms is that the voltage in pilot algorithm is adjusted only when the previous voltage is stable while the voltages in LMS algorithm are updated simultaneously.



(a) LMS algorithm (b) Pilot algorithm

Figure 4.46: Adaptive bias voltages using pilot and LMS algorithms

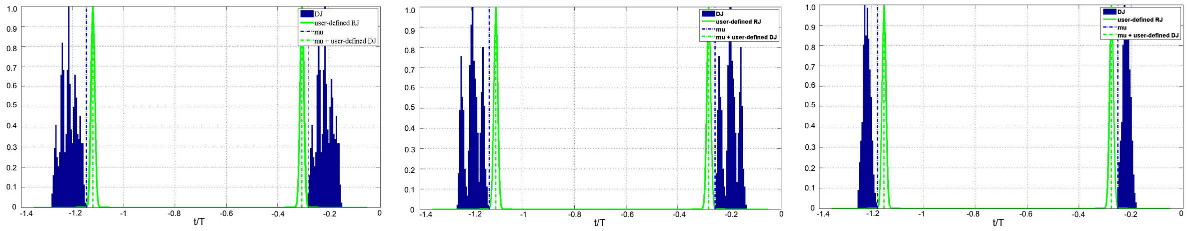


(a) LMS algorithm (b) Pilot algorithm (c) Direct algorithm

Figure 4.47: Eye diagrams using different algorithms in the adaptive Tx EQs

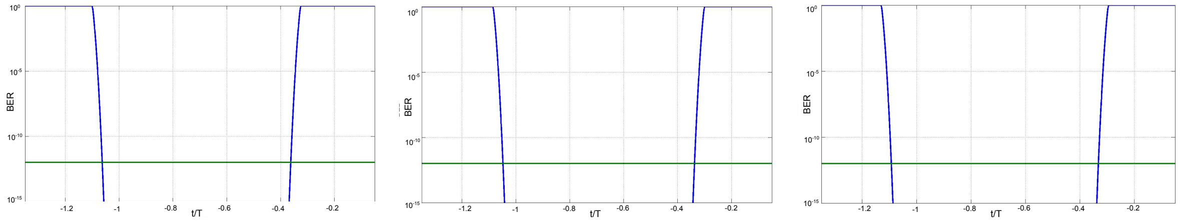
Figure 4.47 shows the eye diagrams based on three different adaptation algorithms and proves that all of them can improve the data quality dramatically compared with the eye diagram without equalizer. Among these three algorithms, direct algorithm gives the most promising performance in terms of jitter and bathtub curves shown in Figure 4.48 and Figure 4.49.

Table 4.9 shows the comparison among the three different algorithms. The sequential calculation of floating coefficients in the direct algorithm leads to the largest digital area



(a) LMS algorithm (b) Pilot algorithm (c) Direct algorithm

Figure 4.48: Jitters using different algorithms in the adaptive Tx EQs



(a) LMS algorithm (b) Pilot algorithm (c) Direct algorithm

Figure 4.49: Bathtub curves using different algorithms in the adaptive Tx EQs

and its complexity constrains the data rate (up to 1 GHz) after synthesis. Pilot algorithm occupies the smallest area due to its simplicity. However, it has consumed the longest training time, which can be optimized by about 50% reduction, if faster clock for the control unit is used. Regarding to the coefficients, both pilot and direct algorithms have their limitation in canceling the ISI components caused by the pre-cursor in the channel.

Table 4.9: Comparison of Tx EQs with three algorithms

	LMS algorithm	Pilot algorithm	Direct algorithm
Area (mm^2)	0.03	0.00218	0.17
Training time (μs)	2.4	3.4	0.153
Data rate (Gbps)	5	5	1
Coefficients	0, -0.0021, -0.1642, 1.6895, -0.7791, 0.1435, 0, 0	1.65, -0.5, -0.073, 0.008, -0.025	1.633, -0.629, 0.043, 0, -0.352

4.6 Summary

In this chapter, a simplified system architecture for high-speed communication systems and the comparison of different adaptive receiver and transmitter equalizers are presented. In contrast to the conventional architecture using adaptive equalizers, the proposed architecture simplifies the design in the slave IC in terms of hardware overheads. To verify the proposed architecture, receiver equalizers are designed with analog and semi-digital circuits while the transmitter equalizers are implemented with semi-digital and full-digital circuits. Performances such as eye diagram, jitter, area and training time are used to evaluate different implementations of receiver and transmitter equalizers. Results show that with the proposed system architecture SI is significantly improved. Compared with analog receiver equalizer, the semi-digital receiver equalizer shows better eye diagram, although it needs more training time. Three different algorithms in the transmitter equalizers are also compared in this paper. Direct algorithm takes the least training time, but it consumes the largest area and runs at relatively slow data rate. Among three algorithms, LMS shows the best performance with reasonable area and acceptable training time at high data rate. It differs with the other two algorithms in the cancellation of both pre- and post-cursors.

Chapter 5

I/O Design

Contents

5.1 State-of-the-art 68
- 5.1.1 I/O Driver Structure 68
- 5.1.2 PVT Compensation Schemes 70

5.2 Proposed Hybrid Digital Impedance Calibration 74
- 5.2.1 Calibration Architecture 74
- 5.2.2 Hybrid Calibration Algorithm 76

5.3 Simulation Results 78

5.4 Summary 80

As the demands for high volume data exchanges increase in many fields such as multimedia and telecommunications, significant efforts have been devoted to improve chip I/O performances to achieve high bandwidth. However, impedance mismatch between I/O driver and a transmitting channel causes signal reflections which interfere with the incoming data in terms of overshoot, undershoot or ringing. To ensure high quality SI at the data rate beyond several gigabits per second, high-speed interfaces require minimum variations of output voltage level and slew rate over PVT variations.

Figure 5.1 depicts the P-channel Field-Effect Transistor (PFET) impedance variation from target impedance of 120Ω of a GDDR5 I/O interface from 76.4Ω to 173.8Ω caused by the PVT changes without calibration in TSMC 65nm LP technology. Each label is composed of a mark number, a temperature and an impedance value. The large impedance variation is not acceptable in the circuit. Therefore, impedance calibration becomes key factors in the high-speed interfaces such as GDDR5. By controlling the impedance of off-chip driver (OCD) and on-die terminator (ODT), signal reflections can be effectively absorbed, and thus maintain the SI.

In this chapter, different algorithms for digital impedance calibration are compared for impedance calibration in terms of power and calibration time.

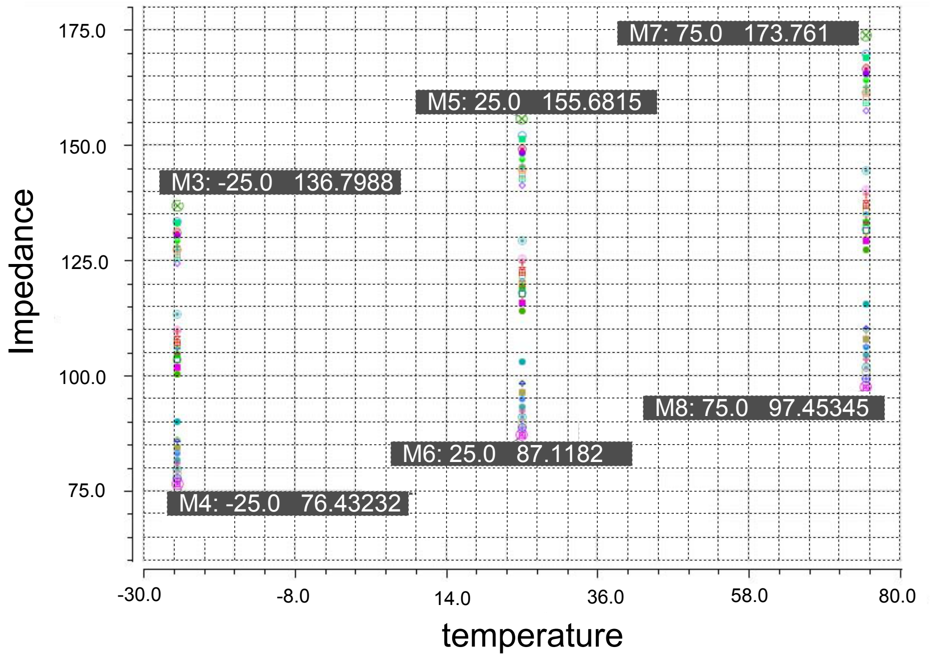


Figure 5.1: PFET impedance variations over PVT variations

5.1 State-of-the-art

5.1.1 I/O Driver Structure

Authors in [65] presented different driver architectures such as simple programmable I/O driver, current assist I/O driver and poly resistance I/O driver. The simple programmable I/O driver depicted in Figure 5.2 consists of an array of binary weighted

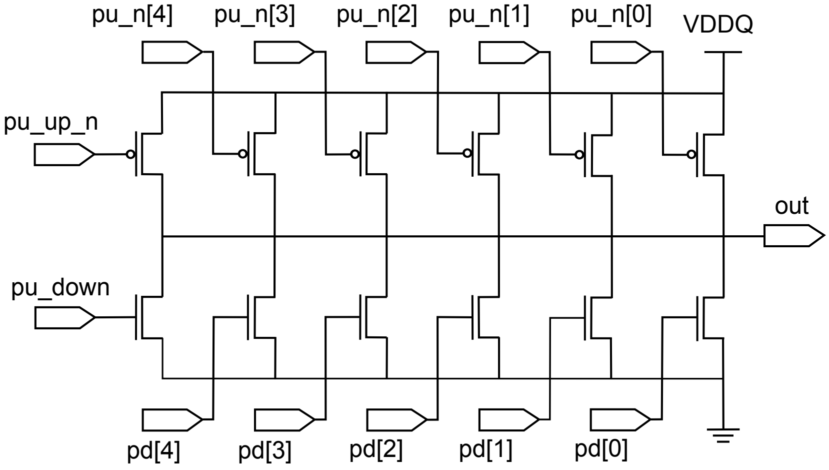


Figure 5.2: Schematic of simple programmable I/O driver

PMOS transistors and NMOS transistors. The impedance matching is achieved by enabling different combination of transistors in array. The first branch is a reference circuit in the nominal condition without any PVT variations while the other branches are for the calibration purpose. When driving high at the output the impedance of the driver is equivalent to the resistance of PMOS array. Similarly, when driving low at the output the impedance of the driver is equivalent to the resistance of the NMOS array. The total impedance decreases when the number of "on" transistors increases. However, the non-linearity of the simple programmable I/O driver causes the change of the driver impedance and its output voltage, which may lead to the matching issues at VDD/2.

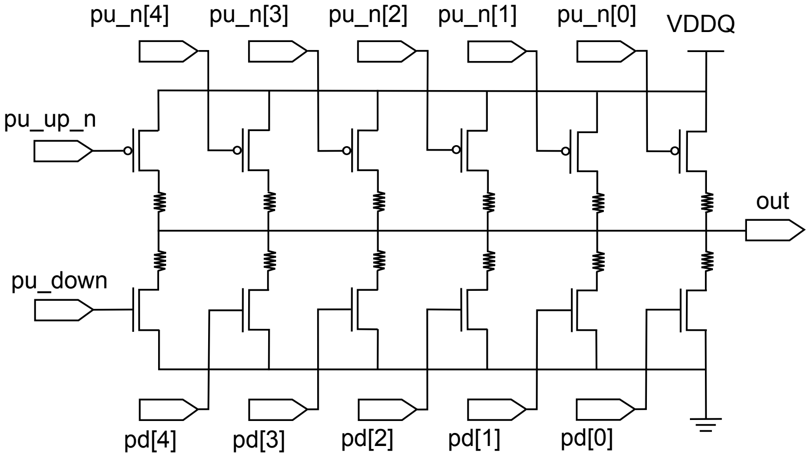


Figure 5.3: Schematic of poly resistance I/O driver

To flatten the output impedance curve, the current assist I/O driver shown in Figure 5.4 is used by extending the simple programmable I/O driver with additional transistors. Current assist pull-up NMOS is always on when driving high at the output. In the similar way, current assist pull-down PMOS is always on when driving low at the output. By inserting this additional current NMOS or PMOS, the linearity of the I/O driver can be improved dramatically. When the number of current assist transistors increase, the output impedance is close to a constant.

As depicted in Figure 5.3 poly resistance I/O driver architecture is an alternative option for flattening the output impedance curve by inserting a poly resistor in each branch of the driver. The non-linearity of the simple programmable I/O driver is compensated by the poly resistor. The percentage of the poly resistor value in the total output impedance indicates the linearity of the driver, but at the cost of high output capacitance, which limits the speed of data transmission.

Combination of poly resistor I/O driver architecture and current assist I/O driver architecture is also a possible solution for ensuring the linearity of the I/O driver.

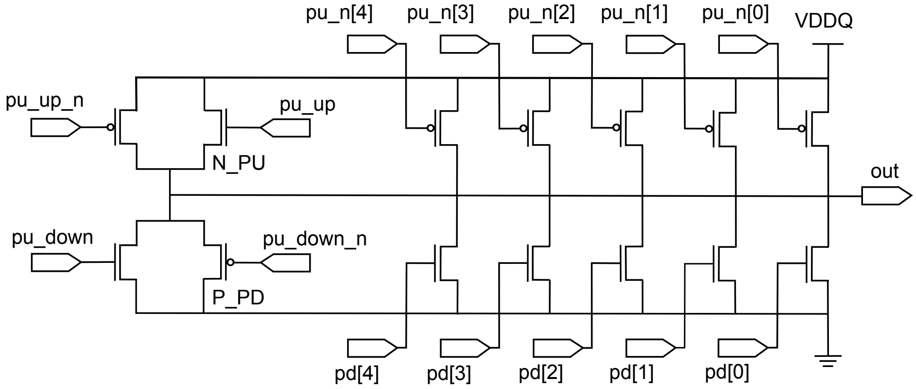


Figure 5.4: Schematic of current assist I/O driver

5.1.2 PVT Compensation Schemes

Figure 5.5 illustrates the conventional calibration logic. After the initial state, it is first checked if the output voltage is matched within the specified voltage range. When it is out of that range, match signal switches the state from high to low. As soon as the match signal goes low, the calibration starts in the next clock cycle. The next step is to check whether the output voltage of I/O is above upper bound of the specified range or below the lower bound of that, indicated by the "reference check" box. Depending on that, the reference transistor will be turned on or off. During the calibration, decision will be made whether to turn on or off each branch until the match signal goes high.

