Source-synchronous I/O Links using Adaptive Interface Training for High Bandwidth Applications

Vom Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines
Doktor–Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

M. Tech.
Ashok Kumar Jaiswal
geboren am 04. März 1981
in Pratapgarh, Indien

Referent:	Prof. Dr. Ing. Klaus Hofmann *Technische Universität Darmstadt*
Korreferent:	Prof. Dr. sc. techn. Andreas Herkersdorf *Technische Universität München*
Tag der Einreichung:	22.04.2014
Tag der mündlichen Prüfung:	16.07.2014

D17

Darmstadt, 2014

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.

Darmstadt, 22.04.2014

Acknowledgments

This thesis is the outcome of my work as a research assistant at the Institute of Integrated Electronic Systems, Technische Universität Darmstadt. First of all, I want to sincerely thank my supervisor Prof. Klaus Hofmann for his kind advice and guidance. He supported me not only in technical matters but was also available as a local guardian whenever I needed any personal guidance. I also express my gratitude towards Prof. Andreas Herkersdorf, who kindly accepted to act as co-supervisor for this thesis. His comments have been very valuable for improving the quality of the work. Furthermore, I would like to thank Prof. Franko Küppers, Prof. Ralf Steinmetz, and Prof. Christian Hochberger for acting as members of the examination committee.

This whole journey would have not been so interesting without all colleagues at the institute who made a great working atmosphere. I would like to thank Haoyuan Ying, Prof. Thomas Hollstein, Prof. Mohamed Salem, Leandro Möller and Faizal Samman for all the interesting discussions on Networks-on-Chip topic. I would specially like to thank Yuan Fang who worked together as a great team member on high-speed chip-to-chip communication topic. I would also like to thank Prof. Peter Gregorius for his finishing comments on various papers. I thank my colleagues Boris Traskov, Christian Küller, Katrin Schaechtle, Dr. Roland Steck who helped me in proof reading and German translation. I also want to thank all the students who did their seminar, thesis under my supervision and created a mutual-learning experience. Many thanks to Andreas Schmidt, Roland Brand and Silvia Hermann for making our day-to-day life easy at our institute. I want to thank my ex-Qimonda colleague Martin Maier regarding GDDR5 training hints. I want to sincerely thank my mentor Andreas Jakobs (from Qimonda days) for all the suggestions to improve the quality of the thesis.

I want to thank my wife Sapna for her love, patience and continuous support without which firstly I could have not made the decision to join academia and secondly reach at this stage of writing acknowledgment. Last but not the least, I would like to thank my parents and entire family for their blessings.

Kurzfassung

Die Forderung nach permanenter Erreichbarkeit der Nutzer setzt eine ununterbrochene Anbindung an das Internet und somit an zentrale Server voraus. Mit der exponentiellen Zunahme von mobilen Endgeräten wie Smartphones, Tablets und Laptops wird der Datentransfer zum Jahr 2018 voraussichtlich die Exabyte-Schwelle überschreiten. Zusätzlich werden Anwendungen wie Videostreaming, Video-on-Demand, Online-Gaming und Soziale Netzwerke das Datenvolumen weiter erhöhen. Zukünftige Anwendungsszenarien wie Smart Cities, das Internet der Dinge, Industrie 4.0 und Machine-to-machine (M2M) Kommunikation stellen darüber hinaus höchste Anforderungen an die Kommunikationsinfrastruktur, wie z.B. hohe Datenraten bei gleichzeitig niedriger Leistungsaufnahme. Wissenschaftliche Untersuchungen wie die Erforschung des Weltalls sowie die Ölförderung werden im Jahr 2018 voraussichtlich Rechengeschwindigkeiten im Exaflops/s-Bereich benötigen, was einen Datendurchsatz pro Speicherschnittstelle im TB/s-Bereich erfordert. Um einen solchen Datendurchsatz zu erreichen, müssen die I/O-Link-Geschwindigkeiten zwischen zwei Geräten in den GB/s-Bereich erhöht werden.

Bei solch hohen Datenraten können Informationen sowohl über komplexe, serielle Clock-Data-Recovery (CDR) als auch über einfachere, parallele Quellen-synchrone Verbindungen übertragen werden. Obwohl CDR im Vergleich zur Quellen-synchronen Alternative effizienter ist, sind für das Erreichen einer TB/s-Datenrate mehrere serielle Verbindungen notwendig. Abgesehen davon kann die parallele Quellen-synchrone Übertragung hinsichtlich Leistungsaufnahme und Silizium-Flächenbedarf Vorteile für sich verbuchen, da zusätzliche I/Os keine weiteren Hardwareressourcen erfordern. Bei hohen Datenraten treten bei Quellen-synchronen Verbindungen jedoch Probleme wie Rauschen der Versorgungsspannung, Übersprechen, Inter-Symbol-Interferenzen (ISI) usw. auf, die Laufzeitunterschiede zur Folge haben. Um diesen Problemen zu entgegnen, kann die Methode des adaptiven Trainings im Zeitbereich angewandt werden, um weiterhin mit höchsten Datenraten zu kommunizieren.

In dieser Dissertation werden zwei neue Architekturen für adaptive Quellen-synchrone Verbindungen vorgestellt, die gegenüber bisherigen Implementierungen signifikante Vorteile bezüglich Leistungsverbrauch und Siliziumfläche aufweisen. Die erste Architektur basiert auf einer Verzögerungseinheit, die inkrementell kleinste Verzögerungen zur Phase eines Takts addiert. Eine zweite Architektur basiert auf einem in der PLL (Phase Locked Loop) integrierten Phaseninterpolator (PI). Dieser kann kleinste Verzögerungen zur Phase eines Takts sowohl hinzufügen als auch subtrahieren. Dadurch kann die Synchronisation auf Kosten höherer Komplexität schneller erreicht werden.Gerade bei Double Data Rate (DDR) Systemen, die für Quellen-synchrone Systeme mit hohen Datenraten üblicherweise eingesetzt werden, reduzieren auch ein nicht-optimaler Taktbaum sowie ein unausbalanciertes Tastverhältnis die Timingmarge. Um dem entgegenzuwirken, werden in dieser Dissertation auch ein neuartiger Algorithmus zur Taktbuffergenerierung sowie ein neuartiges Tastverhältnis-Korrekturglied vorgestellt. Dadurch kann auch eine Reduzierung der Leistungsaufnahme Quellen-synchroner Systeme erreicht werden.

Abstract

Mobility is the key to the global business which requires people to be always connected to a central server. With the exponential increase in smart phones, tablets, laptops, mobile traffic will soon reach in the range of Exabytes per month by 2018. Applications like video streaming, on-demand-video, online gaming, social media applications will further increase the traffic load. Future application scenarios, such as Smart Cities, Industry 4.0, Machine-to-Machine (M2M) communications bring the concepts of Internet of Things (IoT) which requires high-speed low power communication infrastructures. Scientific applications, such as space exploration, oil exploration also require computing speed in the range of Exaflops/s by 2018 which means TB/s bandwidth at each memory node. To achieve such bandwidth, Input/Output (I/O) link speed between two devices needs to be increased to GB/s.

The data at high speed between devices can be transferred serially using complex Clock-Data-Recovery (CDR) I/O links or parallely using simple source-synchronous I/O links. Even though CDR is more efficient than the source-synchronous method for single I/O link, but to achieve TB/s bandwidth from a single device, additional I/O links will be required and the source-synchronous method will be more advantageous in terms of area and power requirements as additional I/O links do not require extra hardware resources. At high speed, there are several non-idealities (Supply noise, crosstalk, Inter-Symbol-Interference (ISI), etc.) which create unwanted skew problem among parallel source-synchronous I/O links. To solve these problems, adaptive trainings are used in time domain to synchronize parallel source-synchronous I/O links irrespective of these non-idealities.

In this thesis, two novel adaptive training architectures for source-synchronous I/O links are discussed which require significantly less silicon area and power in comparison to state-of-the-art architectures. First novel adaptive architecture is based on the unit delay concept to synchronize two parallel clocks by adjusting the phase of one clock in only one direction. Second novel adaptive architecture concept consists of Phase Interpolator (PI)-based Phase Locked Loop (PLL) which can adjust the phase in both direction and achieve faster synchronization at the expense of added complexity. With an increase in parallel I/O links, clock skew which is generated by the improper clock tree, also affects the timing margin. Incorrect duty cycle further reduces the timing margin mainly in Double Data Rate (DDR) systems which are generally used to increase the bandwidth of a high-speed communication system. To solve clock skew and duty cycle problems, a novel clock tree buffering algorithm and a novel duty cycle corrector are described which further reduce the power consumption of a source-synchronous system.

Contents

1 Introduction **1**

1.1 Problem description and motivation 2
1.2 Source-synchronous I/O links: background 5
1.3 Source-synchronous I/O links: challenges 9
  1.3.1 Unwanted Board Skew 9
    1.3.1.1 Supply (Vref) noise / SSO noise 9
    1.3.1.2 Crosstalk 10
    1.3.1.3 Inter-Symbol-Interference 11
    1.3.1.4 Ci Mismatch 11
    1.3.1.5 Termination Mismatch 12
    1.3.1.6 Trace Length Mismatch 12
  1.3.2 Clock Skew and Duty Cycle 13
1.4 Thesis Overview 14

2 Source-Synchronous I/O Links: Adaptive Training **15**

2.1 Introduction 16
2.2 Current state-of-the-art architecture 22
2.3 Novel Architectures 23
  2.3.1 Unit-delay phase incrementer 25
  2.3.2 PI-based PLL 25
    2.3.2.1 Functioning of the system 28
2.4 Conclusion 31

3 Source-Synchronous I/O Links: Clock Skew and Duty Cycle **33**

3.1 Clock Skew 34
  3.1.1 Background 34
  3.1.2 Power optimized buffering 36
    3.1.2.1 Preparing the clock-tree 38
    3.1.2.2 Pre-Buffering PVT evaluation 38
    3.1.2.3 Buffering with clock-skew and slew-rate specifications . . . 39
    3.1.2.4 Polarities 41
    3.1.2.5 Post-Buffering Optimization 41

3.1.3 Scalability 43
3.1.3.1 Improvements because of better clock-skew 43
3.1.3.2 Improvements within the algorithm. 43
3.2 Duty Cycle 45
3.2.1 Circuit structure of the considered DCC 47
3.2.2 Novel Architecture with programmable charge pump 49
3.3 Conclusions 50

4 Simulation Results 51
4.1 Adaptive trainings: Results 52
4.1.1 Comparison with state-of-the-arts 52
4.1.2 Results using circuit level simulation 54
4.1.2.1 Low Frequency Simulations 54
4.1.2.2 High Frequency Simulations 54
4.2 Clock Skew: Results 55
4.2.1 Scalability 55
4.2.2 Benchmarks 56
4.3 Duty Cycle: Results 57

5 Summary and Future Works 63
5.1 Summary 64
5.2 Future Works 64

A Design Methodology 67
1.1 Design Methodology using Matlab/Simulink-Cadence co-simulation 68
1.1.1 Related Work 69
1.1.2 Design Flow 70
1.1.3 Simulation Setup 71
1.1.4 Simulation Results 74
1.2 Conclusions 76

B Implementation of PI-based PLL architecture 77
2.1 Introduction 78
2.2 Implementation 78
2.2.1 Phase-Frequency Detector 79
2.2.2 Charge-Pump 80
2.2.3 Operational Transconductance Amplifier 81
2.2.4 Voltage-to-Current Converters 81
2.2.5 Current Controlled Oscillator 83
2.2.6 Phase Selector and Frequency Divider 84
2.2.7 Programmable Filter 85
2.2.8 Comparator 87

2.2.9 Digital Control 87
2.2.10 Lock Detector 88

References **91**

Invention Disclosures **99**

List of Own Publications **101**

List of Unrelated Publications **103**

Supervised Master Theses **105**

Supervised Bachelor Theses **107**

Supervised Seminars **109**

Glossary **111**

Resume **115**

List of Tables

1.1 Comparison of Compound Annual Growth Rate (CAGR) of Global Device Units and Global Mobile Data Traffic [1] 2
1.2 Comparison of supercomputers with projected Exascale Machine [8] 3
1.3 High-speed CDR architectures 4
1.4 High-Speed Source-Synchronous Architectures 5
1.5 Source-Synchronous Memories 9

2.1 Propagation delays inserted the system. 29
2.2 Theoretical worst case error for the different configurations at 2 GHz Write Clock (WCK). 31
2.3 Theoretical worst case error of 5GHz WCK configuration. 31

3.1 Weight parameters for controlled solution pruning 45

4.1 Comparison with other work 54
4.2 Measured error of 2 GHz WCK configuration. 55
4.3 Measured error of 5GHz WCK configuration. 55
4.4 Run-time comparison of the novel buffering algorithm with seven different randomly generated circuits. 56
4.5 Comparison between the results generated by Reliable Low Power Buffering and Optimization algorithm (RLPBO) and Contango 2 57
4.6 Comparison with the previous work 59

A.1 Modeling approaches and performance trade-off 76

B.1 PLL specifications. 78
B.2 Coarse-loop Voltage-to-Current (V2I) converter performance parameter. . . 82
B.3 Fine-loop V2I converter performance parameter. 83
B.4 Current Controller Oscillator (CCO) output performances. 85
B.5 Parameters of the components of the PLL. The Voltage Controller Oscillator (VCO) gains were measured employing a filter resistor of 7 kΩ. 86

B.6 Laplace-domain features of the PLL systems and respective resistance values. 87

List of Figures

1.1 Conventional CDR . . . 3
1.2 Source-Synchronous System . . . 4
1.3 SSD-SSR in the Network Interface . . . 6
1.4 Intel Quickpath Interconnect . . . 7
1.5 Physical Layer of Intel Quickpath Interconnect . . . 7
1.6 HyperTransport Interconnect . . . 8
1.7 Physical Layer of HyperTransport Interconnect . . . 8
1.8 Source-Synchronous Skew Distribution [33] . . . 10
1.9 Crosstalk [33] . . . 10
1.10 ISI [33] . . . 11
1.11 Ci-Mismatch [33] . . . 12
1.12 Termination-Mismatch [33] . . . 12
1.13 Clock Skew . . . 13
1.14 Duty Cycle Error . . . 13

2.1 Source-Synchronous Clock Forward (SSCF) link concept. . . . 16
2.2 Adaptive Training Concept . . . 16
2.3 High-speed chip-to-chip communication . . . 17
2.4 PI example [46] . . . 17
2.5 QuickPath Receiver with two PIs [49] . . . 18
2.6 Digital CDR with two PIs [50] . . . 18
2.7 High-speed chip-to-chip communication with feedback . . . 19
2.8 Intel QuickPath with Receiver Feedback . . . 19
2.9 High-speed Master-Slave communication . . . 20
2.10 High-speed chip-to-chip communication in Master-Slave configuration . . . 20
2.11 GDDR5 Example [35] . . . 21
2.12 GDDR5 training sequence [35] . . . 21
2.13 Error Detection and Correction (EDC) functionality during WCK2CK training [35] . . . 22

2.14 Current state-of-the-art [35] 22
2.15 Current state-of-the-art: (A) Phase-detector (B) Phase-adjuster 23
2.16 Novel Architecture based on Unit-Delay Incrementer 24
2.17 Novel Architecture with PI-based PLL 24
2.18 Custom Delay Cells in TSMC 65nm: (a) Voltage controlled buffer (28ps typ. delay); (b) Voltage controlled inverter (10ps typ. delay); (c) Transmission gate (3ps typ. delay) 25
2.19 PI-based-PLL [57] 26
2.20 Dual Loop PLL (DLPLL) architecture [59] 26
2.21 PI-based PLL in CDR architecture [57] 27
2.22 PLL architecture utilized in this design. 28
2.23 Bode-plot of the fine loop, coarse loop and open loop transfer functions. . . 29
2.24 step-response. 30

3.1 Overview of the clock generation 35
3.2 Flowchart 37
3.3 Adjustment of the tree topology prior to buffering 38
3.4 Flowchart of the post-buffering optimization of the clock-tree to achieve clock-skew constraints 42
3.5 Analog Duty Cycle Corrector (DCC) [69] 46
3.6 Digital DCC [69] 46
3.7 Mixed-signal DCC [80] 47
3.8 Gain-boosting Charge Pump (CP) [80] 48
3.9 Loop Analysis [80] 48
3.10 CP characteristics simulated in TSMC 65nm for 3 GHz - 5 GHz 49
3.11 Programmable CP 49

4.1 Schematic:Unit-Delay incrementer based architecture 53
4.2 Run-time of the buffering algorithm for different random circuits shows linearity in the number of buffering positions 56
4.3 Output Duty cycle for 70% input duty cycle 58
4.4 Programmable CP and Buffer 58
4.5 CP characteristics for 6 GHz & 7 GHz 59
4.6 Output Duty cycle for 70% input duty cycle 60
4.7 Output Duty cycle for 30% input duty cycle 60

A.1 Conventional Design Flow 68
A.2 Adopted Design Flow 70
A.3 Abstract Specification 71

A.4 Level 1: Matlab Simulation 72
A.5 Level 2: Controller Block-Level Specification 73
A.6 Level 2: Matlab-HDL Co-simulation 74
A.7 Level 3: Verilog-AMS and SPICE co-simulation 74
A.8 DQ in read training 75
A.9 DQ in write training 75
A.10 DQ before and after channel 75

B.1 Schematic of the charge-pump employed in this design. 80
B.2 Schematic of the sub-threshold Operational Transconductance Amplifier (OTA) employed in this design. 80
B.3 Schematic of the coarse loop V2I converter. 81
B.4 Schematic of the fine loop V2I converter. 82
B.5 Schematic of the four phases CCO. 83
B.6 Schematic of the delay cell.[57] 84
B.7 Duty cycle corrector [92]. 84
B.8 Waveforms showing the difference between the selection of an anticipated *a)* or a delayed *b)* VCO output in respect to the current one. 86
B.9 Waveforms of the control signals after the lock-in of the PLL. 88
B.10 Schematic of the lock detector. 88

Chapter 1

Introduction

1.1 Problem description and motivation

The growth in mobile devices like smartphones, tablets, laptops leads to the exponential growth in mobile data traffic (Table 1.1) which may reach in the range of Exabytes by 2018 [1] due to high-definition videos, on-demand media, gaming, image search, social media applications. This growth in mobile connectivity has initiated the paradigm shift of not only connecting humans via mobile devices, but also connecting all the objects via Machine-to-Machine (M2M) or Device-to Device (D2D) communications using Internet of Things (IoT) infrastructure [2], [3] which will further enable smart cities [4] and industry 4.0 [5]. Table 1.1 shows that Compound Annual Growth Rate (CAGR) of traffic due to M2M module will be 113% till 2018. With such data traffic increase, new optimized data mining [6] and deployment schemes [7] are being investigated which will need energy efficient computing platforms.

Table 1.1: Comparison of CAGR of Global Device Units and Global Mobile Data Traffic [1]

Device Type	**Growth in Devices 2013-2018 CAGR**	**Growth in Mobile Traffic 2013-2018 CAGR**
Smart Phone	18%	63%
Tablet	41%	87%
Laptop	13%	30%
M2M Module	43%	113%

Scientific applications like space explorations, oil exploration demand high computation in the range of ExaFlop/s. For space exploration, the Square Kilometer Array (SKA) project [8] is initiated to build a next generation radio telescope which will require in excess of one ExaFlop/s in order to process and reduce the massive amount of data generated by the sensors. Table 1.2 shows the comparison among projected Exascale machine and two recent top-of-the-line supercomputers where Theoretical Peak Performance (Rpeak) of the system is measured in number of Flops e.g. petaFlops (PF). The memory bandwidth per node is conservatively projected at 2-4 TB/s and the total power required is 20 MW. To realize tomorrow's Exascale systems, the chips need to bring energy efficiency down from about 3000 pJ per flop to about 20 pJ per flop.

Due to this rising demand on computation, high bandwidth communication is required at all levels of the system design for example processor-to-processor or processor-to-memory communications. The bandwidth of electrical communication links can be increased either through increasing pin counts or increasing interface speed per link (or some combination of the two). The 2010 International Technology Roadmap for Semiconductors (ITRS) roadmap [9] predicts that the number of Application Specific Integrated Circuit (ASIC) pins and microprocessor pins will increase 5% per year, respectively. It also predicts that the high-performance off-chip Input/Output (I/O) speed will increase 10% per year up to over 50 Gbps in 2020 [9]. Each additional pin causes more power as charging and discharging of pin capacitance is needed. It additionally increases package and Printed Circuit Board (PCB) complexity.

Table 1.2: Comparison of supercomputers with projected Exascale Machine [8]

	2009 **Jaguar**	**2011** **K computer**	**2018** **projected**	**2009 Vs 2018** **factor**
System Rpeak	2 PF	10 PF	1 EF	O(1000)
Node Rpeak	125 GF	128 GF	1 - 11 TF	O(10-100)
Power	6 MW	10 MW	20 MW	O(10)
Energy/Flop	3 nJ/F	1 nJ/F	20 pJ/F	- O(100)
System Memory	0.3 PB	1 PB	32-64 PB	O(100)
Memory/Flop	0.6 B/F	0.1 B/F	0.03 B/F	- O(10)
Memory BW/node	25 GB/s	64 GB/s	2-4 TB/s	O(100)
Memory BW/Flop	0.2 B/s/F	0.5 B/s/F	0.002 B/s/F	-O(100)
Total concurrency	225,000	548,352	O(10^9)	O(10^5)

Due to the limitation on pin count, the current focus is on increasing link speed. The data is sent over these high speed links using mainly two different clocking schemes: Clock-Data-Recovery (CDR) [10],[11] and source-synchronous [12],[13]. Conventional CDR (Figure 1.1) includes Phase Detector (PD), Phase Interpolator (PI), filter, Phase Locked Loop (PLL) etc., which consume a significant amount of chip area and power. To achieve fast and the accurate phase relationship between clock and data, CDR clocking schemes employ different techniques, e.g. dual edge injection locking [14], eye-tracking [15], phase-tracking [11], [16], [17], [18] while maintaining the power budget.

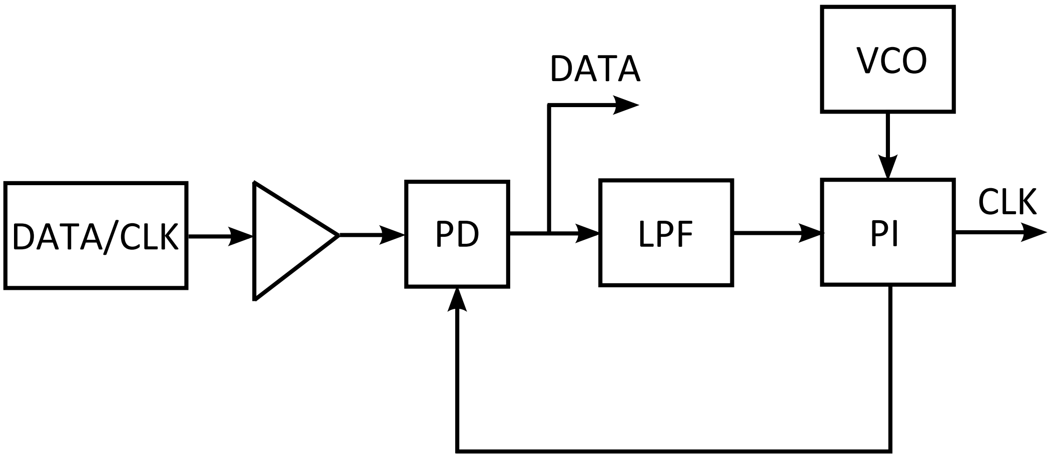

Figure 1.1: Conventional CDR

On the other hand, source-synchronous clocking system (Figure 1.2) is used for high-speed parallel interfaces due to its simple clock distribution and fast power control techniques [13]. As the clock is forwarded along with the data to the receiver, it eliminates the need of receiver PLL and the data coding. However, non-idealities such as noise in channels, crosstalk, Inter-Symbol-Interference (ISI) affect both forwarded clock and data. To solve these problems, additional techniques e.g. equalization circuits [12], common-mode clocking [19], I/O calibration [13] are used.

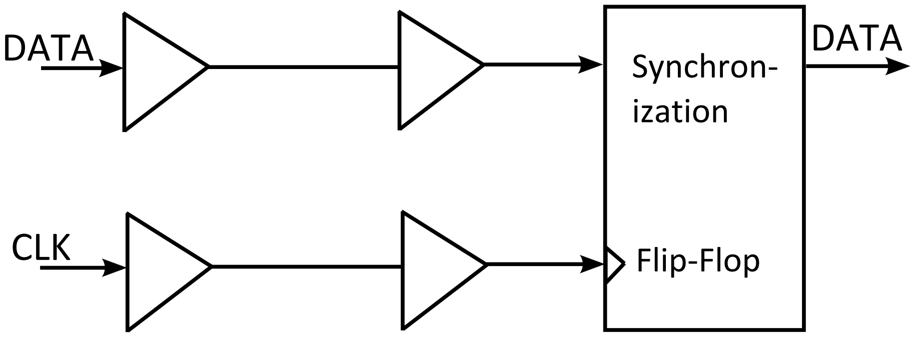

Figure 1.2: Source-Synchronous System

Table 1.3: High-speed CDR architectures

CDR Architecture	technology	Data rate (Gbps)	power (mW)	area (mm2)	BER	Power Efficiency (mW/Gbps)
[10]	65nm CMOS	5	280	0.71	10^{-12}	56.00
[14]	32nm SOI CMOS	5.814	32.2	0.58	10^{-10}	5.54
[15]	90nm CMOS	8	232	0.29	10^{-12}	29.00
[16]	90nm CMOS	8	160	0.48	10^{-15}	20.00
[17]	40nm CMOS	8	235	0.45	10^{-10}	29.38
[18]	65nm CMOS	10	102.56	0.25	-	10.26
[20]	32nm CMOS	11.8	78	0.16	$2*10^{-15}$	6.61
[21]	65nm CMOS	12.5	12.3	0.24	10^{-12}	0.98
[11]	65nm CMOS	40	655	1.78	10^{-12}	16.38

Table 1.3 and Table 1.4 show the CDR and source-synchronous architecture characteristics in recent publications, respectively. The shown numbers belong to a link representing single electrical channel interface including transmitter and receiver circuitries. Furthermore, following conclusions can be made from these tables:

1. CDR architectures achieve in general higher data rates per link than source synchronous architectures.

2. CDR architectures have better power efficiency per link for the same technology with similar data-rate.

Table 1.4: High-Speed Source-Synchronous Architectures

Source Synchronous Architecture	**technology**	**Data rate (Gbps)**	**power (mW)**	**area (mm2)**	**BER**	**Power Efficiency (mW/Gbps)**
[12]	0.13um CMOS	8	280	0.13	10^{-10}	35.00
[19]	40nm LP CMOS	5	25	-	10^{-15}	5.00
[22]	32nm SOI CMOS	8	85	4.83	$3.3*10^{-12}$	10.63
[23]	65nm CMOS	10	140	-	10^{-12}	14.00
[13]	45nm SOI	11.5	34.4	-	10^{-9}	4.3

With these observations, CDR architectures are the preferred option to achieve higher data-rates at high power efficiency per link. To fulfill Exascale bandwidth, single CDR link will not be sufficient and parallel links (32 or more channels) need to be considered. Therefore, the total power consumption and complexity of CDR architectures will increase linearly with an increase in the number of links as a CDR is needed for every link. In other words, source synchronous architectures can reduce circuit complexity and power for parallel link systems.

At high-speed, source-synchronous electrical interfaces are mainly limited by channel non-idealities i.e. crosstalk, ISI etc. and need better control over the phase relationship among clock and data and signal integrity. These channel non-idealities can be handled by interface trainings where phases and signal quality of data/clocks can be controlled according to the channel characteristics which lead to lower the Bit-Error-Rate (BER). These adaptive interface trainings need to have a simple architecture with low power consumption.

1.2 Source-synchronous I/O links: background

Source-synchronous systems have evolved from synchronous systems where transmitter and receiver Integrated Circuits (ICs) use the same clock source. The speed of a synchronous system is restricted by the fact that the flight time (timing delay) between transmitter and receiver can not be more than the defined clock period. Therefore, to increase the bandwidth, number of pins has to be increased which further increases the power consumption and complexity of the chip. To overcome the flight time issue, the clock is also forwarded along with the data from transmitter IC (source) to receiver IC (destination) that is why this system is called source-synchronous (Figure 1.2). In such systems, bandwidth can be increased directly by increasing the link frequency.

The source-synchronous concept has been adopted by many I/Os and memory interfaces to fulfill high-bandwidth requirements. Scalable Coherent Interface (SCI) [24], SGI-Craylink interface [25] and High-Performance Parallel Interface (HIPPI-6400-PH) [26] have used Source Synchronous Drivers/Receivers (SSD/SSR) in the network interface (Figure 1.3) for high speed network communication. SSD sends wide bus (e.g. 80 bits) information with core frequency (e.g. 200 MHz) to high frequency (e.g. 800 MHz) network with small bus (20 bits) while SSR performs the reverse operation.

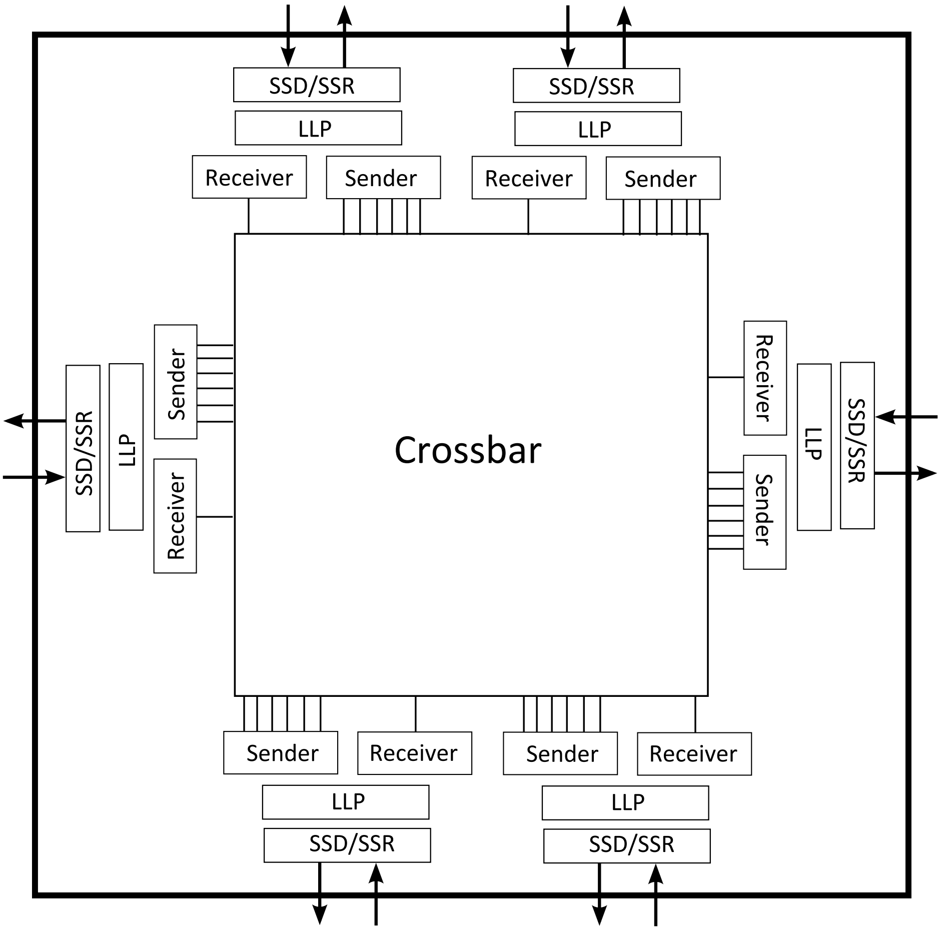

Figure 1.3: SSD-SSR in the Network Interface

Quickpath interconnect [27] (Figure 1.4) is a high-speed point-to-point source-synchronous interconnect used in Intel microprocessors. Physical layer (Figure 1.5) of Quickpath interconnect consists of 20 bits wide two unidirectional data bus with 1 bit clock in each direction which is able to provide 25.6 GB/s of low-latency bandwidth. Physical layer comes with additional functionality of lane/polarity reversal, data recovery, deskew at the receiver and waveform equalization, which fulfills Reliability, Availability, Serviceability (RAS) requirements of high-performance systems.

On the other hand, HyperTransport 3 [28] (Figure 1.6) uses 32 bit full duplex point-to-point source-synchronous interconnect (Figure 1.7) to achieve 51.2 GB/s bandwidth. HyperTransport provides dynamic link clock/width adjustment on the fly and hot plugging for server applications. It also uses training and equalization schemes for high performance and signal reliability.