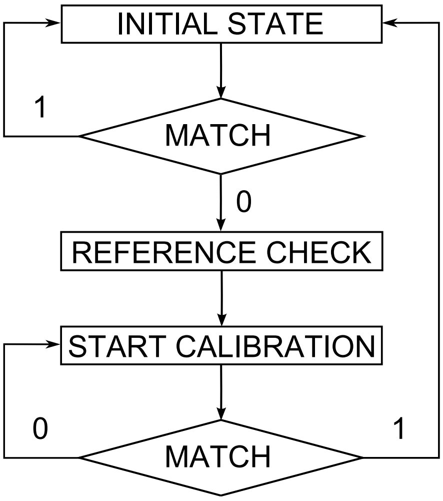


Figure 5.5: Conventional impedance calibration algorithm

A review of PVT compensation circuits is given by the authors in [66]. Figure 5.6 shows the types of compensation circuits in I/O driver, which can either be analog or digital. The analog impedance calibration is based on adjusting the gate voltage of the compensating transistor until the drain voltage matches to the reference voltage which is equal to the voltage drop across a known external resistor. In digital impedance calibration, several transistors constituting the I/O driver are connected in parallel. Each leg of the transistor in the I/O driver can be turned on or off. Same as the analog calibration, the drain voltage of the parallel transistors is compared with the reference voltage through a comparator. Depending on the comparator output, the on or off state of the current device is determined. Until the state of the last device is known, the total driver strength is presented by all the "on" transistors.

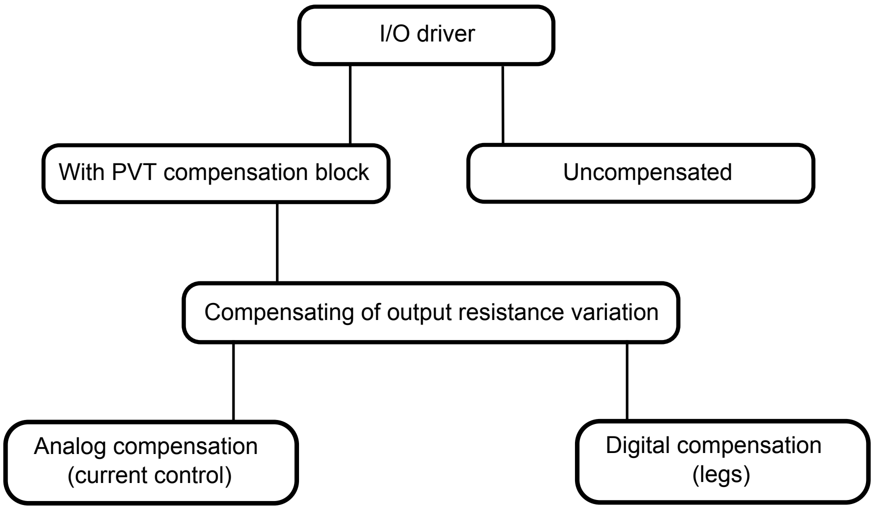

I/O driver
With PVT compensation block
Uncompensated
Compensating of output resistance variation
Analog compensation (current control)
Digital compensation (legs)

Figure 5.6: Compensation schemes [66]

The primary difference between analog and digital impedance calibration is that in the digital impedance calibration the transistors served as I/O driver are fully turned on or off while in the analog scheme the transistor is partially on. The gate voltage in the analog impedance calibration is dependent on the current change due to PVT variation. In other words, analog impedance calibration is sensitive to noise. Compared to the analog scheme, digital impedance calibration is more immune to noise and suitable for chips with high I/O counts. Therefore, digital impedance calibration under PVT variations is focused and further explored.

Authors in [67] proposed two circuits to detect PVT variations: DLL based and ring oscillator based PVT compensation. The DLL based circuit detects the PVT variations by generating the phase error between the input clock signal and the output of an I/O driver while a ring oscillator based circuit converts the current variations into a clock signal by employing a current controlled oscillator (CCO). Compared to the DLL based circuit, the ring oscillator based circuit is able to detect even skewed corners with the help of a modified current reference circuit by diode connected PMOS and NMOS transistors.

In their work, linear search using equal-sized transistors as I/O driver is applied during the impedance calibration. The linear search starts from the center value of the total

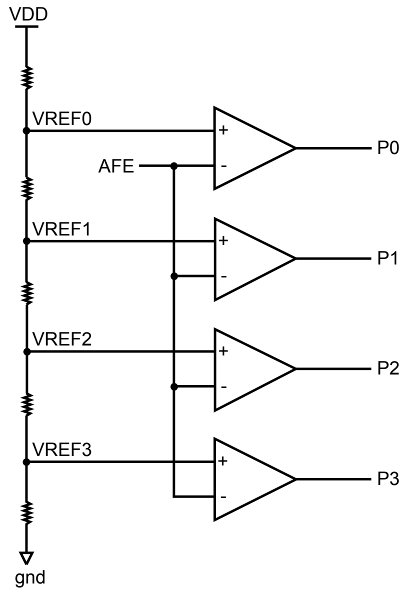


Figure 5.7: Analog Comparator block

driver strengths. Based on the output of the comparator, the desired value is searched sequentially. In the linear search, the number of the calibration transistors is high, which leads to a high number of comparators illustrated in Figure 5.7. Figure 5.8 illustrates the digital comparator in its digital logic.

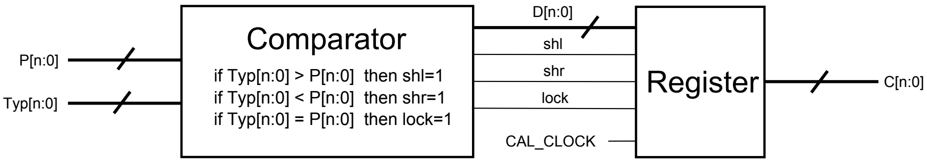

P[n:0]
Typ[n:0]
Comparator
if Typ[n:0] > P[n:0] then shl=1
if Typ[n:0] < P[n:0] then shr=1
if Typ[n:0] = P[n:0] then lock=1
D[n:0]
shl
shr
lock
CAL_CLOCK
Register
C[n:0]

Figure 5.8: Digital block for calibration

However, both DLL based and ring oscillator based PVT compensation circuits require a large number of comparators to generate thermometer codes for a high resolution when compensating the PVT variations by comparing the thermometer codes with a reference code in the nominal condition. The number of comparators increases exponentially with the number of calibration legs. For different I/O driver circuits, using diode connected PMOS and NMOS transistors are not able to detect skewed corners.

D. bhattacharya et al. [68] suggested a compensation circuit in Figure 5.9 including a monitor circuit indicating the status of pull-up or pull-down impedance over variations in PVT conditions and a control circuit in Figure 5.10 to generate PVT bits. In the monitor circuit, a reference current over a precision resistor is fed into the duplicated monitor device. The resulted voltage over the impedance is then converted into digital control

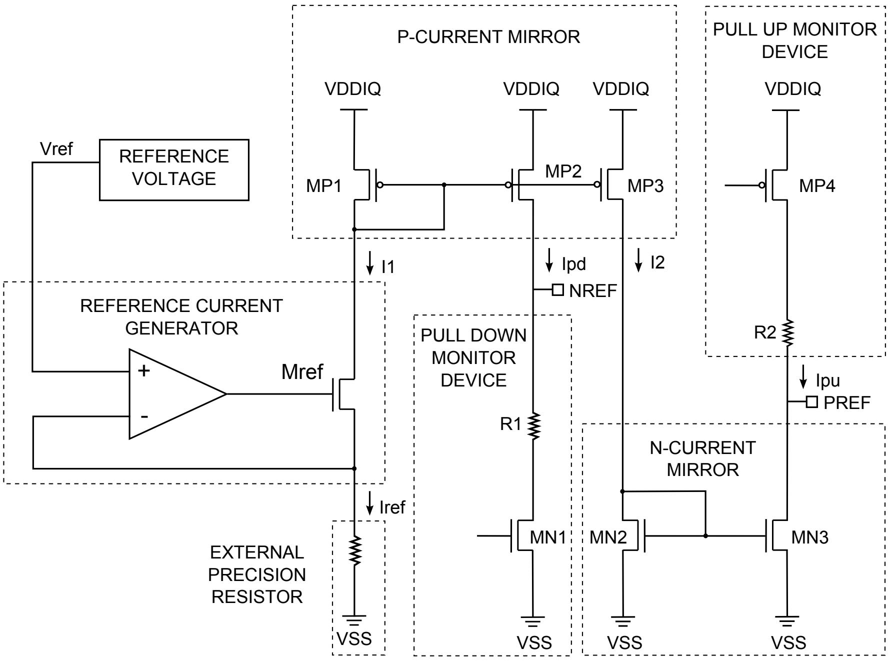


Figure 5.9: PVT variation detect circuit

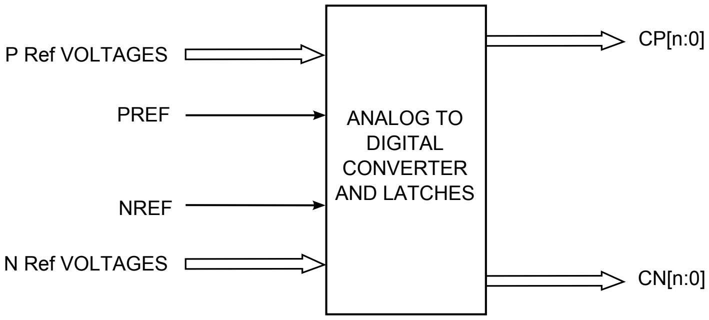


Figure 5.10: Analog to digital converter

bits with given threshold voltages for making comparison decisions. The drawback of this circuit is its high power consumption and area penalty by using a large number of comparators in the ADC circuit when high resolution is required.

S. Mei [69] suggested a compensation circuit including a driver circuit, a comparator circuit and a binary searcher [70]. Figure 5.11 illustrates the binary search algorithm for four bits. The advantage of binary search algorithm is its fast convergence. For example, the binary search takes at maximum six clock cycles for 64 driver strength values (6-bit) to determine a particular strength of the target impedance. On the other hand, it has the drawback of its higher power consumption by switching each bit sequentially compared to the linear search. Furthermore, the resulted switching noise may affect the SI. On the

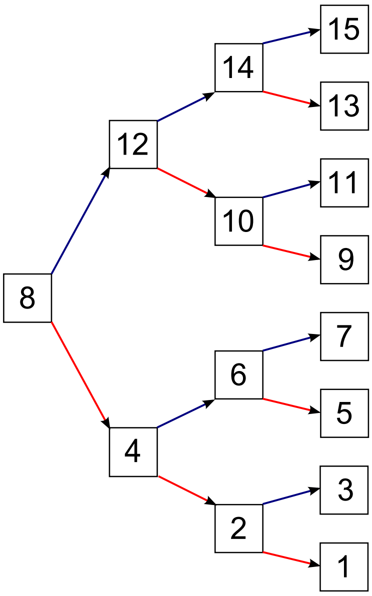


Figure 5.11: Illustration of binary search algorithm for 4 bits

other hand, the step size of binary search halves in each clock cycle, thus reducing the switching noise.

To compensate those disadvantages, the need for a power-efficient calibration method with less complexity and less silicon area arises.

5.2 Proposed Hybrid Digital Impedance Calibration

5.2.1 Calibration Architecture

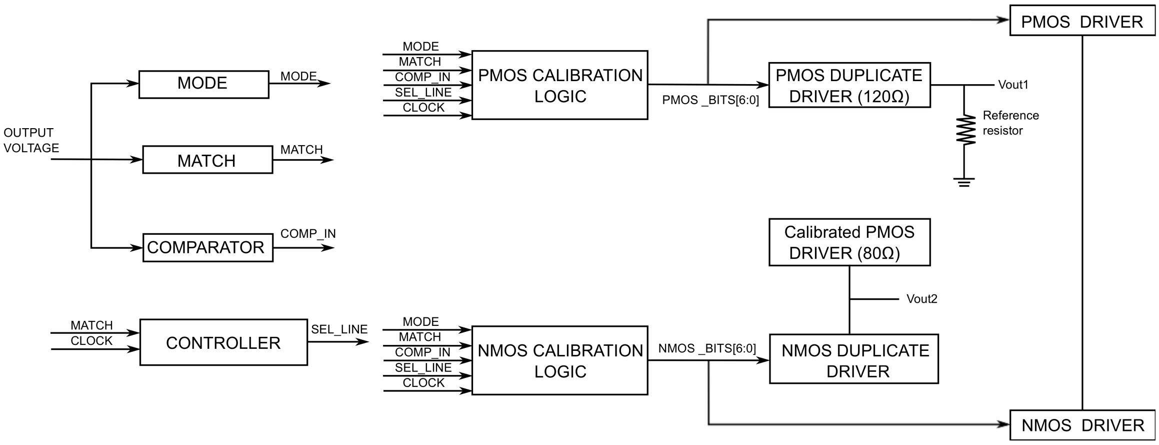


Figure 5.12: Proposed calibration architecture

Figure 5.12 depicts the proposed hybrid architecture for digital impedance calibration. The main components in Figure 5.12 are mode, match, comparator and controller. The match block in Figure 5.12 provides the information to the calibration block about calibration start and stop. Its internal architecture is the window comparator architecture shown in Figure 5.13. The upper and lower limits set by two comparators are $V_{REF} + 1\% * V_{REF}$ (=Ref_{up}) and $V_{REF} - 1\% * V_{REF}$ (=Ref_{low}), respectively, where V_{REF} is half of the power supply VDD. The comparator block provides the information to the calibration block about the status of the output voltage. The negative input of the comparator is set to be V_{REF}.