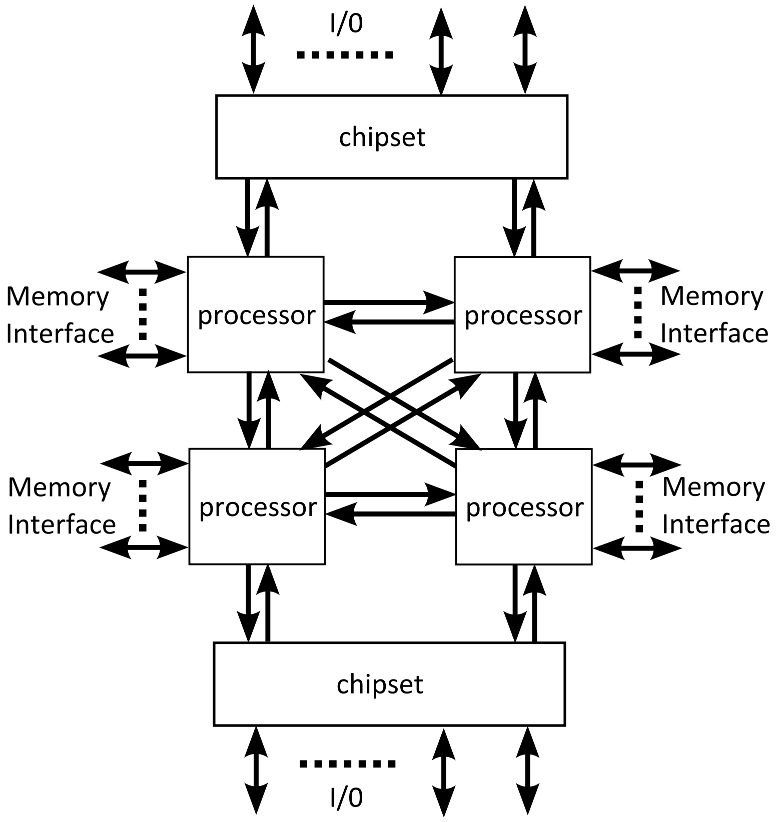

Figure 1.4: Intel Quickpath Interconnect

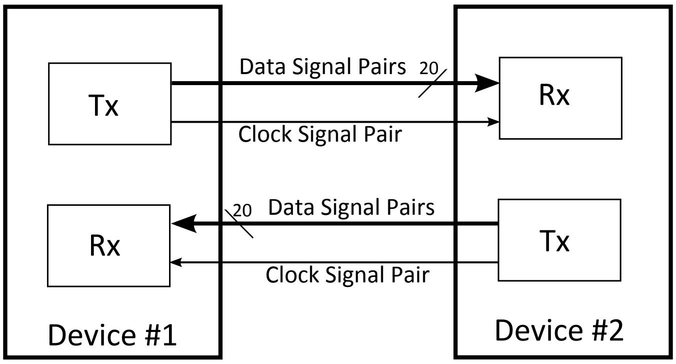

Figure 1.5: Physical Layer of Intel Quickpath Interconnect

Table 1.5 shows the source-synchronous memories and the respective achievable bandwidth for graphics applications where connections are point-to-point and chips are directly soldered on PCB board. The first generation of Source-Synchronous Dynamic Random Access Memory (SDRAM) was Single Data Rate (SDR) where data is sampled at each positive edge of the system (memory) clock. The next generations of SDRAM follow DDR scheme where data is sent on both positive and negative edges of the system clock. Graphics Dynamic Random Access Memory (GDRAM) is the variation of SDRAM working at higher voltage than SDRAM to achieve better data-rate. Each improved GDRAM generation [29], [30], [31] (GDDR, GDDR2, GDDR3, GDDR5) works at a faster memory clock, hence achieving greater bandwidth (represented as Transfer rate in Table 1.5). GDDR5 is still

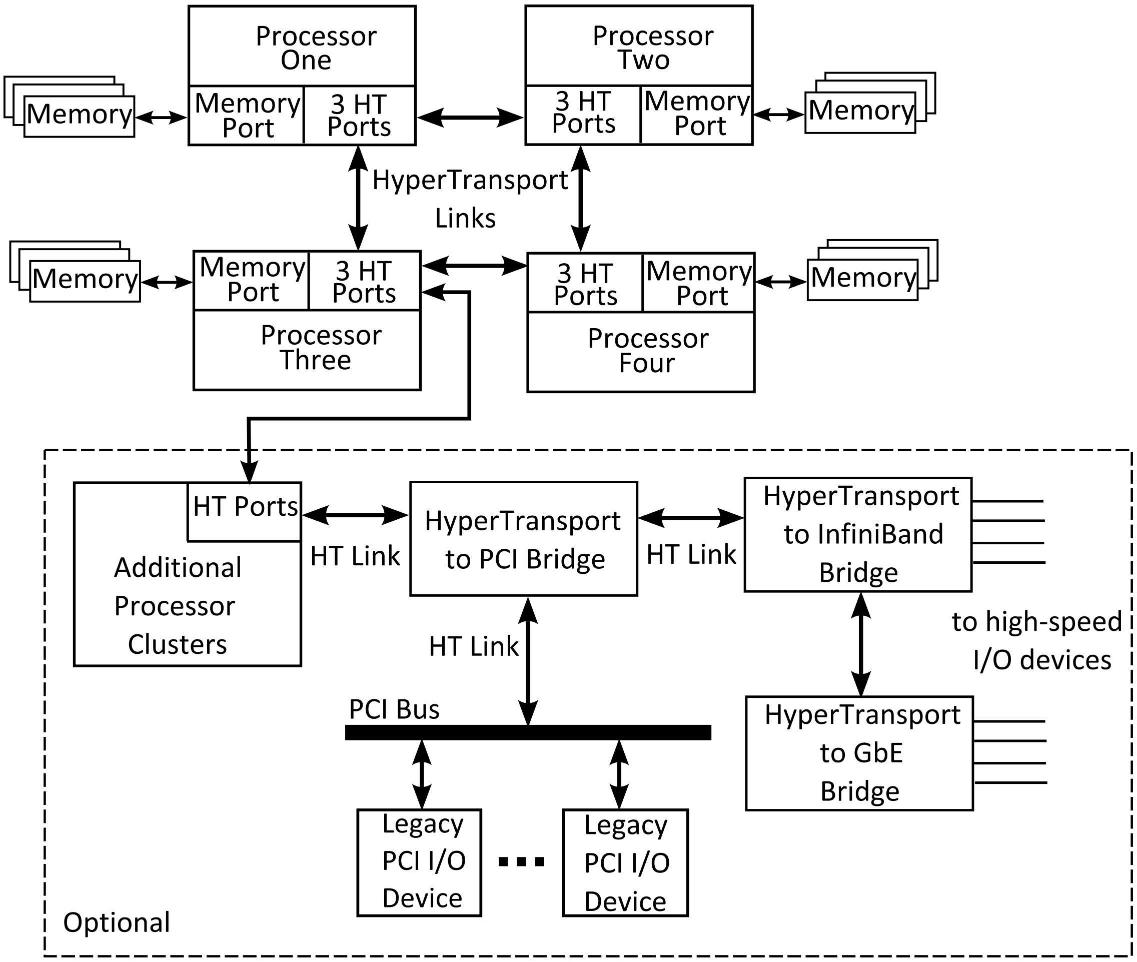

Figure 1.6: HyperTransport Interconnect

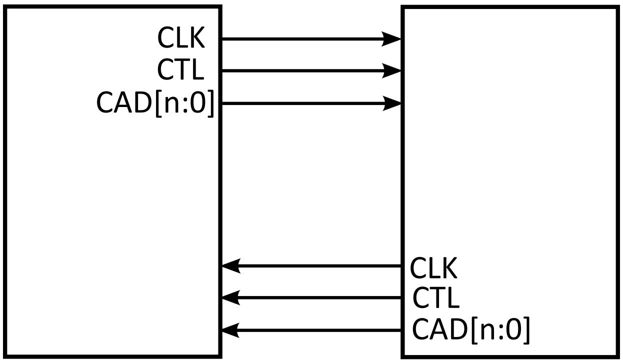

Figure 1.7: Physical Layer of HyperTransport Interconnect

considered in double data rate memory group as there is a separate WCK clock carrying the data at both edges. If we compare the GDDR5 transfer rate (6 GT/s) to the system clock (1500 MHz) then it is represented as a quad data rate memory. Extreme data rate dynamic random-access memory (XDR) is a proprietary memory family from Rambus [32] which sends 8 (XDR) and 32 (XDR2) data respectively on each system clock, hence providing highest bandwidth (57.6 GB/s and 160 GB/s) among available memory chips. Due to the high cost, XDR memories are mainly used in niche applications e.g. game consoles.

In summary, to achieve high bandwidth, high-speed interfaces have used point-to-point connections. Some interfaces e.g. HyperTransport, Quickpath interconnect, XDR have used differential signaling to reduce noise, crosstalk while other interfaces e.g. GDRAM have used single-ended (SE) signaling due to limited pin constrains. Both types of interfaces use some kind of interface trainings in the initialization process to make sure correct signal transmission and use programmable on chip termination to reduce the reflection problem.

Table 1.5: Source-Synchronous Memories

Bus Size	**No. of chips**	**Memory type**	**Memory clock**	**Transfers /s**	**Transfer rate**
64 bits	4	SDR	133 MHz	0.13 GT/s	0.53 GB/s
64 bits	4	GDDR	350 MHz	0.7 GT/s	5.6 GB/s
64 bits	4	GDDR2	600 MHz	1.2 GT/s	9.6 GB/s
64 bits	4	GDDR3	1400 MHz	2.8 GT/s	22.4 GB/s
64 bits	2	GDDR5	1750 MHz	7 GT/s	56 GB/s
64 bits	4	XDR	900 MHz	7.2 GT/s	57.6 GB/s
64 bits	2	XDR2	625 MHz	20 GT/s	160 GB/s

1.3 Source-synchronous I/O links: challenges

1.3.1 Unwanted Board Skew

Source-synchronous system solves the problem of flight delay between transmitter and receiver, but there are skew problems which limit the speed of the system. DDR SDRAM also uses separate bidirectional DDR strobe (DQS) to latch the data properly. The skew contribution of the system can be divided into three main components: Transmitter (Tx) skew, Board skew and Receiver (Rx) skew as shown in Figure 1.8. In general, Tx and Rx skew numbers are provided by the transmitter and receiver chip vendors. It is the board skew which creates main deviation between the data and strobe timing at the receiver as the skew is dependent on noise, crosstalk, ISI, Ci mismatch, termination mismatch, trace-length mismatch and Process, Voltage and Temperature (PVT) variations.

1.3.1.1 Supply (Vref) noise / SSO noise

In high-speed data communication, due to the non-ideal power distribution network, simultaneous I/O switching generates deterministic supply noise. Considering the worst case slew rate and $\pm 50mV$ Vref noise, strobe to data skew can be as large as 200ps [33]. There are mainly two techniques which can minimize the supply noise. The one approach

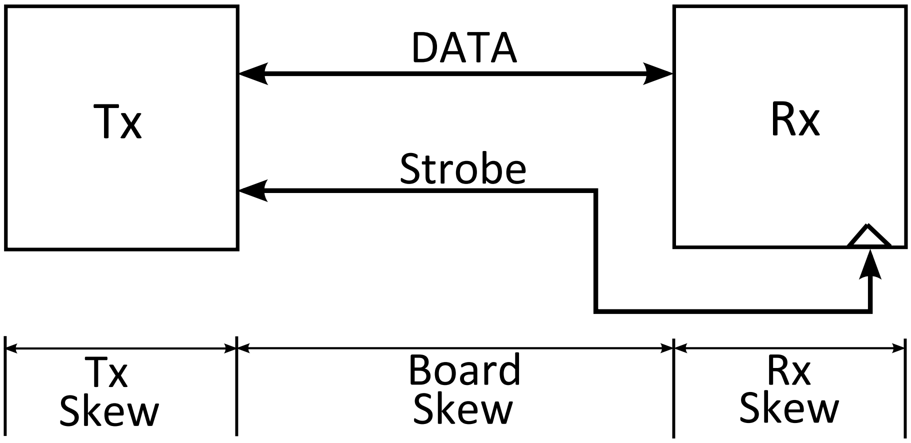

Figure 1.8: Source-Synchronous Skew Distribution [33]

is based on using the on-chip bypass capacitors [34]. The other approach is to reduce the switching frequency by encoding data bits. For example, GDDR5 systems [35] use Data Bit Inversion (DBI) pin which takes advantage of Pseudo Open Drain Logic (PODL) termination where Direct current (DC) is only drawn when zero logic level is transmitted. These techniques cannot completely eliminate the switching noise [36]. An advanced data encoding technique or I/O signaling can be employed to further decrease SSO.

1.3.1.2 Crosstalk

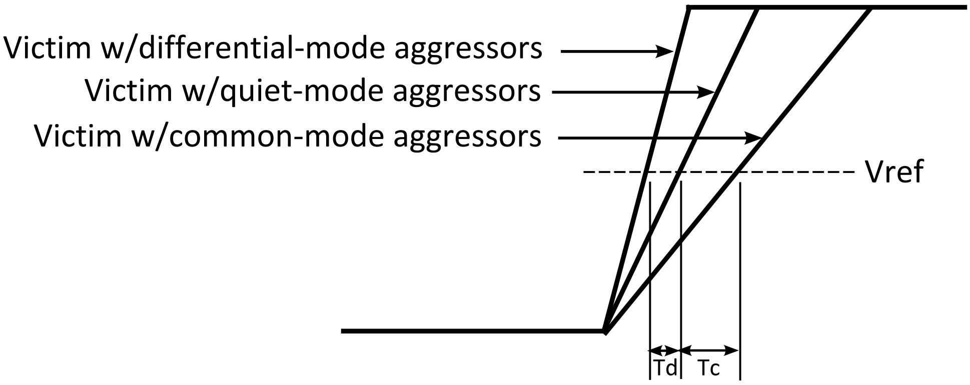
Victim w/differential-mode aggressors
Victim w/quiet-mode aggressors
Victim w/common-mode aggressors
Vref
Td
Tc

Figure 1.9: Crosstalk [33]

Crosstalk [37], [38], [39] is an another limiting factor for high-speed data communication systems. Crosstalk happens when switching behavior of the observed signal (victim) is affected by the switching activity of neighboring signals (aggressors) which results in unwanted skew. Crosstalk is measured in three modes : Differential Mode where the victim and the aggressor have the different polarity; Common Mode where the victim and the aggressor have the same polarity and Quite Mode where the aggressor is not active which can be taken as a reference point to measure the skew Td and Tc caused by differential and common modes (Figure 1.9) respectively. Crosstalk is mainly caused in the package and PCB routing. The package needs to be designed specifically to minimize crosstalk and supply noise by using adequate signal-to-power ratio for both Ball Grid

Array (BGA) package and PCB via patterns. For the PCB routing, stripline layers are usually preferred compared to the micro-strip for high speed routing to minimize the far-end crosstalk. To reduce the near-end crosstalk, signals switching on the same clock should be routed together. However, due to the large number of signals, using only stripline wires becomes very difficult with the conventional PCB technology. The line-to-line spacing within the PCB needs to be chosen carefully in order to have minimal impact of crosstalk to achieve high data-rates. Strobe signals should be isolated to reduce the skew between strobe and data.

1.3.1.3 Inter-Symbol-Interference

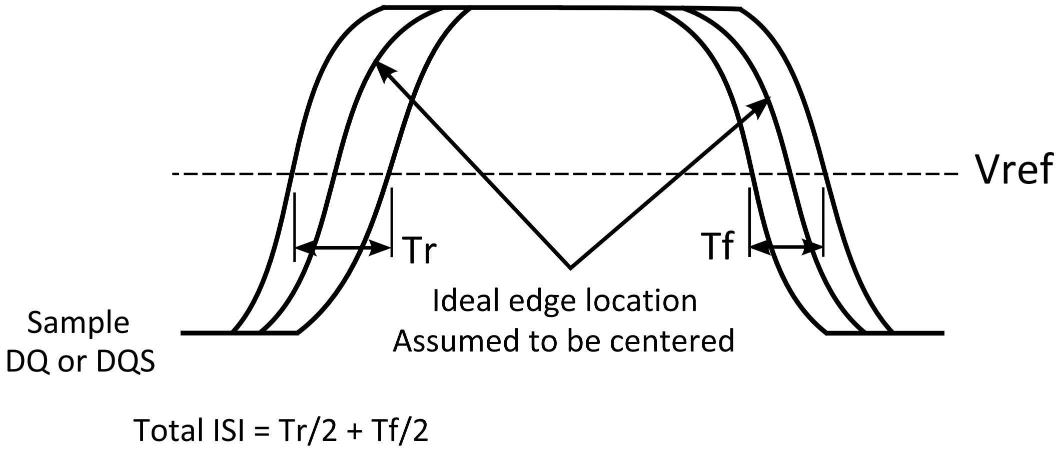

Figure 1.10: ISI [33]

ISI [40], [41] refers to the symbol interference caused by switching while the signal representing one data symbol is not settled properly, which may further lead to wrong interpretation of the data at the receiver side. It reduces eye-opening at high data rate significantly. As it affects both rising and falling edges, total ISI is given by Tr/2+ Tf/2 as shown in Figure 1.10. Impedance matching and equalization techniques are two main solutions to overcome ISI problem. A compact voltage-mode equalizer with impedance calibration was proposed in [42], which can be utilized in write transactions. A very fast and efficient 1-tap Decision Feedback Equalizer (DFE) was proposed in [43] that can be used in the read transactions in conjunction with a linear equalizer. Furthermore, input capacitance (Ci) of I/O drivers should be kept low to reduce the reflections on the pad while a large number of segments are needed for impedance matching which in-turn increases Ci. A compromise has to be made between the number of segments and Ci value.

1.3.1.4 Ci Mismatch

Another source of skew comes from Ci mismatch (Figure 1.11) at different pins. It can be measured by using Min/Max of Ci given in vendor datasheets.

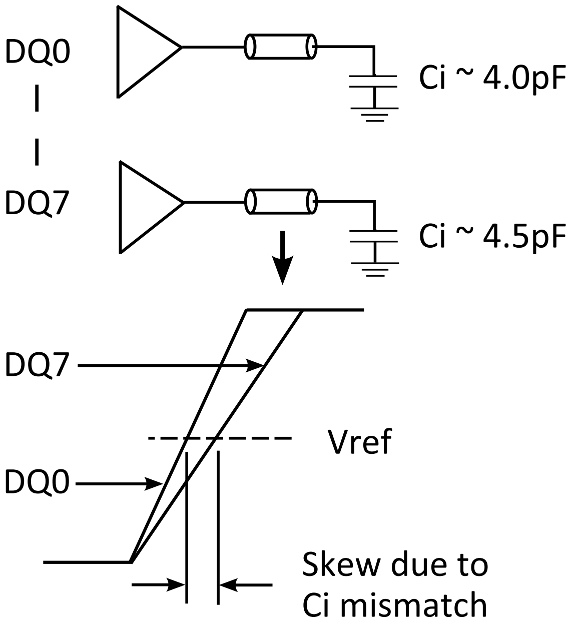

Figure 1.11: Ci-Mismatch [33]

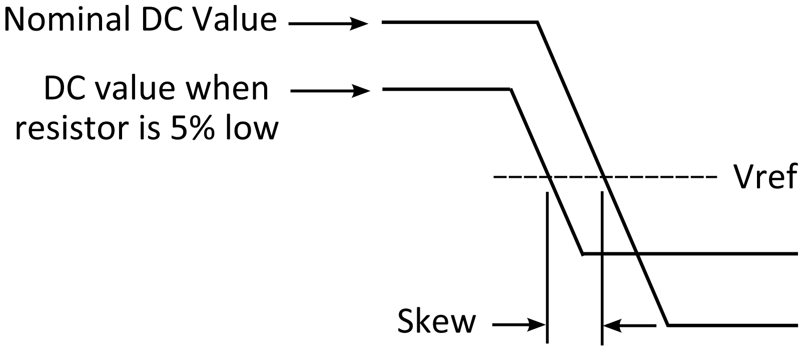

Figure 1.12: Termination-Mismatch [33]

1.3.1.5 Termination Mismatch

Termination mismatch occurs due to the PVT variation effect on the termination resistance. It can be measured by simulating time required for the rise/fall time between DC value and Vref level at different tolerance point of the termination resistor as shown in Figure 1.12.

1.3.1.6 Trace Length Mismatch

PCB routing can be different for parallel links which results in trace length mismatch. In case of point-to-point short connection (5 cm or below), serpentine trace patterns can be used to change the trace length. Serpentine traces can also cause additional skew due to self coupling. Therefore, the distance between serpentine traces has to be carefully decided. The change in the PCB dielectric constant and the trace impedance can also change the propagation delay [33].

1.3.2 Clock Skew and Duty Cycle

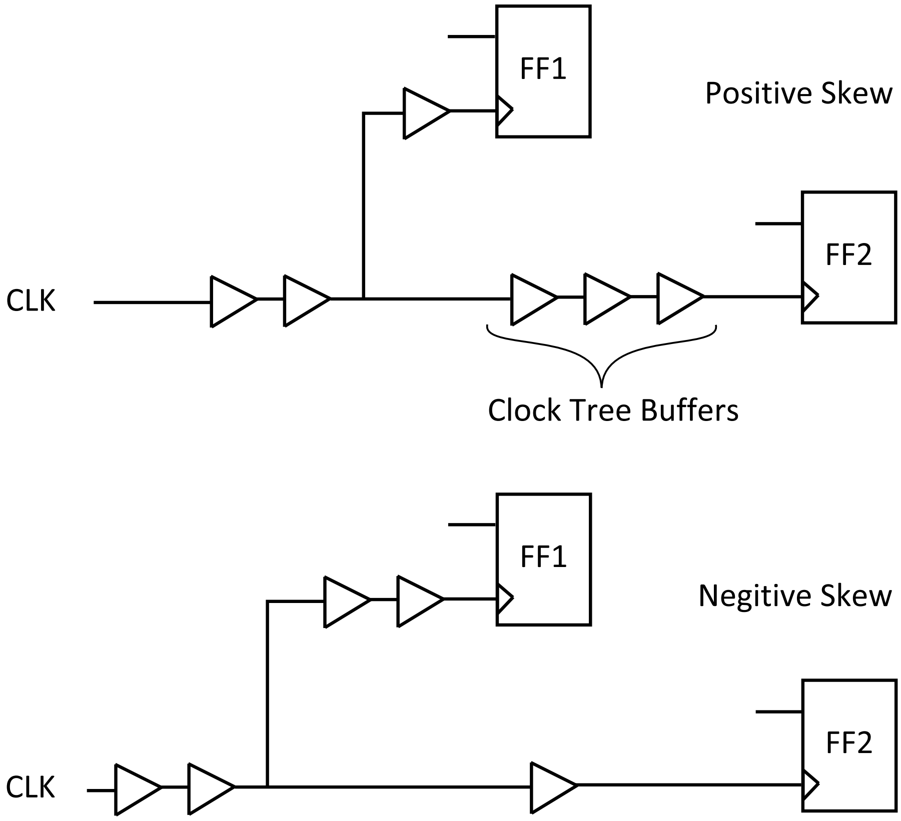

Figure 1.13: Clock Skew

In source-synchronous systems many parallel data bits have to be synchronized with the clock. In other words, the clock will have a large number of end-points (sinks) so the clock tree has to be optimized to achieve low clock skew. Figure 1.13 shows that improper clock buffering can also add additional skew in any direction (positive or negative). As the demand for higher performance increases and the technology scales down, the clock-tree synthesis has to be performed at higher frequencies and under larger process variations while consuming the least possible amount of energy [44], [45]. Going to higher frequencies lowers the acceptable levels of clock-skew (3 - 4% of the clock period) and slew-rate (10% of the clock period) significantly so the impact of process variations on these parameters needs to be analyzed properly.

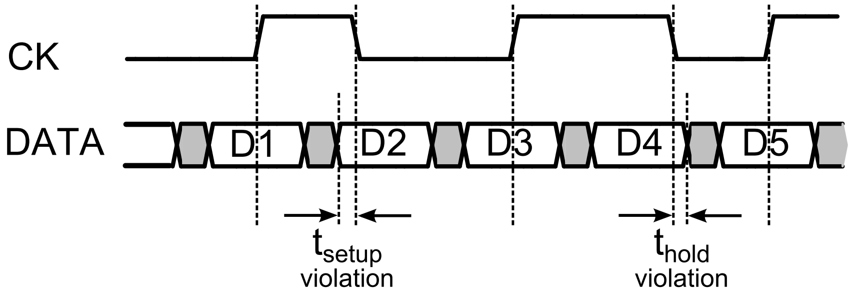

Figure 1.14: Duty Cycle Error

In DDR SDRAM systems, 50% duty cycle of the signal is also important to achieve same setup and hold margin at both rising and falling edges. Figure 1.14 shows if duty cycle is not 50%, then there would be setup and hold time violations. Therefore, a DCC has also to be investigated which can work at wide frequency range.

1.4 Thesis Overview

To address the challenges involved in high-speed source-synchronous I/O links, this thesis focuses on adaptive interface training concept and its implementation. Chapter 2 first describes the need of adaptive interface training in high-bandwidth source synchronous communication systems and then presents two low power novel architecture: Unit-delay and PI-based PLL architectures [99], [103]. In Chapter 3, a novel buffering algorithm [100] for reliable low power clock tree is presented. It also describes a novel low power wide range duty cycle corrector [101], [104] which provides 50% duty cycle for applications like double data-rate memories. Chapter 4 provides the simulation results achieved in this thesis. Finally, Chapter 5 concludes the thesis and summarizes the key contributions of this thesis along with future directions. Appendix A explains the design methodology [105] used to realize the concepts. Appendix B describes the standard building blocks used for PI-based PLL adaptive training architecture.

Chapter 2

Source-Synchronous I/O Links: Adaptive Training

2.1 Introduction

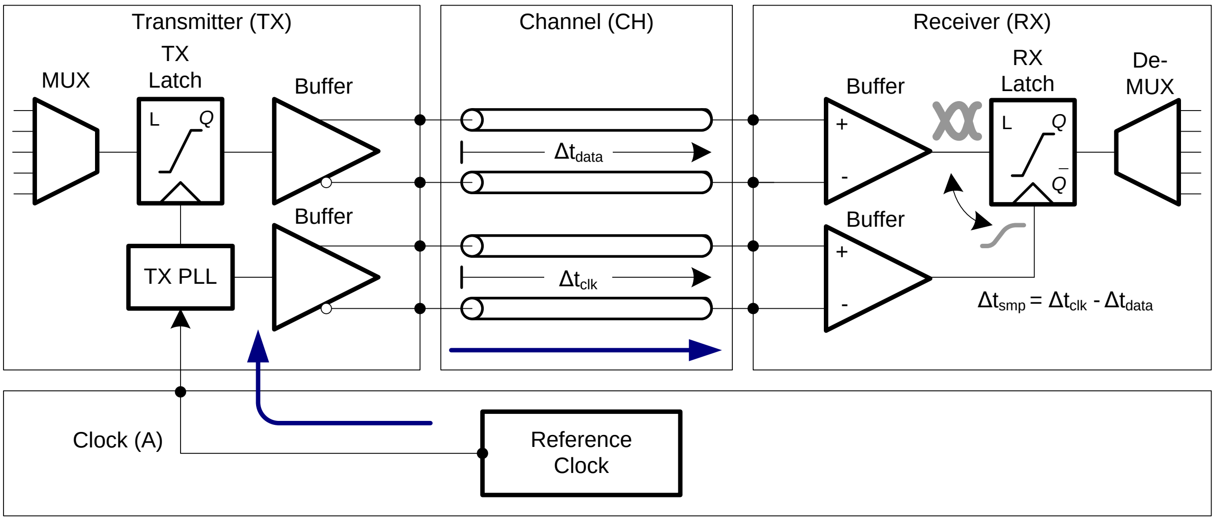

Figure 2.1: Source-Synchronous Clock Forward (SSCF) link concept.

Many high-speed interfaces e.g. QuickPath Interconnect, HyperTransport interconnect, XDR Dynamic Random Access Memory (DRAM), GDDR5 DRAM are using a Source-Synchronous Clock Forward (SSCF) (Figure 2.1) technique to synchronize data and clock by sending in parallel from transmitter to receiver. The Source-Synchronous Clock Forward (SSCF) concept could make synchronization less complex at the receiver side as long as the remaining timing uncertainty Δt_{smp} at the receivers sampling latch is negligible. However, distributed timing recovery concepts as they are used in Gbps data transmission schemes are suffering from trace-length matching errors, skew, termination variations as described in section 1.3. These challenging system conditions are increasing the system cost significantly due to the need of complex synchronization schemes.

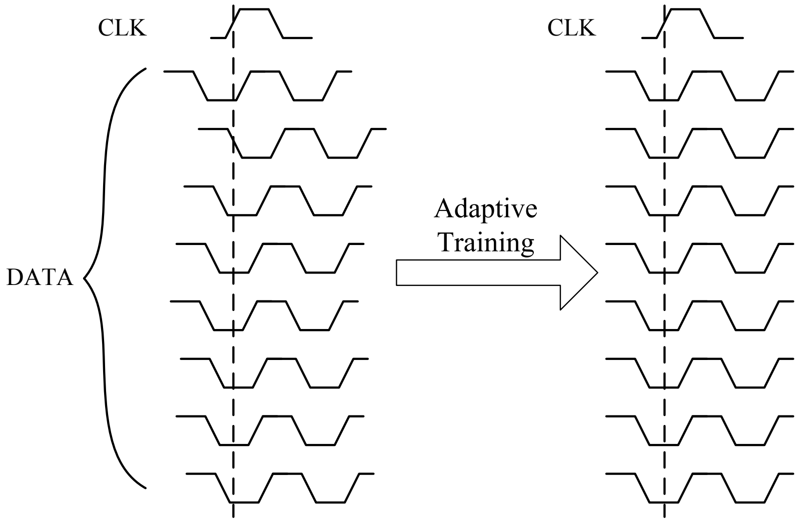

Figure 2.2: Adaptive Training Concept

The current state-of-the-art handles these issues using adaptive training. Adaptive training as shown in Figure 2.2 is used to synchronize the skewed data to the forwarded clock so that the latching of data with respect to clock should be correct and have largest possible setup and hold time which in turn reduces the possibility of errors in the system.

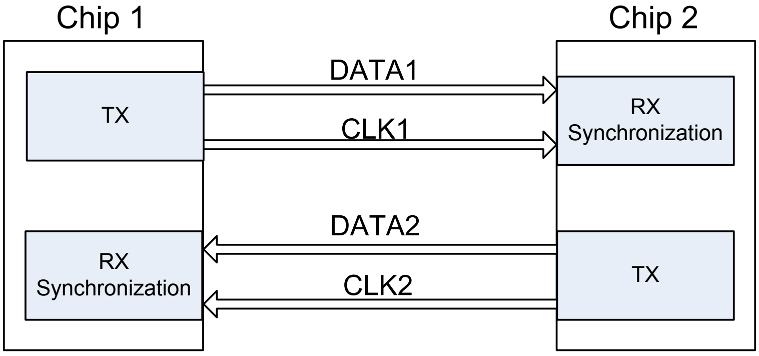

Figure 2.3: High-speed chip-to-chip communication

In high-speed chip-to-chip connection (Figure 2.3), first known data patterns (DATA1, DATA2) are sent along with the clock (CLK1, CLK2) and at the receiver side, received data is compared to the known pattern. In case of mismatch between the received data and known pattern, the clock phase is adjusted at the receiver side using concept similar to conventional analog CDR (Figure 1.1). In this case, forwarded clocks (CLK1, CLK2) are used to generate different phases of the clock at the receiver so that the receiver synchronization circuitry can select the correct phase of the clock based on the phase detector to sample the DATA (DATA1, DATA2). Once training is complete, system can start its normal operation.

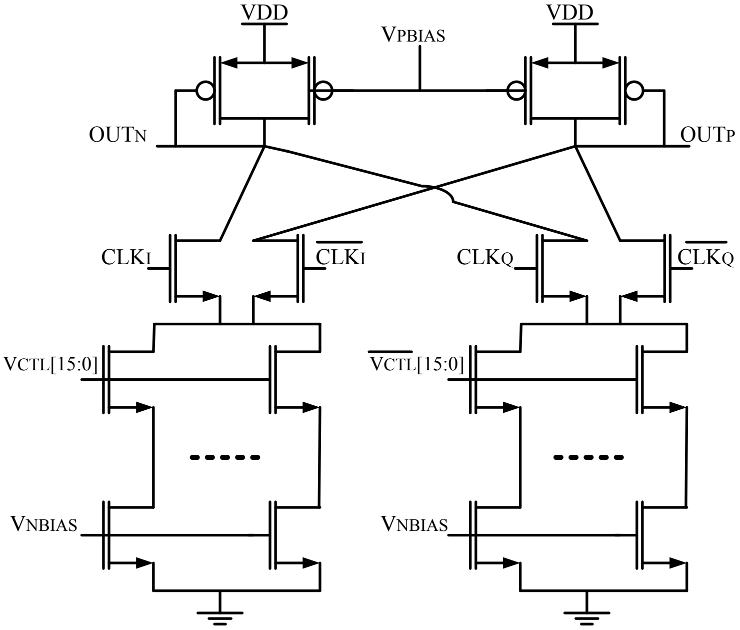

Figure 2.4: PI example [46]

In general, analog CDR [47, 48, 46] comprise PI based techniques which consumes considerable area and power. Figure 2.4 shows a conventional analog PI [46]. The output of the PI is a weighted sum of input clocks CLK_I and CLK_Q. V_{CTL}[15:0] are the thermometer code to control the PI phase-resolution (phase-steps) . As soon as CLK frequency increases, fine phase resolution is needed to get the proper data-eye. To achieve such fine phase resolution, large number of V_{CTL} bits are required, which increase the PI area and consume power exponentially.

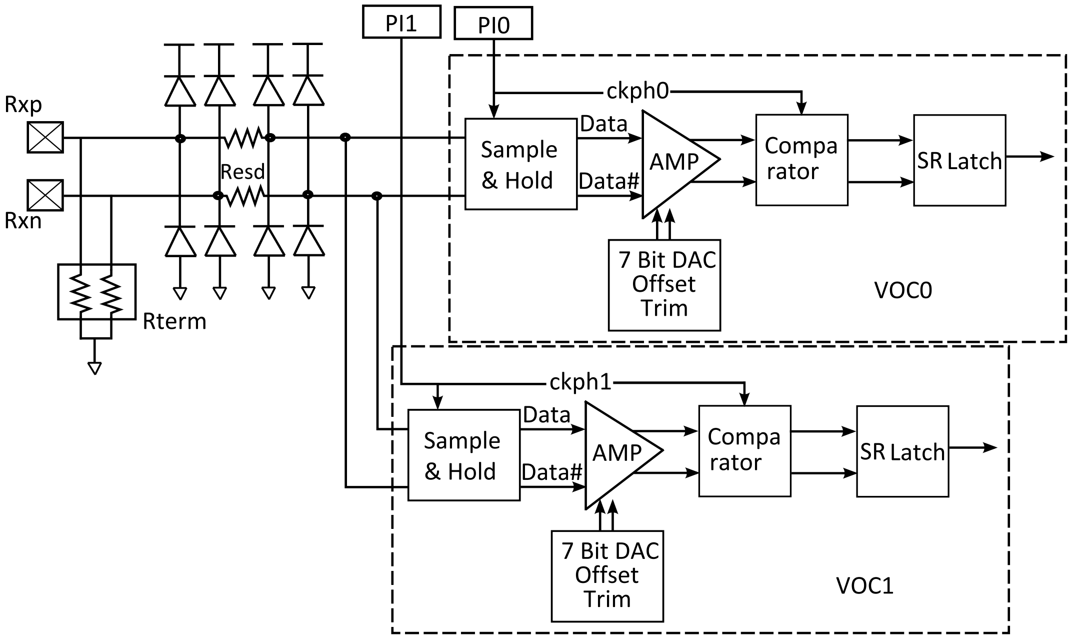

Figure 2.5: QuickPath Receiver with two PIs [49]

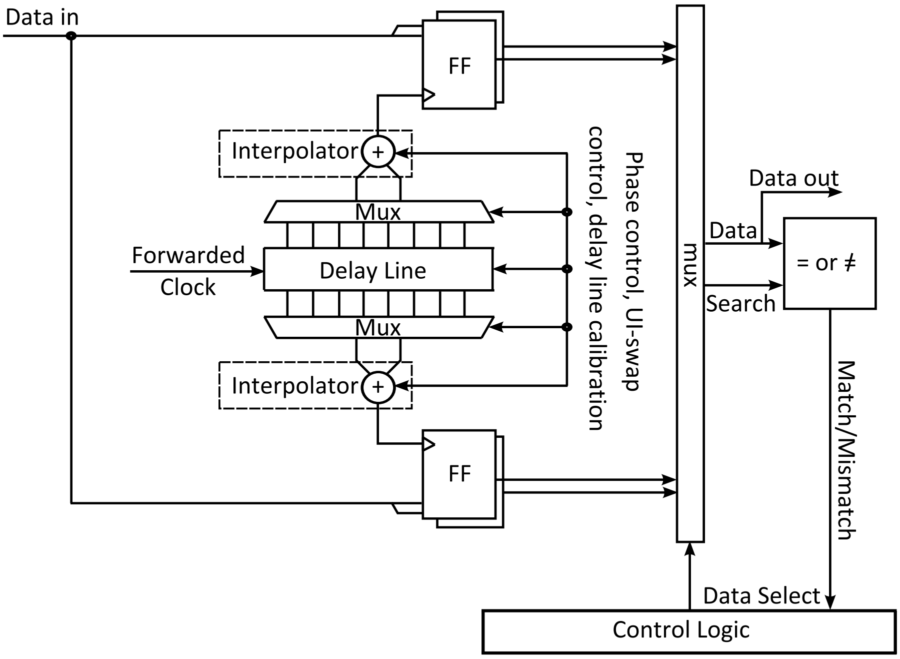

Figure 2.6: Digital CDR with two PIs [50]

Intel's QuickPath Interconnect [49] uses two PIs for each lane at receiver side (Figure

2.5) to latch the data properly. The clock outputs from PI0 and PI1 samples the odd and even data respectively at each falling edge. It eliminates the need of DCC circuit for incoming data and PI output. The PI training is used to find the center of the received data.

Loh and Emami-Neyestanak [50] presented all digital CDR (Figure 2.6) and showed that a digital approach can reduce area, power and complexity in comparison to analog circuits. The simplified digital CDR can also help to increase the yield of the system. A simplified calibrated delay line is also used instead of PLL or Delay Locked Loop (DLL) for generating multiple clock phases, but it uses two PIs for center data-eye and one for edge-search.

Dan Oh et. al. [51] has proposed full time calibration approach at high-speed where the timing center drift is tracked periodically. This calibration approach is complex and may take longer time to converge.

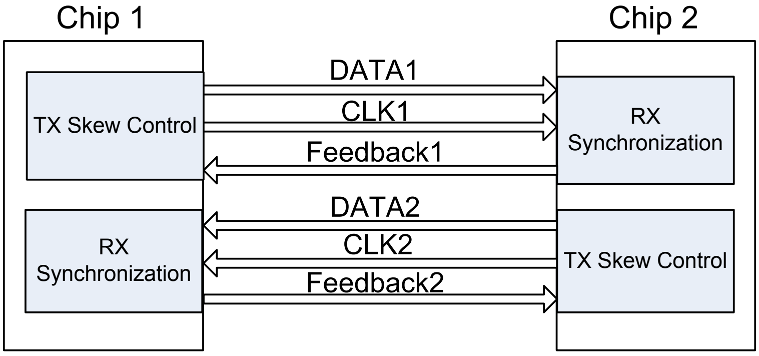

Figure 2.7: High-speed chip-to-chip communication with feedback

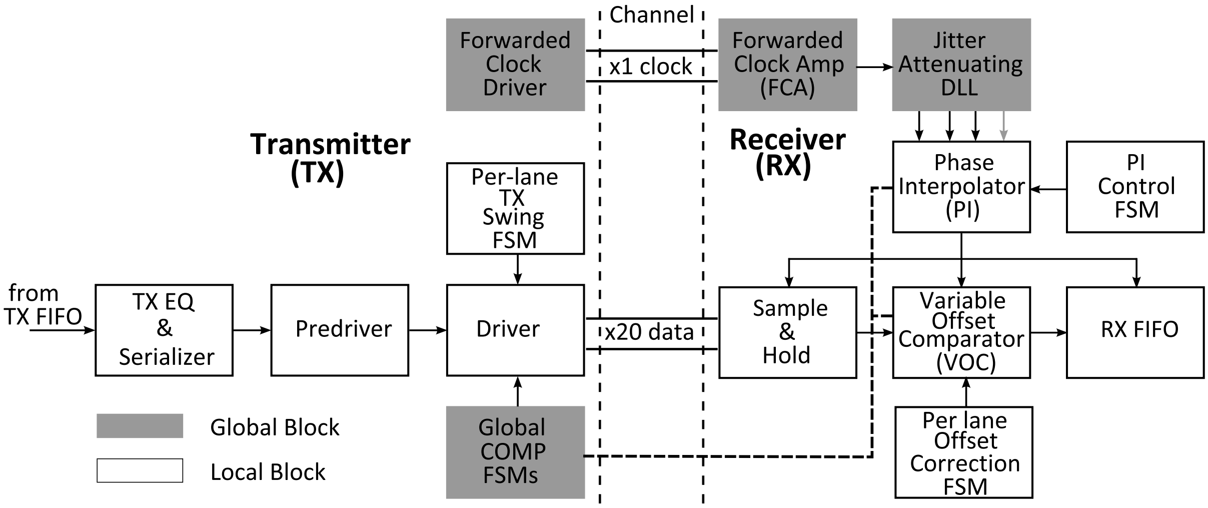

Figure 2.8: Intel QuickPath with Receiver Feedback

Another way to increase the efficiency of the synchronization process in adaptive training is to enable the transmitter chip to control the skew of data or clock based on synchronization feedback from receiver side (Figure 2.7). This kind of feedback is useful to track the changes due to PVT variations. Intel QuickPath interconnect [49] uses such feedback mechanism to track the voltage and temperature drifts and control per lane transmitter skew. In Figure 2.8, depending upon the feedback from PI and Variable Offset Comparator (VOC), global compensation Finite State Machine (FSM) controls the transmitter skew.

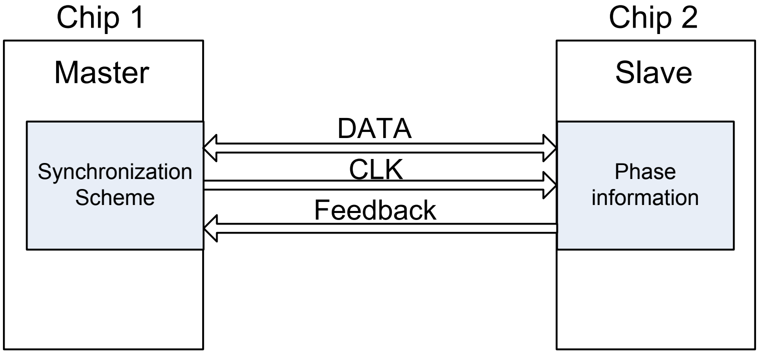

Figure 2.9: High-speed Master-Slave communication

For high-speed master-slave communication (Figure 2.9), master chip (transmitter) takes the full responsibility of data and clock synchronization depending upon feedback (phase information) from slave chip (receiver). Memory systems use this configuration where memory controller (master) communicates with single memory (slave e.g. GDDR5) or memory banks (multiple slaves e.g. DDR3, DDR4). In this case, a lot of area and power can be saved due to simplified slave chip. Even this concept can be applied to any chip-to-chip communication system (Figure 2.10) where transmitter keeps the skew control of the transmitted signal depending upon the phase information at the receiver.

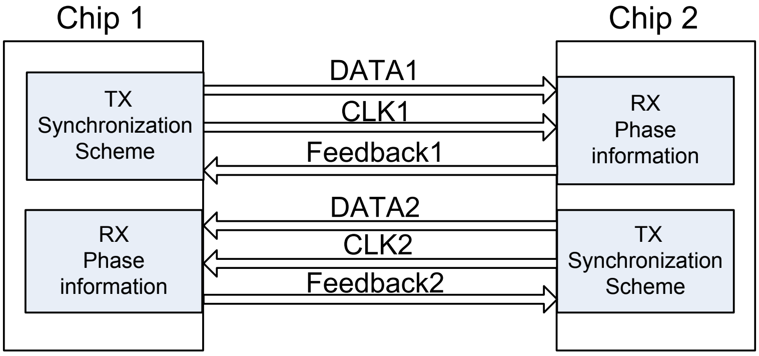

Figure 2.10: High-speed chip-to-chip communication in Master-Slave configuration

Researchers have used the same complex CDR circuits for the adaptive synchronization trainings in all the configurations (Figure 2.3, 2.7, 2.9). In this contribution, with the feedback information from receiver chip, a digital transmitter synchronization scheme has been employed to save considerable area and power. With digital scheme, it is less sensitive to PVT variations, hence increase the system yield.

For simplicity, the example of a GDDR5 system (Figure 2.11) is used which comprises a memory controller (transmitter) connected to a GDDR5 memory (receiver) via channel. From controller side, the channel consists of unidirectional control signals like ADDR/CMD running at CLK and bidirectional data signals like DQ running at WCK.

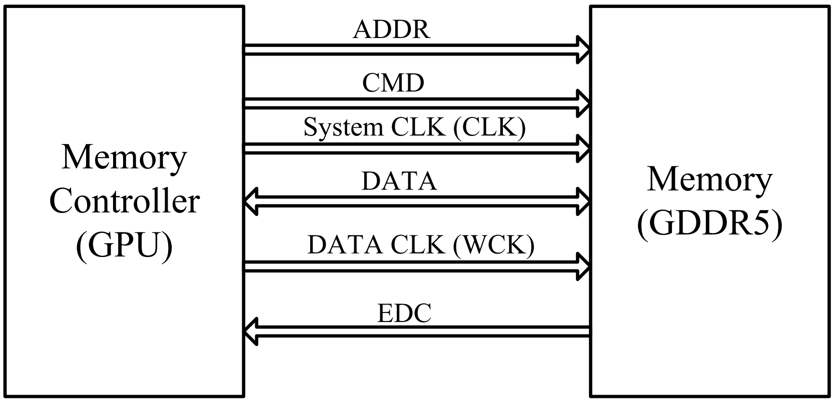

Figure 2.11: GDDR5 Example [35]

WCK is running at double frequency with respect to CLK. The communication system may have many other control signals, e.g. scan enable etc. From memory side, the channel consists of unidirectional EDC, which provides different feedback information depending on different memory mode settings.

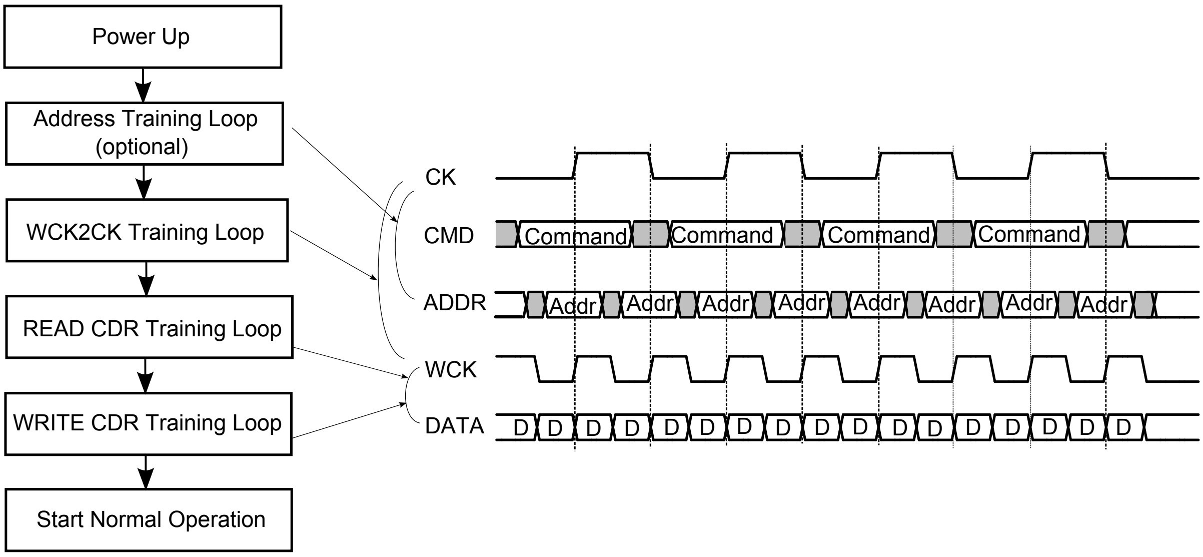

Figure 2.12: GDDR5 training sequence [35]

At the GDDR5 memory power up sequence (Figure 2.12), different trainings e.g. address training, WCK2CK training, read training, write training are required for proper read/write operations at a very high-speed (2 Gbps or more). In GDDR5, ADDR is double data rate i.e. ADDR is send at both positive and negative edges of CLK. ADDR training is required to find the center of ADDR data at CLK edges. It is optional [35] as this can be handled at circuit level. Similarly, read/write clock timing training is required to latch DATA properly with respect to WCK.

The WCK2CK training refers to a defined timing relationship between both clocks which should be within the timing margin determined by the GDDR5 memory specification. During the WCK2CK training, the memory sends back internal phase alignment

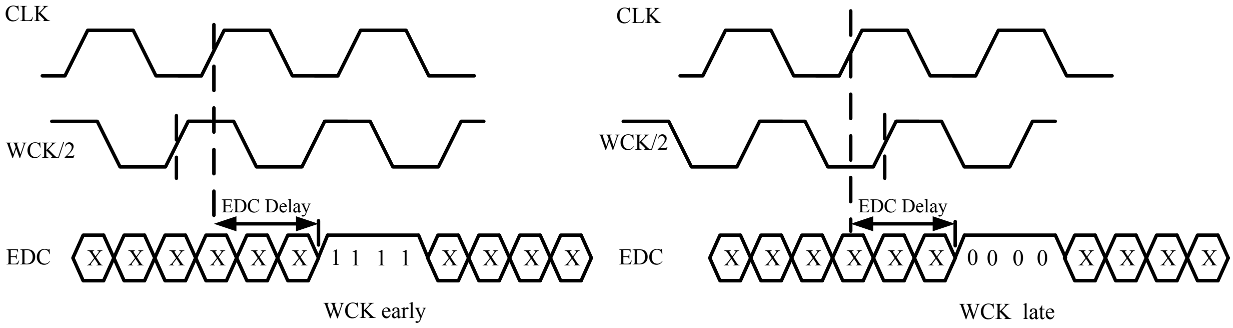

Figure 2.13: EDC functionality during WCK2CK training [35]

information to the memory controller via EDC pin. In Figure 2.13, WCK/2 is sampled at the rising edge of CLK and corresponding result is sent to EDC bits after EDC delay. If WCK is early, then EDC is "1111" and if WCK is late, EDC is "0000". The controller can either increment or decrement the phase of WCK based on EDC information. The described timing recovery scheme is based on a timing error detector at the receiver (an early late detector scheme), the feedback path EDC and the loop control plus phase alignment functionality at the transmitter side.

In this chapter, two novel architectures based on unit-delay incrementer and PI-based PLL for WCK2CK training are described which save considerable area and power. These novel architectures are not specific to GDDR5 and can be applied to any high-speed source-synchronous communication system which uses multiple high-speed clocks, hence needs synchronization among these clocks.

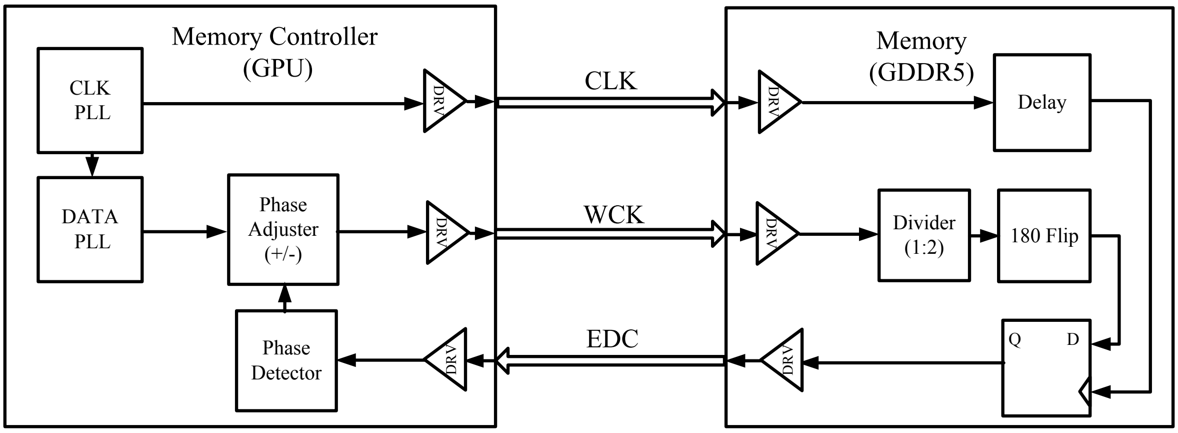

Figure 2.14: Current state-of-the-art [35]

2.2 Current state-of-the-art architecture

In the current state-of-the-art, Figure 2.14 shows the block diagrams how WCK phase is synchronized with CLK in GDDR5 memory system. Due to different speeds of CLK and WCK, a separate DATA PLL is used for WCK path. At memory side, CLK path and WCK path delays are different. CLK path includes driver and compensation delay and WCK

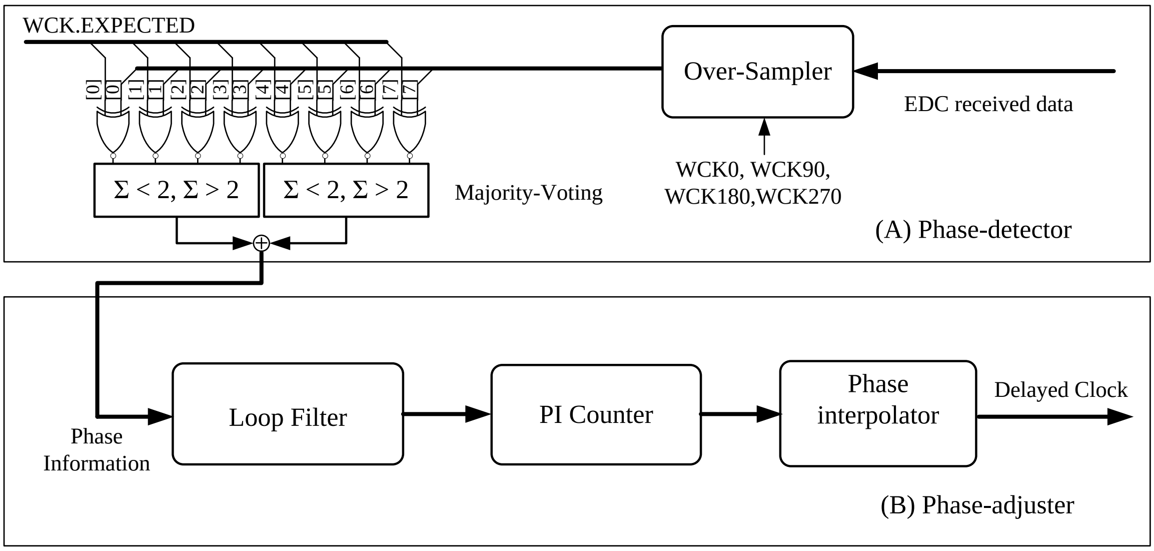

Figure 2.15: Current state-of-the-art: (A) Phase-detector (B) Phase-adjuster

path includes driver, divider and 180 degree flip (to make WCK and WCK/2 polarity same). Compensation delay in CLK path only can not guarantee required timing margin between WCK and CLK at phase detector. Therefore, some control mechanism at the controller side is required to adjust the phase of WCK.

The feedback EDC pins are asynchronous to the controller since there is no clock or strobe signal along with the EDC data from the memory. The controller needs to track the expected time of the EDC data accurately after sending the WCK2CK training commands. Due to PVT variations, a conventional way for the exact information is to over-sample EDC data (Figure 2.15(A)). Oversampling is done with respect to different phases of WCK clock (WCK0, WCK90, WCK180, WCK270) to get multiple samples. After oversampling, majority voting is used to detect phase information from over-sampled EDC data. Loop-filter, PI counter with the special coding for PI control and PI itself (Figure 2.15(B)) are used for phase adjustment. The loop Filter is used to avoid the false result due to the glitch in the majority-voting circuit and the special coding in the PI counter is used for coarse and fine phase shift control of PI to achieve faster and accurate phase adjustment.

2.3 Novel Architectures

To reduce area and power, two novel architectures were presented for multiple clock synchronization. Figure 2.16 shows the first novel architecture based on the unit-delay circuit where shift registers together with comparator work as a phase detector and unit-delay circuit performs the phase adjustment functionality. In GDDR5, WCK is free running, so the timing relation between WCK and CLK can be adjusted in one direction.

At the start of WCK2CK training, EDC is sampled at each positive edge of CLK and stored in shift registers which are 4 bits wide and have a depth of 3. Assuming WCK is early, then EDC is ″1111″, after 3 samples, shift register data would be ″1111 1111 1111″.

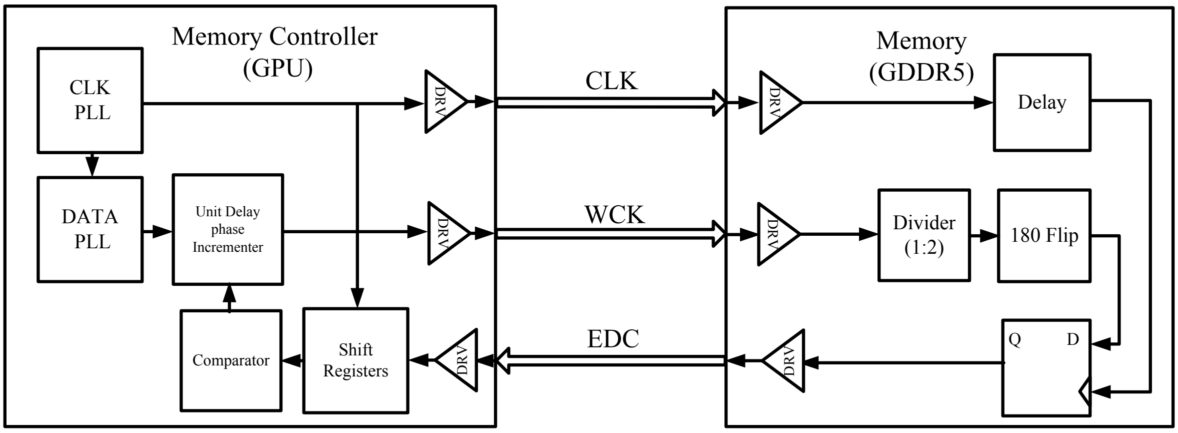

Figure 2.16: Novel Architecture based on Unit-Delay Incrementer

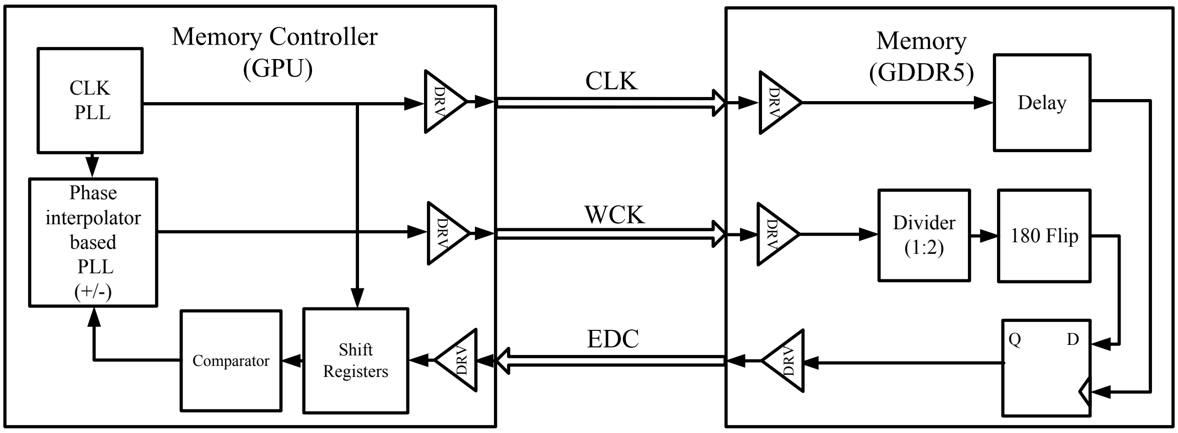

Figure 2.17: Novel Architecture with PI-based PLL

Since EDC data comes from the memory and it has its own latency which is the sum of EDC latency defined in the mode register of the memory, channel delay, other receiving circuit delay. The phase-increment of WCK is done only at these latency intervals. After some increment when WCK becomes late, EDC data changes to "0000" and shift register values become "1111 1111 0000", which means early-to-late transition has happened at the phase detector in the memory and it is the right time to stop the training. A comparator is needed, which compares this pattern "1111 1111 0000" to the stored pattern in shift registers. Once matched, the comparator generates a training stop signal. How closely WCK and CLK are matched, depends on the resolution of the unit-delay circuit, at which it increments the phase of WCK clock. In this novel architecture, the WCK phase is only incremented in one direction until the comparator produces the training stop signal.

In the other novel architecture as shown in Figure 2.17, data PLL can be implemented as a phase interpolated based PLL that can work as a phase adjuster circuit itself instead of a separate unit delay circuit. The system utilizes the available PI in PLL for unit delay increment. The WCK phase can be changed in any one direction ("+ or -") or both directions (" ± ").

2.3.1 Unit-delay phase incrementer

Unit-delay in Figure 2.16 can be implemented by as simple circuits as single transistor load, inverter or Flip-flop and as complex circuits as a DLL or PI. The t_{WCK2CK} timing refers to the delay margin of WCK and CLK which can be handled by the memory phase detector, thereby the delay of unit-delay incrementer circuit should be within the t_{WCK2CK} specification. GDDR5 specification [52] shows that the delay margin between WCK and CK at the phase detector is 0.4^*t_{CK}. Therefore, for the reference frequency (CLK) of 1.5 GHz, the delay margin should be +/- 266 ps. TSMC 65nm low power library [53] has many cells e.g. inverter (delay 15ps), buffer (delay 30ps), delay cell (delay 45ps), D flip-flop (delay 80ps) which can be used as unit-delay incrementer circuit. Figure 2.18 shows 3 custom delay cells designed in TSMC 65nm: voltage controlled buffer, voltage controlled inverter, transmission gate which can achieve granularity of 28ps, 10ps, 3ps delay respectively in typical operating condition.

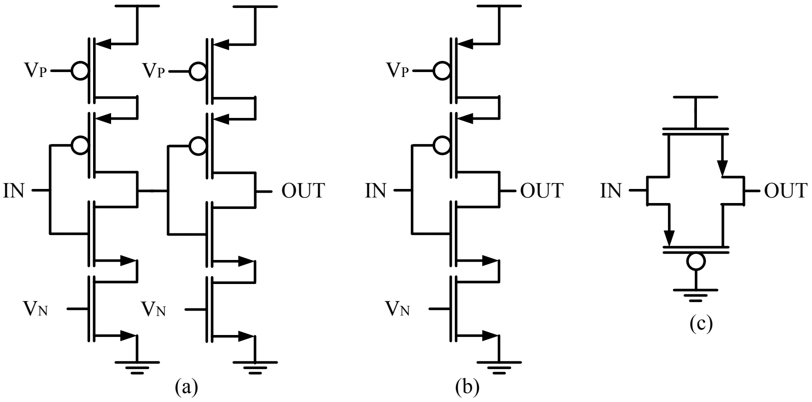

Figure 2.18: Custom Delay Cells in TSMC 65nm: (a) Voltage controlled buffer (28ps typ. delay); (b) Voltage controlled inverter (10ps typ. delay); (c) Transmission gate (3ps typ. delay)

2.3.2 PI-based PLL

Figure 2.17 shows the architecture with PI-based PLL. Many researchers ([54], [55], [56], [57], [58]) have used DLL or PLL based CDR which includes a phase interpolation method for fast data recovery. In [57], the authors used high resolution PI-based PLL to achieve low dithering jitter as well as to achieve precise tracking of frequency modulated input data. Figure 2.19 shows the detailed diagram of PI-based-PLL [57]. PI can be a simple multiplexer controlled by comparator output to choose the one of the phases generated by VCO. As this PI architecture consists digitally controlled circuitry and a multiplexer, it is mostly immune to PVT variations. If the fine resolution is needed, which is not provided by VCO, then an analog PI (Figure 2.4) can be used instead of simple multiplexer. In the PLL feedback loop, divide by 2 is used as WCK is of twice the frequency of CLK. In Figure 2.19, the phase of WCK can be incremented or decremented depending upon comparator output. The comparator would have two fixed patterns ″0000 0000 1111″ to detect late-to-early transition and ″1111 1111 0000″ to detect early-to-late transition.

A low supply voltage and a wide output frequency range make PI-based PLL design

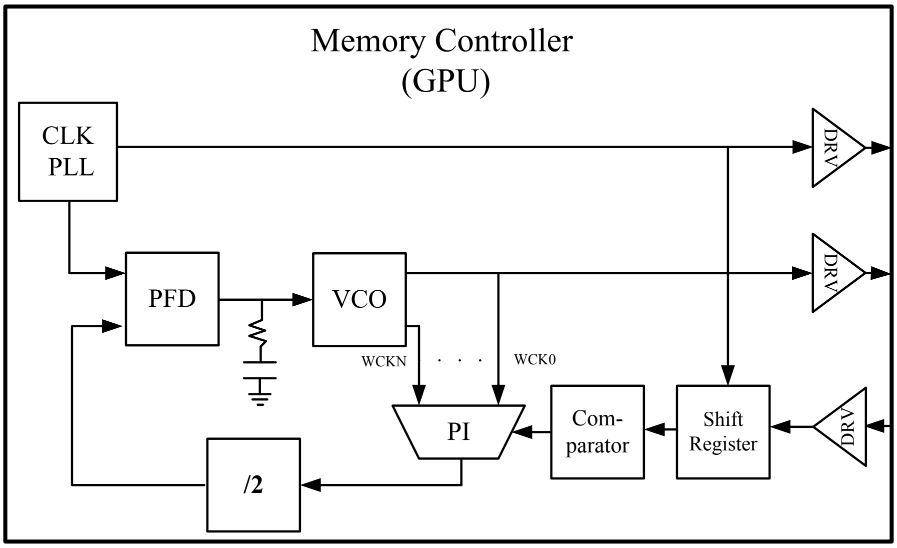

Figure 2.19: PI-based-PLL [57]

challenging. In order to achieve higher frequencies, the VCO gain has to be increased. Another factor to increase the VCO gain is the lower filter output voltage due to low supply voltage. Hence, the gain of the VCO should be as large as feasible with the technology advancement. On the other hand, the increase in VCO gain makes it more sensitive to the noise. To achieve low output noise, the loop bandwidth needs to be as small as possible, which in turn makes the system response too slow. Hence, in a single loop PLL architecture a compromise between responsiveness of the system and output noise is needed. This compromise can be avoided by using DLPLL architecture instead of single loop PLL architecture. One loop with narrow bandwidth and high gain is called coarse-loop and other loop with wide bandwidth and low gain is called fine-loop. The coarse-loop with narrow bandwidth reduces the noise, therefore the gain can be increased. The fine-loop with a wide bandwidth generates a large noise on the output, hence, the gain has to be very small. The DLPLL architecture has advantages of fast lock-in due to the large coarse-loop bandwidth and low output noise due to the narrow fine-loop bandwidth.

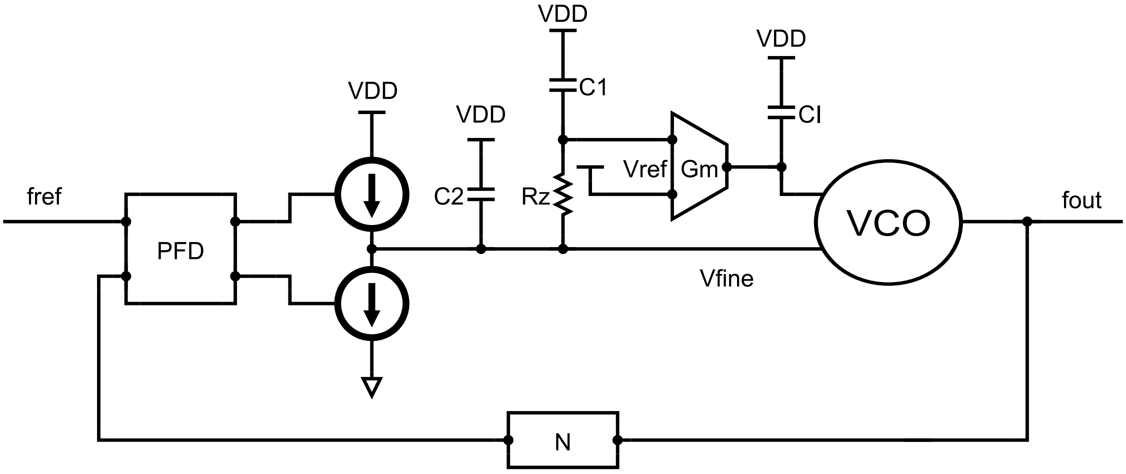

Figure 2.20: DLPLL architecture [59]

Williams et. al. have described the DLPLL architecture [59] as presented in Figure 2.20. The advantages delivered by the DLPLL architecture are a higher output frequency at the lower supply voltage, and, as shown by [60] a lower output jitter than an equivalent single loop topology. It is composed of the same components of a CP PLL but there is an additional integrator and VCO is controlled by two voltages instead of one. This forms two different loops: coarse loop and fine loop, which are correlated.

As described in Figure 2.17, PLL architecture should also include PI for multiple source-synchronous clocks synchronization technique. Hanumolu et. al. has described a CDR architecture utilizing such PI-based PLL [57] shown in Figure 2.21. It employs a $\Delta\Sigma$ modulator to suppress quantization noise by generating extremely small phase steps. A Phase Selector (PS) is used to select phases in the PLL. Basically, the PS is switched at high frequency by the $\Delta\Sigma$ modulator which varies the output frequency until the optimum phase to sample the data is achieved.

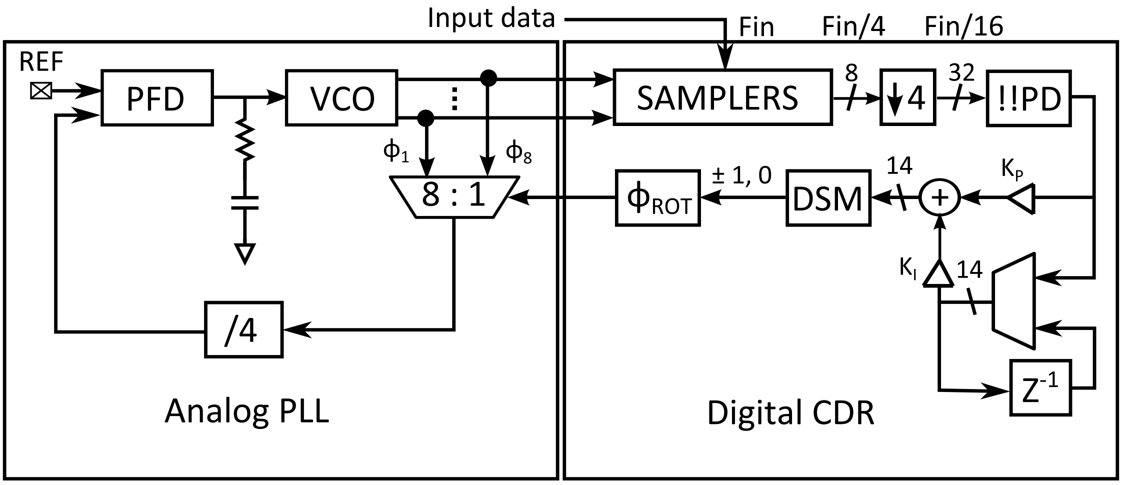

Figure 2.21: PI-based PLL in CDR architecture [57]

The PI-based PLL architecture (Figure 2.22) combines the advantages of PLL architectures shown in Figure 2.20 [59] and Figure 2.21 [57]. VCO is implemented by a combination of Voltage-to-Current (V2I) converters and a Current Controller Oscillator (CCO) which are described in details in Appendix sections 2.2.4 and 2.2.5 respectively.

Coarse loop in Figure 2.22 has the following transfer function:

$$G_C(s) = \frac{K_c}{s}\frac{I_{cp}}{N}\frac{1}{s^2C_2R_zC_1 + s(C_1 + C_2)}\frac{G_m}{sC_I}\frac{\omega_c}{s + \omega_c} \tag{2.1}$$

where K_c is the VCO coarse loop gain, I_{cp} is the CP current, N is the frequency division ratio, C_2, C_1 and R_z1, R_z2, R_z3 are the capacitances and resistors of the filter, as shown in Figure 2.22, G_m is the transconductance of the OTA, C_I is the integrator capacitance and ω_c is the pole frequency of the coarse loop V2I converter. This loop has the function to set the output frequency, therefore a high coarse-loop gain is necessary to cover the output frequency range. To minimize the noise at the VCO input, its bandwidth is designed much smaller than the system bandwidth.

Fine loop in Figure 2.22 has the following transfer function:

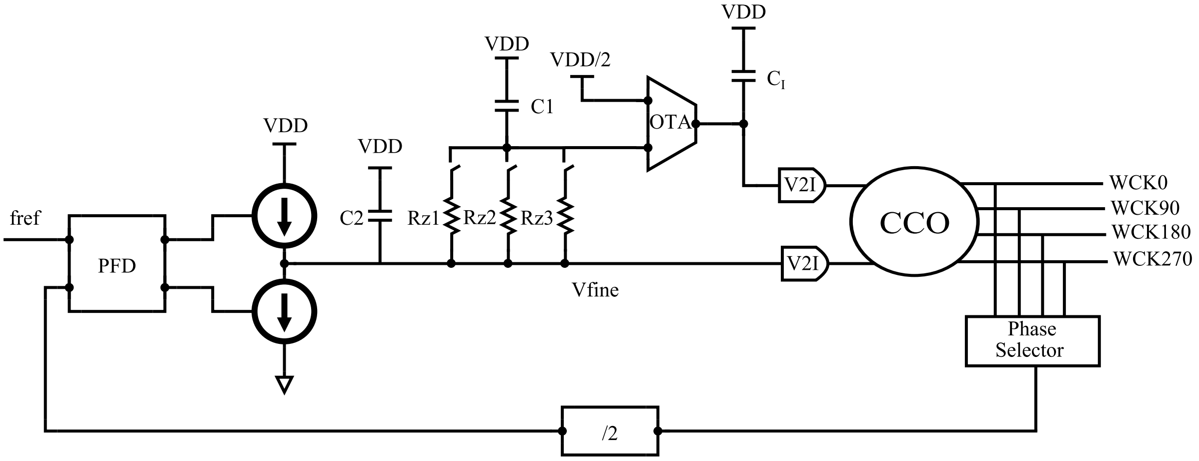

Figure 2.22: PLL architecture utilized in this design.

$$G_F(s) = \frac{K_f}{s}\frac{I_{cp}}{N}\frac{sR_zC_1 + 1}{s^2C_2R_zC_1 + s(C_1 + C_2)}\frac{\omega_f}{s + \omega_f} \tag{2.2}$$

where K_f is the VCO fine loop gain and ω_f is the pole frequency of the fine loop V2I converter. This loop has a bandwidth equal to the system bandwidth. To minimize the noise at the VCO input, the gain is designed very small in this case. The final open loop transfer function is given by the sum of the transfer functions of the coarse loop and fine loop.