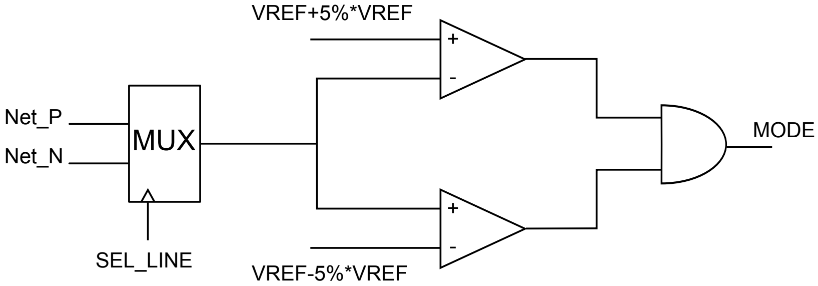


Figure 5.13: Mode and match block architecture

Similar to the match circuit, the mode block is composed of two comparators, but with different upper and lower limits at the inputs of the comparators. The upper and lower bounds are determined by the strength of the calibration transistors. The function of the mode block is to select between the conventional impedance calibration algorithm (binary search) and an alternative calibration algorithm. Compared to the previous work [67] [68], this architecture contains only five comparators independent of the required resolution, which saves power and chip area.

As defined in [71], the calibration sequence starts with the PFET impedance consisting of only PMOS transistors. The PFET impedance is calibrated against the external precision resistor. The voltage across the resistor is then fed to the match circuit. If this voltage is within the specified range in the match circuit, no calibration is required. Once PVT varies which causes the voltage across the resistor to drift beyond the specified voltage range, the calibration process starts. At the beginning of the calibration, the strength control is first set to the center value by the control codes. During the calibration process, the drive strength is increased or decreased according to the comparator output by adjusting the calibration codes. The process lasts until the match signal goes high. When the calibration stops, calibration codes are saved. After the PFET impedance calibration, the N-channel Field-Effect Transistor (NFET) impedance calibration starts. Controller enables the NFET calibration, disables the PFET and selects the corresponding output to each sub block by using an analog multiplexer. The NFET calibration is performed against a calibrated PFET section. In case that the target impedances of NFET and PFET are not equal, the setting of the widths of the PFET or NFET may vary.

Following the calibration sequence, controller sequentially selects the PMOS, NMOS and terminator section indicated by its output "SEL_LINE" (2 bits) during the calibration process. When match signal is high, the controller waits two clocks for allowing the output of comparators to settle down before switching the "SEL_LINE" signal.

Figure 5.14 illustrates the proposed hybrid calibration logic. As match signal goes to low, mode signal is checked if hybrid calibration is used as opposed to the conventional calibration logic shown in Figure 5.5.

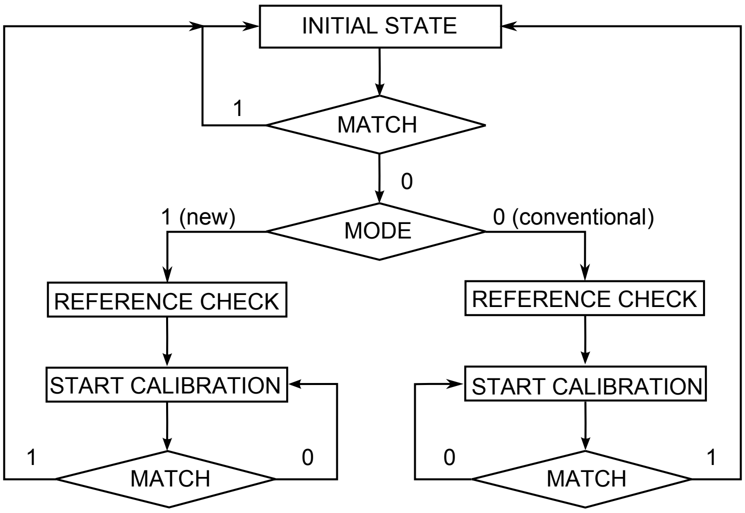


Figure 5.14: Proposed impedance calibration algorithm

5.2.2 Hybrid Calibration Algorithm

In the proposed hybrid calibration, poly resistor I/O architecture is used, as it reduces the effect of PVT variations and provides the linearity in the I/O driver. To ensure good linearity, 75% of the total impedance is contributed by poly resistors while 25% by transistors. As defined in [71], the I/O cell for a GDDR5 memory consists of 60Ω PFET and 40Ω NFET in pseudo open drain (POD) interface. For the symmetry reason, the same specification is suitable for the I/O cell in the memory controller. In order to mitigate the effects of SSO noise on SI, two 120Ω PFET sections and two 80Ω NFET impedance sections connected in parallel are implemented to control the slew rate [72] and to meet the specification for I/O cell impedance as well.

Figure 5.15 presents the PFET part in the I/O driver using binary search algorithm. Similar to the programmable I/O driver, the first branch is a reference circuit for no PVT variations. Depending on both the required resolution and CMOS technology, an example of six employed calibration bits is shown for a GDDR5 interface in TSMC 65nm LP technology as with five calibration bits not all corners under PVT variations can be calibrated. The transistors in the calibration branches are weighted 8W-4W-2W-1W-0.5W-

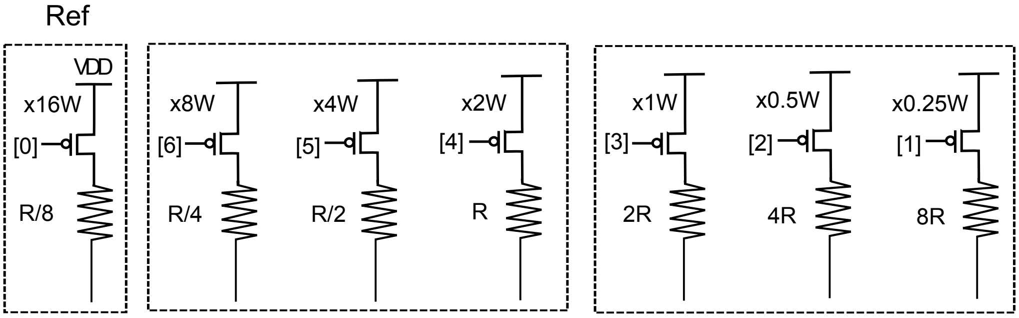


Figure 5.15: PFET part in the I/O driver

0.25W while the resistors are weighted 1/4R-1/2R-R-2R-4R-8R where W represents the width of a finger in each transistor and R is the unit poly resistor resistance in each branch. The strength of the branch decreases with the bit priority order. Most significant bit (MSB) has the highest strength while least significant bit (LSB) has the lowest strength. It should be noted that the width of calibrated PMOS driver in Figure 5.12 is increased by 1.5 times with the same calibration code delivered by the PFET impedance calibration to meet the target impedance.

5.2.2.1 Hybrid Linear Search

One approach of hybrid impedance calibration is to apply linear search and binary search. If the output voltage of the driver for a corner is within the voltage window specified in the mode circuit, the linear search is applied. For the linear search three additional branches with a constant step size are used. The width of each transistor is 0.5W connected in series with a resistance 4R for each branch. The linear step size is set small enough to calibrate those corners close to $\pm1\%*V_{REF}$ variations. The linear search starts from the center value and then traverses sequentially in the direction of $\pm1\%*V_{REF}$ variations based on the result of the comparator. During the linear search in case that the PFET output voltage is smaller than Ref_{low} or the NFET output voltage larger than Ref_{up}, the reference transistor is turned on and all calibration transistors are turned off. For the other case that the PFET output voltage is larger than Ref_{up} or the NFET output voltage smaller than Ref_{low}, the reference transistor is turned off and three most significant bit transistors are turned on simultaneously to act as a reference. The calibration is performed using linear step transistors.

5.2.2.2 Hybrid Reduced Binary Search

An alternative hybrid impedance calibration is to use reduced binary (RB) and binary search. The RB approach reuses the same calibration transistors. When the mode signal is high, the calibration logic uses only last three calibration transistors to calibrate the impedance. During the RB search in case that the PFET output voltage is smaller than Ref_{low} or the NFET output voltage larger than Ref_{up}, the reference transistor is turned on. Three most significant bit transistors are turned off. The calibration is performed using three least significant bit transistors. If the PFET output voltage is larger than Ref_{up} or the NFET output voltage smaller than Ref_{low}, the reference transistor is turned off. In this case the three most significant bit calibration transistors are turned on simultaneously. These transistors act as a reference while the remaining three calibration transistors pull back the voltage within the range of match circuit.

5.3 Simulation Results

The GDDR5 I/O calibration circuit using TSMC 65nm LP technology is simulated in Cadence custom IC 6.1. The total number of corners for poly resistor I/O architecture driver adds up to 135. Following are the specifications on which the I/O driver is subjected to:
Transistor corners: FF, FS, SF, SS, TT
Voltage corners: 1.4, 1.5, 1.6
Temperature corners: -25, 25, 75
Poly resistor corners: FF, TT, SS

Table 5.1: Worst active power consumed for different algorithms

Parameter	Binary Search	Hybrid Linear	Hybrid RB
Power in PFET (mW)	10.73	1.62	1.83
Power in NFET (mW)	17.48	2.31	2.57

Table 5.1 shows the significant power gains when the reference transistor is on and three most significant bit transistors are off. The hybrid linear algorithm consumes less power than hybrid RB model since hybrid linear algorithm provides less strengths.

Table 5.2: Statics for PFET section of driver

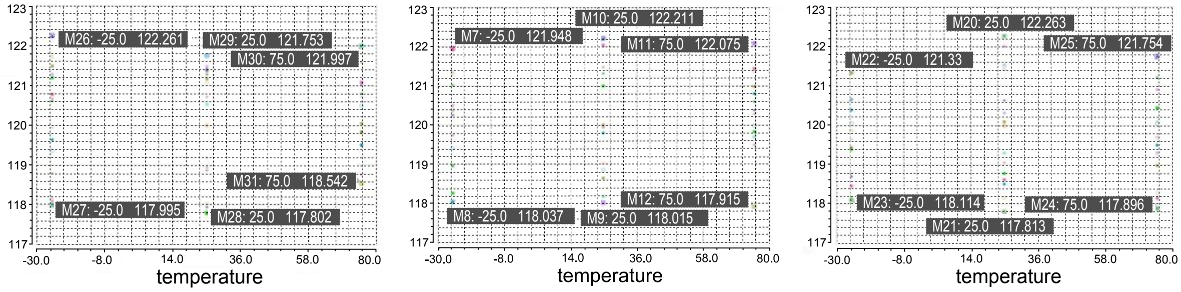
Parameter	Binary Search	Hybrid Linear	Hybrid RB
Corners within +-1% on initialization	11	11	11
Total time (clocks)	492	433	364
Average time (clocks)	3.97	3.49	2.94
Time improvement	–	12%	26%
Corners calibrated by (linear or RB)	–	31	55
Corners calibrated by six bit binary	124	93	69
Corners below Ref_{low} within mode range (linear or RB)	–	10	29

Table 5.2 and 5.3 show the statistics for PFET and NFET, respectively. In the TSMC CMOS 65nm LP technology, the number of corners which need to be calibrated for NFET is more than that for PFET due to the fact that PVT variation in NFET is more severe than that in PFET. Compared to the conventional binary search algorithm, an improvement

Table 5.3: Statics for NFET section of driver

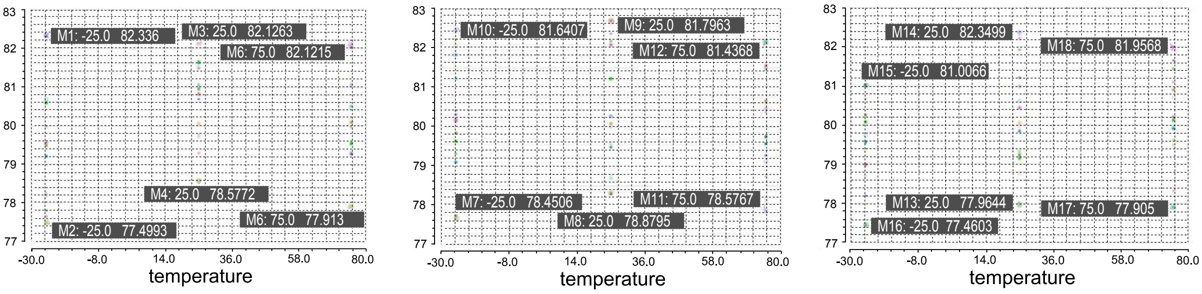
Parameter	Convention	Hybrid Linear	Hybrid RB
Corners within +-1% on initialization	7	7	7
Total time (clocks)	611	524	483
Average time (clocks)	4.77	4.09	3.77
Time improvement	–	14.3%	21%
Corners calibrated by (linear or RB)	–	39	57
Corners calibrated by six bit binary	128	89	71
Corners below Ref_{up} within mode range (linear or RB)	–	27	29

of calibration time in PFET is 12% for hybrid linear and 26% for hybrid RB, respectively. In the case of NFET, the calibration time is reduced by 14.3% and 21%, respectively. For both PFET and NFET impedance calibration, power is saved for more corners when using hybrid RB compared to hybrid linear as it has more strengths than the linear search.