In Figure 2.23 the fine loop, the coarse loop and the open loop transfer functions are displayed. The coarse loop gives the larger contribution at lower frequencies while the fine loop affects the high frequency part of the bandwidth of the system. Hence, the fine loop determines the phase margin and the bandwidth of the system. As reported by [59] and [60], the system has to be designed in such a way that the coarse loop does not affect the fine loop bandwidth or phase margin. Therefore, the crossover frequency, i.e. the frequency at which the fine loop becomes dominant on the coarse loop, has to be designed at a frequency much lower than the zero introduced in the fine loop by the loop filter. This can be achieved only by reducing the coarse loop gain. Since the VCO coarse loop gain K_c has to be as large as possible, the ratio between G_m and C_I has to be minimized.

2.3.2.1 Functioning of the system

Figure 2.24 shows the step-response of DLPLL and a first order system which have the pole frequency equal to the PLL bandwidth. The error between them is within 20%. In system simulations, the controller switches the VCO output phase depending upon the system delay. The delays are shown in Table 2.1.

The del_{mem_CLK} path includes transmitter driver, receiver driver, channel delay, custom delay blocks. The del_{mem_WCK} path includes transmitter driver, receiver driver, channel delay and divider. The del_{EDC} path includes EDC delay, transmitter driver, receiver driver, channel delay, shift registers and comparator.

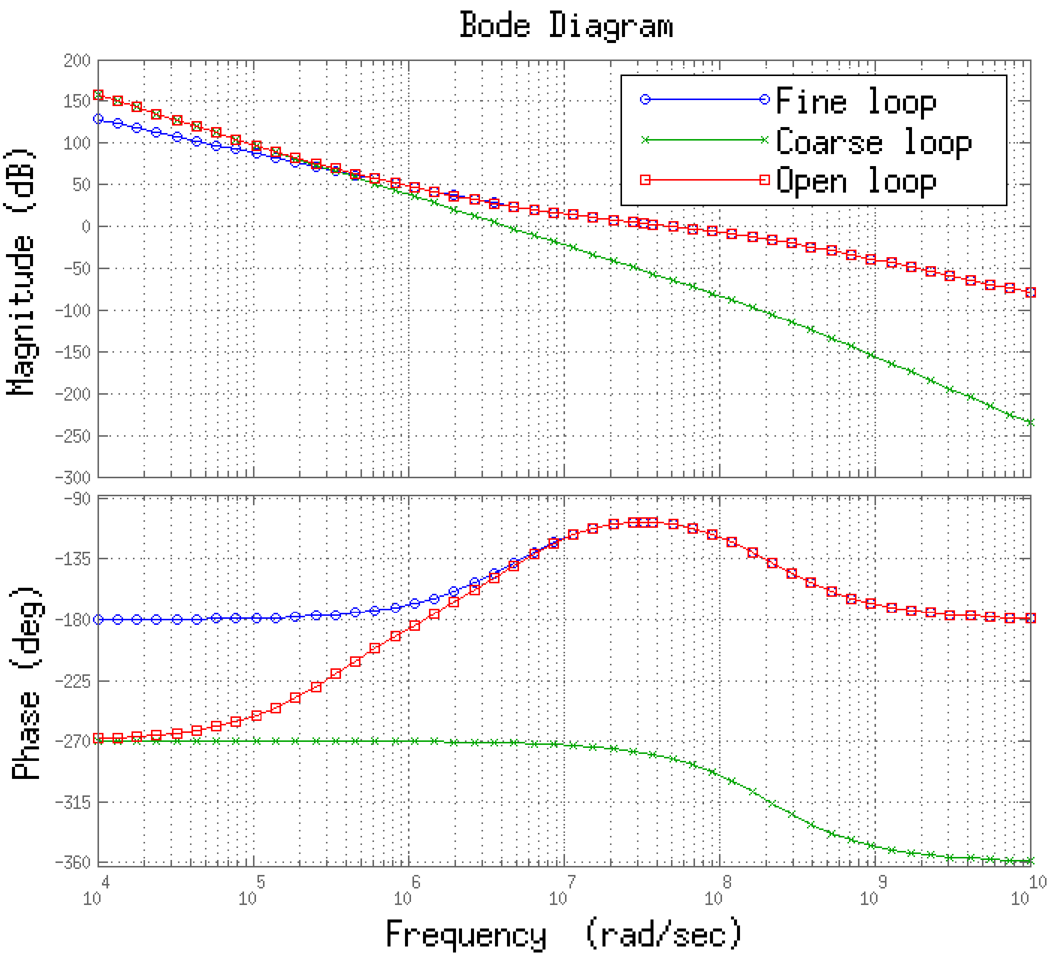

Figure 2.23: Bode-plot of the fine loop, coarse loop and open loop transfer functions.

Table 2.1: Propagation delays inserted the system.

path	delay (ps)
del_{mem_CLK}	4197
del_{mem_WCK}	2730
del_{EDC}	7597

$$del_{TOT} = del_{EDC} + del_{mem\_WCK} + 2 * t_{CLK} \tag{2.3}$$

Equation 2.3 shows the total system delay which is needed to update VCO output phase. Considering the worst case scenario, 2 clock periods are added due to the unknown-phase relationship between the rising edge of *mem_WCK* or EDC and the sampling clock. In case of 1GHz CLK, *del_TOT* will be 12.327 ns. As VCO output phase will be updated in the span of short time, the system response can be approximated with first order system.

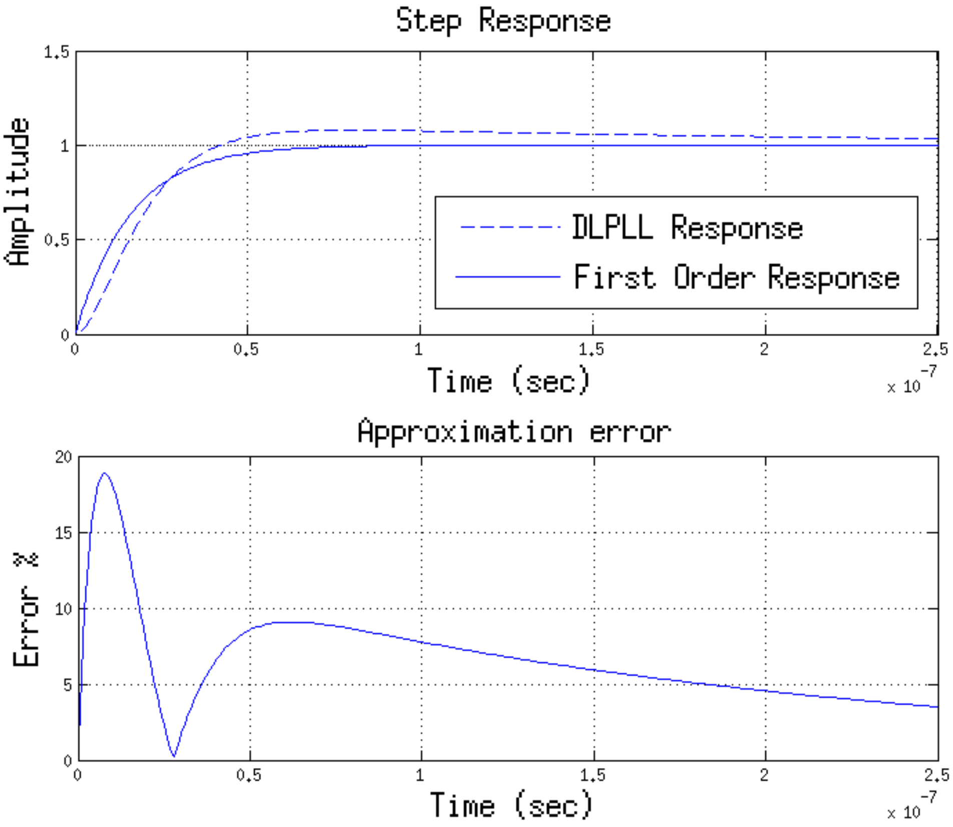

Figure 2.24: step-response.

The amplitude of the first order response after τ seconds is approximately equal to the 63.2 % of the final value of Δ_{phase}. For a 4 phase system, Δ_{phase} is 90° and in case of a 8 phase system, Δ_{phase} is 45°.

$$delay\_rate = \frac{\Delta_{phase} * (1 - e^{-1})}{\tau} \tag{2.4}$$

The time constant of the dominant pole of the system is inversely proportional to the bandwidth of the PLL, which is equal to:

$$\tau = \frac{1}{2\pi f_{BW}} \tag{2.5}$$

The delay rate of the system is given by the phase change during the τ. Therefore, the peak error can be derived by multiplying the delay rate and *del_TOT*.

$$\Delta_{error} = delay\_rate * del_{TOT} \tag{2.6}$$

From Equation 2.6 can be inferred that the slower the response of the PLL, i.e. the smaller the bandwidth of the PLL, the smaller the maximum error. Another improvement can be delivered by increasing the number of phases, since a smaller step causes a smaller delay variation.

Table 2.2: Theoretical worst case error for the different configurations at 2 GHz WCK.

# phases	bandwidth (MHz)	Worst case error (ps)
4	10	61.16
4	6.4	39.14
8	10	30.58
8	6.4	19.57

Table 2.3: Theoretical worst case error of 5GHz WCK configuration.

# phases	bandwidth (MHz)	Worst case error (ps)
4	6.4	11.14

Table 2.2 and Table 2.3 shows the theoretical results based on equations 2.3-2.6.

2.4 Conclusion

In this chapter, two novel architectures based on unit-delay incrementer and PI-based PLL are described for high-speed communication systems consisting of multiple source-synchronous clocks. The unit-delay incrementer circuit can be implemented using the custom delay techniques depending on the technology characteristics. The PI-based PLL architecture can be used for the phase delay control of data clock which avoids the need of extra time required by the unit-delay incrementer based architecture due to uni-directional phase change, but this advantage comes at the expense of addition complexity which leads to more area and power.

Chapter 3

Source-Synchronous I/O Links: Clock Skew and Duty Cycle

3.1 Clock Skew

For high-performance synchronous circuits, well-designed clock-trees that fulfill various constraints (skew, slew-rate, power) are necessary. As the demand for higher performance increases and the technology scales down, the clock-tree synthesis has to be performed at higher frequencies and under larger process variations while consuming the least possible amount of energy. Going to higher frequencies lowers the acceptable levels of clock-skew (3-4% of the clock period) and slew-rate (10% of the clock period) significantly. Therefore, the impact of process variations on these parameters plays a critical role. In source-synchronous systems, many parallel data are transmitted together with the clock and the clock skew directly affects the skew among different data. If clock skew is beyond the given specification, the data skew will also be out of specification. Therefore, a clock-tree design is critical for any high-speed source-synchronous system.

For clock-tree design, the basic question is the choice of topology. While mesh structures are almost exclusively used in industry due to their robustness against process and design variations [61], the recent ISPD 2010 [62] contest has shown that the tree topology has its own advantages. Due to the simpler structure and easier mathematical problem description of the tree topology, it allows good prediction and fine-tuning and as such, it was shown to be more efficient than the mesh topology in the contest.

The construction of a clock-tree generally consists of multiple steps beginning with a topology generation and an initial clock-tree. In the next step, buffers are inserted into the clock-tree. This step is very important, as the buffers control the susceptibility against process variations, but also increase the overall power consumption. While academia extensively studied the clock buffering problem, the industry is seeking fast and scalable solutions that are highly power efficient.

A fast buffering algorithm is presented in this chapter that reduces the number of inserted buffers, while meeting skew and slew specifications. Specific contributions include:

- Estimating susceptibility to process variations as the first step and then buffering the clock-tree accordingly
- Generation of solution options during buffering that have low clock-skew to reduce post-buffering optimization
- High scalability by reducing the run-time complexity of buffering to $O(n)$
- Power reduction of on average 14% with respect to the best known clock-tree for the ISPD 2010 benchmarks.
- Slew-rate dependent delay estimation for higher accuracy

3.1.1 Background

The overall workflow of the clock generation algorithms is shown in figure 3.1. The workflow begins with an algorithm to create a clock topology from a given set of sinks. In

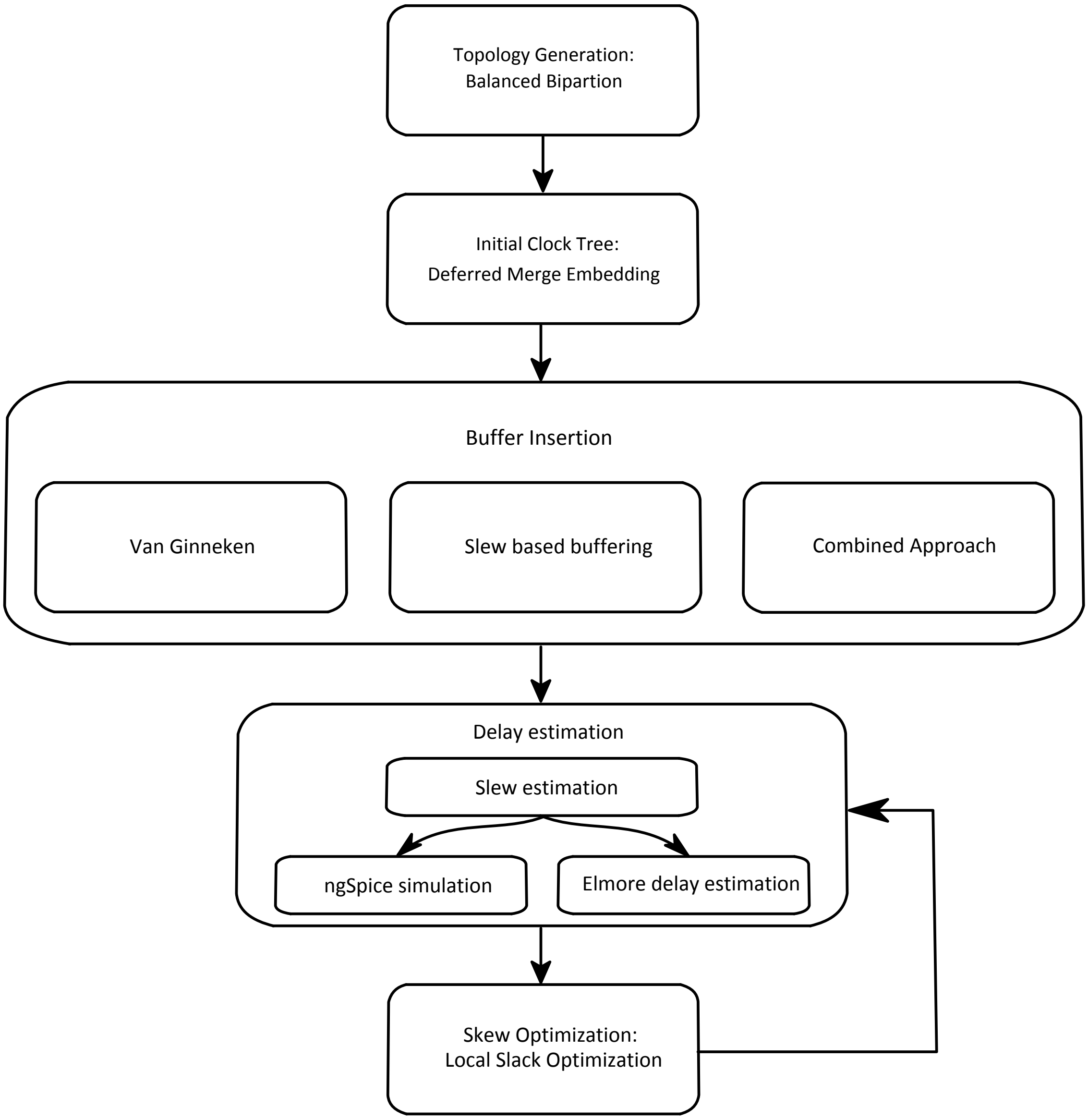

Figure 3.1: Overview of the clock generation

this case, the algorithm for topology generation is the Balanced Bi-partition [63] algorithm. In essence, this algorithm divides a set of sinks into two subsets that have equal capacitance. Of all subsets that satisfy this condition, the solution with the smallest cost (which is defined as the diameter) is chosen.

Then, an initial clock-tree is created by using the Deferred-Merge Embedding (DME) algorithm [63]. This clock-tree is a zero-skew clock-tree. By using a linear delay model, optimum wire-length can be achieved but it is exhaustive. On the other hand, by using the Elmore-delay, a good heuristic solution can be achieved. The algorithm computes the clock-tree in two phases, by first finding all possible placements of nodes in a bottom-up phase and then placing the nodes in a top-down phase.

For the buffer insertion, three algorithms are considered: The Van Ginneken [64] algorithm, the Slew based buffering [65] and the combined buffering approach.

In Van Ginneken's algorithm, a clock-tree of tree topology is buffered for minimum source-to-sink delay in two stages. The *option* with number k at node n is defined as the set $Opt_{k,n} = \{t_{req,n}, C_{sub,n}\}$ of required arrival time $t_{req,n}$ and subtree capacitance $C_{sub,n}$ at node n. The required arrival time at every node is defined such that the clock arrives at the sinks at $t = 0$. Typically, each node has multiple options, representing the different buffering choices. By recursively computing these options in a bottom-up phase and pruning inferior ones, a set of options for the root-node of the clock-tree is computed. From this set of options at the root-node, the algorithm chooses the one that minimizes the required arrival time t_{req} and hence minimizes the source-to-sink delay. In a top-down phase, this option is then implemented at every node. This buffering technique might use abundant amounts of buffers, as the only target is to achieve minimum delay from source to sink, and therefore is not power-efficient. Nonetheless, variations of this technique are effectively employed by researchers on clock-tree synthesis [61, 45, 44].

In the algorithm developed by Hu et. al. [65] the clock-tree is buffered to achieve a given slew-rate specification with minimum buffering. The clock-tree is processed similarly to Van Ginneken's algorithm by bottom-up iteration of nodes and selecting options that satisfy the slew-rate specifications. Finally, the solution with least cost (e.g. number of buffers or area) is implemented.

Accurate estimation of the slew-rate consists of two main steps: Estimating the slew-rate degradation along a wire and the slew-rate at the output of a buffer. To iterate a tree, the options for two sub-trees must also be merged at internal nodes. To guarantee that the worst-case slew-rate is below the specification, the maximum slew-rate of these two options is chosen.

The estimation of slew-rate degradation $t_{slew,w}$ along a wire w is given in equation 3.1 which is based on Bakoglu's metric [66]. It gives an estimate for the increase in slew-rate depending on the Elmore-delay of the wire.

$$t_{slew,w} = ln(9) * t_{elmore,w} \tag{3.1}$$

The slew-rate at the output of a buffer at node v is proportional to the load $C(v)$, a proportionality constant R_b and an intrinsic slew-rate K_b, as given in equation 3.2. Both R_b and K_b are characteristics of a buffer and can be determined through SPICE simulations.

$$t_{slew,b} = C(v) * R_b + K_b \tag{3.2}$$

In the combined buffering approach, both skew and slew are considered in the constrains from the beginning. After buffering, local skew optimization by the means of local clock slack optimization is applied as described in 3.1.2.5, to fine-tune the clock-tree with regard to process-variations and local clock skew. In this phase, wire-snaking and delay buffering are used to balance the clock-tree based on Elmore-delay or Spice simulations.

3.1.2 Power optimized buffering

The target of automated clock-tree generation is the creation of a buffered clock-tree that satisfies the slew-rate and clock-skew specifications with minimum power consumption

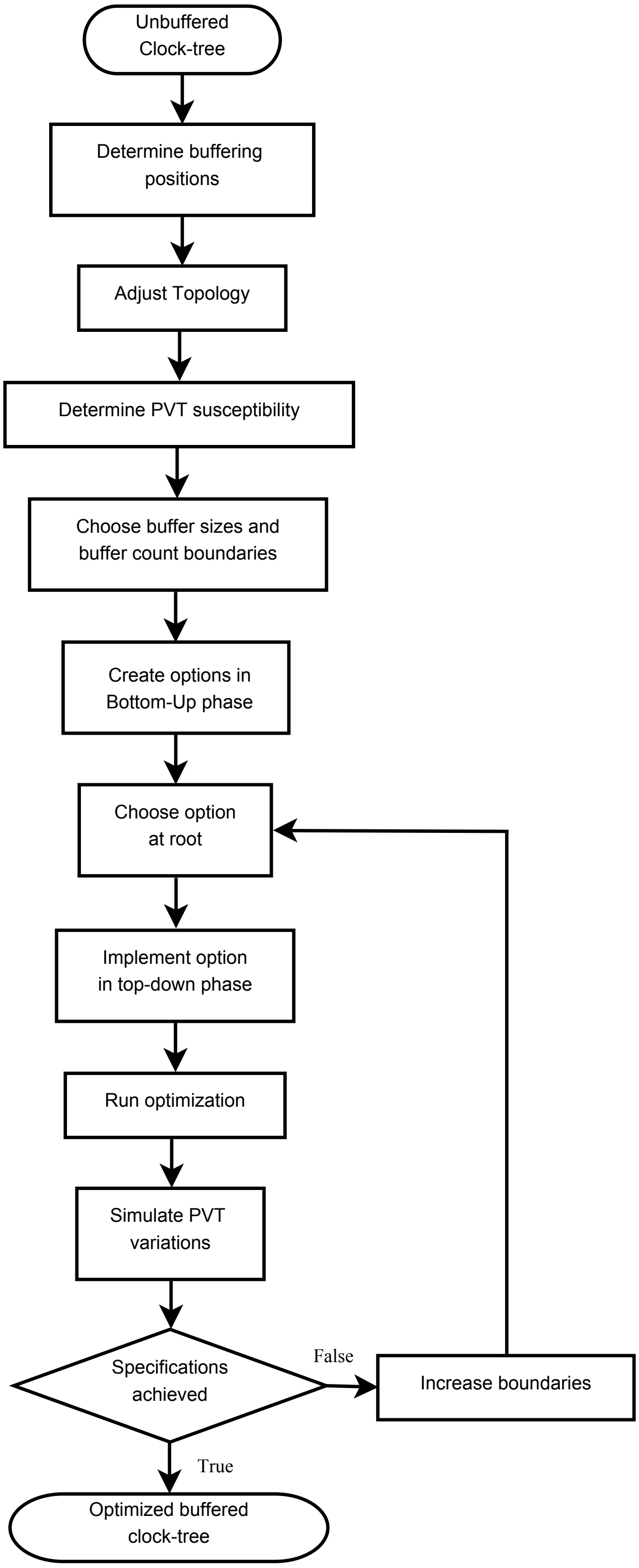
Unbuffered Clock-tree
Determine buffering positions
Adjust Topology
Determine PVT susceptibility
Choose buffer sizes and buffer count boundaries
Create options in Bottom-Up phase
Choose option at root
Implement option in top-down phase
Run optimization
Simulate PVT variations
Specifications achieved
False
Increase boundaries
True
Optimized buffered clock-tree

Figure 3.2: Flowchart

and area usage. In this problem specification, buffers play the main role, as on one hand, buffer sizing and placement can improve timing quality, but on the other hand, each added buffer also increases power and area of a circuit.

The main steps of this buffering approach are shown in Figure 3.2. These are:

- Pre-buffering estimation of the least amount of buffers and adequate buffer sizes based on process knowledge
- Creating and extending possible points of buffer insertion
- Skew oriented buffering algorithm
- Iterative choice of a solution until targets for skew, slew and buffer number are met

3.1.2.1 Preparing the clock-tree

Before the buffering of the clock-tree can begin, suitable buffering positions in the clock-tree have to be determined. In this implementation, placement of buffers in the initial clock-tree is possible at each distance $d_{placement}$. Additionally, each node with two child branches can be buffered at three positions as shown in figure 3.3: Directly at the node, or at the top of the branches. The latter is especially important for clock-skew tuning described in section 3.1.2.5. In that section, wire snaking is one of the methods to change the delay characteristics of a source-to-sink path. To allow wire snaking without affecting other nodes in the clock-tree, the wire segment where wire-snaking is used has to be directly buffered.

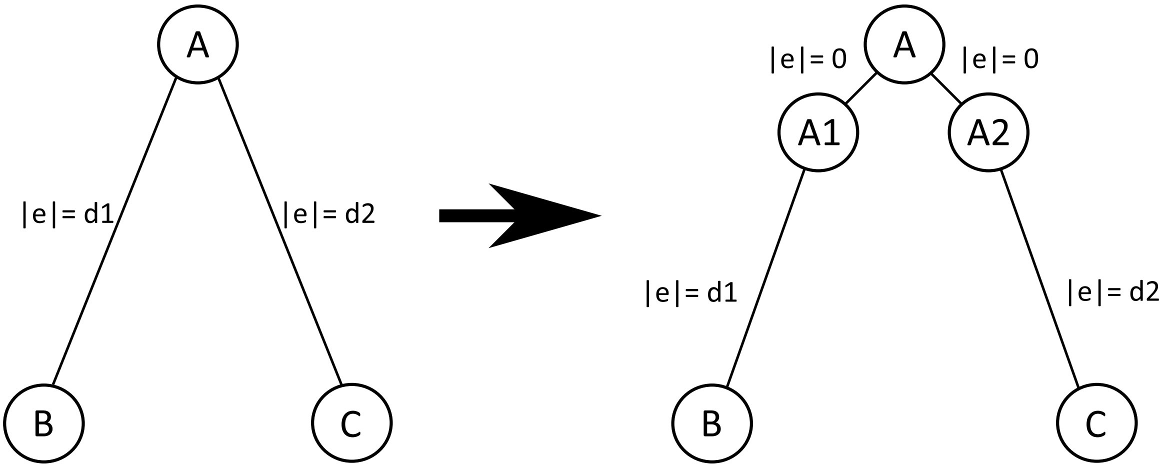

Figure 3.3: Adjustment of the tree topology prior to buffering

3.1.2.2 Pre-Buffering PVT evaluation

The effect of PVT variations can have a severe impact on the clock-skew and slew-rate, as cells and metal layers behave differently than assumed in the clock generation. To be able to handle process variations without a large power overhead, these variations are considered as early as possible in the clock generation flow. This algorithm uses existing knowledge of the given technology and the range of expected process variations to determine how strongly, the clock-tree will be affected by variations.

The estimation has three requirements: Knowledge of the technology, understanding of the circuit, and minimum post-buffering optimization. Knowledge of the technology comes not only from accurate knowledge of process parameters, but has to come through repeated buffering at that technology. In the agile development model, clock buffering has to be performed repeatedly because of the continuous floor plan and placement changes. Therefore, such parameters can be accurately modeled without additional cost. Understanding of the circuit determines the estimated susceptibility to process variations depending on circuit properties such as number of sinks, capacitances, and circuit sizes. For this algorithm, a simple characterization based on Elmore delay estimation after the initial clock-tree is used.

The pre-estimation based on the initial clock-tree directly leads to the requirement of least amount of post-buffering optimization. If pre-estimation is not considered then wire-snaking and similar techniques for optimization will lead to the change in the topology and it deviates the actual circuit properties from those initially expected. Therefore, it is a key achievement that this buffering algorithm requires only little tuning.

Depending on this estimation, possible buffer-sizes and a projected minimum amount of required buffers are chosen. To illustrate this further, benchmarks 4 and 5 from the ISPD 2010 contest [62] are chosen, which are also used for evaluation in section 4.2. Based on fast Elmore delay calculations, it is possible to determine that the initial clock-tree is much more susceptible for benchmark 4. As this benchmark also has a large number of sinks, the technology based minimum buffer estimation was 2200. In benchmark 5, the observed variation of the initial tree is much lower, which lead to a technology based buffer estimation of 1500. With the techniques described in subsection 3.1.2.3 and in section 3.1.3, the buffering algorithm finally leads to a solution with 2340 and 1659 buffers, respectively. In this case, the buffer is constructed with 15 or 33 parallel inverters together based on the buffer strength requirement against variations. These characterizations can be extracted from a mapping generated after a multiple buffering runs.

The algorithm will buffer the clock-tree with at least the specified buffer count. If the minimum buffer limit is chosen too high or the buffer sizes are too large, the power consumption will not be minimized. Choosing very low values for the minimum buffer count will lead to more computation time, as the algorithm will have to process the clock-tree in multiple iterations due to large clock-skew. Despite this minor difficulty, estimating the susceptibility of the clock-tree before buffering has the major advantage that the buffered clock-tree will already have a natural resistance against process variations and will not require excessive tuning in later stages. This directly relates to reduced power consumption and required area. Furthermore, this will reduce the amount of iterations in the post-buffering optimization stage described in subsection 3.1.2.5, which will greatly reduce the computation time.

3.1.2.3 Buffering with clock-skew and slew-rate specifications

Once the appropriate buffer sizes are chosen, the clock-tree can essentially be buffered similar to Van Ginneken's algorithm [64] or Hu's algorithm [65]. In a bottom-up phase, a limited number of viable buffering solutions are determined for each node. These solutions are characterized by low bottom-up skew and delay. Every solution is linked to its child solutions and together, they represent one possible buffering of the whole subtree rooted at that solution. A buffering solution is defined with index k at a node n as

$$Sol_{k,n} = \{C_{subtree}, W, B, t_{slew}, t_{req}, t_{skew}\} \tag{3.3}$$

which reflects the following parameters:

- Sub-tree capacitance $C_{subtree}$ and weight W of the solution (e.g. ratio of power consumption or area to a minimum size buffer)
- Buffer type B (zero-type if no buffer is present)
- Slew-rate t_{slew} and required-arrival time t_{req} at the node when using this option
- Accumulated clock-skew t_{skew}

A solution $Sol_{k,n_{sink}}$ for a sink-node n_{sink} is created from the subtree-capacitance of the sink and initializing clock-skew, slew-rate, weight and buffer-type as zero-values:

$$Sol_{k,n_{sink}} = \{C_{subtree,n_{sink}}, 0, 0, 0, 0, 0\} \tag{3.4}$$

A node is called merging-node $n_{merging}$ if it has two child-nodes $n1$ and $n2$, since these two branches have to be merged in the bottom-up process. Each of the child nodes $n1$ and $n2$ has a set of solutions, $S_{Sol,n1}$ and $S_{Sol,n2}$. The minimum skew $min(\delta_{skew})$ between any two compatible solutions of $S_{Sol,n1}$ and $S_{Sol,n2}$ is calculated by iterating both sets. The implementation uses a maximum $\delta_{skew,acceptable}$ to define an acceptable level of introduced skew at each node. If it is chosen too small, this constant will harm the generated output, since too many options are pruned. A very large value also makes the whole skew consideration redundant and thereby the computation time. In this evaluation, $\delta_{skew,acceptable} = 2 * min(\delta_{skew})$ with a lower bound of $10ps$ performed well. In practice, good values for the respective technology can easily be found by starting with a high value of $\delta_{skew,acceptable}$ and then, iterative reducing it until run-time goals are met.

In an iteration of both sets, all solution pairs that have less than δ_{skew} are merged to form a new solution $Sol_{k,n_{merging}}$. This is described in equations 3.5, 3.6, 3.7. The new t_{skew} is found by calculating the local clock skew in a nested iteration of both sets.

$$C_{subtree,n_{merging}} = C_{subtree,n1} + C_{subtree,n2} \tag{3.5}$$

$$W_{n_{merging}} = W_{n1} + W_{n2} \tag{3.6}$$

$$t_{slew,n_{merging}} = max(t_{slew,n1}, t_{slew,n2}) \tag{3.7}$$

New options for the parent-node are generated by adjusting the parameters of all options at node n by taking the wire-delay, slew-rate degradation and wire-capacitance into account:

$$C_{subtree,p} = C_{subtree,n} + c * l \tag{3.8}$$

$$t_{slew,p} = t_{slew} \tag{3.9}$$

3.1.2.4 Polarities

The original buffering algorithm by Van Ginneken considers only non-inverting buffers. In some publications [44], buffers are added at the root of inverted trees to change the polarity where required. In [44], the authors propose to handle polarity mismatches by inverter insertion, as the skew introduced by new inverters can be fixed by downstream optimization. However, the effect of optimization on run time and power can be significant and algorithmic consideration has to be taken into account for favorable results. In this implementation, each option has an additional bit for the polarity. When buffering with an inverting buffer, this bit is inverted. When merging two branches at a node, only those options that have equal polarity can be merged. Similarly, only solutions of the same polarity are considered for purging.

3.1.2.5 Post-Buffering Optimization

The buffering algorithm described in this chapter provides a buffered clock-tree that has low clock-skew and was inherently built for PVT resistance. This results in two very important and useful implications for post-buffering optimization:

- As the clock-skew is already low, we do not require extensive optimization after buffering as seen in other publications that use Van Ginneken's algorithm for buffering. Less optimization in this context directly implies less capacitance, as the wire-length that would be used for wire-snaking is saved and fewer buffers are inserted at this stage.
- Low initial skew means that this post-buffering optimization requires fewer simulation iterations, which is crucial as the simulations (e.g. SPICE) require extensive processing and are the main component of used CPU-time.

The flow of clock-tree tuning after buffering is shown in figure 3.4. As an early step, the slew-rate of all nodes is calculated based on Bakoglu's metric [66] as described in [65]. This is very important in the following steps, as it allows to precisely estimate the delay of wires and buffers, which further reduces the amount of iterations. Then, groups of sinks that are local, i.e. all within a specific distance of each other, are iterated and the slack is determined, similarly to [44]. The slack of node k of local set S is given by equation 3.10.

$$t_{slack}(k) = max_{n \in S}\{t_{delay}(n)\} - t_{delay}(k) \tag{3.10}$$

The slack denotes the amount of delay that should be added to the delay of sink k to balance the delay within local set S. The delay of sinks can either be determined by estimation with the Elmore-delay and consideration of slew-rate, or with simulation. In evaluations, Spice simulation is used for more accurate results.

Once the slack is determined for every sink, the slack at intermediate nodes is calculated. If the node has only one child node, the slack is equal to the slack of the child-node. Otherwise, the slack is the minimum of both child slacks and represents the common delay that can be added to both lower subtrees without either of them exceeding the maximum delay within the local group. As the optimization concentrates on connections between

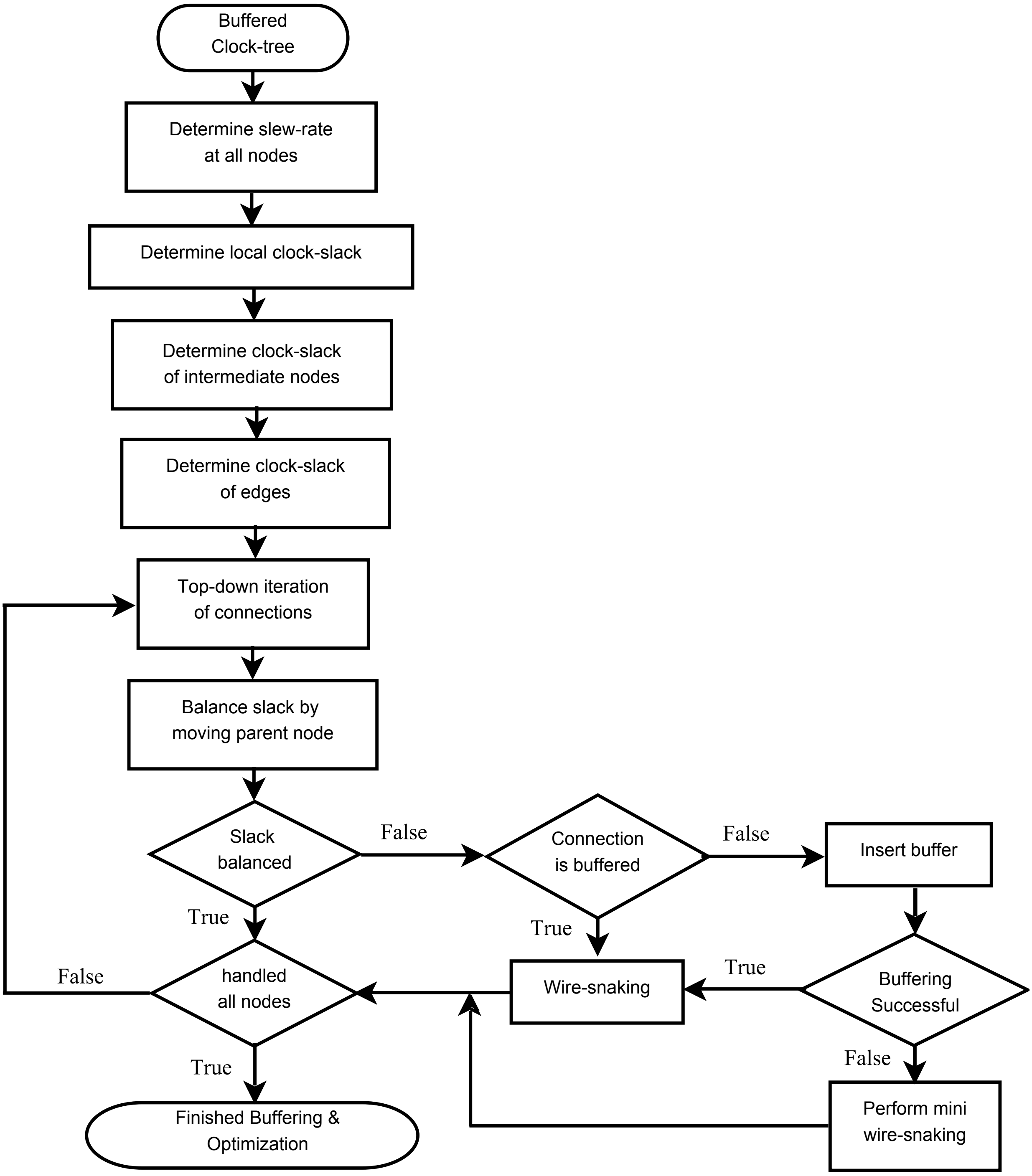

Figure 3.4: Flowchart of the post-buffering optimization of the clock-tree to achieve clock-skew constraints

nodes, the slack of edges has to be determined. The slack of a connection $t_{slack}(k, v)$ is defined to be the difference of the slack of the lower node v and the slack of the higher node k:

$$t_{slack}(k, v) = t_{slack}(k) - t_{slack}(v) \tag{3.11}$$

Then, the delay is iteratively balanced. The following scheme is used for this purpose: If a connection is buffered, then the wire-snaking technique is allowed. Theoretically, wire-sizing may also be employed, but our implementation does not consider this technique as wire-sizing techniques can lead to practical difficulties [44]. If the connection is not buffered, using wire-snaking will also affect the delay of surrounding nodes that have a common ancestor with the upper node of this connection. Therefore, the algorithm calculates whether it is possible to add a buffer without changing the upstream capacitance. If this is possible, the buffer is inserted. Otherwise, mini-wire-snaking is employed by drastically reducing the amount of slack at the node by multiplying the computed slack value by a reduction factor and then employing wire-snaking based on the reduced slack targets. This will lead to higher overall delay, but helps to handle situations where the buffering algorithm gets stuck at a sub-optimal solution because no connection can be adjusted.

3.1.3 Scalability

The novel buffering algorithm does not only achieve the best results for a buffered-clock-tree, but is also highly scalable and faster than comparable algorithms. As described in [67], scalability is an increasingly important problem in clock-tree synthesis.

Compared to Van Ginneken's buffering algorithm [64], this algorithm directly targets slew-rate and clock-skew. The computational complexity reduction and run-time improvements of this algorithm can be divided into two categories:

3.1.3.1 Improvements because of better clock-skew

As the algorithm leads to a buffered clock-tree with low clock-skew, the amount of optimization required after buffering is very limited.

In the evaluation, clock skew after buffering was around $13-18ps$, which greatly reduces the amount of optimization iterations in comparison to Van Ginneken algorithm, which typically lead to skew larger than $30ps$. This is particularly important, as the optimization iterations require costly SPICE simulations and may ultimately lead to increased wire length and power overhead due to wire-snaking and additional delay buffers.

3.1.3.2 Improvements within the algorithm.

The key improvement with regard to Van Ginneken's algorithm is the reduction of complexity. Given B possible buffering positions, Van Ginneken's algorithm leads to $B + 1$ solutions at the root of the tree and because of repeated iterations a general complexity is order of $O(B^2)$. This algorithm uses an adaptive heuristic to drastically reduce the amount of solutions by heavily constraining the solutions while maintaining excellent solution quality as shown in section 4.2. By adaptively adjusting buffer-number, capacitance, delay and slew limits, the amount of solutions at every node can be bounded by a constant k_{sol}. Instead of calculating and iterating many thousand solutions at each intermediate node, the novel algorithm computes only at most k_{sol} solutions per node. Therefore, the bottom-up phase has a complexity of $O(k_{sol} * B)$ resulting in linear complexity with respect to the amount of buffering positions. The top-down phase is a simple iteration of all buffering positions and therefore also has linear time complexity, resulting in overall linear complexity.

Algorithm 1 Purging algorithm.

```
Require: SolutionList
Ensure: worstSolution
  $\delta_{min} = 0$;
  Solution worstSolution;
  for all Solution s1 in SolutionList do
      for all Solution s2 in SolutionList do
          if $s1.polarity \neq s2.polarity$ then
              continue;
          end if
          $\delta = pd(s1.capacitance, s2.capacitance) * w_c$;
          $\delta+ = pd(s1.skew, s2.skew) * w_{skew}$;
          $\delta+ = pd(s1.rat, s2.rat) * w_{rat}$;
          $\delta+ = pd(s1.weight, s2.weight) * w_{weight}$;
          $\delta+ = pd(s1.slew, s2.slew) * w_{slew}$;
          if $\delta < \delta_{min}$ then
              $\delta_{min} = \delta$;
              $worstSolution = s2$;
          end if
      end for
  end for
```

In Algorithm 1, the pseudo-code is shown for a simplified purging process to effectively delete a solution from the *SolutionList* at a buffering position. In a nested iteration of the solution list, the minimal difference between any two solutions is calculated. For this purpose, a method $pd(k_1, k_2)$ is used that computes the percentage difference between two numbers k_1 and k_2 as shown in equation 3.12.

$$pd(k_1, k_2) = \frac{abs(k_1) - abs(k_2)}{max(abs(k_1), abs(k_2))} \tag{3.12}$$

The weights for capacitance, skew, required arrival time, buffer weight and slew rate are w_c, w_{skew}, w_{rat}, w_{weight}, and w_{slew} respectively. These weights reflect the importance of a fine granularity for the specific parameter. They have to be chosen with respect to the actual application domain. In a typical implementation, skew and required arrival time would be assigned high weights, because even minor differences can be highly important. However, the buffering would not be assigned a high weight w_{weight}, because small differences do not have a high impact on power consumption and therefore, it is acceptable for the buffer count to have coarse granularity. Table 3.1 shows an example of weights chosen for the evaluation. In actual implementation, dynamic programming can be employed to fasten the purging algorithm and purge all overhead nodes at the same time.

Table 3.1: Weight parameters for controlled solution pruning

Parameter	Value
w_c	3
w_{skew}	5
w_{rat}	5
w_{weight}	1
w_{slew}	1.5

3.2 Duty Cycle

In high-speed source-synchronous systems, along with a strict clock skew requirement, a duty cycle (on time per clock period) of 50% is also required to have a good setup and hold timing margin. Systems (e.g. DDR3, GDDR5, XDR) which work on both rising and falling edges of clock have stringent requirement of 50% duty cycle because a duty cycle error of 5% may lead to 10% performance degradation. Multi-phase clocking systems also require symmetrical wave shapes which demand a precise 50% duty cycle [68]. Furthermore, the clock distribution with different components like buffers and inverters, deteriorates the clock duty cycle due to PVT variations, inaccurate modeling etc.

In [69], the authors compared an analog DCC with a digital DCC. Analog DCC (Figure 3.5) uses the voltage mode correction technique. In the feedback path, clamp and integrator covert the duty cycle error into differential voltage. In the forward path, the duty cycle corrector uses resulting differential voltage of the feedback path to adjust the rise time and fall time of the input clock to achieve 50% duty cycle.

Digital DCC (Figure 3.6) uses the current mode correction technique. In the feedback path, integrator (CP), pre-amplifier, comparator and FSM convert duty cycle error into digital up/down bits. In the forward path, the duty cycle corrector controls the programmable current source/sink using up/down bits which in turn control the bias point of the inverter to correct duty cycle.

Analog DCC provides more accuracy in the feedback path while digital DCC is more suitable for low voltage applications and can provide wide duty correction range. The result in [69] shows that digital DCC works at 6.4 Gbps data-rate (3.2 GHz clock frequency) for ±10% input duty cycle error.

In [70], the authors utilize a binary search algorithm with Successive Approximation Register (SAR) to control the duty cycle adjuster. The duty cycle adjuster controls the duty cycle by delaying the rising/falling edge of the input clock based on SAR control bits. However, this DCC works for 300 MHz - 1 GHz range which cannot be applied to high frequency range applications such as GDDR5 and XDR.

In [71], [72], [73], the authors considered digital DCC techniques which work for wide input duty cycle range (10%-90%) and achieve 1 GHz - 2 GHz operating frequency at 1.8 V power supply. In [74], the authors used dual-loop DCC architecture which works for 1 GHz -2 GHz at 0.9 V - 1.4 V power supply and handles 25% -75% input duty cycle range.

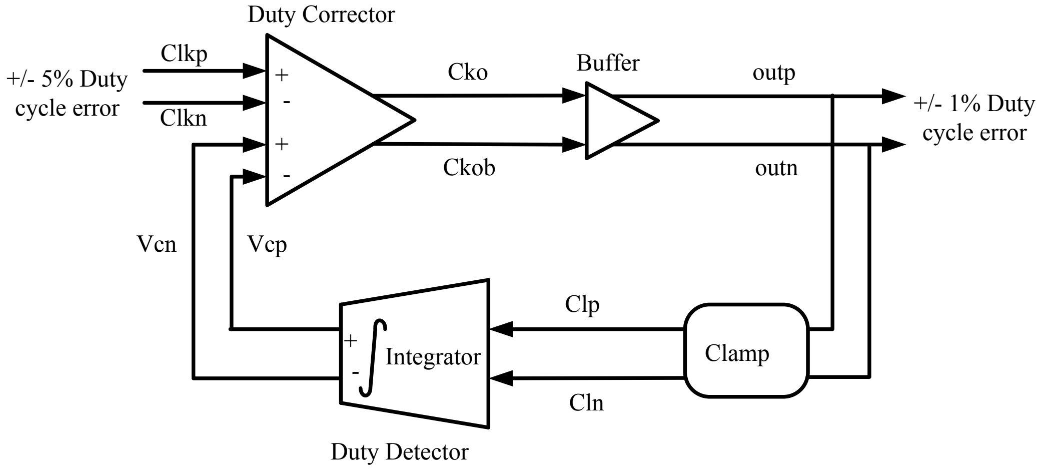

Figure 3.5: Analog DCC [69]

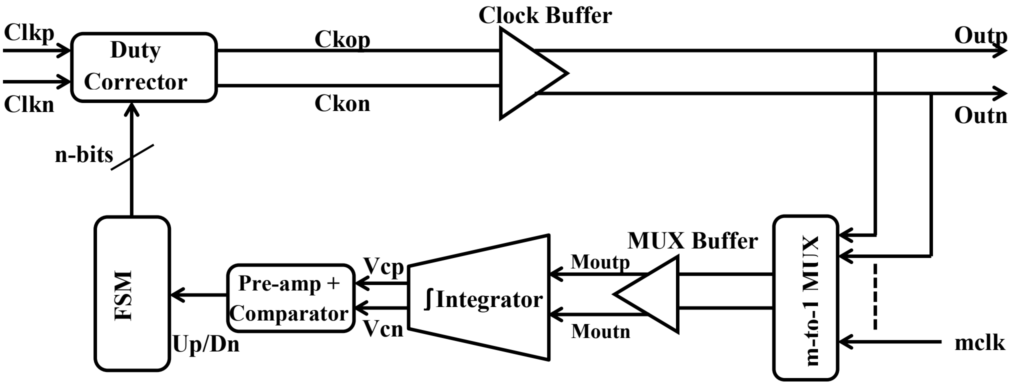

Figure 3.6: Digital DCC [69]

In [75], the authors used hybrid feedback (digital as well as analog) to achieve better loop stability and wide input duty cycle range (± 30%). This hybrid DCC works from 0.5 GHz - 2 GHz operating frequency range at 1.8 V of power supply. Similarly, in [76], the authors also used mixed techniques where the coarse correction is done by a half cycle delay line and the fine correction is done by analog feedback. They have achieved 200 MHz - 2 GHz operating frequency for ± 30% input duty cycle range at 1.2 V power supply.

In [77], the authors proposed all digital non-feedback duty cycle corrector which includes a delay unit based on precharge logic gates. This DCC works for 400 MHz - 2 GHz operating range and 20% - 80% input duty cycle range at 1.8 V power supply.

In [78], the authors presented all digital duty cycle corrector with pulse-width detector. This DCC can handle 10% - 90% input duty cycle error and can work for 100 MHz - 3.6 GHz operating frequency at 1 V power supply. In [79], the authors demonstrated wide range duty cycle corrector specifically for GDDR5 SDRAM which can handle ± 100 ps input duty error for 800 MHz - 3.5 GHz at 1.5 V power supply. It uses an anti-harmonic asynchronous binary search (ABS) circuit for fast correction.

In [80], the authors proposed a mixed signal DCC which consists of a control stage, a buffer stage and a gain boosting CP. This simple DCC architecture works for wide frequency range (20 MHz - 2.5 GHz). For low frequency range (upto 800 MHz) at 1.3 V, it can handle 1% - 99% input duty, while at high frequency (GHz), 20% - 80% of input duty can be handled at 1.8 V power supply. The main limitation of this DCC architecture is that the power supply has to be increased at higher frequencies from 1.3 V to 1.8 V due to the cascode structure of the charge pump. Furthermore, the output duty cycle can not be corrected beyond 2.5 GHz. Due to the simplicity of this DCC architecture with wide frequency and input duty cycle range, this architecture has been investigated in details to overcome its limitations.

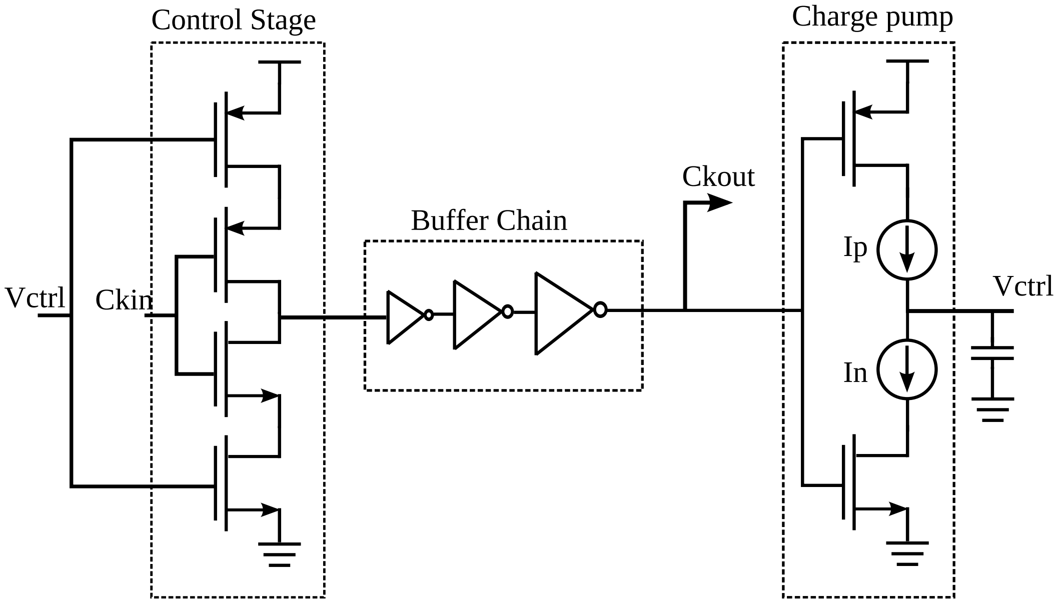

Figure 3.7: Mixed-signal DCC [80]

3.2.1 Circuit structure of the considered DCC

Figure 3.7 shows the circuit architecture of the considered DCC [80] which consists of three stages: The control stage, The buffer chain, The charge pump. The control stage is a simple push-pull structure where charging and discharging path are controlled by Vctrl (analog voltage). Depending upon the Vctrl level, the duty cycle of the input clock (Ckin) increases or decreases. This push-pull structure leads to faster locking time as it adjusts the rising and falling simultaneously and doubles the gain. The buffer chain is the classical inverter chain which is used to make the output of the control stage rail-to-rail. Figure 3.8 shows the gain boosting CP. Current sources implemented by MP4 and MN3 transistors isolate Vctrl from the Ckout to reduce the coupling effect, hence there will be less ripple on Vctrl which leads to less jitter.

Figure 3.9 shows the loop analysis of the DCC (Figure 3.7) where K_{CS} is the sensitivity of the control stage, K_{BC} is the gain of the clock buffer chain, K_{CP} is the gain of charge pump, Wout is the target pulse width of the output clock signal and Win is the pulse width of the input clock signal. R and C are the equivalent output resistance of the charge pump and the output capacitance of the charge pump respectively. τ_0 is the time constant.

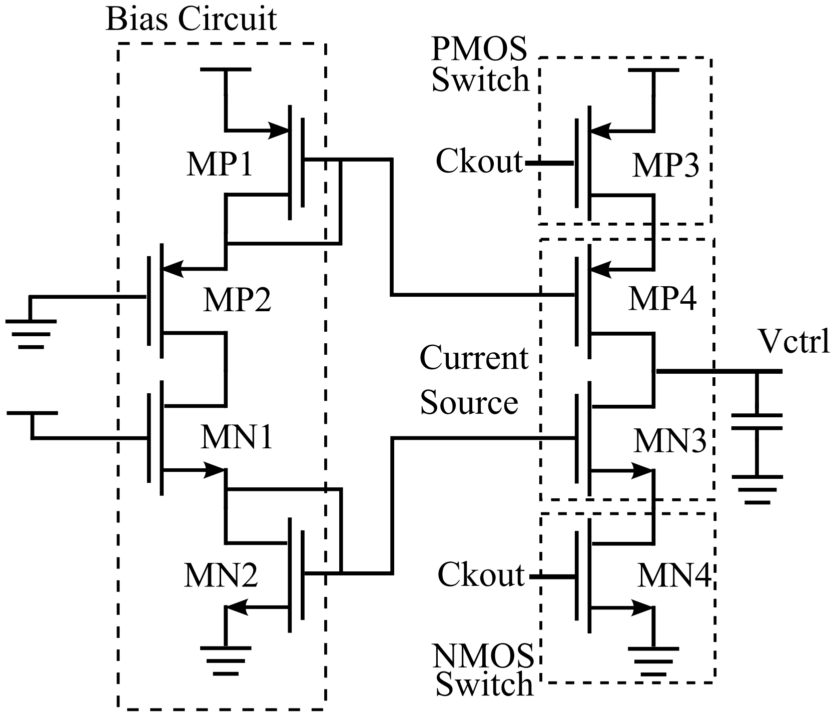

Figure 3.8: Gain-boosting CP [80]

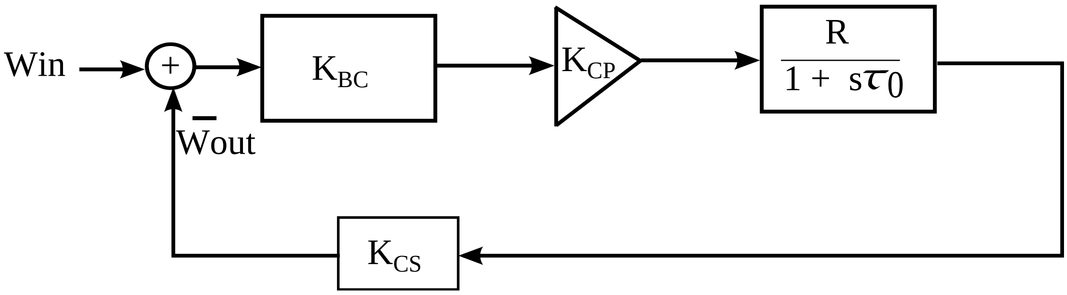

Figure 3.9: Loop Analysis [80]

$$F(s)|closed = \frac{H(s)}{1+H(s)} = \frac{\frac{K_A}{1+K_A}}{1+\frac{s}{W_0(1+K_A)}}; \tag{3.13}$$

$$K_A = K_{CS}K_{BC}K_{CP}R; \tag{3.14}$$

$$W_0 = \frac{1}{RC}; \tag{3.15}$$

Eq. 3.13 shows the closed loop transfer function of Figure 3.9. With the single pole, the circuit is unconditionally stable. From Eq. 3.14 and Eq. 3.15, the parameters W_0 and K_A are dependent on R. As the gain bandwidth product is constant for the control loop, R has to be decreased to reduce the open loop gain K_A and to increase the loop bandwidth.

In Figure 3.8, R is mainly dependent on current sources (MP4 and MN3) as MP3 and MN4 transistors work as simple switches which will be on and off depending on Ckout.

To increase the bandwidth (to decrease R), sizes of MP4 and MN3 transistors have to be increased.

3.2.2 Novel Architecture with programmable charge pump

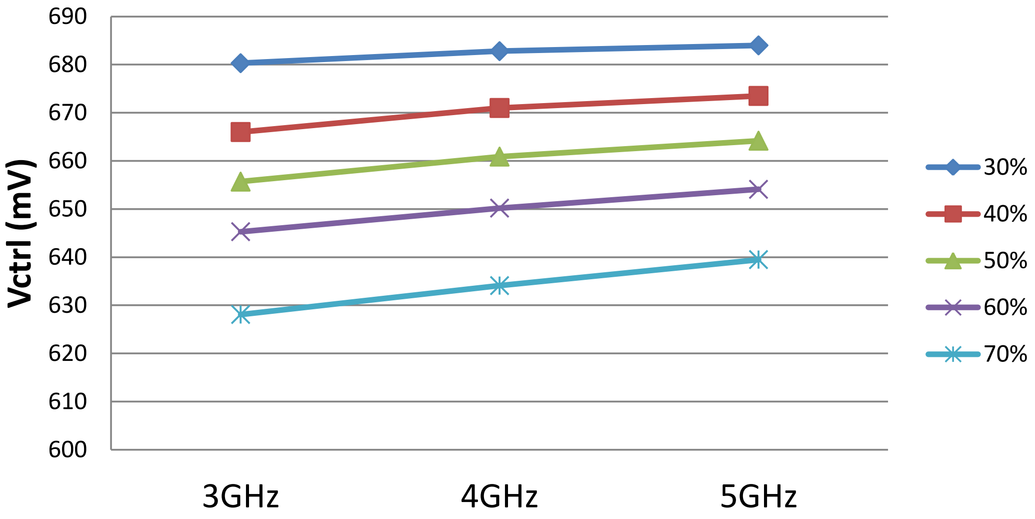

Figure 3.10: CP characteristics simulated in TSMC 65nm for 3 GHz - 5 GHz

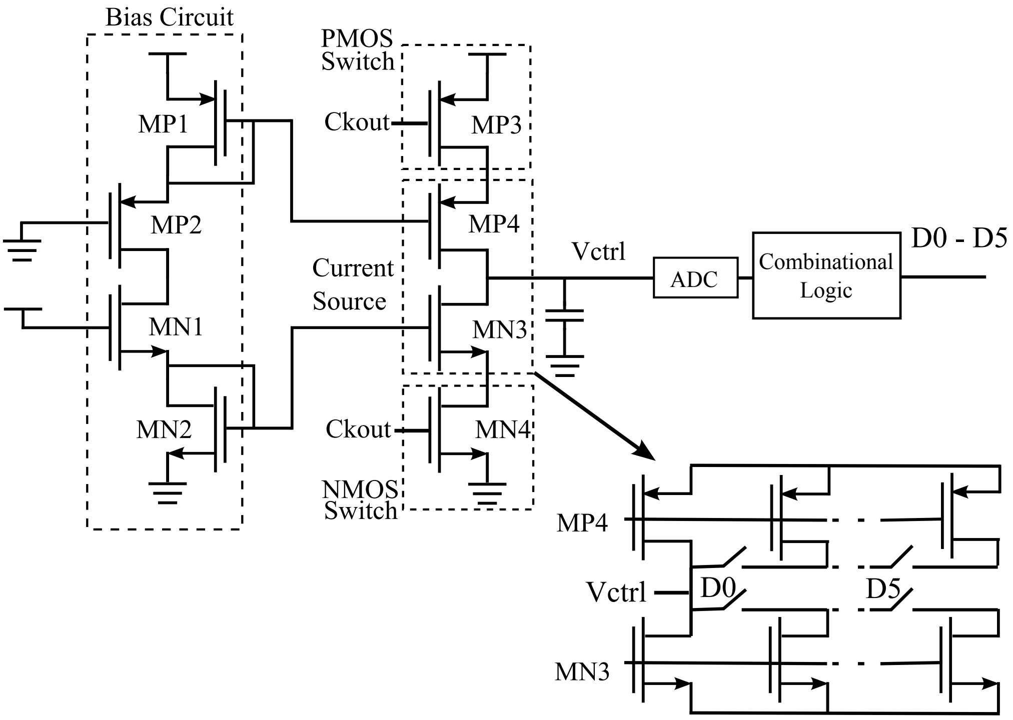

Figure 3.11: Programmable CP

The DCC architecture [80] in Figure 3.7 can work from 20 MHz - 2.5 GHz. With the presented architecture based on [80], the operating frequency range has been extended from 3 GHz - 7 GHz with 30% - 70% input duty range as it can satisfy the specification range of DDR3/GDDR5/XDR applications.

Figure 3.10 shows the Vctrl characteristics of CP simulated in TSMC 65nm for 3 GHz to 5 GHz. Vctrl has good linearity for different frequencies and has a unique value for a particular input duty cycle and a particular frequency. In other words, if the Vctrl value is known, the operating frequency and the input duty cycle can be derived from it. This property of Vctrl is used to make the CP programmable.

To increase the operating frequency range, MN3 and MP4 of Figure 3.8 can be replaced by a set of switchable impedances (parallel transistors) to adjust resultant impedance and thereby loop bandwidth depending on the operational frequency. Due to the particular frequency characteristics of Vctrl this can be used to derive the necessary control information.

Figure 3.11 shows the programmable CP where Vctrl value (analog value) is converted into digital bits using 6-bit Analog-to-Digital Converter (ADC) and combinational logic to decode the value (e.g. D0-D5) which will in turn control the parallel transistors of MP4 and MN3. These parallel transistor can be designed in many ways: with same size; with increase in unit size of each subsequent transistor (linear coding); with each subsequent transistor doubling its size (binary weighted coding); with any customized coding based on ADC readout for particular Vctrl value. As Vctrl is unique for particular frequency and input duty cycle, ADC does not need to run at higher speed for quick sampling. ADC sampling frequency depends on the fact how often the input frequency and the duty cycle changes due to PVT variations or desired functionality.

3.3 Conclusions

This chapter presented novel algorithm features a buffering algorithm for a tree-structured clock-tree. Along with accurate delay estimation by slew-rate consideration and early preparation of the clock-tree for wire-snaking, the buffered clock-tree is very robust against process variations. This chapter also described a novel programmable wide range duty cycle corrector.

Chapter 4

Simulation Results

This chapter provides the simulation results and comparison with state-of-the-arts for adaptive training, clock skew and duty cycle concepts.

4.1 Adaptive trainings: Results

4.1.1 Comparison with state-of-the-arts

In comparison to the current state-of-the-art architecture (Figure 2.15), the novel architecture based on unit-delay (Figure 2.16) further reduces complexity as it does not need separate phase generation units. The unit-delay architecture needs only unit-delay incrementer instead of large calibrated delay line. Phase-detector circuit is also simpler as it only needs shift-registers and comparator which can be implemented using combination logic while the current state-of-the art (Figure 2.15(A)) needs oversampling, majority voting. Phase incrementer part in Figure 2.16 is a simple unit-delay and a multiplexer for path control. It is obvious that in comparison with Figure 2.15(B) which includes the loop filter, PI counter and the PI, the novel architecture based on unit-delay incrementer saves considerable area and power.

However, the simplicity in the unit-delay architecture comes with the penalty of extra time needed for synchronization as unit-delay can be applied only in the positive direction (early to late transition). In the worst case, transition edge search from early to late needs to be done for one reference clock cycle i.e. one CLK period. Nevertheless, this worst case synchronization time is minimal in comparison to the whole training sequence including power-up sequence and other read and write synchronization algorithms.

In the case of the second novel architecture with PI-based PLL, the timing penalty is removed as a PI-based PLL has the advantage of correcting the phase in positive as well as negative directions. If phase correction is needed in the negative direction (for example -10 degree), PI-based PLL will need less time to synchronize (only 2 steps with the resolution of 5 degree) while unit-delay incrementer needs longer time (-10 degree = 350 degree i.e. 70 steps with the resolution of 5 degree) as it can correct phase in only positive direction. Each phase-delay step consumes additional latency which is equal to the sum of EDC latency (5ns [52]), channel delay (100ps - 200ps based on RLGC channel model), other receiving circuit delay including shift-registers and comparator (3ns - 5ns). In the given example of -10 degree phase difference, the PI-based architecture will take 20ns (2 steps * 10ns latency) to complete WCK2CK training while the unit-delay architecture will consume approximately 700ns (70 steps * 10ns latency) which is still negligible in comparison to GDDR5 initialization time (400μs - 500μs [35]). A PI-based PLL has the disadvantage of additional PLL complexity. The decision to choose one architecture over the other is based on synchronization time Vs PLL complexity.

For the simulation setup, the first novel architecture is described in Verilog and synthesized as shown in Figure 4.1 with TSMC 65nm low-power library ([53]). Figure 4.1 used 12 shift registers only for 3 samples of 4 bit EDC value and few logic gates for comparison and multiplexer.

Table 4.1 compares the synthesis result with other works. The novel unit-delay architecture only consumes 0.89 mW power and 100 $(\mu m)^2$ area which is 16.8 times less power

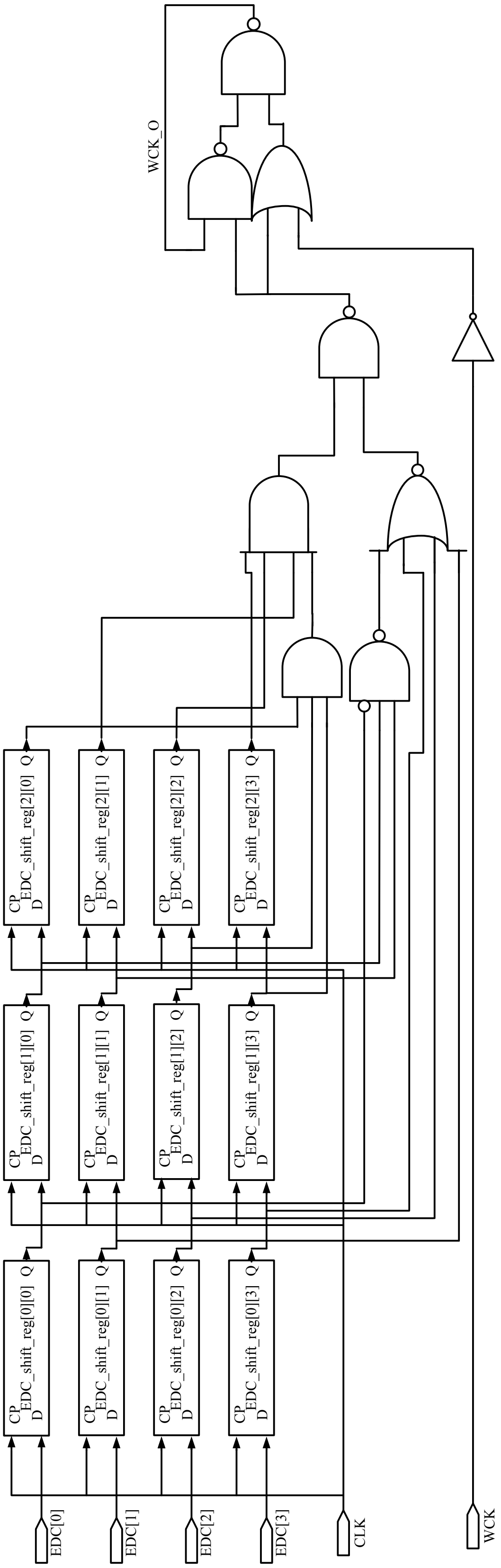

Figure 4.1: Schematic:Unit-Delay incrementer based architecture

Table 4.1: Comparison with other work

	[50]	[48]	[46]	Unit-delay architecture	PI-based PLL architecture
Process (nm)	90	65	65	65	65
Supply (V)	1.25	1.2	1	1.32	1.32
Data rate (Gbps)	9	1-6	10	10	10
Power (mW)	34.2	22	15	0.89	1.28
Area (μm)^2	150000	3500	-	100	130

and 35 times less area than other works. If we use customized cells instead of TSMC library cells, then further area and power can be saved. Considering PI as 8:1 multiplexer in Figure 2.19, the architecture consumes 1.28 mW power and 130 (μm)^2 area which are in similar range as the unit-delay incrementer architecture. If higher resolution is required, then PI-based PLL architecture will consume more power and area depending upon PI (e.g. Figure 2.4) used in the architecture.

4.1.2 Results using circuit level simulation

In this section the results obtained from the circuit level simulation of PI-based PLL architecture in cadence environment using TSMC 65nm low power library are shown and compared. PI-based PLL block used in this architecture was part of master thesis [115]. The simulations are divided into the low frequency tests with a WCK frequency of 2 GHz and the high frequency tests with WCK set equal to 5 GHz.

4.1.2.1 Low Frequency Simulations

The simulations consist of two main cases: one employing the VCO generating eight phases and another one utilizing the VCO producing four phases. Each of these systems is tested for the minimum and maximum PLL bandwidth. This will show the advantage and disadvantage of each configuration. In the simulations the focus will be mainly laid on the delay error between *mem_CLK* and *mem_WCK* because this quantity characterizes the best the performance of the whole system. The results of the simulations are shown in Table 4.2.

4.1.2.2 High Frequency Simulations

The second part of the simulations lies in testing the system working at a much higher frequency than in the previous case. The reference clock CLK is set at 2.5 GHz resulting in a WCK frequency of 5 GHz. As shown in the previous cases, the PLL with lower bandwidth performs the best. Therefore, for the high frequency simulation the low bandwidth resistor will be selected. Also in this case the error, i.e. the delay between *mem_CLK* and *mem_WCK*, is measured and reported in Table 4.3.

Table 4.2: Measured error of 2 GHz WCK configuration.

# phases	bandwidth (MHz)	Peak error (ps)	RMS error (ps)
4	10.3	44.79	24.13
4	7.26	31.02	16.0
8	10.3	16.02	10.22
8	7.26	9.194	5.76

Table 4.3: Measured error of 5GHz WCK configuration.

# phases	bandwidth (MHz)	Peak error (ps)	RMS error (ps)
4	4.9	7.72	4.245

Table 4.3 provides interesting information. The system performs better than the low frequency scenario from both absolute and relative points of view. This improvement is due to the higher frequency of the CLK, which has two consequences: first, the digital circuits work faster providing the control signals sooner and, second, the sampling rate is higher. These two aspects result in both quicker control signals and faster recognition of the crossing of the alignment between the signals. However, the improvement is hampered by the propagation delays of the signals and by the synchronous logic of the comparator and the control.

Simulation results show that the PI-based PLL architecture can be used for the phase delay control of data clock. The error generated by the system is dependent on the bandwidth of the system, phase steps and the system delay. At 2 GHz WCK, with 8 phases 5.76 ps rms error is achieved while at 5 GHz WCK, with 4 phases rms error is 4.245 ps.

4.2 Clock Skew: Results

In this section, the evaluation of solutions generated with the novel buffering and optimization technique is given. The clock tree generation using the novel algorithm was implemented in a bachelor thesis [122] using C++ language. In subsection 4.2.1 the simulated data that illustrates the high scalability of the algorithm is provided. In subsection 4.2.2 it is shown that the algorithm performs well in recent benchmarks and brings extensive run-time and power reductions in comparison to other algorithms.