(a) Vdd=1.4V (b) Vdd=1.5V (c) Vdd=1.6V

Figure 5.16: Calibrated PFET impedance for PVT variations



(a) Vdd=1.4V (b) Vdd=1.5V (c) Vdd=1.6V

Figure 5.17: Calibrated NFET impedance for PVT variations

Figure 5.16 and 5.17 illustrate the calibrated PFET impedance and NFET impedance for three operating temperatures under all process and voltage variations. The calibrated impedance for PFET is within +/-2% of 120Ω while the calibrated impedance for NFET is within +3.3% and -2.8% of 80Ω. The larger variation of he NFET impedance compared to the PFET impedance is due to the fact that it is calibrated against the calibrated PMOS section.

As to the required chip area, [48] requires 2^n comparators for a n-bit calibration code. The number of comparators increases exponentially with the number of calibration legs. However, the proposed method needs only 5 comparators independent on the number of calibration legs.

5.4 Summary

The implemented architecture requires less comparators for impedance calibration than the previous work, which saves considerable amount of power and area. Furthermore, the output voltage of the driver is kept analog in the proposed calibration architecture, which avoids the inaccuracy in the digital domain.

In this chapter, two hybrid impedance calibration algorithms are proposed to provide significant advantages with minimized calibration time and reduced power consumption as compared to the conventional binary search algorithm. Furthermore, with the hybrid scheme switching noise is reduced in the circuit to maintain the SI. Compared to the hybrid linear algorithm, the hybrid RB gives better timing performance. Hybrid linear requires extra calibration transistor branches, thus increasing the area. However, power difference between both hybrid algorithms is not significant.

Chapter 6

Summary and Outlook

Contents

6.1 Summary 81

6.2 Directions for Future Work 82

This thesis has presented a PHY link design and optimization for high-speed low-power communication systems. The architectural design, system-level and circuit-level verification of primary components in the PHY link, i.e., CDR, EQs and I/O interface, have been described and considered so far in the previous chapters. Investigation on those components is aimed to improve the entire system performance. This chapter will summarize the contribution of the work (Section 6.1) and some aspects related to potential future research investigations (Section 6.2).

6.1 Summary

The contributions of this thesis can be summarized in the following points:

- **Introduction of new design flow using Matlab/Simulink Cadence co-simulation in mixed-signal design** [17]: A complex mixed signal design requires an overview of all the system building blocks (i.e. memory controller, memory, PCB design in a memory system) in the early design phase. To achieve realistic modeling, all the non-idealities i.e. noise, cross talk etc., should be considered to fulfill the jitter and setup-hold specifications. Matlab/Simulink provides customizable components for noise sources, statistical models and communication blocks and early validation of the system behavior without the need of a higher refinement level. In addition, replacing partially building blocks with analog blocks and synthesizable HDL for digital blocks are desired for a more accurate simulation. To combine the advantages of both methodologies, Matlab/Simulink Cadence co-simulation is applied to mixed-signal design in the thesis.

- **Design and Analysis of low-jitter CDR with low power consumption** [12]: Phase interpolator based CDRs, i.e., modified CDR, adaptive CDR and proposed PI^2 CDR, are implemented for read and write training. A good compromise between power consumption and phase jitter is achieved in the proposed CDR. Furthermore, a mechanism to stop CDR training using 3TS scheme is also proposed and implemented.

- **Introduction of additional adaptive EQ trainings in high-speed communication systems with a novel system architecture and verification of proposed architecture by exploring different adaptive algorithms** [13], [14], [15], [16]: As EQ dramatically improves the SI in the high-speed system environment, two sequential adaptive EQ trainings, i.e., Rx EQ training and Tx EQ trainings using LMS algorithm, pilot signal/peak detection and direct calculation are implemented and compared. Meanwhile, a simplified system architecture of unsymmetrical placement of EQs in EQ trainings is verified in terms of eye diagram, jitter, area and training time. Among three algorithms, LMS shows the best system performance at high data rate.

- **Design and implementation of hybrid digital impedance calibration over PVT variations for high-speed I/O driver** [18]: Three impedance calibration algorithms, i.e., binary search, hybrid linear search and hybrid RB search are investigated over PVT variations in the digital calibration scheme. Compared to the binary search, proposed hybrid algorithms have advantages in reduction of calibration time and power consumption. Furthermore, considerable power and chip area are saved with the proposed architecture.

6.2 Directions for Future Work

In accordance with the current status of the research results presented in this thesis, some future work that can be focused on is summarized in the following points:

- Further adaptive trainings, e.g. power management training via dynamic voltage and frequency scaling, can be integrated into the training scheme. Through the trainings, consumed electrical power is able to be adapted to the currently needed performance of the system to maximize the power efficiency for various bandwidth applications.

- Trainings might be applicable in 3D-IC technology as well as in the optical domain. 3D-IC technology is considered as necessary to meet the growing demands of high bandwidth by increasing the number of I/O pins while optical I/O is motivated to replace the conventional electrical interconnect to achieve scaling data rates in a power-efficient manner.

- Re-training mechanism is to be developed to maintain specified BER since BER may strongly be affected by PVT variations as well as additional random noise in the systems during normal operations. One possible solution to retrain the system is, for example, to monitor the current BER.

Appendix A

Improvement of Signal Integrity by Adaptive Equalizer Trainings

A.1 Jitter and Bathtub

The jitter distribution is thought as the histogram of the crossing-point of a eye diagram, having three regions: at the crossing point of eye diagram deterministic jitter (DJ) dominates the distribution, at time-delays farther from the crossing-point random jitter (RJ) dominates the distribution until far from the crossing point, in the asymptotic limit, the trails follow the Gaussian RJ distribution. The dual-Dirac model [73] in Figure A.1 provides the simplest possible distribution of jitter: the crossing point is separated into two Dirac-delta functions positioned at μ_L and μ_R, the DJ dominated region, followed by an artificially abrupt transition to the RJ dominated tails. RJ is characterized by a Gaussian distribution [42]:

$$RJ(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{t-\mu}{\sigma})^2} \tag{A.1}$$

where μ is center of the gaussian-distribution and σ is standard deviation of the distribution. In the dual-Dirac model, the standard deviation has to be converted with the multiplication-factor α_{BER} [55], which can be expressed by:

$$\alpha_{BER} = 2\sqrt{2} \cdot erfc^{-1}(2 \cdot BER) \tag{A.2}$$

A bathtub plot BER(x), as shown in Figure A.2 is derived from the jitter distribution. The eye opening $t(BER)$ is resulted by the separation of the left and right BER curves at a given BER expressed by:

$$t(BER) = x_R(BER) - x_L(BER) \tag{A.3}$$

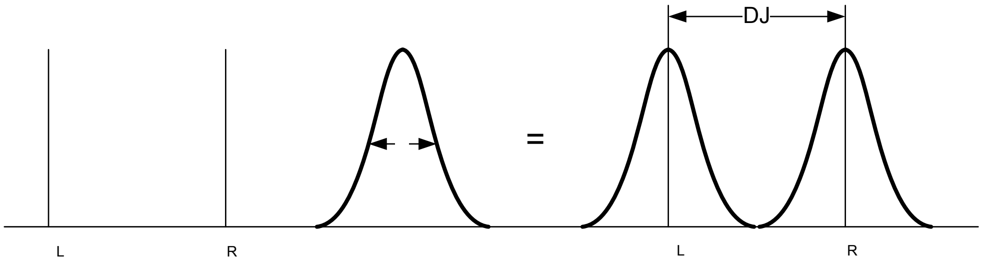


Figure A.1: The Dual-Dirac model for distribution of jitter

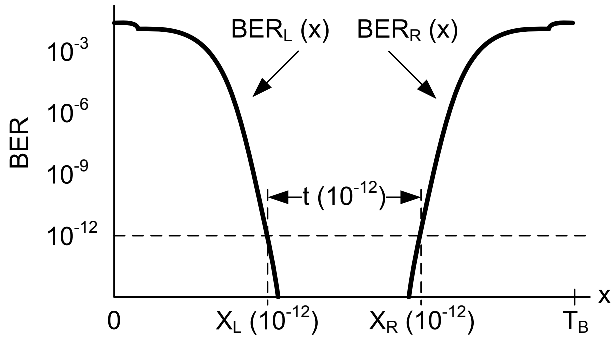


Figure A.2: Bathtub cursor

A.2 MATLAB Code for Signal Evaluation in Adaptive Trainings

```
function cadence_Tx_eye(enable, leq_c1, leq_c2, leq_c3, leq_c4, leq_c5, leq_c6, leq_c7, leq_c8)
%function cadence_Rx_eye(enable, dfe_c1, dfe_c2, dfe_c3, dfe_c4, ffe_c1, ffe_c2, ffe_c3, ffe_c4)
if(enable==1)
%---------------------------------------------------------------------------
%                                   initialization
%---------------------------------------------------------------------------
fd = 5e09;    %data rate
osr=20;     %oversampling rate
fs = fd*osr;
nStop = 1024;
Max = 1;
Min = 0;
BER= 1e-12;
RJ=0.1;       %user defined random jitter
DJ=0.05;      %user defined determinitic jitter


fidRise = fopen(['/victim','_impulse4','.txt']);
[chData_impulse, t] = readFData(fidRise);    %read channel coefficients
chData_impulse = dataResample(chData_impulse, t, fs);

%---------------------------------------------------------------------------
taps=[leq_c1,leq_c2,leq_c3,leq_c4,leq_c5,leg_c6,leg_c7,leg_c8]; %Tx EQ coefficients
%taps=[ffe_c1, ffe_c2, ffe_c3, ffe_c4, dfe_c1, dfe_c2, dfe_c3, dfe_c4]; %Rx EQ coefficients
```

```
%-----------------------------------------------------------------------
%                           eye diagram
%-----------------------------------------------------------------------
order=10;
nTruncBegin=50;
nTruncEnd=50;
prb=prbs(order,nStop);
prb=double(prb);
new_pattern=conv(prb*(Max-Min)+Min,taps,'same'); %signal with EQ
signal3=prbsResample(prb*(Max-Min)+Min,fd,fs);%resampled signal without EQ
signal4=prbsResample(new_pattern,fd,fs);%resampled signal with EQ
signal3=conv(signal3,chData_impulse)/osr;%signal after channel without EQ
signal4=conv(signal4,chData_impulse)/osr;%signa lafter channel with EQ

eyeshift1= -0.5;
eyeshift2= -0.35;
eyeshift1=addEyeshift(eyeshift1,getEyeshift(chData_impulse,osr));
eyeshift2=addEyeshift(eyeshift2,getEyeshift(chData_impulse,osr));
eye1=calc_eye(signal3,osr,eyeshift1*osr);
eye2=calc_eye(signal4,osr,eyeshift2*osr);
eye1=calc_trunc_eye(eye1,nTruncBegin,nTruncEnd);
eye2=calc_trunc_eye(eye2,nTruncBegin,nTruncEnd);
figure
plot_eye(eye1);
title(['cadence␣Eyediagram␣without␣equalizer']);
ylim([0 1]);

figure
plot_eye(eye2);
title(['cadence␣Eyediagram␣␣equalizer']);
ylim([0 1]);

eyeLevel1=calc_eyeLevel(eye1,'mean'); %calculation the mean level of the eye
eyeLevel2=calc_eyeLevel(eye2,'mean');

%-----------------------------------------------------------------------
%                   calculate and plot the bathtub
%-----------------------------------------------------------------------
try
   bathtub1=calc_bathtub(eye1,eyeLevel1,BER,RJ,DJ);

catch
    warning(['Could␣not␣calculated␣bathtub-plot␣for␣' ...
               'TP', num2str(i), '.␣Reason␣could␣be␣'...
               'no␣eye␣opening␣or␣a␣wrong␣shifted␣eye']);
end

figure
        plot_bathtub(bathtub1);
        title(['cadence␣Bathtub␣TP␣without␣EQ']);



try
   bathtub2=calc_bathtub(eye2,eyeLevel2,BER,RJ,DJ);

catch
    warning(['Could␣not␣calculated␣bathtub-plot␣for␣' ...
               'TP', num2str(i), '.␣Reason␣could␣be␣'...
               'no␣eye␣opening␣or␣a␣wrong␣shifted␣eye']);
end

figure
        plot_bathtub(bathtub2);
        title(['cadence␣Bathtub␣TP␣with␣EQ']);
```

```
%-----------------------------------------------------------------------
%                       calculate and plot the jitter
%-----------------------------------------------------------------------
try
   jitter1=calc_jitter(eye1,eyeLevel1,BER,RJ,DJ);

catch
    warning(['Could␣not␣calculated␣jitter-plot␣for␣' ...
               'TP', num2str(i), '.␣Reason␣could␣be␣'...
               'no␣eye␣opening␣or␣a␣wrong␣shifted␣eye']);
end

 figure
        plot_jitter(jitter1,RJ,DJ,BER);
        title(['cadence␣Jitter␣TP␣without␣EQ']);

try
   jitter2=calc_jitter(eye2,eyeLevel2,BER,RJ,DJ);

catch
    warning(['Could␣not␣calculated␣jitter-plot␣for␣' ...
               'TP', num2str(i), '.␣Reason␣could␣be␣'...
               'no␣eye␣opening␣or␣a␣wrong␣shifted␣eye']);
end

 figure
        plot_jitter(jitter2,RJ,DJ,BER);
        title(['cadence␣Jitter␣TP␣with␣EQ']);
```

A.2.1 Resample of prbs Data

```
function [out, time] = prbsResample(in, Fd, Fs)

if (Fs >= Fd)
    out = [false];
    out(:) = [];

    time = 0:1/Fs:(length(in))/Fd - 1/Fs;

    j = 1;
    for i=1:length(time)
        if (((i-1)/Fs) >= (j/Fd))
            j = j+1;
        end
        out = [out, in(j)];
    end
else
    error('Fs␣is␣should␣be␣larger␣than␣Fd');
end
```

A.2.1.1 Eye Shift Obtained from Impulse Response

```
function eyeshift = getEyeshift(imp, osr)

[maxValue, maxPos] = max(imp);
eyeshift = - mod(maxPos, osr)/osr;
```

A.2.2 Eye Shifts Addition

```
function eyeshift = addEyeshift(shift1, shift2)

eyeshift = mod(shift1 + shift2 + 1, 2) - 1;
```

A.2.3 Eye Data Calculation

```
function [Eye] = calc_eye(signal,osr,shift_osr)

osr1=round(osr/8);
osr2=2*osr1;
LL=length(signal);
shift_osr=-shift_osr;
if abs(shift_osr)>osr
    warning('eyeshift␣>␣+/-␣T␣is␣not␣allowed.');
else
    x=signal(round(shift_osr+1-osr1+3*osr/2):LL);
    num=round(shift_osr+1-osr11+3*osr/2);
    L=floor(LL/osr)-3;
    A=zeros(osr,L);
    A(:)=x(1:L*osr);

    for i=1:w2+1
        A=[A;[A(i,2:L) x(L*osr+i)]];
    end
    t=[-osr/2:osr/2+osr2]'+shift_osr-osr1;
    t=t/osr;
end

Eye = [t,A];
```

A.2.4 Eye Data Truncation at Start and Stop Points

```
function Eye_new = calc_trunc_eye(Eye, Start, Stop)
if (Start >= 0)&&(Stop >=0)
    if (Start + Stop) < length(Eye-1-Start-Stop)
        Eye_new = [Eye(:,1), Eye(:,(Start + 2):(end - Stop))];
    else
         error('eye␣data␣that␣have␣to␣be␣truncated␣are␣too␣large');
    end
else
    error('Start␣and␣Stop␣points␣must␣be␣positive␣integer␣values');
end
```

A.2.5 Bathtub Data Calculation

```
function bathtub = calc_bathtub(eye,eyeLevel,BER,RJ,DJ)
level = eyeLevel;
minNPoints = 2e4;           % minimum number of bathtub-points
minRJ = 1e-3;               % minimum value for random jitter

[muL, muR, crossVect, time] = getDJ(eye,level);
x = time;

% add the user-defined deterministic jitter to the borders of the
muL = muL+DJ/2;
muR = muR-DJ/2;

% create minimum number of points for x if there are too few
if length(x) < minNPoints
    x = x(1):(x(end) - x(1))/minNPoints:x(end)+1e-15;
```

```
end

if RJ < minRJ
    RJ = 0;
end

if RJ = 0
         for i = 1:length(x)
        if x(i) <= muL || x(i) >= muR
            data(i) = 1;
        elseif (x(i) > muL && x(i-1) <= muL) ...
                || (x(i) < muR && x(i+1) >= muR)
            data(i) = 10^(-20);
        else
            data(i) = NaN;
        end
    end

else
    alpha = 2*sqrt(2)*erfcinv(2*BER);
    distrRJLeft = normpdf(x,(muL),RJ/alpha);
    distrRJRight = normpdf(x,(muR),RJ/alpha);
    distrRJ = distrRJRight + distrRJLeft;
    data = erf(distrRJ);

    % correct the bathtub curve beyond muL .. muR
    for i = 1:length(x)
        if x(i) < muL || x(i) > muR
            data(i) = 1;
        elseif data(i) < 10^(-20)
            data(i) = NaN;
        end
    end
end

bathtub.xAxes = x;
bathtub.data = data;
bathtub.BER = BER;
```

A.2.6 Plot Bathtub

```
function plot_bathtub(bathtub)
if length(bathtub) ~= 0
    x = bathtub.xAxes;
    data = bathtub.data;
    BER = bathtub.BER;

    lim = BER*(1-x)./(1-x);

    semilogy(x,data,x,lim); grid on;
    axis([x(1) x(end) 10^(log10(BER)-3) 1e0]);
    title('Bathtub-Plot');
    xlabel('t/T');
    ylabel('BER');
end
```

A.2.7 Jitter Calculation

```
function jitter = calc_jitter(eye,eyeLevel,BER,RJ,DJ)

level = eyeLevel;
```

```
[muL, muR, crossVect, time] = getDJ(eye, level);

alpha = 2*sqrt(2)*erfcinv(2*BER);

% random jitter distribution
distrRJLeft = normpdf(time,(muL+DJ/2),RJ/alpha);
distrRJRight = normpdf(time,(muR-DJ/2),RJ/alpha);
distrRJ = distrRJRight + distrRJLeft;

jitter.distribution.RJ = distrRJ';
jitter.distribution.DJ = crossVect;
jitter.distribution.mu = [muL, muR];
jitter.distribution.xAxes = time;
```

A.2.8 Plot Jitter

```
function plot_jitter(jitter,RJ,DJ,BER)

if isfield(jitter, 'distribution')
    muL = jitter.distribution.mu(1);
    muR = jitter.distribution.mu(2);
    xAxes = jitter.distribution.xAxes;
    DJDistr = jitter.distribution.DJ;
    RJDistr = jitter.distribution.RJ;

    bar(xAxes,DJDistr/max(DJDistr));
    hold on;

    if RJ == 0 && DJ == 0
        legend('DJ');
    elseif RJ == 0 && DJ ~= 0
        line([muL,muL],[0,1],'Color','b','Linestyle','-.')
        line([muL+DJ/2,muL+DJ/2],[0,1],'Color','g','Linestyle','-.')
        line([muR,muR],[0,1],'Color','b','Linestyle','-.')
        line([muR-DJ/2,muR-DJ/2],[0,1],'Color','g','Linestyle','-.')
        legend('DJ','mu','mu␣+␣user-defined␣DJ');
    elseif RJ ~= 0 && DJ == 0
        plot(xAxes,RJDistr/max(RJDistr),'g');
        line([muL,muL],[0,1],'Color','b','Linestyle','-.')
        line([muR,muR],[0,1],'Color','b','Linestyle','-.')
        legend('DJ','user-defined␣RJ','mu');
    elseif RJ ~= 0 && DJ ~= 0
        plot(xAxes,RJDistr/max(RJDistr),'g');
        line([muL,muL],[0,1],'Color','b','Linestyle','-.')
        line([muL+DJ/2,muL+DJ/2],[0,1],'Color','g','Linestyle','-.')
        line([muR,muR],[0,1],'Color','b','Linestyle','-.')
        line([muR-DJ/2,muR-DJ/2],[0,1],'Color','g','Linestyle','-.')
        legend('DJ','user-defined␣RJ','mu','mu␣+␣user-defined␣DJ');
    end

    title('Jitter');
    xlabel('t/T');
    grid on;
    hold off;
end
```

A.3 Verilog Code for LMS Engine and Direct Tx EQ

A.3.1 Multiplier

```
module multi_10bit( clk,rst,in_a,in_b,y_out );

 input clk,rst;
 input  [9:0]   in_a,in_b;
 output [9:0]   y_out;

 reg [9:0]   y_out;
 reg [8:0]   x1,x2,x3,x4;
 reg         x5;
 reg [15:0]  x6;
 reg [9:0]   x7;

 reg  tmp1, tmp2, tmp3, tmp4,tmp5;

 always @ (posedge clk)
 begin
    if (rst==1)
                begin
                        tmp1 <= 1'b0;
                        tmp2 <= 1'b0;
                        tmp3 <= 1'b0;
                end
    else
        begin
                tmp1 <= in_a[8];
                tmp2 <= tmp1;
                tmp3 <= tmp2;
        end
        end

 always @ (posedge clk or posedge rst )
 begin
        if (rst)
        begin
                x1     <= 9'b0;
                x2     <= 9'b0;
                x3     <= 9'b0;
                x4     <= 9'b0;
                x5     <= 1'b0;
                x6     <= 16'b0;
                x7     <= 10'b0;
                y_out <= 10'b0;
        end
    else if(tmp3 ==1)
        begin
                y_out <= in_b;
                x7     <= {x5,1'b0,x6[15:8]};
                x6     <= x1[7:0]*x2[7:0];
                        x5     <= x1[8] ^ x2[8];
        end
    else if(tmp3 ==0)
        begin
          x1     <= {in_a[9],in_a[7:0]};
          x2     <= {in_b[9],in_b[7:0]};
          x5     <= x1[8] ^ x2[8];
          x6     <= x1[7:0]*x2[7:0];
          x7     <= {x5,1'b0,x6[15:8]};
          if (x7==10'b1000000000)
                y_out<= 10'b0;
          else
            y_out<= x7;
         end
 end
 endmodule
```

A.3.2 Adder

```
module adder(out_abs,sign,out,in1,in2,keep,clk,rst);
   input[9:0] in1,in2;
   input clk,rst;
   input keep;
   output[9:0] out_abs,out;
   output sign;
   reg[9:0] out_abs,out;
   reg sign;
   reg[9:0] temp,temp1,temp2,temp3;

always @(posedge clk or posedge rst)
   begin
     if (rst==1) begin
                temp <= 10'b0;
                temp1 <= 10'b0;
                temp2 <= 10'b0;
                temp3 <= 10'b0;
                out_abs  <= 10'b0;
                out <= 10'b0;
            sign <= 1'b0;
     end
     else if (keep==0) begin
        temp1<= (in1[9]==0)? in1:{in1[9],~in1[8:0]+1'b1};
        temp2<= (in2[9]==0)? in2:{in2[9],~in2[8:0]+1'b1};
        temp3 <= temp1 + temp2;
        temp <= (temp3[9]==0)? temp3:{temp3[9],~temp3[8:0]+1'b1};
        if (temp[8:0]> 9'b110000000)
        begin
            out  <= {temp[9],9'b110000000};
                        sign <= temp[9];
            out_abs <= {1'b0, 9'b110000000};
         end else begin
            out  <= temp;
            sign <= temp[9];
                out_abs<= {1'b0, temp[8:0]};
         end
     end else if (keep==1) begin
          temp1<=10'b0;
          temp2<=(in2[9]==0)? in2:{in2[9],~in2[8:0]+1'b1};
          temp3<=temp1+temp2;
          temp <= (temp3[9]==0)? temp3:{temp3[9],~temp3[8:0]+1'b1};
      if (temp[8:0]> 9'b110000000) begin
         out  <= {temp[9],9'b110000000};
         sign <= temp[9];
             out_abs<= {1'b0, 9'b110000000};
      end else begin
         out  <= temp;
         sign <= temp[9];
         out_abs<= {1'b0, temp[8:0]};
      end
     end
        end
endmodule
```