4.2.1 Scalability

The high scalability of the novel *Reliable Low Power Buffering and Optimization algorithm* (RLPBO) in section 3.1.3 is elaborated. Based on the ISPD 2010 CNS benchmark suite, circuits of varying size are randomly generated and the RLBPO buffering time along with initial clock-skew is measured. The results are presented in table 4.4. It is clearly visible that directly after buffering, the clock-network already has good nominal skew. This is a very important characteristic and helps to reduce further optimization time, as described in 3.1.3. Figure 4.2 visually shows the linear relation between run-time and the number of buffering positions for a wide range from 2,010 to 40,052 buffering positions.

Table 4.4: Run-time comparison of the novel buffering algorithm with seven different randomly generated circuits.

Sinks	Buffering Positions	Nominal Skew [ps]	Run-time [s]
20000	40052	1.88868	171
15000	30037	4.5465	120
10000	20035	2.14846	82
7500	15024	1.33554	62
5000	10016	1.382	43
2500	5018	1.66268	21
1000	2010	4.0756	10

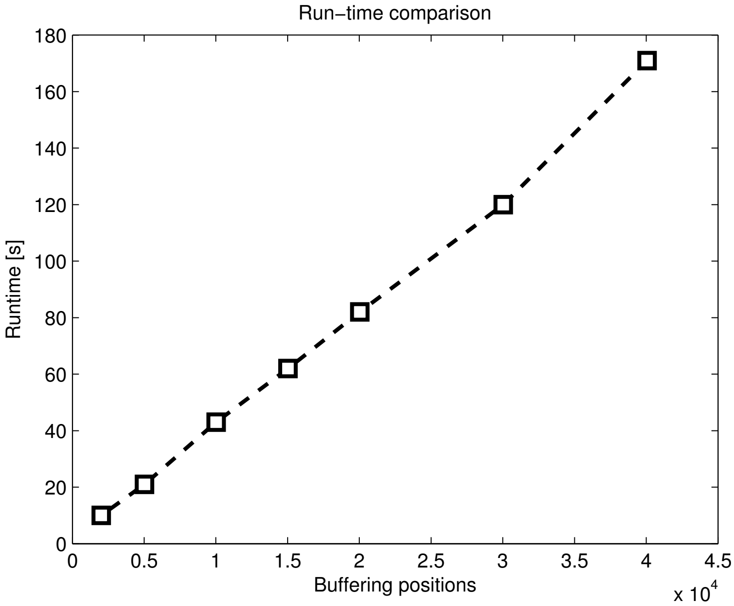

Figure 4.2: Run-time of the buffering algorithm for different random circuits shows linearity in the number of buffering positions

4.2.2 Benchmarks

For evaluation of the buffering algorithm, the ISPD 2010 CNS benchmark suite [62] was used. In five-hundred simulations, the output of the RLPBO running on a simple DME implementation is shown in comparison to contest participants and the *Contango 2* solution, to strongest result for the ISPD 2010 benchmark suite featuring a tree topology to the best of current knowledge.

Table 4.5: Comparison between the results generated by RLPBO and Contango 2

	RLPBO			**Contango 2**			
Benchmark	**Capacitance (fF)**	**LCS (ps)**	**CPU (s)**	**Capacitance (fF)**	**LCS (ps)**	**CPU (s)**	**PC (C)**
3	49517	6.7553	578	55860	4.18	3840	-0.11355
4	61650.7	6.8867	1306	71840	4.46	6075	-0.14183
5	32756.1	6.11438	546	37690	4.41	2406	-0.13091
6	42011.5	7.1836	1044	47810	6.05	2660	-0.12128
7	62316.4	6.13	1385	72660	4.58	2351	-0.14236
8	42550.2	6.9502	636	52490	5.15	1987	-0.18937

The ISPD benchmarks are based on real designs and target a local clock-skew of $7.5ps$ within a distance of $600\mu m$. The slew-rate is required to be below $100ps$. For simulation of PVT variations, the benchmarks randomly vary the power supply voltage of buffers (max. +-7.5%) and width of wire segments (max. +-5%) with a uniform distribution. In Table 4.5, a detailed comparison between RLBPO and Contango 2 is provided with regard to capacitance (a simple metric for power consumption), Local Clock Skew (LCS) and CPU run-time for six different benchmarks. RLBPO outperforms Contango 2 by providing a solution much closer to the desired constraints and leads to a capacitance reduction of 14% on average and an overall run-time reduction of 68% on average. Although simulations and optimization steps were performed in Spice (ngspice simulator), due to the high quality of our initial buffering, the number of optimization iterations reduces to approximately 3 to 6. Both results emphasize the importance of estimating the effect of process variations in the very early buffering stage and show that the post-optimization that is necessary if the susceptibility to process-variations is not considered properly before buffering, strongly affects the quality of the generated clock network. Note that the main contribution is buffering and light post-buffering optimizations. Therefore, the comparison is performed against other implementations with the DME as the baseline. As this algorithm targets the buffering stage of clock-network synthesis, it can therefore be combined with recent research achievements in the area of mixed-topology clock-trees.

4.3 Duty Cycle: Results

The novel DCC architecture (Figure 3.11) is simulated in cadence environment using TSMC 65nm low-power technology at 1.2 V power supply. Figure 4.3 shows that with programmable CP, DCC can produce output duty cycle with error well below ±1% for the operating frequency up to 5 GHz while convention CP (Figure 3.8) produces more than 10% output duty cycle error at 5 GHz. For converting Vctrl voltage to digital bits, a low power 6-bit ADC [81] can be used at 1 MS/s which consumes only 40 μW.

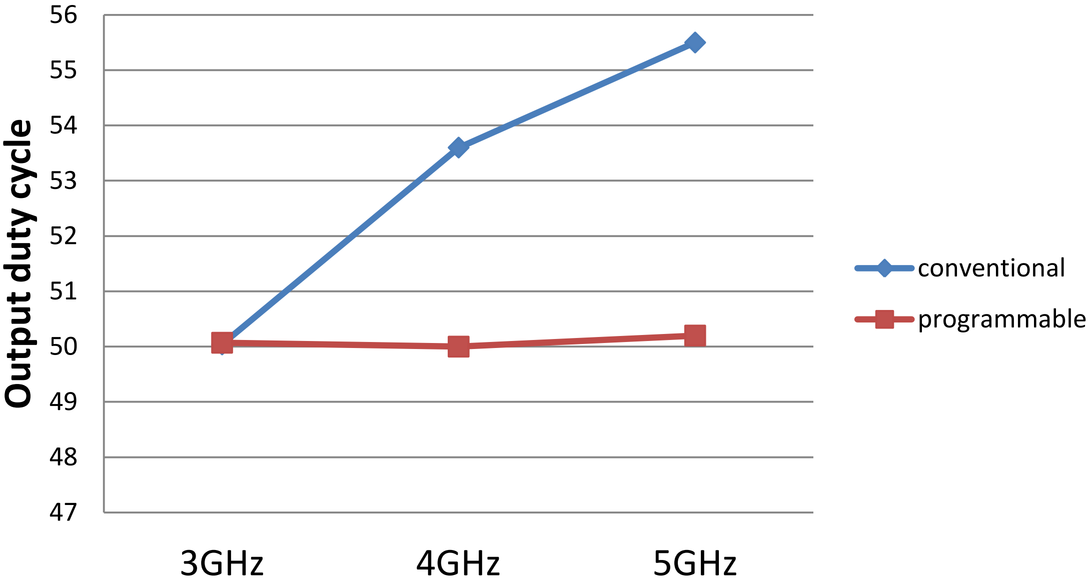

Figure 4.3: Output Duty cycle for 70% input duty cycle

When the operating frequency is increased beyond 5 GHz, then the novel DCC has shown unstable behavior. The limiting factor was the classical buffer chain design (increasing sizes of consequent inverter). At a frequency of 6 GHz (clock period- 166 ps) with the input duty cycle of 30% (49.8 ps on time), the input gate capacitance of big inverters cannot be charged or discharged properly by the previous stage inverter. Therefore, sizes of large inverters have to be reduced beyond 5 GHz. This can be easily achieved by using an additional bit A0 as shown in Figure 4.4. The A0 bit can be an external input (in this case) or can be generated by a frequency detector.

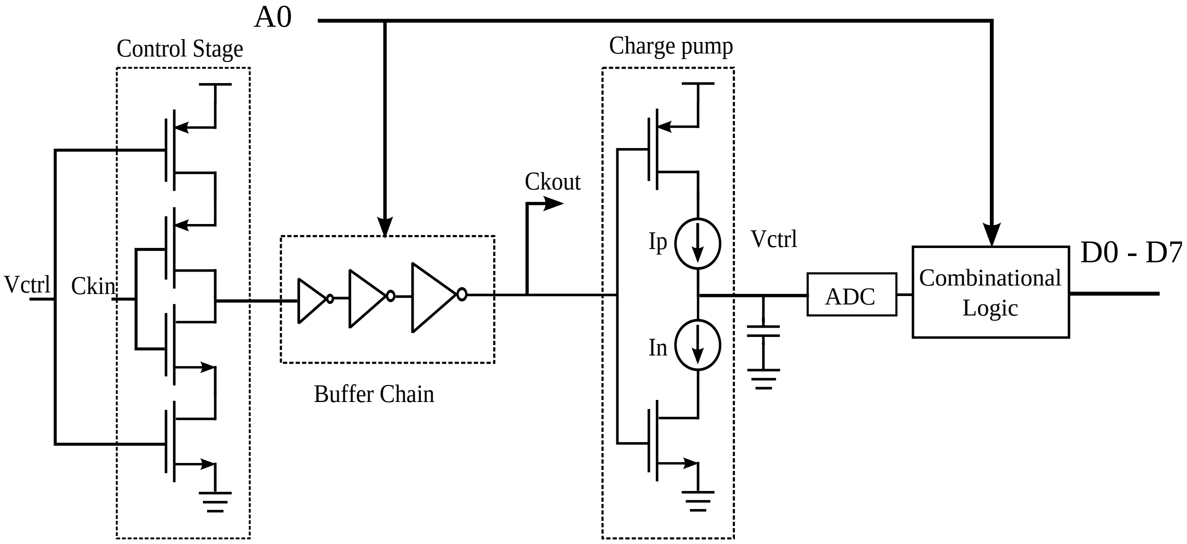

Figure 4.4: Programmable CP and Buffer

Till 5 GHz, A0 bit is 1 and normal operation works with parallel transistors on in large inverters. Beyond 5 GHz, A0 bit is 0 and some parallel transistors are switched off to lower

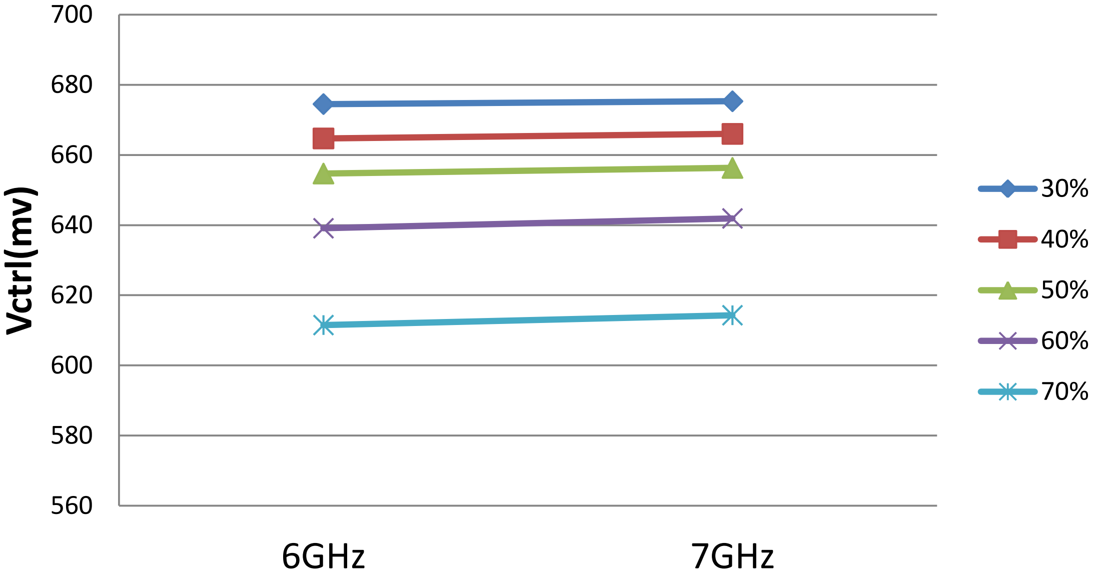

Figure 4.5: CP characteristics for 6 GHz & 7 GHz

the effective size of large inverters and therefore the gate capacitance of the programmed inverters are also reduced.

Figure 4.5 shows the Vctrl characteristics for 6-7 GHz. This Vctrl characteristic is similar to the Figure 3.10 i.e. if Vctrl is known then the frequency and the input duty can be derived from it. Figure 4.5 and Figure 3.10 differ in the absolute Vctrl voltage level e.g. for 70% input duty cycle at 5 GHz operating frequency, Vctrl is 640 mV while at 6 GHz, Vctrl is 615 mV which is lower than 5 GHz value. To differentiate between them A0 bit is also used in combinational logic (Figure 4.4).

Figure 4.6 and 4.7 shows that with the programmable CP and the programmable buffer chain, the output duty cycle of 50% can be maintained upto 7 GHz operating frequency for 30% - 70% input duty cycle with error well below ±1%.

TABLE 4.6 compares this work with the previous work [80] and shows that with the novel programmable DCC architecture can achieve a wide frequency range.

Table 4.6: Comparison with the previous work

	[80]	**this work**
Process	180nm	65nm
Supply	1.8 V	1.2 V
Input Duty Range	20%-80% ≥ 1 GHz	30%-70% ≥ 3 GHz
Max. Frequency	2.5 GHz	7 GHz
Min. Frequency	20 MHz	20 MHz
Power	0.36 mW@1 GHz	0.11 mW@1 GHz

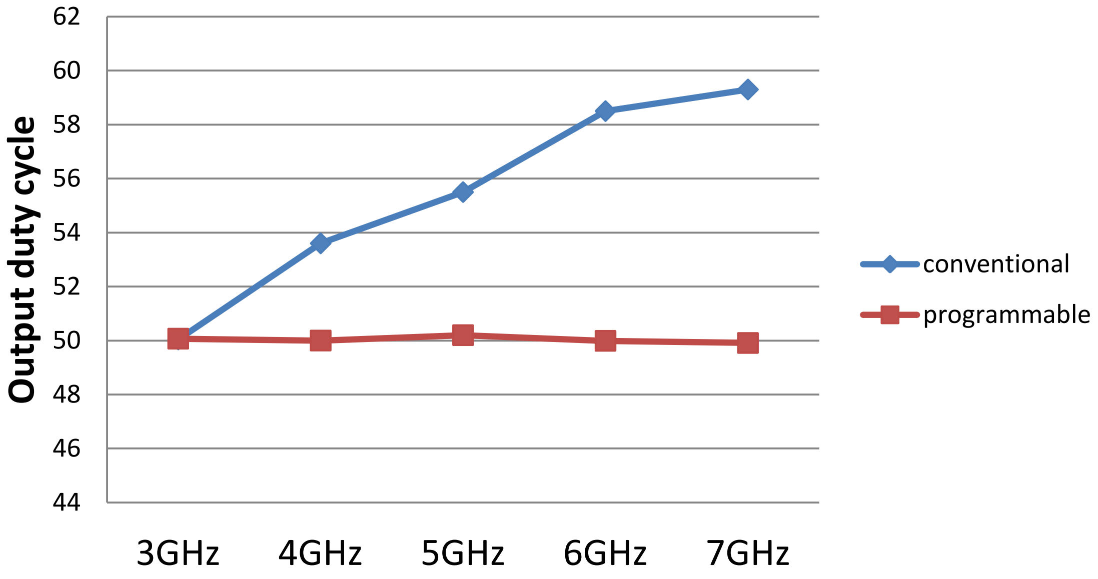

Figure 4.6: Output Duty cycle for 70% input duty cycle

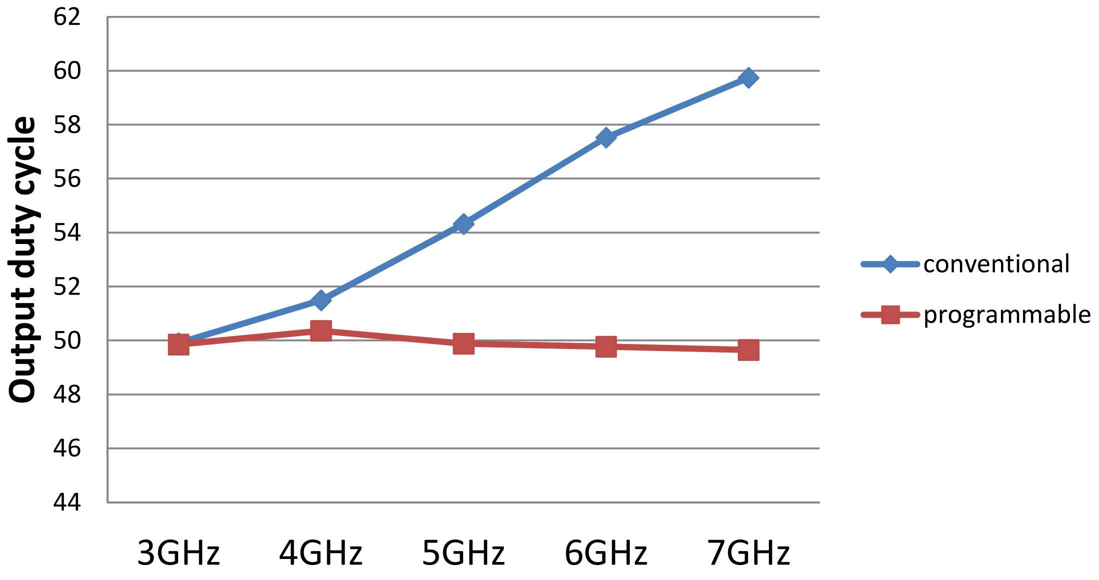

Figure 4.7: Output Duty cycle for 30% input duty cycle

The novel DCC architecture has two main limitations. First limitation is that the CP output (Vctrl) should reach a stable state to perform proper operation which needs approximately 200 ns in this case. However; this time is much less than the PLL locked-in time, which is in the range of 50 μs - 100 μs generally. Second limitation is that the novel programmable DCC architecture is technology-dependent. If new technology is employed, the whole architecture (including ADC and combination logic) has to be re-characterized as Vctrl value is dependent on the supply voltage and the transistor characteristics.

This novel programmable mixed signal DCC architecture works up to 7 GHz and produces output for 30% - 70% input duty cycle with an error below ±1% in TSMC 65nm technology. This DCC is suitable for applications e.g. DDR3/GDDR5/XDR which perform operations at a wide frequency range.

Chapter 5

Summary and Future Works

5.1 Summary

In future, mobile traffic and supercomputing applications will demand Exascale bandwidth which relates to GB/s bandwidth requirement per I/O link. It has been shown in chapter 1 that source-synchronous I/O links will perform better in terms of area and power if many high speed links are used in parallel. It has also been shown that the state-of-the-art systems use adaptive trainings to overcome non-idealities faced by parallel source-synchronous I/O links. In this thesis, novel efficient adaptive training architectures have been discussed. A novel clock tree buffering algorithm and a novel duty cycle corrector have also been presented to compensate for clock skew and duty cycle error which are critical for source-synchronous I/O links. Major contributions of this thesis are as follows.

- **Novel Adaptive Trainings:** Chapter 2 has shown that most of the state-of-the-art source synchronous system used complex adaptive trainings based on analog approach. Two novel efficient adaptive architectures based on unit-delay and PI-based PLL concepts [99], [103] with an example of multiple clock synchronization have been explained in detail. First these concepts were analyzed using high level behavioral models and compared with state-of-the-arts. PI-based PLL adaptive training is implemented at the circuit level and the building blocks are described in appendix B. Results in chapter 4 show that synchronization between multiple clocks can be achieved with very low phase error (4.245 ps at 5 GHz) using PI-based PLL concept.
- **Novel Clock Tree Buffering Algorithm:** Chapter 3 described that at high speed, the clock tree is the major source of power consumption and PVT variations affect the clock skew. A novel reliable low power buffering algorithm [100] for clock tree has been explained. This algorithm considers the PVT variations at topology stage itself and inserts buffers into the clock tree accordingly. Results in chapter 4 show that presented algorithm have achieved better results than the state-of-the-arts.
- **Novel Duty Cycle Corrector:** Chapter 3 also described the importance of the duty cycle for high-speed DDR systems. A novel wide range duty cycle corrector [101], [104] has been explained. Results in chapter 4 show that presented duty cycle corrector can work up-to 7 GHz for 30% - 70% input duty cycle error.

5.2 Future Works

Novel concepts explained in this thesis has been implemented at the circuit level using low power TSMC 65nm technology library. Next logical step is to realize these concepts in hardware. A high speed board including transmitter, receiver chips and the possibility of incorporating links with different lengths between transmitter and receiver have to be designed. A novel testing concepts have to be developed for precise measurements. Apart from hardware realization, following things can also be considered for future work.

- **Re-trainings Mechanism:** Even though adaptive training synchronize signals with high resolution, external factors such as noise, PVT variations can still drift the phase. A novel on-chip monitoring concept has to be developed on the basis of phase difference detection or BER so that as soon as synchronization goes out of boundary condition, re-training can be initiated.

- **Adaptation to Optical links:** Electric links will soon reach their maximum speed limit and to increase the bandwidth, optical links will be incorporated. Adaptive trainings have to be adjusted according to the optical link characteristics.
- **Equalization Trainings:** Non-idealities in transmitter, receiver and channel not only affect the signal in the time domain (in terms of skew) but also in the voltage domain (signal degradation). Signal degradation can result in false propagation of data. Novel adaptive trainings presented in this thesis can solve the synchronization problem only in time domain. To eliminate signal degradation, equalization trainings can be added to adaptive training sequences.

Appendix A

Design Methodology

1.1 Design Methodology using Matlab/Simulink-Cadence co-simulation

The development of high performance integrated circuits using advanced technologies has lead to an increased complexity in circuit design, particularly in the mixed-signal domain. An early detection of design errors can avoid higher design cost and minimize time-to-market. Therefore, a systematic methodology on system level ranging from concept to implementation stage, has become increasingly important. It enables designers to explore the system behavior at different abstraction levels, and thus to detect design flaws in the early phase.

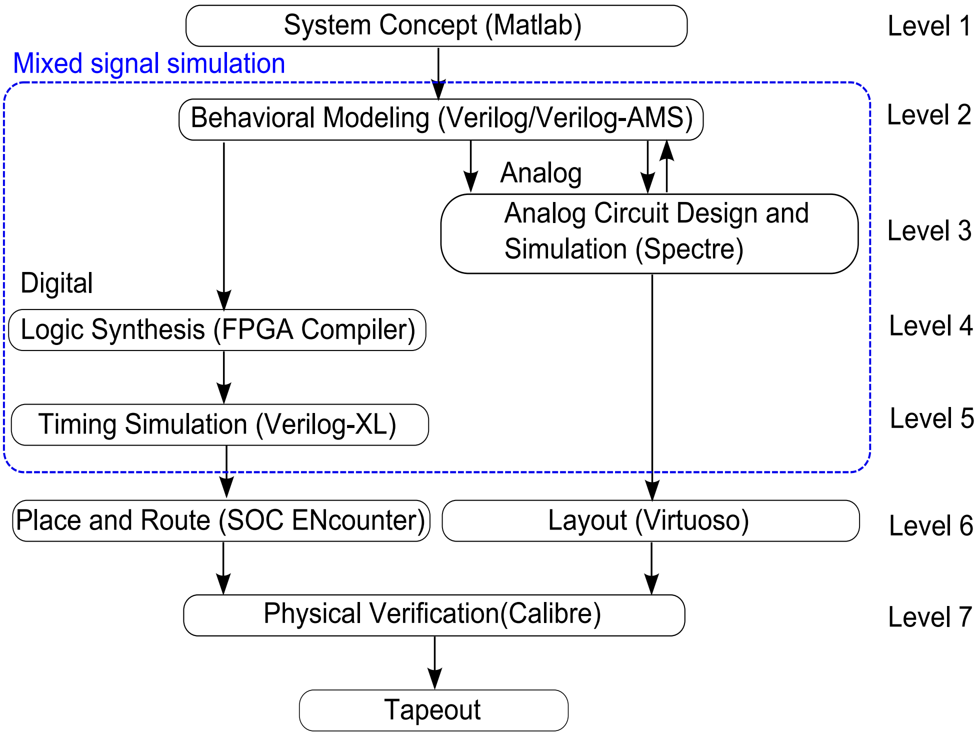
System Concept (Matlab)
Level 1
Mixed signal simulation
Behavioral Modeling (Verilog/Verilog-AMS)
Level 2
Analog
Analog Circuit Design and Simulation (Spectre)
Level 3
Digital
Logic Synthesis (FPGA Compiler)
Level 4
Timing Simulation (Verilog-XL)
Level 5
Place and Route (SOC ENcounter)
Layout (Virtuoso)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

Figure A.1: Conventional Design Flow

Figure A.1 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, the system is divided into digital and analog blocks. 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. Levels of abstraction are decreased from level-1 to level-7.

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 [82], Power Electronics [83]. W. Hassairi et al. [84] proposed a solution to SystemC/Matlab co-simulation for functional verification of multi-abstraction level designs. B. Gestner et al. [85] developed Modelsim-Matlab interface for RTL debugging and verification.

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. [86] explained the semantics of Verilog-AMS that 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 the 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 presented.

This chapter aims to present an efficient methodology for mixed-signal design based on Matlab/Simulink-HDL co-simulation and describes solutions found for overcoming the known pitfalls. GDDR5 memory system application is used as a case study to verify the presented method.

1.1.1 Related Work

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 the 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. [87] 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. [88] in terms of 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 [89] [90] are proposed. Matlab-VHDL [89] 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 [90] 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.

To verify the functionality of RTL-level models from Matlab/Simulink, the presented

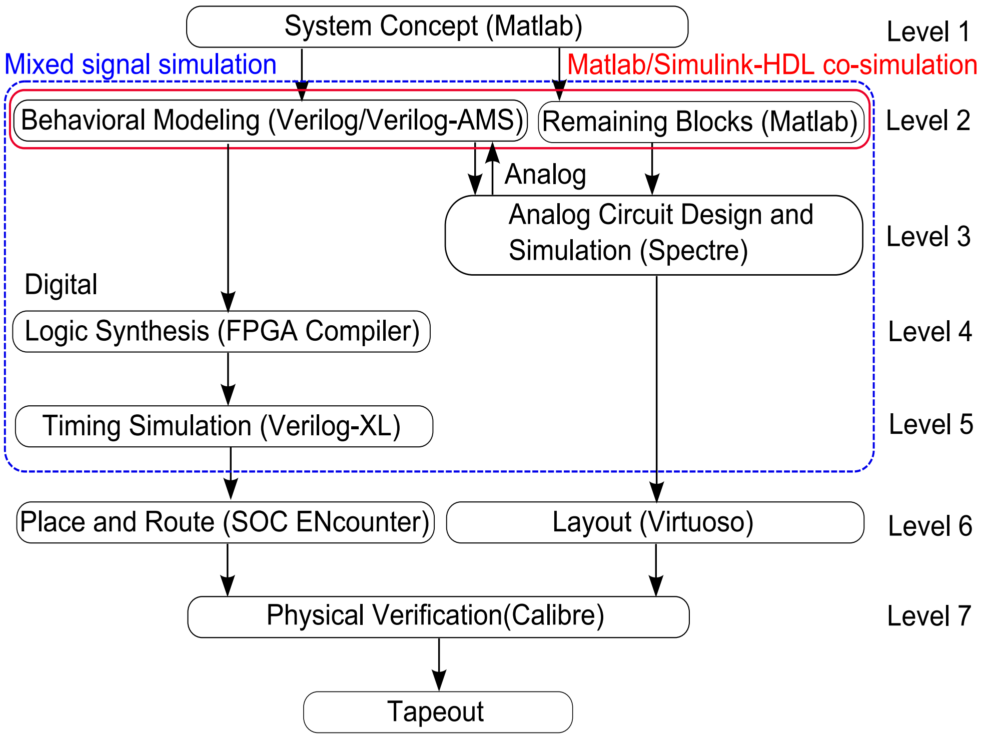
System Concept (Matlab)
Level 1
Mixed signal simulation
Matlab/Simulink-HDL co-simulation
Behavioral Modeling (Verilog/Verilog-AMS)
Remaining Blocks (Matlab)
Level 2
Analog
Analog Circuit Design and Simulation (Spectre)
Level 3
Digital
Logic Synthesis (FPGA Compiler)
Level 4
Timing Simulation (Verilog-XL)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

Figure A.2: Adopted Design Flow

method using Matlab/Simulink-HDL co-simulation for mixed-signal design is preferred. It benefits from both Matlab/Simulink and Verilog-AMS, and can be easily extended to include SystemC-AMS component models.

1.1.2 Design Flow

Figure A.2 illustrates the adopted design flow [105]. Compared to the conventional design flow, Matlab/Simulink-HDL co-simulation is added at level-2. Each level in presented design flow is explained in detail in the following section.

- **Level 1**: System concept is developed using Matlab/Simulink. At this level, system specifications and requirements are clarified and mathematical functional verification of system behavior is performed at the highest abstraction level.
- **Level 2**: Abstract implementation is done at block-level using Matlab/Simulink-HDL co-simulation. After system modeling, some design blocks are refined in Verilog-AMS. Co-simulation between Matlab and Verilog-AMS can accelerate block level implementation.
- **Level 3**: This level is the refinement of level-2. Verilog-AMS models of analog circuits can be replaced by Spectre netlist, if available. The Verilog-AMS and SPICE co-simulation could be extremely time-consuming for complex mixed-signal circuit

designs. Multiple iterations are needed between level-2 and level-3 to make Verilog-AMS model fine-tuned with real circuits.

- **Level 4**: Logic is synthesized at gate-level for digital circuits. This level can be moved to level-3 if co-simulation in level-3 is not executed.
- **Level 5**: Timing simulation takes place for digital circuits. By replacing the model with generated gate-level netlist in level-4, timing behavior is simulated with consideration of propagation delays due to logic elements and interconnecting.
- **Level 6**: Layout of digital circuits are obtained using place-and-route tools and layout of analog circuits is custom-made.
- **Level 7**: Designed layout is integrated for mixed-signal.

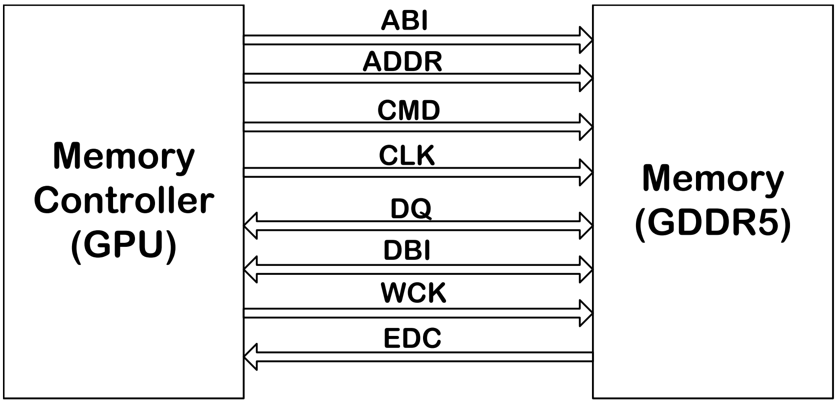

Figure A.3: Abstract Specification

1.1.3 Simulation Setup

An example of GDDR5 memory system [35] is used for a case study. GDDR5 is a standard for graphics synchronous Dynamic Random Access Memory (DRAM) which is currently the highest bandwidth memory available and accepted in the market. It is suitable for server applications and has minimized system cost by comparably low requirements for the PCB design. GDDR5 runs at GHz range frequency and provides additional features such as EDC, data and address bus inversion for power reduction (DBI/ABI) as depicted in Figure A.3. At GHz speed, communication between controller and memory becomes very challenging. For example, for a data rate of 5 Gbps, the max data window is 200 ps. In this small window (data-eye), data would be affected by transmitter/receiver noise, package/channel non-idealities or any kind of noise sources.

To find the center of the data-eye, JEDEC has come up with training sequences [35] as illustrated in Figure 2.12 which run at the initialization of the GDDR5 system to verify if data can be read/write successfully from/to the memory. In GDDR5 memory system, 2 clocks are provided: CK for command/address and WCK for data. WCK runs twice the CK frequency. After power up, different trainings start sequentially in the initialization phase. The address training is optional since it runs at lower frequency CK which is not as critical as data. In WCK2CK training, WCK needs to be aligned with CK on the memory side for synchronization issues. After that, all DQ pins are trained to guarantee that the sample clock can meet the center of the data-eye diagram with some predefined margin. Since write training depends on the right read training, the latest has to be operated first. Due to the high complexity in GDDR5 memory system, prototype in ASIC may take some time.

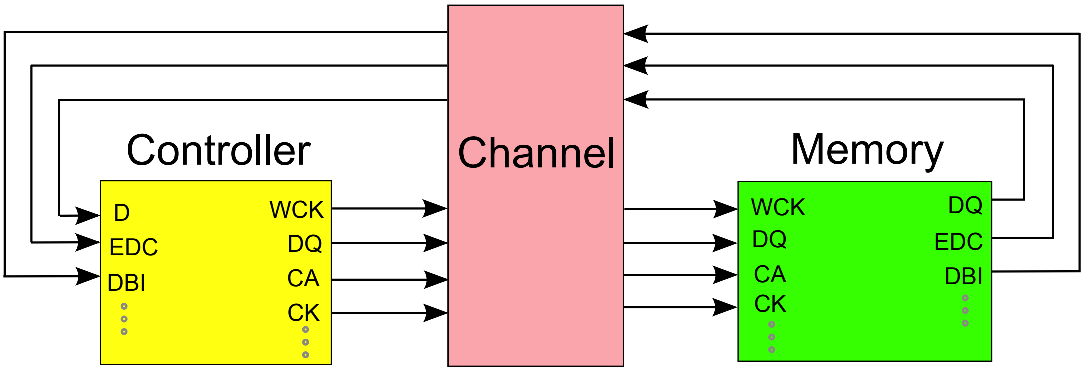

Figure A.4: Level 1: Matlab Simulation

Figure A.4 represents level-1: Matlab simulation setup. Write and read paths are separated to verify write and read commands independently. No external testbench is required since controller uses the built-in test. In both memory controller and memory, subsystems such as PLL, global clock distribution and transceivers are implemented in Matlab/Simulink. Communication channels in terms of S-Parameters are converted into Simulink models and Additive White Gaussian Noise (AWGN) in the channel is modeled as well.

After a successful run of MATLAB setup in level-1, level-2 starts with design blocks implementation. For example, the memory controller at block-level is illustrated in Figure A.5 in detail. It consists of memory controller core and physical (PHY) I/O Interface. The submodules in the memory controller core include scheduler, address mapper, I/O handler, Control and Settings Register (CSR), training engine, etc. Scheduler, as the main module in the core, has the task to control the other submodules for different modes for example read and write. Address mapper and I/O handler take care of the incoming/outgoing address and data. CSR contains all settings for the memory controller as well as for the attached memory devices. Training engine deals with different trainings after power up. In the PHY I/O interface, submodules such as FIFOs, serializers, deserializers, phase-interpolators and I/O drivers are implemented.