A.3.3 Divider

```
module pipelined_divi2 (res,abs_res,sign,dividend,divisor,clk);

input [9:0] dividend;
input [9:0] divisor;
input clk;
output reg [9:0] res = 10'b0; //result of quotient
output reg [9:0] abs_res = 10'b0;
output reg sign;
reg [8:0] rem = 9'b0;// remainder
reg[17:0] res18,dd18;
reg[17:0] res17,dd17;
reg[17:0] res16,dd16;
reg[17:0] res15,dd15;
reg[17:0] res14,dd14;
reg[17:0] res13,dd13;
reg[17:0] res12,dd12;
reg[17:0] res11,dd11;
reg[17:0] res10,dd10;
reg[17:0] res9,dd9;
reg[17:0] res8,dd8;
reg[17:0] res7,dd7;
reg[17:0] res6,dd6;
reg[17:0] res5,dd5;
reg[17:0] res4,dd4;
reg[17:0] res3,dd3;
reg[17:0] res2,dd2;
reg[17:0] res1,dd1,dd0;
reg[8:0] rem18,dr18,dt18;
reg[8:0] rem17,dr17,dt17;
reg[8:0] rem16,dr16,dt16;
reg[8:0] rem15,dr15,dt15;
reg[8:0] rem14,dr14,dt14;
reg[8:0] rem13,dr13,dt13;
reg[8:0] rem12,dr12,dt12;
reg[8:0] rem11,dr11,dt11;
reg[8:0] rem10,dr10,dt10;
reg[8:0] rem9,dr9,dt9;
reg[8:0] rem8,dr8,dt8;
reg[8:0] rem7,dr7,dt7;
reg[8:0] rem6,dr6,dt6;
reg[8:0] rem5,dr5,dt5;
reg[8:0] rem4,dr4,dt4;
reg[8:0] rem3,dr3,dt3;
reg[8:0] rem2,dr2,dt2;
reg[8:0] rem1,dr1,dt1,dr0,dt0;
reg sign18,sign17,sign16,sign15,sign14,
    sign13,sign12,sign11,sign10,sign9,
    sign8,sign7,sign6,sign5,sign4,
    sign3,sign2,sign1,sign0;

always @ (posedge clk)      //18
begin
        rem18 <= 4'b0;
        res18 <= 8'b0;
        dd18  <= {dividend[8:0],9'b0};
        sign18 <= dividend[9]^divisor[9];
        dt18  <= divisor;
        if (divisor[8]==1)
            dr18  <= divisor[8:0]>>1;
        else
            dr18  <= divisor[8:0];
end
always @ (posedge clk)      //17
begin
        rem17 <= ((rem18-((rem18<dr18)? 9'b0:dr18))<<1)+dd18[17];
        res17 <= (res18<<1)+((rem17<dr17)?1'b0:1'b1);
        dd17  <= dd18;
        dr17  <= dr18;
        dt17   <= dt18;
        sign17   <= sign18;
end
always @ (posedge clk) //16
begin
         rem16 <= ((rem17-((rem17<dr17)? 9'b0:dr17))<<1)+dd17[16];
         res16 <= (res17<<1)+((rem16<dr16)?1'b0:1'b1);
         dd16   <= dd17;
         dr16   <= dr17;
         dt16   <= dt17;
         sign16   <= sign17;
end
always @ (posedge clk)//15
begin
         rem15 <= ((rem16-((rem16<dr16)? 9'b0:dr16))<<1)+dd16[15];
         res15 <= (res16<<1)+((rem15<dr15)?1'b0:1'b1);
         dd15   <= dd16;
         dr15   <= dr16;
         dt15   <= dt16;
         sign15   <= sign16;
end
always @ (posedge clk) //14
begin
         rem14 <= ((rem15-((rem15<dr15)? 9'b0:dr15))<<1)+dd15[14];
         res14 <= (res15<<1)+((rem14<dr14)?1'b0:1'b1);
         dd14   <= dd15;
         dr14   <= dr15;
         dt14   <= dt15;
         sign14   <= sign15;
end
always @ (posedge clk) //13
begin
         rem13 <= ((rem14-((rem14<dr14)? 9'b0:dr14))<<1)+dd14[13];
         res13 <= (res14<<1)+((rem13<dr13)?1'b0:1'b1);
         dd13   <= dd14;
         dr13   <= dr14;
         dt13   <= dt14;
         sign13   <= sign14;
end
always @ (posedge clk)  //12
begin
         rem12 <= ((rem13-((rem13<dr13)? 9'b0:dr13))<<1)+dd13[12];
         res12 <= (res13<<1)+((rem12<dr12)?1'b0:1'b1);
         dd12   <= dd13;
         dr12   <= dr13;
         dt12   <= dt13;
         sign12   <= sign13;
end
always @ (posedge clk)  //11
begin
         rem11 <= ((rem12-((rem12<dr12)? 9'b0:dr12))<<1)+dd12[11];
         res11 <= (res12<<1)+((rem11<dr11)?1'b0:1'b1);
         dd11   <= dd12;
         dr11   <= dr12;
         dt11   <= dt12;
         sign11   <= sign12;
end
always @ (posedge clk)  //10
begin
         rem10 <= ((rem11-((rem11<dr11)? 9'b0:dr11))<<1)+dd11[10];
         res10 <= (res11<<1)+((rem10<dr10)?1'b0:1'b1);
         dd10   <= dd11;
         dr10   <= dr11;
         dt10   <= dt11;
        sign10  <= sign11;
end
always @ (posedge clk) //9
begin
        rem9 <= ((rem10-((rem10<dr10)? 9'b0:dr10))<<1)+dd10[9];
        res9 <= (res10<<1)+((rem9<dr9)?1'b0:1'b1);
        dd9  <= dd10;
        dr9  <= dr10;
        dt9  <= dt10;
        sign9  <= sign10;
end
always @ (posedge clk)//8
begin
        rem8 <= ((rem9-((rem9<dr9)? 9'b0:dr9))<<1)+dd9[8];
        res8 <= (res9<<1)+((rem8<dr8)?1'b0:1'b1);
        dd8  <= dd9;
        dr8  <= dr9;
        dt8  <= dt9;
        sign8  <= sign9;
end
always @ (posedge clk) //7
begin
        rem7 <= ((rem8-((rem8<dr8)? 9'b0:dr8))<<1)+dd8[7];
        res7 <= (res8<<1)+((rem7<dr7)?1'b0:1'b1);
        dd7  <= dd8;
        dr7  <= dr8;
        dt7  <= dt8;
        sign7  <= sign8;
end
always @ (posedge clk)    //6
begin
        rem6 <= ((rem7-((rem7<dr7)? 9'b0:dr7))<<1)+dd7[6];
        res6 <= (res7<<1)+((rem6<dr6)?1'b0:1'b1);
        dd6  <= dd7;
        dr6  <= dr7;
        dt6  <= dt7;
        sign6  <= sign7;
end
always @ (posedge clk)    //5
begin
        rem5 <= ((rem6-((rem6<dr6)? 9'b0:dr6))<<1)+dd6[5];
        res5 <= (res6<<1)+((rem5<dr5)?1'b0:1'b1);
        dd5  <= dd6;
        dr5  <= dr6;
        dt5  <= dt6;
        sign5  <= sign6;
end
always @ (posedge clk)   //4
begin
        rem4 <= ((rem5-((rem5<dr5)? 9'b0:dr5))<<1)+dd5[4];
        res4 <= (res5<<1)+((rem4<dr4)?1'b0:1'b1);
        dd4  <= dd5;
        dr4  <= dr5;
        dt4  <= dt5;
        sign4  <= sign5;
end
always @ (posedge clk)    //3
begin
        rem3 <= ((rem4-((rem4<dr4)? 9'b0:dr4))<<1)+dd4[3];
        res3 <= (res4<<1)+((rem3<dr3)?1'b0:1'b1);
        dd3  <= dd4;
        dr3  <= dr4;
        dt3  <= dt4;
        sign3  <= sign4;
```

```
end
always @ (posedge clk)     //2
begin
         rem2 <= ((rem3-((rem3<dr3)? 9'b0:dr3))<<1)+dd3[2];
         res2 <= (res3<<1)+((rem2<dr2)?1'b0:1'b1);
         dd2  <= dd3;
         dr2  <= dr3;
         dt2  <= dt3;
         sign2  <= sign3;
end
always @ (posedge clk)      //1
begin
         rem1 <= ((rem2-((rem2<dr2)? 9'b0:dr2))<<1)+dd2[1];
         res1 <= (res2<<1)+((rem1<dr1)?1'b0:1'b1);
         dd1  <= dd2;
         dr1  <= dr2;
         dt1  <= dt2;
         sign1  <= sign2;
end
always @ (posedge clk)
begin
         rem <= rem1-((rem1<dr1)? 9'b0:dr1);
         dd0 <= dd1;
         dr0 <= dr1;
         dt0  <= dt1;
         sign0<= sign1;
         if ((dt0[8]==1)&&(!dd0[17]==1))
         begin
              res <= {sign0,res1[8:0]>>1};
              sign<= sign0;
              abs_res<= {1'b0,res1[8:0]>>1};
         end
         else if ((dt0[8]==1)&&(dd0[17]==1))
         begin
              res <= {sign0,res1[9:1]};
              sign<= sign0;
              abs_res<= {1'b0,res1[9:1]};
         end
         else
         begin
              res <= {sign0,res1[8:0]};
              sign<= sign0;
              abs_res<= {1'b0,res1[8:0]};
         end
end
endmodule
```

References

[1] N. Krishnapura, M. Barazande-Pour, Q. Chaudhry, J. Khoury, K. Lakshmikumar, and A. Aggarwal. “A 5Gb/s NRZ transceiver with adaptive equalization for backplane transmission”. In: *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International* (Feb. 2005), Vol. 1, 60–585, ISSN: 0193-6530.

[2] J.F. Bulzacchelli, C. Menolfi, T.J. Beukema, D.W. Storaska, J. Hertle, D.R. Hanson, P.H. Hsieh, S.V. Rylov, D. Furrer, D. Gardellini, A. Prati, T. Morf, V. Sharma, R. Kelkar, H.A. Ainspan, W.R. Kelly, L.R. Chieco, G.A. Ritter, J.A. Sorice, J.D. Garlett, R. Callan, M. Brandli, P. Buchmann, M. Kossel, T. Toifl, and D.J. Friedman. “A 28-Gb/s 4-Tap FFE/15-Tap DFE Serial Link Transceiver in 32-nm SOI CMOS Technology”. In: *Solid-State Circuits, IEEE Journal of* (Dec. 2012), Vol. 47, No. 12, 3232–3248, ISSN: 0018-9200.

[3] A.L.S. Loke, B.A. Doyle, S.K. Maheshwari, D.M. Fischette, C.L. Wang, T.T. Wee, and E.S. Fang. “An 8.0-Gb/s HyperTransport Transceiver for 32-nm SOI-CMOS Server Processors”. In: *Solid-State Circuits, IEEE Journal of* (Nov. 2012), Vol. 47, No. 11, 2627–2642, ISSN: 0018-9200.

[4] D. Ziakas, A. Baum, R.A. Maddox, and R.J. Safranek. “Intel® QuickPath Interconnect Architectural Features Supporting Scalable System Architectures”. In: *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on* (Aug. 2010), pp. 1–6.

[5] A. Amirkhany, J. Wei, N. Mishra, J. Shen, W. Beyene, T. Chin, C. Huang, V. Gadde, K. Kaviani, P. Le, M. M, C. Madden, S. Mukherjee, L. Raghavan, K. Saito, D. Secker, F. Shuaeb, S. Srinivas, T. Wu, C. Tran, A. Vaidyanathan, K. Vyas, M. Jain, K. Chang, and C. Yuan. “A 12.8-Gb/s/link tri-modal single-ended memory interface for graphics applications”. In: *VLSI Circuits (VLSIC), 2011 Symposium on* (June 2011), pp. 232–233. ISSN: 2158-5601.

[6] K. Kaviani, T. Wu, J. Wei, A. Amirkhany, J. Shen, T. J. Chin, C. Thakkar, W.T. Beyene, N. Chan, C. Chen, B.R. Chuang, D. Dressler, V.P. Gadde, M. Hekmat, E. Ho, C. Huang, P. Le, Mahabaleshwara, C. Madden, N.K. Mishra, L. Raghavan, K. Saito, R. Schmitt, D. Secker, Xudong Shi, S. Fazeel, G.S. Srinivas, S. Zhang, C. Tran, A. Vaidyanath, K. Vyas, M. Jain, K.Y.K. Chang, and X.C. Yuan. “A Tri-Modal

20-Gbps/Link Differential/DDR3/GDDR5 Memory Interface". In: *Solid-State Circuits, IEEE Journal of* (Apr. 2012), Vol. 47, No. 4, 926–937, ISSN: 0018-9200.

[7] A. Amirkhany, W. Beyene, C. Madden, A. Abbasfar, D. Secker, O. Dan, M. Hekmat, R. Schmitt, and C. Yuan. "On overcoming the limitations of single-ended signaling for graphics memory interfaces". In: *Solid State Circuits Conference (A-SSCC), 2011 IEEE Asian* (Nov. 2011), pp. 25–28.

[8] *GRAPHICS DOUBLE DATA RATE (GDDR5) SGRAM STANDARD*. http://www.jedec.org/standards-documents/docs/jesd212B.

[9] C.C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari. "Bridging the processor-memory performance gap with 3D IC technology". In: *Design Test of Computers, IEEE* (Nov. 2005), Vol. 22, No. 6, 556–564, ISSN: 0740-7475.

[10] J.H. Lau. "TSV manufacturing yield and hidden costs for 3D IC integration". In: *Electronic Components and Technology Conference (ECTC), 2010 Proceedings 60th* (June 2010), pp. 1031–1042. ISSN: 0569-5503.

[11] J.S. Kim, C.S. Oh, H. Lee, D. Lee, H.R. Hwang, S. Hwang, B. Na, J. Moon, J.G. Kim, H. Park, J.W. Ryu, K. Park, S.K. Kang, S.Y. Kim, H. Kim, J.M. Bang, H. Cho, M. Jang, C. Han, J.B. Lee, K. Kyung, J.S. Choi, and Y.H. Jun. "A 1.2V 12.8GB/s 2Gb mobile Wide-I/O DRAM with 4x128 I/Os using TSV-based stacking". In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International* (Feb. 2011), pp. 496–498. ISSN: 0193-6530.

[12] Y. Fang, J. Bargon, A. Jaiswal, and K. Hofmann. "A Low-Jitter Clock and Data Recovery Using Proportional-Integral Controller for GDDR5 Interface Trainings". In: *Integrated Circuit Design and Technology, 2014. ICICDT '14. International Conference on* (May 2014).

[13] Y. Fang, L. Chen, A. Jaiswal, K. Hofmann, and P. Gregorius. "Adaptive Equalizer Training for High-Speed Low-Power Communication Systems". In: *Digital System Design (DSD), 2013 Euromicro Conference on* (Sept. 2013), pp. 745–751.

[14] Y. Fang, A. Jaiswal, and K. Hofmann. "Low-power signal integrity trainings for multi-clock source-synchronous memory systems". In: *SOC Conference (SOCC), 2013 IEEE 26th International* (Sept. 2013), pp. 319–324. ISSN: 2164-1676.

[15] Y. Fang, L. Chen, A. Jaiswal, and K. Hofmann. "Transmitter equalizer training based on pilot signal and peak detection". In: *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on* (Dec. 2013), pp. 377–380.

[16] Y. Fang, A. Jaiswal, and K. Hofmann. "Method and Means for Improving the Data Transfer Integrity In a Time Synchronised System". In: *Patent, filed on 10th Sep.* (2012).

[17] A. Jaiswal, Y. Fang, K. Hofmann, and P. Gregorius. "An efficient methodology for mixed-signal high-speed memory design using Matlab/Simulink-HDL co-simulation". In: *Solid-State and Integrated Circuit Technology (ICSICT), 2012 IEEE 11th International Conference on* (Oct. 2012), pp. 1–3.

[18] Y. Fang, U. Muhammad, A. Jaiswal, and K. Hofmann. "Low-power design of hybrid digital impedance calibration for process, voltage, temperature compensations". In: *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on* (Dec. 2013), pp. 37–40.

[19] *Intel® Z87 Chipset Platform Diagram*. `http://www.intel.com/content/www/us/en/chipsets/performance-chipsets/z87-chipset-diagram.html`.

[20] `http://www.pcisig.com`.

[21] `http://www.hypertransport.org`.

[22] *HyperTranspor™ I/O Link Specification*. `http://www.hypertransport.org/default.cfm?page=HyperTransportSpecifications31`.

[23] `http://www.jedec.org`.

[24] H. Yamaguchi, H. Tamura, Y. Doi, Y. Tomita, T. Hamada, M. Kibune, S. Ohmoto, K. Tateishi, O. Tyshchenko, A. Sheikholeslami, T. Higuchi, J. Ogawa, T. Saito, H. Ishida, and K. Gotoh. "A 5Gb/s transceiver with an ADC-based feedforward CDR and CMA adaptive equalizer in 65nm CMOS". In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International* (Feb. 2010), pp. 168–169. ISSN: 0193-6530.

[25] K. Fukuda, H. Yamashita, F. Yuki, M. Yagyu, R. Nemoto, T. Takemoto, T. Saito, N. Chujo, K. Yamamoto, H. Kanai, and A. Hayashi. "An 8Gb/s Transceiver with 3x-Oversampling 2-Threshold Eye-Tracking CDR Circuit for -36.8dB-loss Backplane". In: *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International* (Feb. 2008), pp. 98–598.

[26] A. Hayashi, M. Kuwata, K. Suzuki, T. Muto, M. Tsuge, K. Nagashima, D. Hamano, T. Usugi, K. Nakajima, M. Ogihara, N. Mikami, and K. Watanabe. "A 21-channel 8Gb/s transceiver macro with 3.6ns latency in 90nm CMOS for 80cm backplane communication". In: *VLSI Circuits, 2008 IEEE Symposium on* (June 2008), pp. 202–203.

[27] W.C. Chen, C.C. Tsai, C.H. Chang, Y.C. Peng, F.L. Hsueh, T.H. Yu, J.Y. Chien, W.H. Huang, C.C. Lu, M.S. Lin, C.M. Fu, S.C. Yang, C.W. Wong, W.T. Chen, C.H. Wen, L.Y. Wang, and C. Pu. "A 2.5-8Gb/s transceiver with 5-tap DFE and Second order CDR against 28-inch channel and 5000ppm SSC in 40nm CMOS technology". In: *Custom Integrated Circuits Conference (CICC), 2010 IEEE* (Sept. 2010), pp. 1–4. ISSN: 0886-5930.

[28] M.S. Chen, Y.N. Shih, C.L. Lin, H.W. Hung, and J. Lee. "A Fully-Integrated 40-Gb/s Transceiver in 65-nm CMOS Technology". In: *Solid-State Circuits, IEEE Journal of* (Mar. 2012), Vol. 47, No. 3, 627–640, ISSN: 0018-9200.

[29] M.T. Hsieh and G. Sobelman. "Architectures for multi-gigabit wire-linked clock and data recovery". In: *Circuits and Systems Magazine, IEEE* (Apr. 2008), Vol. 8, No. 4, 45–57, ISSN: 1531-636X.

[30] M.H. Perrott, Y.T. Huang, R.T. Baird, B.W. Garlepp, D. Pastorello, E.T. King, Q.C. Yu, D.B. Kasha, P. Steiner, L.G. Zhang, J. Hein, and B. Del Signore. "A 2.5-Gb/s Multi-Rate 0.25-μm CMOS Clock and Data Recovery Circuit Utilizing a Hybrid Analog/Digital Loop Filter and All-Digital Referenceless Frequency Acquisition". In: *Solid-State Circuits, IEEE Journal of* (Dec. 2006), Vol. 41, No. 12, 2930–2944, ISSN: 0018-9200.

[31] H.H. Chang, R.J. Yang, and S.I. Liu. "Low jitter and multirate clock and data recovery circuit using a MSADLL for chip-to-chip interconnection". In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* (Dec. 2004), Vol. 51, No. 12, 2356–2364, ISSN: 1549-8328.

[32] R. Kreienkamp, U. Langmann, C. Zimmermann, T. Aoyama, and H. Siedhoff. "A 10-Gb/s CMOS clock and data recovery circuit with an analog phase interpolator". In: *Solid-State Circuits, IEEE Journal of* (Mar. 2005), Vol. 40, No. 3, 736–743, ISSN: 0018-9200.

[33] K. Maruko, T. Sugioka, H. Hayashi, Zhou Z.W., Y. Tsukuda, Y. Yagishita, H. Konishi, T. Ogata, H. Owa, T. Niki, K. Konda, M. Sato, H. Shiroshita, T. Ogura, T. Aoki, H. Kihara, and S. Tanaka. "A 1.296-to-5.184Gb/s Transceiver with 2.4mW/(Gb/s) Burst-mode CDR using Dual-Edge Injection-Locked Oscillator". In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International* (Feb. 2010), pp. 364–365. ISSN: 0193-6530.

[34] F. Spagna, L.D. Chen, M. Deshpande, Y.P. Fan, D. Gambetta, S. Gowder, S. Iyer, R. Kumar, P. Kwok, R. Krishnamurthy, C.C. Lin, R. Mohanavelu, R. Nicholson, J. Ou, M. Pasquarella, K. Prasad, H. Rustam, L. Tong, A. Tran, J. Wu, and X.G. Zhang. "A 78mW 11.8Gb/s serial link transceiver with adaptive RX equalization and baud-rate CDR in 32nm CMOS". In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International* (Feb. 2010), pp. 366–367. ISSN: 0193-6530.

[35] Z. Gao, Yu. H., P. Chiang, Y. Yang, and F. Zhang. "A 10Gb/s wire-line transceiver with half rate period calibration CDR". In: *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on* (May 2009), pp. 1827–1830.

[36] K. Fukuda, H. Yamashita, G. Ono, R. Nemoto, E. Suzuki, N. Masuda, T. Takemoto, F. Yuki, and T. Saito. "A 12.3-mW 12.5-Gb/s Complete Transceiver in 65-nm CMOS Process". In: *Solid-State Circuits, IEEE Journal of* (Dec. 2010), Vol. 45, No. 12, 2838–2849, ISSN: 0018-9200.

[37] S. Sidiropoulos and M.A. Horowitz. "A semidigital dual delay-locked loop". In: *Solid-State Circuits, IEEE Journal of* (Nov. 1997), Vol. 32, No. 11, 1683–1692, ISSN: 0018-9200.

[38] J. Park, J.F. Liu, L.R. Carley, and C.P. Yue. "A 1-V, 1.4-2.5 GHz Charge-Pump-Less PLL for a Phase Interpolator Based CDR". In: *Custom Integrated Circuits Conference, 2007. CICC '07. IEEE* (Sept. 2007), pp. 281–284.

[39] S.J. Hu, C. Jia, K. Huang, C. Zhang, X.Q. Zheng, and Z.H. Wang. "A 10Gbps CDR based on phase interpolator for source synchronous receiver in 65nm CMOS". In: *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on* (May 2012), pp. 309–312. ISSN: 0271-4302.

[40] K. Desai, R. Nagulapalli, V. Krishna, R. Palwai, P. Kumar Venkatesan, and V. Khawshe. "High Speed Clock and Data Recovery Circuit with Novel Jitter Reduction Technique". In: *VLSI Design, 2010. VLSID '10. 23rd International Conference on* (Jan. 2010), pp. 300–305. ISSN: 1063-9667.

[41] M.Y. He and J. Poulton. "A CMOS mixed-signal clock and data recovery circuit for OIF CEI-6G+ backplane transceiver". In: *Solid-State Circuits, IEEE Journal of* (Mar. 2006), Vol. 41, No. 3, 597–606, ISSN: 0018-9200.

[42] P. Gregorius, ed. *Verfahren zur Symbolsynchronisation in der schnellen seriellen Datenuebertragung*. Shaker, 2007.

[43] S. Kumaki, A.H. Johari, T. Matsubara, I. Hayashi, and H. Ishikuro. "A 0.5V 6-bit scalable phase interpolator". In: *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on* (Dec. 2010), pp. 1019–1022.

[44] Y. Tomita, M. Kibune, J. Ogawa, W.W. Walker, H. Tamura, and T. Kuroda. "A 10-Gb/s receiver with series equalizer and on-chip ISI monitor in 0.11-μm CMOS". In: *Solid-State Circuits, IEEE Journal of* (Apr. 2005), Vol. 40, No. 4, 986–993, ISSN: 0018-9200.

[45] X.F. Lin, J. Liu, H. Lee, and H. Liu. "A 2.5- to 3.5-Gb/s Adaptive FIR Equalizer With Continuous-Time Wide-Bandwidth Delay Line in 0.25-μm CMOS". In: *Solid-State Circuits, IEEE Journal of* (Aug. 2006), Vol. 41, No. 8, 1908–1918, ISSN: 0018-9200.

[46] Y. Hidaka, W.X. Gai, H. Takeshi, J.H. Jiang, Y. Koyanagi, and H. Osone. "A 4-Channel 1.25-10.3 Gb/s Backplane Transceiver Macro With 35 dB Equalizer and Sign-Based Zero-Forcing Adaptive Control". In: *Solid-State Circuits, IEEE Journal of* (Dec. 2009), Vol. 44, No. 12, 3547–3559, ISSN: 0018-9200.

[47] J. Zerbe, F. Chen, A. Ho, R. Farjad-Rad, J. Poulton, K. Donnelly, and B. Leibowitz. "High-speed signaling systems with adaptable pre-emphasis and equalization". In: (Aug. 2006). US Patent App. 11/336,045.

[48] J.L. Zerbe, C.W. Werner, V. Stojanovic, F. Chen, J. Wei, G. Tsang, D. Kim, W.F. Stonecypher, A. Ho, T.P. Thrush, R.T. Kollipara, M.A. Horowitz, and K.S. Donnelly. "Equalization and clock recovery for a 2.5-10-Gb/s 2-PAM/4-PAM backplane transceiver cell". In: *Solid-State Circuits, IEEE Journal of* (Dec. 2003), Vol. 38, No. 12, 2121–2130, ISSN: 0018-9200.

[49] A. Amirkhany, J. Wei, N.K. Mishra, J. Shen, W.T. Beyene, C. Chen, T. J. Chin, D. Dressler, C. Huang, V.P. Gadde, M. Hekmat, K. Kaviani, H. Lan, P. Le, Mahabaleshwara, C. Madden, S. Mukherjee, L. Raghavan, K. Saito, D. Secker, A. Sendhil, R. Schmitt, S. Fazeel, G.S. Srinivas, T. Wu, C. Tran, A. Vaidyanath, K. Vyas, L. Yang, M. Jain, K.Y.K. Chang, and X.C. Yuan. "A 12.8-Gb/s/link Tri-Modal Single-Ended Memory Interface". In: *Solid-State Circuits, IEEE Journal of* (Apr. 2012), Vol. 47, No. 4, 911–925, ISSN: 0018-9200.

[50] M. Lin and K.T. Cheng. "Testable Design for Adaptive Linear Equalizer in High-Speed Serial Links". In: *Test Conference, 2006. ITC '06. IEEE International* (Oct. 2006), pp. 1–10. ISSN: 1089-3539.

[51] J. Sonntag, J. Stonick, J. Gorecki, B. Beale, B. Check, X.M. Gong, J. Guiliano, K. Lee, B. Lefferts, D. Martin, U.K. Moon, A. Sengir, S. Titus, G.Y. Wei, D. Weinlader, and Y.H. Yang. "An adaptive PAM-4 5 Gb/s backplane transceiver in 0.25 μm CMOS". In: *Custom Integrated Circuits Conference, 2002. Proceedings of the IEEE 2002* (2002), pp. 363–366.

[52] D. Tonietto, J. Hogeboon, E. Bensoudane, S. Sadeghi, H. Khor, and P. Krotnev. "A 7.5Gb/s transmitter with self-adaptive FIR". In: *VLSI Circuits, 2008 IEEE Symposium on* (June 2008), pp. 198–199.

[53] M.Q. Le, P.J. Hurst, and K.C. Dyer. "An analog DFE for disk drives using a mixed-signal integrator". In: *VLSI Circuits, 1998. Digest of Technical Papers. 1998 Symposium on* (June 1998), pp. 156–157.

[54] J.E.C. Brown, P.J. Hurst, I. Agi, and L. Der. "Analog decision-feedback equalizer architectures". In: *VLSI Signal Processing, VII, 1994., [Workshop on]* (1994), pp. 266–275.

[55] G. Han and E. Sanchez-Sinencio. "CMOS transconductance multipliers: a tutorial". In: *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on* (Dec. 1998), Vol. 45, No. 12, 1550–1563, ISSN: 1057-7130.

[56] M. Kargar and M.M. Green. "A 10 Gb/s adaptive analog decision feedback equalizer for multimode fiber dispersion compensation in 0.13 μm CMOS". In: *ESSCIRC, 2010 Proceedings of the* (Sept. 2010), pp. 550–553. ISSN: 1930-8833.

[57] C. Angerer, R. Langwieser, and M. Rupp. "Evaluation and Exploration of RFID Systems by Rapid Prototyping". In: *Personal Ubiquitous Comput.* (Mar. 2012), Vol. 16, No. 3, 309–321, ISSN: 1617-4909.

[58] B. Oraw, V. Choudhary, and R. Ayyanar. "A Cosimulation Approach to Model-based Design for Complex Power Electronics and Digital Control Systems". In: *Proceedings of the 2007 Summer Computer Simulation Conference*. SCSC '07 (2007), pp. 157–164.

[59] B. Gestner and D.V. Anderson. "Automatic generation of ModelSim-Matlab interface for RTL debugging and verification". In: *Circuits and Systems, 2007. MWSCAS 2007. 50th Midwest Symposium on* (Aug. 2007), pp. 1497–1500. ISSN: 1548-3746.

[60] P. Frey and D. O'Riordan. "Verilog-AMS: Mixed-signal simulation and cross domain connect modules". In: *Behavioral Modeling and Simulation, 2000. Proceedings. 2000 IEEE/ACM International Workshop on* (2000), pp. 103–108.