Figure A.6 depicts Matlab-HDL co-simulation setup. At this level, controller and memory are described in Verilog/Verilog-AMS. I/O drivers in controller/memory are analog and implemented in Verilog-AMS while all digital components in Verilog. The channel model is still represented by a Simulink model as that in level-1. Controller and memory HDL-views are generated by Cadence Incisive configuration in Matlab. The HDL-view provides two buttons, one for compilation and the other for elaboration. These two buttons avoid new generation of HDL-views if some changes are made in Verilog-AMS

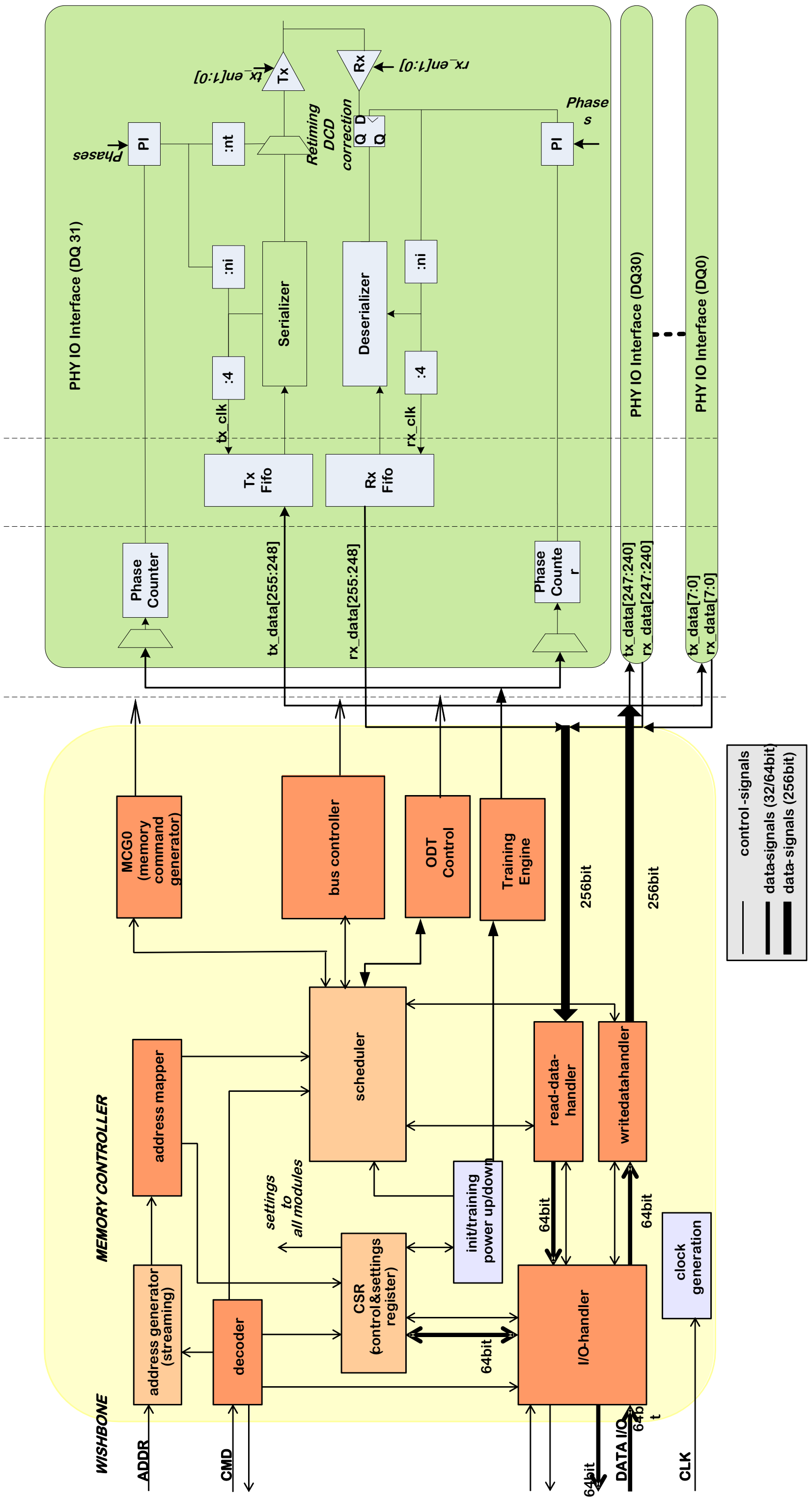

Figure A.5: Level 2: Controller Block-Level Specification

codes. In addition, communication between two HDL blocks needs data type conversion from integer to bus for vector signals such as 32 bits DQ.

Figure A.7 represents level-3: whole memory system in Cadence-AMS environment.

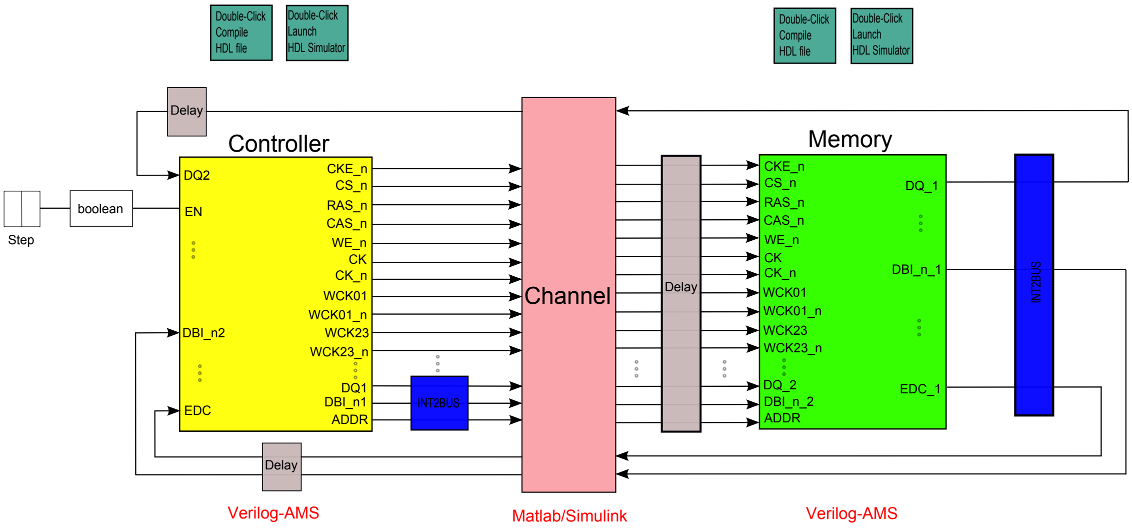

Figure A.6: Level 2: Matlab-HDL Co-simulation

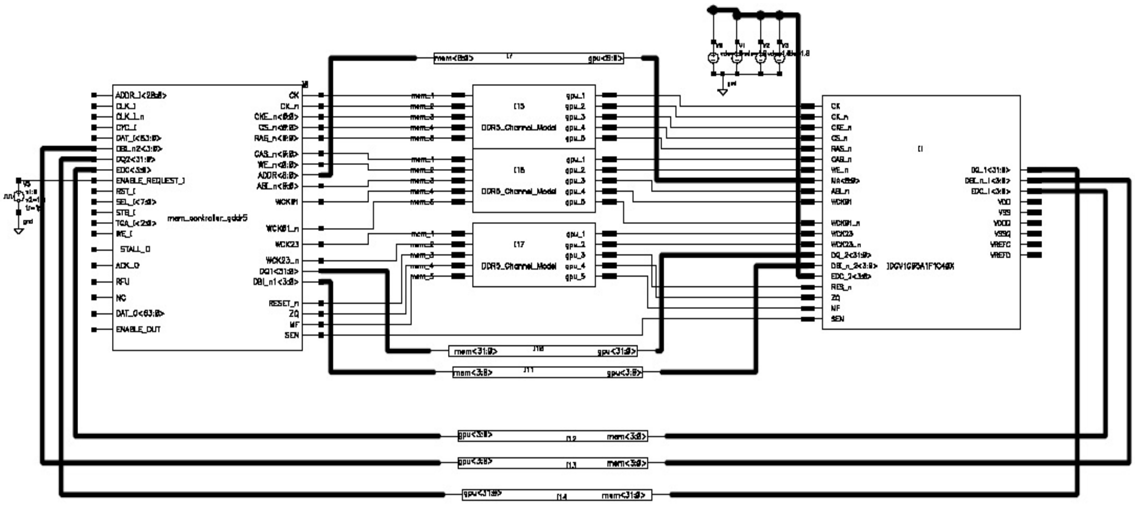

Figure A.7: Level 3: Verilog-AMS and SPICE co-simulation

The difference between Figure A.6 and Figure A.7 is only the channel models, all other block models remain the same. Channels are modeled using multiple transmission lines which used RLGC models derived from the characterization of the real package and the channel.

1.1.4 Simulation Results

The built-in test in the memory controller has followed the training sequences. After training, normal read and write commands are operated. The whole simulation time for the complete test run in Verilog-AMS is 11,000,000ps. The real execution time for

Matlab/Simulink-HDL co-simulation is in the range of several hours while for Verilog-AMS and SPICE co-simulation several days.

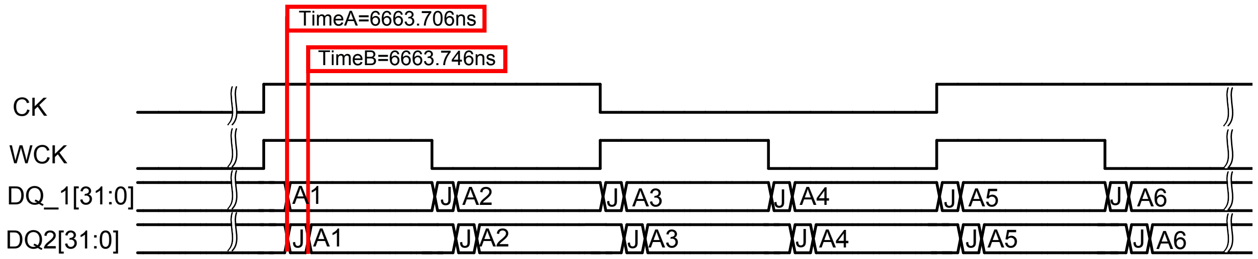

Figure A.8: DQ in read training

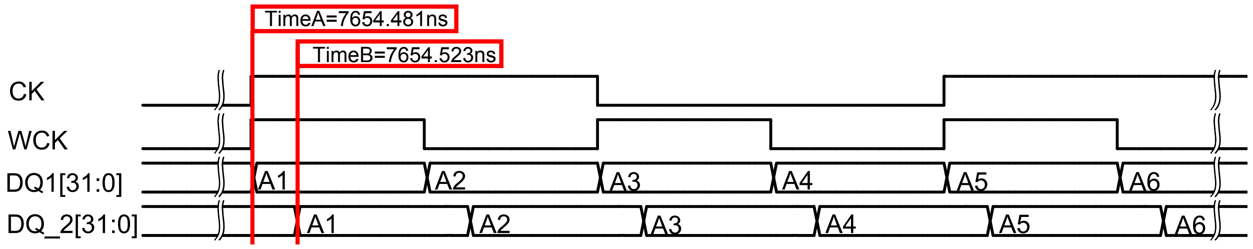

Figure A.9: DQ in write training

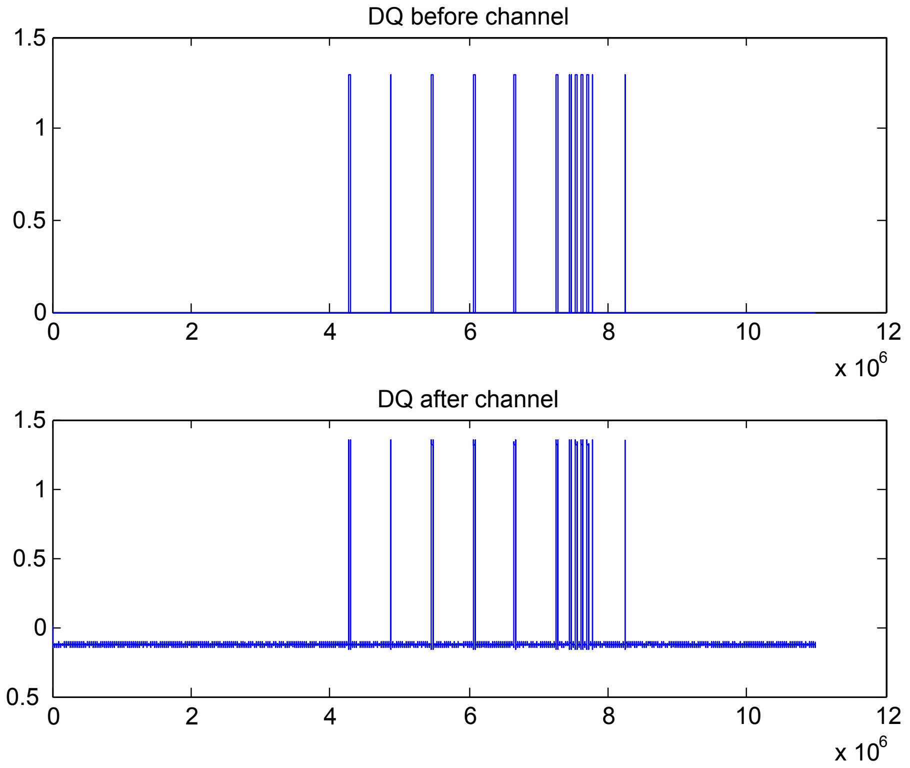

Figure A.10: DQ before and after channel

Figure A.8 and Figure A.9 illustrate the DQ path as an example in two different scenarios: read training and write training. In these figures, symbols A1 to A6 represent

Table A.1: Modeling approaches and performance trade-off

Parameter	**Verilog/ VHDL-AMS**	**SystemC-AMS**	**SPICE**	**Matlab/ Simulink-HDL**
Accuracy	Med-High	Med-High	High	Med-High
Abstraction Level	Any Abstraction Level	Any Abstraction Level	Transistor/ Netlist Level	Any Abstraction Level
Time to model	Low to Medium	Low	High	Low
Simulation Run Time	Low	Medium-High	High	Medium-High
Required Tools	Simulators	AMS libs with GCC/Simulators	Simulators	Simulators

random data while J for jitter in data modeled in the memory. As depicted in Figure A.6, for read training, data DQ_1 on the memory side is sent to DQ2 on the memory controller side through the channel while for write training, data DQ1 on the memory controller side is received by DQ_2 on the memory side. The total delay from transmitter to receiver for both read and write is around 40ps due to the channel. Figure A.10 illustrates a single bit DQ[0] out of 32 bits data before and after the channel. The noise in amplitude is caused by noise modeled in the channel.

Table A.1 compares various modeling approaches. The main parameters considered are accuracy, abstraction levels, time required for modeling the system, simulation time and required tools. SPICE has the highest accuracy compared to other approaches, but it also requires the highest time to model and large simulation run time. Matlab/Simulink-HDL co-simulation has a similar level of accuracy in comparison to Verilog/VHDL/SystemC-AMS, but it provides ready to use system-level tool-set which reduces the time required for modeling.

1.2 Conclusions

An efficient methodology for mixed-signal memory system using Matlab/Simulink-HDL co-simulation is presented in this chapter. As opposed to the conventional design flow, this approach employs Matlab/Simulink-HDL co-simulation framework for the system behavior verification to accelerate design time.

Appendix B

Implementation of PI-based PLL architecture

2.1 Introduction

Chapter 2 has described PI-based PLL architecture for multiple source-synchronous clocks synchronization technique which adopts the phase in both directions to reduce the time required for synchronization training. The desired PLL requirements are listed in Table B.1.

Table B.1: PLL specifications.

Parameter	Value
Process technology	low-power TSMC 65 nm CMOS logic
Supply voltage	Analog and digital supply 1.1 V ± 5%
Reference clock frequency range	100 MHz, 2.5% single ended
Ref. spread spectrum	0.5% down-spread
VCO working range	1.6 GHz - 3.6 GHz
VCO operational frequency range	1.6 GHz - 5GHz
Number of phases	4 with 90° delay and 8 with 45° delay
Output phase D-C	50% ± 1.5% (with 3 σ mismatch)
Long-term (500UI) p-p jitter simulated with a 100 kHz, 10% p-p supply square wave and 5% substrate square wave	15% output cycle
Period p-p jitter simulated with a 100kHz, 7.5 % p-p supply square wave and 1% substrate square wave (max)	± 1.5 % output cycle
Lock-in time	< 5000 reference cycles (50 μs)
Junction temperature (nom,min,max)	65° C, 0°, C 125° C
Closed loop -3dB bandwidth	0.5-0.1 ref. clock and programmable
Damping factor ζ	> 0.7
Output phase load	500 fF each
Power dissipation	35 mW @ 1.1V with load

2.2 Implementation

In this chapter, the implementation of PI-based PLL [115] will be described. Two different systems will be compared: one employs a PLL generating 4 phase while the other one utilizes a PLL producing 8 output phases. The figures and descriptions in the following sections deal for the sake of briefness with the four phases case.

The aim of this design is to find a compromise between power, area and phase error of the synchronization system. For this purpose, the performances of the system employing a 4-phases VCO are compared with those delivered by a system with an 8-phases VCO. In addition to this examination, an analysis of the variation of the phase error as a function of the bandwidth of the PLL is also carried out. To make the last investigation possible a programmable filter is employed. The PLL presented in Figure 2.22 is formed by the following components.

2.2.1 Phase-Frequency Detector

The Phase-Frequency Divider (PFD), on the top left of Figure 2.22, is a classic PFD described in a VHDL module. The pin for the reference frequency of the PFD is connected to the signal CLK, while the pin of the feedback frequency is connected to the Frequency Divider (FD). The PFD VHDL code is described as following:

```
library ieee;
use ieee.std_logic_1164.all;

--------------------------------------------------------------
-- Behavioural description of a 24-counter                  --
--------------------------------------------------------------

entity PFD is
  port(
        f_ref, f_vco : in std_logic;           -- counter intial value
        up, dn : out std_logic                 -- '1' if counter = 0, '0' otherwise
  );
end PFD;

architecture behavioural of PFD is
  signal rst_del, rst : std_logic := '1';
  signal tmp_up, tmp_dn : std_logic;

  begin
    proc_up : process (f_ref,rst)
        begin
        if (rst = '1') then
          tmp_up <= '0';
        elsif (f_ref'event and f_ref = '1') then
          tmp_up <= '1';
        end if;
    end process proc_up;

    proc_dn : process (f_vco,rst)
        begin
        if (rst = '1') then
          tmp_dn <= '0';
        elsif (f_vco'event and f_vco = '1') then
          tmp_dn <= '1';
        end if;
    end process proc_dn;

    up <= tmp_up;
    dn <= tmp_dn;
    rst_del <= tmp_up AND tmp_dn;
    rst <= rst_del after 10 ps;
end behavioural;
```

The signal *rst_del* was added to provide a delay in reset in order to reduce the effects of the CP dead-zone of the PFD. The minimum UP or DN pulse in the simulation is equal to 200 ps.

2.2.2 Charge-Pump

The outputs of the PFD are connected directly to the current CP, which works at the same supply voltage as the rest of the system. In order to minimize the current mismatch a replica-biased CP is employed and its schematic is shown in Figure B.1.

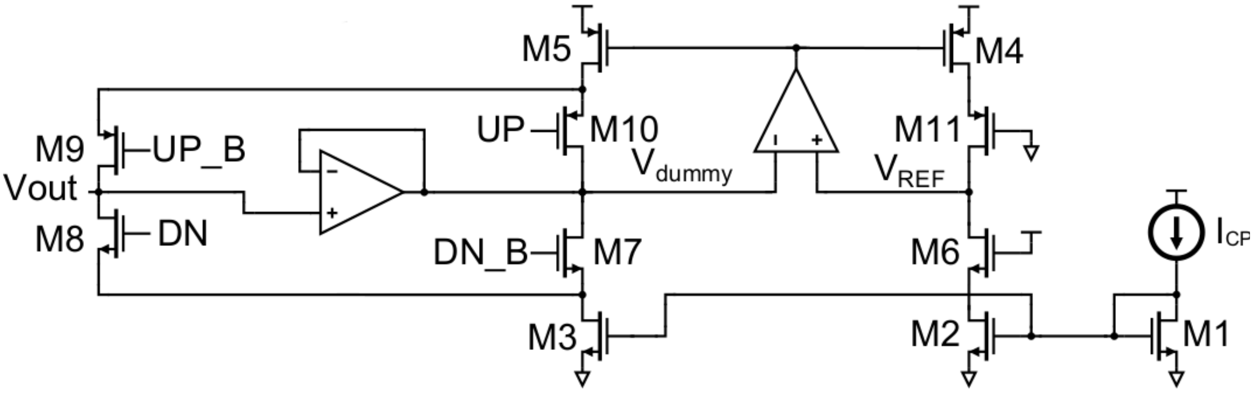

Figure B.1: Schematic of the charge-pump employed in this design.

To reduce the charge-injection phenomenon, an Operational Amplifier (OPAMP) is connected in a voltage-follower fashion between output and dummy switches. Doing so, charge injection is significantly reduced. The OPAMPs have a gain equal to 60 dB, a bandwidth of 10 MHz and a rail-to-rail input stage. The CP output current is injected into the loop filter. The kind of filter utilized in this design is the same shown in [59] but it also presents the feature to make the PLL work with three different bandwidth. This aspect will be discussed in detail later.

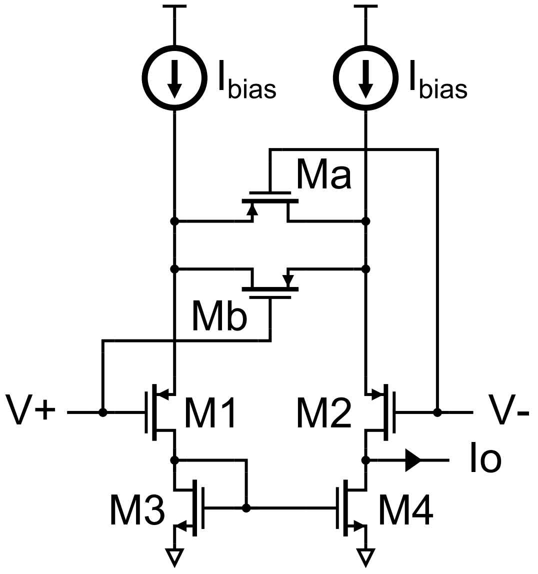

Figure B.2: Schematic of the sub-threshold OTA employed in this design.

2.2.3 Operational Transconductance Amplifier

The filter generates two output signals: one is the fine loop voltage and is sent directly to a V2I converter, while the second output is connected to the OTA input. To have the smaller coarse loop bandwidth, the ratio between G_m and C_I, as shown in Equation 2.1, needs to be designed as small as possible.

As shown by [60], a sub-threshold design is helpful to minimize the transconductance G_m of the OTA and at the same time keeping the physical implementation small. The schematic of the circuit is shown in Figure B.2. The differential pair of the OTA, formed by the transistors M1 and M2, is biased employing two current generators producing a current of 320 nA each. The transistors Ma and Mb, whose gates are connected to the differential input pins, are placed between the sources of the transistors of the differential pair.

This configuration for sub-threshold differential stages is proposed by [91]. However, in this work PMOS transistors are employed instead of NMOS transistors in order to reduce both transconductance and noise at minimum aspect ratio. Another advantage of this configuration is a higher linear range in the characteristic of the OTA. The measured transconductance G_m is 1.122 μS and the output resistance Z_{out} 39.84 MΩ.

2.2.4 Voltage-to-Current Converters

In the coarse loop, the current generated by the OTA loads the capacitance C_I and its voltage is converted into a current by a V2I converter (Figure 2.22). The architectures of the V2I converters implemented in this design are taken from [57]. The V2Is are employed because, according to [57], they can effectively reduce the alteration of phase margin and bandwidth of the system due to process variations on the filter resistor.

The V2I converter for the coarse-loop has the OTA integrator output voltage as input. The output of the circuit is the gate voltage -i.e. the current source bias voltage necessary for the CCO to generate a frequency proportional to the voltage imposed by the control circuit. The gain of the coarse-loop converter needs to be maximized as required by the DLPLL design.

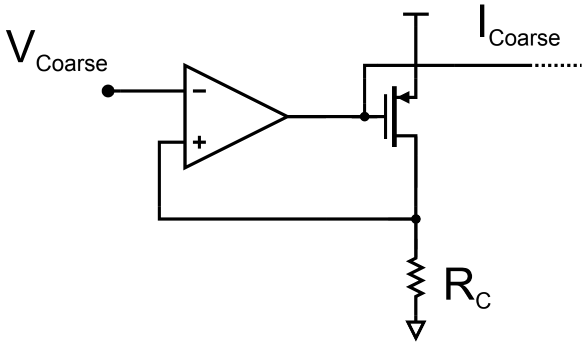

Figure B.3: Schematic of the coarse loop V2I converter.

The circuit, shown in Figure B.3, is composed by:

- an OPAMP
- a PMOS transistor of the same size of the VCO coarse-loop current drivers
- a resistor R_C as large as the filter resistor R_z

This approach delivers two main advantages:

- the linear characteristic of the V2I converter compensates for the VCO non-linear gain characteristic
- matching the V2Is resistors with the resistor of the filter, makes the loop bandwidth and phase margin virtually unaffected by process variations of the resistor of the filter.

The designed coarse-loop V2I performance parameter are described in Table B.2.

Table B.2: Coarse-loop V2I converter performance parameter.

R_C	8 kΩ
Gain	10^{-4}A/V
bandwidth	52.515 MHz

However, the same V2I design is not suitable for the fine loop as the pole of the coarse loop V2I converter would deteriorate the phase margin of the entire system. For this reason the transconductor design is utilized, as the pole introduced by it, does not affect the phase margin as well as the bandwidth of the system.

The fine-loop V2I converter shown in Figure B.4 takes the input voltage from the node connecting the CP, the capacitor C_2 and the resistor R_z. The output is the gate voltage necessary to make the CCO produce a current proportional only on a narrow range to the input voltage. It is important to remark that the aim of the fine-loop V2I converter design is to minimize its gain as a requirement of the DLPLL design. The circuit is composed of two source-follower transistors with active load.

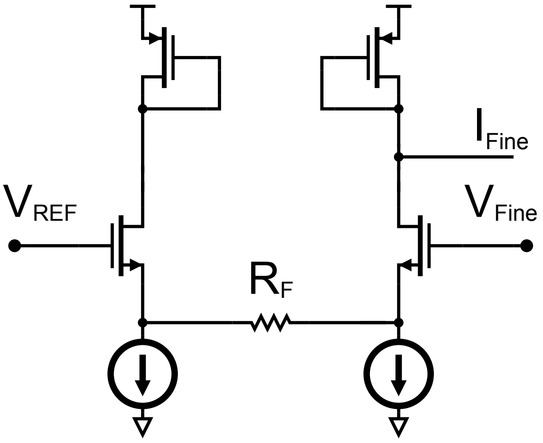

Figure B.4: Schematic of the fine loop V2I converter.

The Fine-loop V2I performance parameters are described in the Table B.3.

Table B.3: Fine-loop V2I converter performance parameter.

R_F	8 kΩ
Gain	17μA/V
bandwidth	2.9 GHz

2.2.5 Current Controlled Oscillator

The CCO has the task to integrate the filtered phase error and generate the proper output frequency. Such element has the following requirements:

- duty cycle: 50% ± 2% with 3 σ mismatch
- minimum load capacitance of 500 pF on each phase
- frequency range 1.2-6 GHz
- t_{rise} and t_{fall} have to be smaller than the 25 % of the period

The phase error voltage is converted into a current by the voltage-to-current converter, and then sent to the CCO and the output frequency is generated. The design of the CCO (Figure B.5) is based on a wide frequency range ring oscillator and duty-cycle correctors as presented by [57].

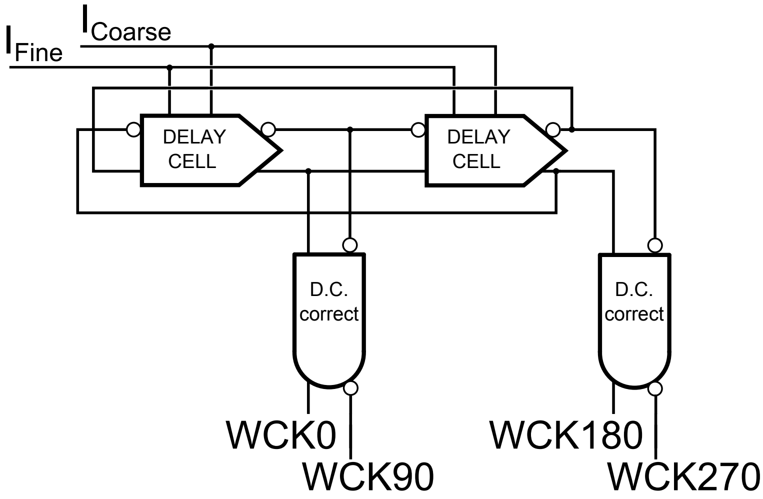

Figure B.5: Schematic of the four phases CCO.

The ring oscillator is made up by delay cells (Figure B.6) introduced by [92] and modified by [57], in order to have two control signals.

Delay cell does not produce equal rise and fall time. Due to the difference between rise-time (fixed by the bias current) and the fall-time (fixed by the circuit), a large D-C error is still present at the CCO output. To correct the signal, each differential output is sent to a dual-path differential-to-single ended amplifier and then to a crossed couple of inverters which forms a duty cycle corrector shown in Figure B.7.

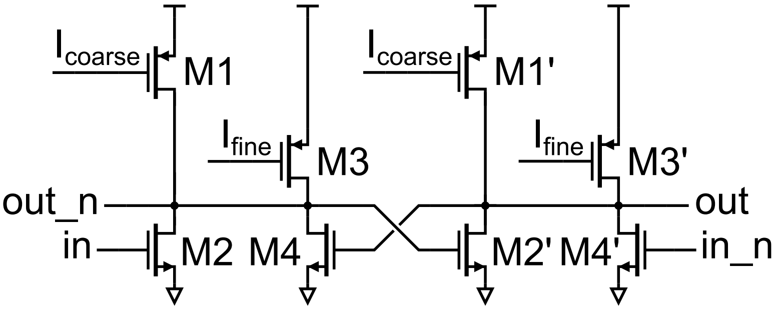

Figure B.6: Schematic of the delay cell.[57]

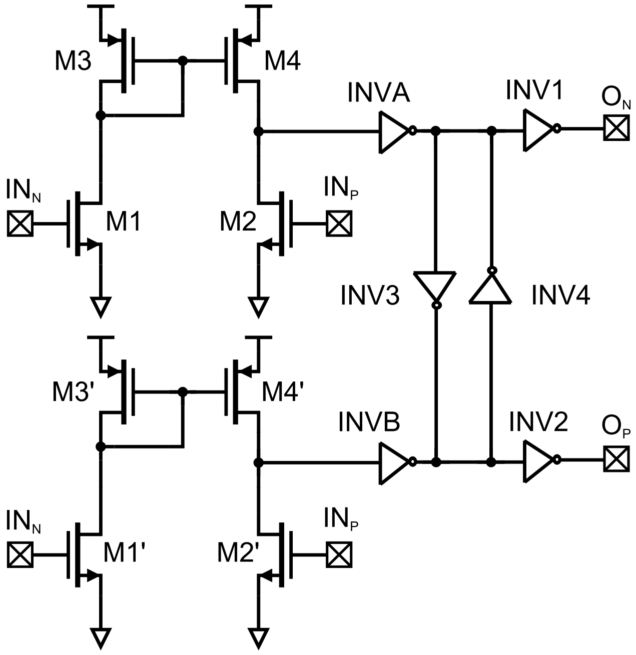

Figure B.7: Duty cycle corrector [92].

The CCO generating eight phases consists of the same components connected in the same fashion but with double number. A buffer stage, able to drive a maximum load capacitance of 500 fF up to 6 GHz operating frequency, is connected to each output of the CCO.

The performances obtained from the circuit are shown in Table B.4. The output frequency satisfies the specifications both for maximum and minimum frequency.

2.2.6 Phase Selector and Frequency Divider

In order to make the PLL shift the output phase, PS is employed. Such circuit is able to choose one among the four or eight VCO output signals either with delayed or with advanced phase with respect to the current one. However, only adjacent outputs can be selected, e.g. from *ph_90* to either *ph_0* or *ph_180*.

The PS is composed by:

Table B.4: CCO output performances.

I_{bias} μA	f GHz	D-C error
5	1.044	0.259%
10	1.793	0.26%
15	2.454	0.415%
20	3.076	0.371%
25	3.674	0.002%
30	4.258	0.238%
35	4.834	0.217%
40	5.427	0.548%
45	5.975	0.472%
50	6.543	0.065%
60	7.66	1.34%

- a multiplexer to select the phase;
- a counter to switch to both higher and lower phases as long as the output is not the desired one;
- a 2-to-4 bits decoder;
- an inverting output buffer;
- a register for the control signals to avoid the switching to happen at any moment.

As shown in Figure B.8, the transition to a signal with a larger delay causes a glitch to happen on the selector output signal *sel_out*. This results into an erroneous count of the FD which makes the PLL to loose the lock. On the contrary, the transition to a waveform with a smaller delay does not result into glitches.

To solve this issue the phase selector needs to use a synchronization circuit, as presented in [93], in order to properly select the VCO outputs, a signal with a delay larger than the current one is selected. The output of the PS is sent to a FD. The frequency division ratio is equal to 2, and it is necessary to generate the signal WCK needed for the functioning of the synchronization system.

2.2.7 Programmable Filter

Another feature of the PLL implemented in this design is the filter programmability. Three different bandwidths can be set by selecting three different resistors for the filter. The capacitors are instead kept constant in order to minimize the area of the programmable filter. Ideally the bandwidths of 5, 7.5 and 10 MHz are meant to be selected. It is also important to mention that, according to the design in [57], each V2I converter employs a resistor whose value is equal to the resistor of the filter. Three equal resistors have to be

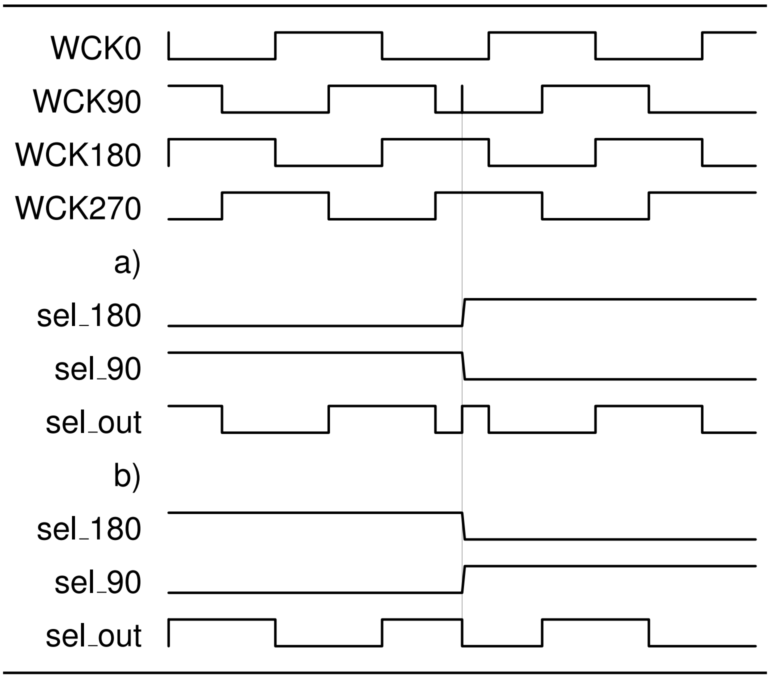

Figure B.8: Waveforms showing the difference between the selection of an anticipated *a)* or a delayed *b)* VCO output in respect to the current one.

employed to make the PLL system work with one desired bandwidth. Since in this system three different bandwidths can be selected and each case utilizes three resistors, the system contains nine resistors. Another consequence of the employment of V2I converters is that the coarse and fine loop gains are affected by the resistance value in the filter which is equal to the resistance value used in the V2I converter. The gains of the components in the loop were measured and are shown in Table B.5.

Table B.5: Parameters of the components of the PLL. The VCO gains were measured employing a filter resistor of 7 kΩ.

# outputs	4	8
I_{cp}	10.416 μA	76.11 μA
K_c	8.6 GHz/V	2.6 GHz/V
K_f	1.3 GHz/V	0.176 GHz/V
G_m	1.122 μS	1.122 μS
C_I	34 pF	20 pF

The filter was designed in the following fashion. First three different filters are designed for the three different bandwidths, i.e. 5, 7.5 and 10 MHz, and a phase margin of 70°. At this point the capacitors for the bandwidth of 7.5 MHz are chosen as a compromise between the two other cases. After this, an iterative design procedure starts. Indeed, a variation of the filter resistance not only varies the PLL bandwidth, but also the VCO gains.