[61] P. Daglio and C. Roma. "A fully qualified top-down and bottom-up mixed-signal design flow for non volatile memories technologies". In: *Design, Automation and Test in Europe Conference and Exhibition, 2003* (2003), pp. 274–279. ISSN: 1530-1591.

[62] D. Potop-Butucaru, C. Lallement, and A. Vachoux. "VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems". In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* (Feb. 2005), Vol. 24, No. 2, 204–225, ISSN: 0278-0070.

[63] A. Lecointre, D. Dragomirescu, and R. Plana. "System Architecture Modeling of an UWB Receiver for Wireless Sensor Network". In: *Proceedings of the 7th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*. SAMOS'07 (2007), pp. 408–420.

[64] A. Vachoux, C. Grimm, and K. Einwich. "Towards analog and mixed-signal SOC design with systemC-AMS". In: *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on* (Jan. 2004), pp. 97–102.

[65] G.J. Esch and T. Chen. "Near-linear CMOS I/O driver with less sensitivity to process, voltage, and temperature variations". In: *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* (Nov. 2004), Vol. 12, No. 11, 1253–1257, ISSN: 1063-8210.

[66] A. Malkov, D. Vasiounin, and O. Semenov. "A Review of PVT Compensation Circuits for Advanced CMOS Technologies". In: *Circuits and Systems* (2011), Vol. 2, No. 3, 162–169,

[67] Q.A. Khan, G. K. Siddhartha, D. Tripathi, S.K. Wadhwa, and K. Misri. "Techniques for on-chip process voltage and temperature detection and compensation". In: *VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on* (Jan. 2006), 6 pp.–. ISSN: 1063-9667.

[68] D. Bhattacharya, A.V. Shukla, J.C. Kriz, and M. Kothandaraman. "Impedance compensation in a buffer circuit". In: (Apr. 2012). US Patent 8,159,262.

[69] S. Mei. "Digital calibration circuits, devices and systems including same, and methods of operation". In: (Oct. 2010). US Patent 7,821,291.

[70] J. Koo, G.S. Kim, J. Song, K.W. Kim, Y.J. Choi, and C. Kim. “Small-area high-accuracy ODT/OCD by calibration of global on-chip for 512M GDDR5 application”. In: *Custom Integrated Circuits Conference, 2009. CICC '09. IEEE* (Sept. 2009), pp. 717–720.

[71] *POD15, 1.5V Pseudo Open Drain I/O*. `www.jedec.org/standards-documents/docs/jesd-8-20`. Accessed: 2013-09-30.

[72] R. Senthinathan and J.L. Prince. “Application specific CMOS output driver circuit design techniques to reduce simultaneous switching noise”. In: *Solid-State Circuits, IEEE Journal of* (Dec. 1993), Vol. 28, No. 12, 1383–1388, ISSN: 0018-9200.

[73] Agilent Technologies. *Jitter Analysis: The dual-Dirac Model, RJ/DJ, and Q-Scale*. White Paper, 2004.

[74] W. El-Reedy, A.A. El-Moursy, and A.H. Fahmy. “High Performance Memory Requests Scheduling Technique for Multicore Processors”. In: *High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on* (June 2012), pp. 127–134.

[75] N. Rafique, W.T. Lim, and M. Thottethodi. “Effective Management of DRAM Bandwidth in Multicore Processors”. In: *Parallel Architecture and Compilation Techniques, 2007. PACT 2007. 16th International Conference on* (Sept. 2007), pp. 245–258. ISSN: 1089-795X.

[76] B.L. Jacob, P.M. Chen, S.R. Silverman, and T.N. Mudge. “An analytical model for designing memory hierarchies”. In: *Computers, IEEE Transactions on* (Oct. 1996), Vol. 45, No. 10, 1180–1194, ISSN: 0018-9340.

[77] A.C. Hsieh, T.T. Hwang, M.T. Chang, M.H. Tsai, C.M. Tseng, and H.C. Li. “TSV redundancy: Architecture and design issues in 3D IC”. In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2010* (Mar. 2010), pp. 166–171. ISSN: 1530-1591.

[78] E.J. Marinissen and Y. Zorian. “Testing 3D chips containing through-silicon vias”. In: *Test Conference, 2009. ITC 2009. International* (Nov. 2009), pp. 1–11.

[79] H.H.S. Lee and K. Chakrabarty. “Test Challenges for 3D Integrated Circuits”. In: *Design Test of Computers, IEEE* (Sept. 2009), Vol. 26, No. 5, 26–35, ISSN: 0740-7475.

[80] S.H. Pan, N. Chang, and T. Hitomi. “3D-IC dynamic thermal analysis with hierarchical and configurable chip thermal model”. In: *3D Systems Integration Conference (3DIC), 2013 IEEE International* (Oct. 2013), pp. 1–8.

[81] Y.S. Huang, Y.H. Liu, and J.D. Huang. “Layer-Aware Design Partitioning for Vertical Interconnect Minimization”. In: *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on* (July 2011), pp. 144–149. ISSN: 2159-3469.

[82] H.L. Chang, H.C. Lai, T.Y. Hsueh, W.K. Cheng, and M.C. Chi. "A 3D IC designs partitioning algorithm with power consideration". In: *Quality Electronic Design (ISQED), 2012 13th International Symposium on* (Mar. 2012), pp. 137–142. ISSN: 1948-3287.

[83] S. Mukherjee, O. Dan, A. Vaidyanath, D. Dressler, and A. Sendhil. "Challenges in extending single-ended graphics memory data rates". In: *Electrical Performance of Electronic Packaging and Systems (EPEPS), 2012 IEEE 21st Conference on* (Oct. 2012), pp. 39–42.

[84] O. Dan, S. Chang, C. Madden, J.H. Kim, R. Schmitt, M. Li, C. Ware, B. Leibowitz, Y. Frans, and N. Nguyen. "Design and characterization of a 12.8GB/s low power differential memory system for mobile applications". In: *Electrical Performance of Electronic Packaging and Systems, 2009. EPEPS '09. IEEE 18th Conference on* (Oct. 2009), pp. 33–36.

[85] K. Yoo and G. GHan. "An adaptation method for FIR pre-emphasis filter on backplane channel". In: *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on* (May 2006), 4 pp.–.

[86] B. Deutschmann and T. Ostermann. "CMOS output drivers with reduced ground bounce and electromagnetic emission". In: *Solid-State Circuits Conference, 2003. ESSCIRC '03. Proceedings of the 29th European* (Sept. 2003), pp. 537–540.

[87] L. Yang and J.S. Yuan. "Output buffer design for low noise and load adaptability". In: *Circuits, Devices and Systems, IEE Proceedings -* (Apr. 2005), Vol. 152, No. 2, 146–150, ISSN: 1350-2409.

[88] R.J. Baker. *CMOS Circuit Design, Layout, and Simulation*. 3rd. Wiley-IEEE Press, 2010. ISBN: 0470881321, 9780470881323.

[89] P.E. Allen and D.R. Holberg. *CMOS Analog Circuit Design*. Oxford series in electrical and computer engineering. Oxford University Press, 2002. ISBN: 9780195116441.

[90] S. Williams, H. Thompson, M. Hufford, and E. Naviasky. "An improved CMOS ring oscillator PLL with less than 4ps RMS accumulated jitter". In: *Custom Integrated Circuits Conference, 2004. Proceedings of the IEEE 2004* (Oct. 2004), pp. 151–154.

[91] P. Raha. "A 0.6-4.2V low-power configurable PLL architecture for 6 GHz-300 MHz applications in a 90 nm CMOS process". In: *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on* (June 2004), pp. 232–235.

[92] S.D. Brown and Z.G. Vranesic. *Fundamentals of digital logic with VHDL design*. McGr aw-Hill, 2000. ISBN: 9780070125919.

[93] A. Jaiswal, Y. Fang, and K. Hofmann. "A Wide Range Programmable Duty Cycle Corrector". In: *Patent, filed on 10 Sep.* (2012).

[94] A. Jaiswal, Y. Fang, and K. Hofmann. "Method and Device for Correcting a Phase Shift In a Time Synchronised System". In: *Patent, filed on 5th 29 Jun.* (2012).

Invention Disclosures

[16] Y. Fang, A. Jaiswal, and K. Hofmann. “Method and Means for Improving the Data Transfer Integrity In a Time Synchronised System”. In: *Patent, filed on 10th Sep.* (2012).

[93] A. Jaiswal, Y. Fang, and K. Hofmann. “A Wide Range Programmable Duty Cycle Corrector”. In: *Patent, filed on 10 Sep.* (2012).

[94] A. Jaiswal, Y. Fang, and K. Hofmann. “Method and Device for Correcting a Phase Shift In a Time Synchronised System”. In: *Patent, filed on 5th 29 Jun.* (2012).

List of Own Publications

[12] Y. Fang, J. Bargon, A. Jaiswal, and K. Hofmann. "A Low-Jitter Clock and Data Recovery Using Proportional-Integral Controller for GDDR5 Interface Trainings". In: *Integrated Circuit Design and Technology, 2014. ICICDT '14. International Conference on* (May 2014).

[13] Y. Fang, L. Chen, A. Jaiswal, K. Hofmann, and P. Gregorius. "Adaptive Equalizer Training for High-Speed Low-Power Communication Systems". In: *Digital System Design (DSD), 2013 Euromicro Conference on* (Sept. 2013), pp. 745–751.

[14] Y. Fang, A. Jaiswal, and K. Hofmann. "Low-power signal integrity trainings for multi-clock source-synchronous memory systems". In: *SOC Conference (SOCC), 2013 IEEE 26th International* (Sept. 2013), pp. 319–324. ISSN: 2164-1676.

[15] Y. Fang, L. Chen, A. Jaiswal, and K. Hofmann. "Transmitter equalizer training based on pilot signal and peak detection". In: *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on* (Dec. 2013), pp. 377–380.

[17] A. Jaiswal, Y. Fang, K. Hofmann, and P. Gregorius. "An efficient methodology for mixed-signal high-speed memory design using Matlab/Simulink-HDL co-simulation". In: *Solid-State and Integrated Circuit Technology (ICSICT), 2012 IEEE 11th International Conference on* (Oct. 2012), pp. 1–3.

[18] Y. Fang, U. Muhammad, A. Jaiswal, and K. Hofmann. "Low-power design of hybrid digital impedance calibration for process, voltage, temperature compensations". In: *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on* (Dec. 2013), pp. 37–40.

List of Unrelated Publications

[95] A. Jaiswal, Y. Fang, P. Gregorius, and K. Hofmann. “Adaptive Low-Power Synchronization Technique for Multiple Source-Synchronous Clocks in High-Speed Communication Systems”. In: *Digital System Design (DSD), 2013 Euromicro Conference on* (Sept. 2013), pp. 752–758.

[96] A. Jaiswal, Y. Fang, K. Nawaz, and K. Hofmann. “A wide range programmable duty cycle corrector”. In: *SOC Conference (SOCC), 2013 IEEE 26th International* (Sept. 2013), pp. 192–196. ISSN: 2164-1676.

Supervised Theses

[97] D. Noll. “A Survey of the High-Speed Chip-to-Chip I/O Circuits Evolution”. In: *Pro-seminar* (2011).

[98] T. Casper. “Design of a PLL for chip-to-chip I/O Circuits”. In: *Pro-seminar* (2011).

[99] X. Ding. “System-level modeling of a phase interpolator based clock and data recovery using Simulink”. In: *Pro-seminar* (2012).

[100] K. Nawaz. “High speed duty cycle corrector”. In: *Project Seminar* (2012).

[101] M.A. Mosabbah. “Level shifters for high speed system”. In: *Project Seminar* (2012).

[102] M. D. Serrano. “Transceivers for high-speed IO”. In: *Project Seminar* (2012).

[103] L. Chen. “Equalization circuits for noise cancellation”. In: *Master Thesis* (2013).

[104] D. Gadler. “Phase interpolator-based PLL”. In: *Master Thesis* (2013).

[105] U. Muhammad. “Digital impedance calibration for PVT compensation”. In: *Master Thesis* (2013).

[106] D. Walk. “Common knowledge of serializer-deserializer”. In: *Pro-seminar* (2013).

[107] N. Eberlein. “A review on Memory Power Management via Dynamic Voltage/Frequency Scaling”. In: *Pro-seminar* (2013).

[108] J. Bargon. “Implementation of clock data recovery”. In: *Master Thesis* (2014).

Curriculum Vitae

Personal Data :

Surname	:	Fang
Name	:	Yuan
Birth date	:	16 October 1983
Location	:	Shanghai, China

School Education :

Year	*School level*	*School name*
1989 – 1994	Primary school	Ri Hui Liu Cun School, Shanghai, China
1994 – 1998	Secondary school	Middle School attached to Shanghai Normal University, Shanghai, China
1998 – 2001	High school	Middle School attached to Shanghai Normal University, Shanghai, China
2003 – 2004	Language school	Technische Universität Dresden Institute of advanced studies, Dresden, Germany

Higher Education :

Year	*Degree*	*Institute*
2004 – 2006	Intermediate Diploma	Technische Universität Dresden, Dresden, Germany
2006 – 2009	Bachelor	Technische Universität München, Munich, Germany
2006 – 2010	Diploma	Technische Universität München, Munich, Germany
2010 – 2014	Doctoral	Technische Universität Darmstadt, Darmstadt, Germany

Work Experience :

Year	*Position*	*Organizations*
$2010 - 2014$	Research Assistant	Technische Universität Darmstadt, Darmstadt, Germany
$2008 - 2009$	Internship	Infineon, Munich, Germany
$2008 - 2009$	Bachelor Thesis	MUNEDA, Munich, Germany