For this reason first a new resistance value for the desired bandwidth and phase margin is set and, then, the new VCO fine loop gain K_f is measured. With the new value of K_f a new resistor value R_z is found and set. The procedure is repeated until suitable values of phase margin and bandwidth are obtained. Although it is not possible to achieve both desired bandwidths and wanted phase margins, however the designer can find a suitable compromise within the specification boundary. The design of the systems has as goal phase margins larger than 65° and resistor values seized in such a way that the worst case process variations cause the variation ranges not to overlap. The attained resistance values and parameters are shown in Table B.6.

Table B.6: Laplace-domain features of the PLL systems and respective resistance values.

System	ϕ_M deg	-3dB bandwidth MHz	R_z kΩ
1	69.1965	6.4	4.1
2	72.3844	8.67	5.6
3	70.7082	10	7

The quantity reported in the second column of Table B.6 is the phase margin measured in degrees, then the closed-loop bandwidth is reported in the third column and the obtained resistance values are listed in the fourth column. The criteria on phase margin and resistance are verified in this design to show that the bandwidth and phase margin are determined by the fine loop alone (Figure 2.23).

2.2.8 Comparator

The comparator receives four EDC signals from the memory. When EDC switches from 'low' to 'high' and it keeps this value for the following two pulses, the signal *l2e* presents a pulse, whose length is equal to the reference clock. On the contrary *e2l* produces a pulse when EDC shifts from 'high' to 'low' and EDC remains at this level for the following two pulses. Hence, the pulses occur on the outputs three reference clock cycles after the transition on EDC. The four different lines are sampled at a reference clock frequency by four flip-flops. The comparison saves the two previous sampled values in eight flip-flops connected downstream. This design has the advantage to insure the stability of the transition on EDC in order to provide a cleaner signal to the digital control.

2.2.9 Digital Control

The digital control selects a phase at the VCO output according to the signals produced by the comparator. In addition to *e2l* and *l2e*, the counter is also varied by a clock whose frequency is smaller than the CLK one. This additional counting is necessary in order to make the PLL approach the needed delay and ultimately to have the phase selector switching solely among two phases. The frequency division ratio depends on the number of phases generated by the VCO and by the reference frequency. The value is chosen in such a way that, once the system starts switching with only two phases, the counter needs to be varied only by the control signals. Therefore, the variation forced by the FD, which is reset each time either *e2l* or *l2e* is 'high', has no chance to occur.

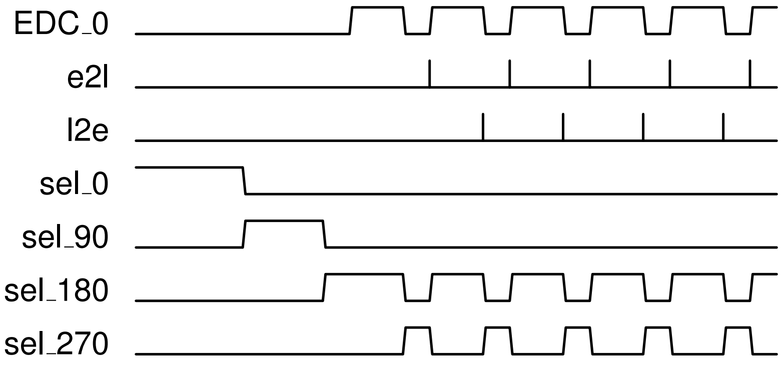

Figure B.9: Waveforms of the control signals after the lock-in of the PLL.

In Figure B.9 are shown the signals that control the loop. In the first part the phase selector is incremented by the FD and the length of the pulse is fixed. As soon as the logic level of EDC changes, the phase selector is varied according to the signals *l2e* and *e2l* generated by the comparator. Henceforth the increment due to the FD does not occur anymore.

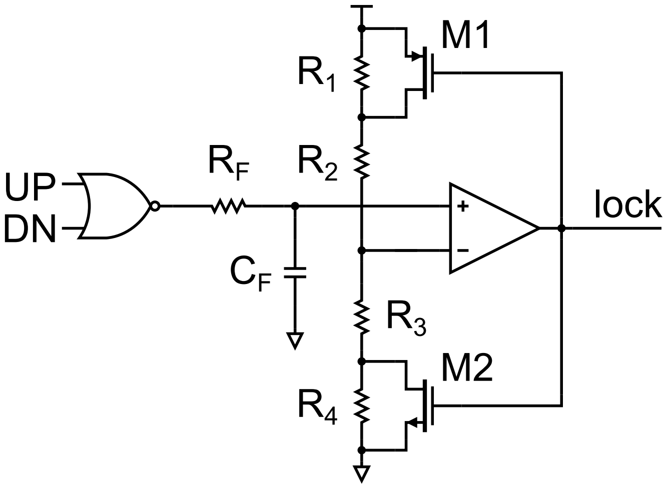

Figure B.10: Schematic of the lock detector.

2.2.10 Lock Detector

The lock detector is needed in order to start the aligning system only after the PLL has locked-in. It is based on an analog architecture shown in Figure B.10.

The circuit lies in integrating the output of a NOR gate driven by the *up* and *dn* signals, generated by the PFD inside the PLL. Once the filter output has crossed the threshold voltage, the *lock* signal acquires the 'high' voltage level. The value of the resistor R_F is 33 KΩ and the value of the capacitor C_F is 10 pF. At the output of the filter a non-inverting

Schmitt-trigger is used as a voltage comparator in order to avoid very quick commutations on the *lock* signal in proximity of the threshold voltage. The schematic of this component was taken from [94]. The resistors were designed in order to have the lower threshold voltage equal to 606 mV and the upper threshold voltage equal to 980 mV.

References

[1] *Cisco VNI Mobile 2014*. www.cisco.com.

[2] L. Atzori, A. Iera, and G. Morabito. “From ”smart objects” to ”social objects”: The next evolutionary step of the internet of things”. In: *Communications Magazine, IEEE* 52.1 (Jan. 2014), pp. 97–105.

[3] O. Bello and S. Zeadally. “Intelligent Device-to-Device Communication in the Internet of Things”. In: *Systems Journal, IEEE* PP.99 (Jan. 2014), pp. 1–11.

[4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. “Internet of Things for Smart Cities”. In: *Internet of Things Journal, IEEE* PP.99 (2014), pp. 1–1.

[5] *Zukunftsprojekt Industrie 4.0*. http://www.bmbf.de/de/19955.php.

[6] C. Tsai, C. Lai, M. Chiang, and L.T. Yang. “Data Mining for Internet of Things: A Survey”. In: *Communications Surveys Tutorials, IEEE* 16.1 (Feb. 2014), pp. 77–97.

[7] J. Huang, Y. Meng, X. Gong, Y. Liu, and Q. Duan. “A Novel Deployment Scheme for Green Internet of Things”. In: *Internet of Things Journal, IEEE* PP.99 (Jan. 2014), pp. 1–1.

[8] P. C. Broekema, R. V. Nieuwpoort, and H. E. Bal. “ExaScale High Performance Computing in the Square Kilometer Array”. In: *Proceedings of the Workshop on High-Performance Computing for Astronomy Date*. Astro-HPC ’12 (2012), pp. 9–16.

[9] *International Technology Roadmap for Semiconductors (ITRS) 2010*. http://public.itrs.net.

[10] 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), IEEE International* (Feb. 2010).

[11] M. Chen, Y. Shih, C. Lin, H. Hung, and J. Lee. “A Fully-Integrated 40-Gb/s Transceiver in 65-nm CMOS Technology”. In: *Solid-State Circuits, IEEE Journal of* 47.3 (Mar. 2012), pp. 627–640.

[12] J.E. Jaussi, G. Balamurugan, D.R. Johnson, B. Casper, A. Martin, J. Kennedy, N. Shanbhag, and R. Mooney. “8*Gb/s* source-synchronous I/O link with adaptive receiver equalization, offset cancellation, and clock de-skew”. In: *IEEE Journal of Solid-State Circuits* 40.1 (Jan. 2005), pp. 80–88.

[13] T.O. Dickson, Y. Liu, S.V. Rylov, B. Dang, C.K. Tsang, P.S. Andry, J.F. Bulzacchelli, H.A. Ainspan, X. Gu, L. Turlapati, M.P. Beakes, B.D. Parker, J.U. Knickerbocker, and D.J. Friedman. “An 8x 10-Gb/s Source-Synchronous I/O System Based on High-Density Silicon Carrier Interconnects”. In: *Solid-State Circuits, IEEE Journal of* 47.4 (Apr. 2012), pp. 884–896.

[14] K. Maruko, T. Sugioka, H. Hayashi, Z. Zhou, 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), IEEE International* (Feb. 2010).

[15] 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 3A-Oversampling 2-Threshold Eye-Tracking CDR Circuit for -36.8dB-loss Backplane”. In: *Solid-State Circuits Conference, ISSCC. Digest of Technical Papers. IEEE International* (Feb. 2008).

[16] 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, IEEE Symposium on* (June 2008).

[17] W. Chen, C. Tsai, C. Chang, Y. Peng, F. Hsueh, T. Yu, J. Chien, W. Huang, C. Lu, M. Lin, C. Fu, S. Yang, C. Wong, W. Chen, C. Wen, L. 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), IEEE* (Sept. 2010).

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

[19] J. Zerbe, B. Daly, L. Luo, W. Stonecypher, W. Dettloff, J.C. Eble, T. Stone, J. Ren, B. Leibowitz, M. Bucher, P. Satarzadeh, Q. Lin, Y. Lu, and R. Kollipara. “A 5 Gb/s Link With Matched Source Synchronous and Common-Mode Clocking Techniques”. In: *Solid-State Circuits, IEEE Journal of* 46.4 (Apr. 2011), pp. 974–985.

[20] F. Spagna, L. Chen, M. Deshpande, Y. Fan, D. Gambetta, S. Gowder, S. Iyer, R. Kumar, P. Kwok, R. Krishnamurthy, C. Lin, R. Mohanavelu, R. Nicholson, J. Ou, M. Pasquarella, K. Prasad, H. Rustam, L. Tong, A. Tran, J. Wu, and X. 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),IEEE International* (Feb. 2010).

[21] 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* 45.12 (Dec. 2010), pp. 2838–2849.

[22] 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* 47.11 (Nov. 2012), pp. 2627 –2642.

[23] D. Walter, S. Hoppner, H. Eisenreich, G. Ellguth, S. Henker, S. Hanzsche, R. Schuffny, M. Winter, and G. Fettweis. “A source-synchronous 90Gb/s capacitively driven serial on-chip link over 6mm in 65nm CMOS”. In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), IEEE International* (Feb. 2012).

[24] *IEEE SCI standard*. `http://standards.ieee.org/findstds/standard/1596-1992.html`.

[25] *SGI interface*. `http://www.sgi.com`.

[26] *HIPPI-6400-PH*. `http://hsi.web.cern.ch/HSI/gsn/gsnhome.htm`.

[27] *Quickpath Interconnect*. `http://www.intel.com/content/www/us/en/io/quickpath-technology/quick-path-interconnect-introduction-paper.html`.

[28] *HyperTransport Interconnect*. `http://www.hypertransport.org`.

[29] *Samsung GDDR products*. `http://www.samsung.com`.

[30] *Hynix GDDR products*. `http://www.skhynix.com`.

[31] *Micron GDDR products*. `http://www.micron.com`.

[32] *Rambus XDR products*. `http://www.rambus.com`.

[33] *Micron Technical Notes*. `http://www.micron.com/products/support/technical-notes`.

[34] D. Oh and C. Yuan. In: Prentice Hall, 2011.

[35] *GDDR5 JEDEC Standard*. `http://www.jedec.org/standards-documents/docs/jesd212`.

[36] K. Ha, L. Kim, S. Bae, K. Park, J. Choi, Jun Y., and K. Kim. “A 0.13- μm CMOS 6 Gb/s/pin Memory Transceiver Using Pseudo-Differential Signaling for Removing Common-Mode Noise Due to SSN”. In: *Solid-State Circuits, IEEE Journal of* 44.11 (Nov. 2009), pp. 3146–3162.

[37] S. Mukherjee, D. Oh, 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), IEEE 21st Conference on* (Oct. 2012).

[38] A. Amirkhany, W. Beyene, C. Madden, A. Abbasfar, D. Secker, D. Oh, M. Hekmat, R. Schmitt, and Chuck Yuan. “On overcoming the limitations of single-ended signaling for graphics memory interfaces”. In: *Solid State Circuits Conference (A-SSCC), IEEE Asian* (Nov. 2011).

[39] D. Oh, S. Chang, C. Madden, J. 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, (EPEPS) IEEE 18th Conference on* (Oct. 2009).

[40] R. Lugannani. “Intersymbol interference and probability of error in digital systems”. In: *Information Theory, IEEE Transactions on* 15.6 (Nov. 1969), pp. 682–688.

[41] W. Xiang and S.S. Pietrobon. “On the capacity and normalization of ISI channels”. In: *Information Theory, IEEE Transactions on* 49.9 (Sept. 2003), pp. 2263–2268.

[42] 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), Symposium on* (June 2011).

[43] 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, X. Shi, S. Fazeel, G.S. Srinivas, S. Zhang, C. Tran, A. Vaidyanath, K. Vyas, M. Jain, K.K. Chang, and Xingchao Yuan. “A Tri-Modal 20-Gbps/Link Differential/DDR3/GDDR5 Memory Interface”. In: *Solid-State Circuits, IEEE Journal of* 47.4 (Apr. 2012), pp. 926–937.

[44] D. Lee, M. Kim, and I.L. Markov. “Low-power clock trees for CPUs”. In: *Computer-Aided Design (ICCAD), IEEE/ACM International Conference on* (Nov. 2010).

[45] D. Lee and I.L. Markov. “Contango: Integrated optimization of SoC clock networks”. In: *Design, Automation Test in Europe Conference Exhibition (DATE)* (Mar. 2010).

[46] S. Hu, C. Jia, K. Huang, C. Zhang, X. Zheng, and Z. Wang. “A 10Gbps CDR based on phase interpolator for source synchronous receiver in 65nm CMOS”. In: *Circuits and Systems (ISCAS), IEEE International Symposium on* (May 2012).

[47] 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: *IEEE Journal of Solid-State Circuits* 40.3 (Mar. 2005), pp. 736–743.

[48] B. Abiri, R. Shivnaraine, A. Sheikholeslami, H. Tamura, and M. Kibune. “A 1-to-6Gb/s phase-interpolator-based burst-mode CDR in 65nm CMOS”. In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), IEEE International* (Feb. 2011).

[49] N. Chowdhury, J. Wight, C. Mozak, and N. Kurd. "Intel Core i5/i7 QuickPath Interconnect receiver clocking circuits and training algorithm". In: *VLSI Design, Automation, and Test (VLSI-DAT), International Symposium on* (Apr. 2012).

[50] M. Loh and A. Emami-Neyestanak. "A 3x9 Gb/s Shared, All-Digital CDR for High-Speed, High-Density I/O". In: *IEEE Journal of Solid-State Circuits* 47.3 (Mar. 2012), pp. 641–651.

[51] D. Oh, A. Vaidyanath, C. Madden, Y. Frans, and W. Kim. "Optimizing the timing center for high-speed parallel buses". In: *Electronic Components and Technology Conference (ECTC), IEEE 62nd* (May 2012).

[52] *Hynix GDDR5 Datasheet*. http://www.hynix.com/datasheet/pdf/graphics/H5GQ1H24AFR(Rev1.0).pdf.

[53] *TSMC 65nm Library*. http://www.tsmc.com/english/dedicatedFoundry/technology/65nm.htm.

[54] C. Hsu, C.Y. Lau, and M.H. Perrott. "A Delay-Locked Loop using a Synthesizer-based Phase Shifter for 3.2 Gb/s Chip-to-Chip Communication". In: *Solid-State Circuits Conference, (ESSCIRC) Proceedings of the 32nd European* (Sept. 2006).

[55] S. Callender and A.M. Niknejad. "A phase-adjustable Delay-Locked Loop utilizing embedded phase interpolation". In: *Radio Frequency Integrated Circuits Symposium (RFIC), IEEE* (June 2011).

[56] P. Larsson. "A 2-1600-MHz CMOS clock recovery PLL with low-Vdd capability". In: *IEEE Journal of Solid-State Circuits* 34.12 (1999), pp. 1951–1960.

[57] P. K. Hanumolu, G. Wei, and U. Moon. "A Wide-Tracking Range Clock and Data Recovery Circuit". In: *IEEE Journal of Solid-State Circuits* 43.2 (2008), pp. 425–439.

[58] M. Zanuso, S. Levantino, C. Samori, and A.L. Lacaita. "A Wideband 3.6 GHz Digital $\Delta\Sigma$ Fractional-N PLL With Phase Interpolation Divider and Digital Spur Cancellation". In: *IEEE Journal of Solid-State Circuits* 46.3 (Mar. 2011), pp. 627–638.

[59] 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, Proceedings of the IEEE* (Oct. 2004).

[60] R. Nonis, N. Da Dalt, P. Palestri, and L. Selmi. "Modeling, design and characterization of a new low jitter analog dual tuning LC-VCO PLL architecture". In: *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on* 4 (May 2004), IV–553–6 Vol.4.

[61] D.J. Lee and I.L. Markov. "Multilevel Tree Fusion for Robust Clock Networks". In: *IEEE/ACM International Conference on Computer-Aided Design* (Nov. 2011).

[62] C. N. Sze. "ISPD 2010 High Performance Clock Network Synthesis Contest: Benchmark Suite and Results". In: *Proceedings of the 19th International Symposium on Physical Design*. ISPD '10 (2010).

[63] T. Chao, Y. Hsu, J. Ho, and A.B. Kahng. "Zero skew clock routing with minimum wirelength". In: *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on* 39.11 (Nov. 1992), pp. 799–814.

[64] L.P.P.P.V. Ginneken. "Buffer placement in distributed RC-tree networks for minimal Elmore delay". In: *Circuits and Systems, 1990, IEEE* (1990).

[65] S. Hu, C.J. Alpert, J. Hu, S. Karandikar, Z. Li, W. Shi, and C. N. Sze. "Fast Algorithms for Slew Constrained Minimum Cost Buffering". In: *Proceedings of the 43rd Annual Design Automation Conference*. DAC (2006).

[66] H. Bakoglu. In: Reading: Addison-Wesley, 1990.

[67] S. Krishnamoorthy. "The Upcoming Golden Age of Placement Research". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2012).

[68] M.-J.E. Lee, W.J. Dally, and P. Chiang. "Low-power area-efficient high-speed I/O circuit techniques". In: *IEEE Journal of Solid-State Circuits* 35.11 (Nov. 2000), pp. 1591–1599.

[69] L. Raghavan and T. Wu. "Architectural Comparison of Analog and Digital Duty Cycle Corrector for High Speed I/O Link". In: *VLSI Design, (VLSID) 23rd International Conference on* (Jan. 2010).

[70] Y. Min, C. Jeong, K. Kim, W.H. Choi, J. Son, C. Kim, and S. Kim. "A 0.31-1 GHz Fast-Corrected Duty-Cycle Corrector With Successive Approximation Register for DDR DRAM Applications". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20.8 (Aug. 2012), pp. 1524–1528.

[71] S. Kao and S. Liu. "A Wide-Range All-Digital Duty Cycle Corrector with a Period Monitor". In: *Electron Devices and Solid-State Circuits, (EDSSC) IEEE Conference on* (Dec. 2007).

[72] S. Kao and Y. You. "Clock buffer with duty cycle corrector". In: *SOC Conference (SOCC), IEEE International* (Sept. 2010).

[73] H. Huang, C. Liang, and W. Chiu. "1-99% input duty 50% output duty cycle corrector". In: *Circuits and Systems, (ISCAS) IEEE International Symposium on* (May 2006).

[74] R. Mehta, S. Seth, S. Shashidharan, B. Chattopadhyay, and S. Chakravarty. "A programmable, multi-GHz, wide-range duty cycle correction circuit in 45nm CMOS process". In: *ESSCIRC, Proceedings of the* (Sept. 2012).

[75] S. Han and J. Kim. "Hybrid duty-cycle corrector circuit with dual feedback loop". In: *Electronics Letters* 47.24 (2011), p. 1311.

[76] J.H. Wu, J.H. Gu, L.Z. Zhang, and M. Zhang. "Full-MOSFET mixed-mode duty cycle corrector". In: *Electronics Letters* 47.19 (2011), p. 1067.

[77] J. Gu, J. Wu, D. Gu, M. Zhang, and L. Shi. “All-Digital Wide Range Precharge Logic 50% Duty Cycle Corrector”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20.4 (Apr. 2012), pp. 760–764.

[78] J. Ke, S. Huang, and D. Kwai. “A high-resolution all-digital duty-cycle corrector with a new pulse-width detector”. In: *Electron Devices and Solid-State Circuits (EDSSC), IEEE International Conference of* (Dec. 2010).

[79] D. Shin, K.J. Na, D. Kwon, J.H. Kang, T. Song, H.D. Jung, W.Y. Lee, K.C. Park, J.H. Park, Y.S. Joo, J.H. Cha, Y. Jung, Y. Kim, D. Han, B.J. Choi, G.I. Lee, J.H. Cho, and Y.J. Choi. “Wide-range fast-lock duty-cycle corrector with offset-tolerant duty-cycle detection scheme for 54nm 7Gb/s GDDR5 DRAM interface”. In: *VLSI Circuits, Symposium on* (June 2009).

[80] H. Huang, C. Liang, and S. Sun. “Low-power 50% duty cycle corrector”. In: *Circuits and Systems, (ISCAS) IEEE International Symposium on* (May 2008).

[81] C.Y. Chen, M. Le, and K.Y. Kim. “A low power 6-bit flash ADC with reference voltage and common-mode calibration”. In: *VLSI Circuits, IEEE Symposium on* (June 2008).

[82] C. Angerer, R. Langwieser, and M. Rupp. “Evaluation and exploration of RFID systems by rapid prototyping”. In: *Personal and Ubiquitous Computing* 16, Issue 3 (Mar. 2012), pp. 309–321.

[83] B. Oraw, V. Choudhary, and R. Ayyanar. “A cosimulation approach to model-based design for complex power electronics and digital control systems”. In: *SCSC* (2007).

[84] W. Hassairi, M. Bousselmi, and M. Abid. “The co-simulation interface SystemC/-Matlab applied in JPEG algorithm”. In: *ReCoSoC* (2011).

[85] B. Gestner and D.V. Anderson. “Automatic Generation of ModelSim-Matlab Interface for RTL Debugging and Verification”. In: *MWSCAS* (2007).

[86] P. Frey and D. O’Riordan. “Verilog-AMS: Mixed-signal simulation and cross domain connect modules”. In: *Behavioral Modeling and Simulation, Proceedings IEEE/ACM International Workshop on* (2000).

[87] P. Daglio and C. Roma. “A fully qualified top-down and bottom-up mixed-signal desgin flow for non volatile memories technologies”. In: *DATE* (2003).

[88] F. Pecheux, C. Lallement, and A. Vachoux. “VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24.2 (Feb. 2005), pp. 204–225.

[89] A. Leconitre, D. Dragomirescu, and R. Plana. “System architecture modeling of an UWB receiver for wireless sensor network”. In: *Proc. of 7th Int. Embedded Computer Systems: Architectures, Modeling, and Simulation(SAMOS) Workshop* (2007).

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

[91] A. Wang, B. H. Calhoun, and A. P. Chandrakasan. In: Boston, MA: Springer, 2007.

[92] 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, Digest of Technical Papers. Symposium on* (June 2004).

[93] N. Krishnapura and P.R. Kinget. "A 5.3-GHz programmable divider for HiPerLAN in 0.25-/spl mu/m CMOS". In: *IEEE Journal of Solid-State Circuits* 35.7 (July 2000), pp. 1019–1024.

[94] *Adding Extra Hysteresis to Comparators*. Application Note 3616, Maxim Integrated Products, Sept. 19 2005.

[95] N. Da Dalt, E. Thaller, P. Gregorius, and L. Gazsi. "A low jitter triple-band digital LC PLL in 130nm CMOS". In: *Solid-State Circuits Conference, (ESSCIRC) Proceeding of the 30th European* (Sept. 2004).

[96] R. Kho, D. Boursin, M. Brox, P. Gregorius, H. Hoenigschmid, B. Kho, S. Kieser, D. Kehrer, M. Kuzmenka, U. Moeller, P. Petkov, M. Plan, M. Richter, I. Russell, K. Schiller, R. Schneider, K. Swaminathan, B. Weber, J. Weber, I. Bormann, F. Funfrock, M. Gjukic, W. Spirkl, H. Steffens, J. Weller, and T. Hein. "75nm 7Gb/s/pin 1Gb GDDR5 graphics memory device with bandwidth improvement techniques". In: *Solid-State Circuits Conference - Digest of Technical Papers, (ISSCC) IEEE International* (Feb. 2009).

[97] R. Kho, D. Boursin, M. Brox, P. Gregorius, H. Hoenigschmid, B. Kho, S. Kieser, D. Kehrer, M. Kuzmenka, U. Moeller, P.V. Petkov, M. Plan, M. Richter, I. Russell, K. Schiller, R. Schneider, K. Swaminathan, B. Weber, J. Weber, I. Bormann, F. Funfrock, M. Gjukic, W. Spirkl, H. Steffens, J. Weller, and T. Hein. "A 75 nm 7 Gb/s/pin 1 Gb GDDR5 Graphics Memory Device With Bandwidth Improvement Techniques". In: *Solid-State Circuits, IEEE Journal of* 45.1 (Jan. 2010), pp. 120–133.

[98] *Understanding Data Eye Diagram Methodology for Analyzing High Speed Digital Signals*. `www.onsemi.com/pub/Collateral/AND9075-D.PDF`.

Invention Disclosures

[99] A. Jaiswal, Y. Fang, and K. Hofmann. “Method and device for correcting a phase shift in a time synchronised system”. In: *Application no. EP12174388.4* (June 2012).

[100] S. Königsmark, A. Jaiswal, and K. Hofmann. “Method for constructing a clock tree”. In: *Application no. EP12174388.4* (Sept. 2012).

[101] A. Jaiswal, Y. Fang, and K. Hofmann. “A wide range programmable duty cycle corrector”. In: *Application no. EP13162421.5* (Apr. 2013).

[102] Y. Fang, A. Jaiswal, and K. Hofmann. “Method and means for improving the data transfer integrity in a time synchronised system”. In: *Application no. EP12183643.1* (Sept. 2012).

List of Own Publications

[103] 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), Euromicro Conference on* (Sept. 2013).

[104] A. Jaiswal, Y. Fang, K. Nawaz, and K. Hofmann. "A wide range programmable duty cycle corrector". In: *SOC Conference (SOCC), IEEE 26th International* (Sept. 2013).

[105] 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), IEEE 11th International Conference on* (Oct. 2012).

List of Unrelated Publications

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

[107] 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), Euromicro Conference on* (Sept. 2013).

[108] Y. Fang, L. Chen, A. Jaiswal, and K. Hofmann. "Transmitter Equalizer Training Based On PilotSignal and Peak Detection". In: *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)* (2013).

[109] Y. Fang, U. Muhammad, A. Jaiswal, and K. Hofmann. "Low-Power Design of Hybrid Digital Impedance Calibration For Process, Voltage, Temperature Compensations". In: *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)* (2013).

[110] H. Ying, A. Jaiswal, T. Hollstein, and K. Hofmann. "Deadlock-free generic routing algorithms for 3-dimensional Networks-on-Chip with reduced vertical link density topologies". In: *Journal of Systems Architecture* 59.7 (2013), pp. 528 –542.

[111] H. Ying, A. Jaiswal, and K. Hofmann. "Deadlock-free routing algorithms for 3-dimension Networks-on-Chip with reduced vertical channel density topologies". In: *High Performance Computing and Simulation (HPCS), International Conference on* (July 2012).

[112] H. Ying, A. Jaiswal, M.A.A. El Ghany, T. Hollstein, and K. Hofmann. "A simulation framework for 3-dimension Networks-on-chip with different vertical channel density configurations". In: *Design and Diagnostics of Electronic Circuits Systems (DDECS), IEEE 15th International Symposium on* (Apr. 2012).

[113] H. Ying, A. Jaiswal, T. Hollstein, and K. Hofmann. "A Fast Congestion-Aware Flow Control Mechanism for ID-Based Networks-on-Chip with Best-Effort Communication". In: *Digital System Design (DSD), 14th Euromicro Conference on* (Aug. 2011).

[114] T. Hollstein, F.A. Samman, A. Jaiswal, H. Ying, M. Glesner, and K. Hofmann. "Invited paper: Design criteria for dependable System-on-Chip architectures". In: *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 6th International Workshop on* (June 2011).

Supervised Master Theses

[115] D. Gadler. “Phase Interpolator Based PLL”. In: (Jan. 2013).

[116] J. Goldeck. “Generalized GDDR5 Controller”. In: (May 2011).

[117] L. Chen. “Equalization Circuits for Noise Cancellation”. In: (Dec. 2012).

[118] M. Uzair. “Digital Impedance Ccalibration for PVT Compensation”. In: (Jan. 2013).

[119] J. Bargon. “Implementation of Phase interpolator-based clock data recovery”. In: (Jan. 2014).

[120] L. Thimmaiah. “Design and Implementation of PAR-BS for GDDR5 Memory Controller in a UVM Verification Environment”. In: (Jan. 2014).

[121] T. Wiemer. “Gem5 verification environment for parallelism-aware memory controller”. In: (Jan. 2014).

Supervised Bachelor Theses

[122] S.T.C. Koenigsmark. “Clock-Tree Optimization techniques”. In: (Aug. 2012).

[123] K. Heid. “Integration and Verification of XHiNoC in the Atlas framework”. In: (Mar. 2011).

[124] D. Wolf. “High Performance DRAM Scheduling Algorithms for Multi Thread-ed Systems”. In: (Oct. 2012).

[125] D. Walk. “High-Speed Serializer and Deserializer”. In: (Sept. 2013).

Supervised Seminars

[126] J. Bargon. “Study of co-simulation with VHDL and SystemC for XHiNoC”. In: (Oct. 2010).

[127] Z.A. Chohan. “Survey of High Speed Communication Protocols and Future Challenges”. In: (Feb. 2012).

[128] M.D. Serrano. “Transceivers for high-speed IOs”. In: (Oct. 2012).

[129] K. Nawaz. “High Speed Duty Cycle Corrector”. In: (Oct. 2012).

[130] M.M. Ahmed. “Level Shifters for High Speed System”. In: (Oct. 2012).

Glossary

ADC Analog-to-Digital Converter. 50, 57, 60, 111

ASIC Application Specific Integrated Circuit. 2, 68, 111

AWGN Additive White Gaussian Noise. 72, 111

BER Bit-Error-Rate. 4, 5, 65, 111

BGA Ball Grid Array. 10, 111

CAGR Compound Annual Growth Rate. ix, 2, 111

CCO Current Controller Oscillator. ix, xiii, 27, 81–85, 111

CDR Clock-Data-Recovery. iii, ix, xi, xii, 3–5, 17–20, 25, 27, 111

Ci input capacitance. 11, 111

CP Charge Pump. xii, 27, 45, 47–50, 57–60, 79, 80, 82, 111

CSR Control and Settings Register. 72, 111

D2D Device-to Device. 2, 111

DBI Data Bit Inversion. 10, 111

DC Direct current. 10, 12, 111

DCC Duty Cycle Corrector. xii, 14, 19, 45–47, 49, 57–61, 111

DDR Double Data Rate. iii, 7, 9, 14, 64, 111

DFE Decision Feedback Equalizer. 11, 111

DLL Delay Locked Loop. 19, 25, 111

DLPLL Dual Loop PLL. xii, 26–28, 81, 82, 111

DME Deferred-Merge Embedding. 35, 56, 57, 111

DRAM Dynamic Random Access Memory. 16, 111

EDC Error Detection and Correction. xi, 21–24, 28, 29, 52, 71, 87, 88, 111

eye The eye diagram [98] is constructed from a digital waveform by folding the parts of the waveform corresponding to each individual bit into a single graph with signal amplitude on the vertical axis and time on horizontal axis. The resultant graph will represent the average statistics of the signal and will resemble an eye. 3, 11, 18, 19, 71, 72, 111

FD Frequency Divider. 79, 85, 88, 111

FSM Finite State Machine. 20, 111

GDRAM Graphics Dynamic Random Access Memory. 7, 9, 111

HIPPI-6400-PH High-Performance Parallel Interface. 5, 111

I/O Input/Output. iii, 2, 4, 5, 9–11, 14, 64, 72, 111

IC Integrated Circuit. 5, 111

IoT Internet of Things. iii, 2, 111

ISI Inter-Symbol-Interference. iii, xi, 4, 5, 9, 11, 111

ITRS International Technology Roadmap for Semiconductors. 2, 111

LCS Local Clock Skew. 57, 111

link electrical channel interface including transmitter and receiver circuitries. iii, xi, 2–6, 12, 16, 64, 65, 111

M2M Machine-to-Machine. iii, 2, 111

OPAMP Operational Amplifier. 80, 82, 111

OTA Operational Transconductance Amplifier. xiii, 27, 80, 81, 111

PCB Printed Circuit Board. 2, 7, 10–12, 111

PD Phase Detector. 3, 111

PF petaFlops. 2, 111

PFD Phase-Frequency Divider. 79, 80, 88, 111

PHY physical. 72, 111

PI Phase Interpolator. iii, xi, xii, 3, 14, 17–20, 22–27, 31, 52, 54, 55, 64, 78, 111

PLL Phase Locked Loop. iii, ix, x, xii, xiii, 3, 4, 14, 19, 22, 24–28, 30, 31, 52, 54, 55, 60, 64, 72, 78–80, 84–88, 111

PODL Pseudo Open Drain Logic. 10, 111

PS Phase Selector. 27, 84, 85, 111

PVT Process, Voltage and Temperature. 9, 12, 19, 20, 23, 25, 38, 41, 45, 50, 57, 64, 111

RAS Reliability, Availability, Serviceability. 6, 111

RLPBO Reliable Low Power Buffering and Optimization algorithm. ix, 56, 57, 111

Rpeak Theoretical Peak Performance. 2, 111

Rx Receiver. 9, 111

SAR Successive Approximation Register. 45, 111

SCI Scalable Coherent Interface. 5, 111

SDR Single Data Rate. 7, 111

SDRAM Source-Synchronous Dynamic Random Access Memory. 7, 9, 14, 111

SE single-ended. 9, 111

sinks end-points. 13, 34–36, 39, 41, 111

SKA Square Kilometer Array. 2, 111

SSCF Source-Synchronous Clock Forward. xi, 16, 111

SSD/SSR Source Synchronous Drivers/Receivers. 6, 111

SSO Simultaneous Switching output. 10, 111

Tx Transmitter. 9, 111

V2I Voltage-to-Current. ix, xiii, 27, 28, 81–83, 85, 86, 111

VCO Voltage Controller Oscillator. ix, xiii, 25–29, 54, 78, 82, 84–87, 111

VOC Variable Offset Comparator. 20, 111

VREF reference voltage. 111

WCK Write Clock. ix, 8, 20–25, 31, 54, 55, 71, 72, 85, 111

XDR Extreme data rate dynamic random-access memory. 8, 9, 111

Resume

Name:	Ashok Kumar Jaiswal
Data of birth:	March 4^{th}, 1981
Place of birth:	Pratapgarh, India

Academic Background

1993-1995	High school at "S. R. M. Inter College Pilibhit", India
1995-1997	Intermediate school at "G. I. C. Raebareli", India
1998-2002	B. Tech. (Bachelor of Technology) at "K. N. I. T. Sultanpur", India
2002-2004	M. Tech. (Master of Technology) at "I. I. T. Delhi", India
2009-2014	Dr. Eng. (Doctor of Engineering) at "T. U. Darmstadt", Germany

Work Experience

2004-2006	Design Engineer at "Infineon Technologies", Bangalore, India
2006-2009	Concept Engineer at "Qimonda Technologies", Munich, Germany
2009-2014	Research assistant at "Institute of Integrated Electronic Systems", Technische Universität Darmstadt, Germany