DESIGN OF A CMOS ASYMMETRIC SERIAL LINK

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Kun-Yung Chang

August 1999

© Copyright by Kun-Yung Chang, 1999

All Rights Reserved

Abstract

In recent years, there has been a growing interest in using high-speed serial links to improve the bandwidth of chip interconnections. However, the use of many serial links in high fanin chips, such as the crossbar chips in network switches, poses design challenges in the clock recovery circuitry within these links. Conventional serial links have timing adjustment circuits containing a Phase-Locked Loop (PLL) at each receiver to recover the data timing. As a result, a large number of PLLs would be required on the high-fanin crossbars, which may cause problems with power, noise management, and the latency of the crossbar chip. A new Asymmetric Serial Link is proposed and implemented to alleviate these problems.

The Asymmetric Serial Link moves the timing adjustment circuits from the high-fanin crossbar chip to its peripheral chips. To synchronize the inbound link to the high-fanin chip, calibration packets are used to feed timing information to the peripheral chips. To track dynamic noise, the link calibration works in conjunction with a local clock generation PLL, which increases the tracking bandwidth of the serial link. In addition, the use of calibration packets for synchronizing the link provides better timing margins for the links because the same receiver can be used for both data reception and phase detection, which completely eliminates the effect of receiver offset and clock buffer mismatch at the receiver end.

A one-channel test chip was implemented in a 0.25μm CMOS process, with a flexible architecture that allows comparison between the performance of periodic and continuous calibration. Experimental results show that both mechanisms achieve 2 Gb/s with almost identical margins. Only DC calibration is needed because of the use of the local PLL on each chip. A 32×32 crossbar chip implemented in a similar process further demonstrates the robustness of the link in a practical environment. The crossbar chip operates successfully with links running at 1.6 Gb/s, limited by the switch core. The measured bit-error-rate is less than 10^{-14} when all channels and the switch core are operating. The crossbar chip consumes 5W and provides a total bandwidth above 50 Gb/s.

Acknowledgments

While pursuing my Ph.D. at Stanford, I faced many challenges that sometimes made me wonder what I was after, especially during my transfer to a different research group in the middle of my seven-year stay at Stanford. But in the end, I enjoyed every moment at Stanford and I shall never regret my decision to pursue a Ph.D. Nevertheless, without the help, support, and encouragement from several people, I would have never come through.

First and foremost, I have to thank my principal advisor Prof. Mark Horowitz for his support and guidance in my research. If there is something called a *dream*, he is certainly the person who made my dream come true. His knowledge about technical materials is beyond any doubt. What really amazes me is his patience (yes) and professionalism in teaching me and all his students. Whenever I have questions or problems in my research, he is always there to provide guidance and encouragement. I feel very grateful and fortunate that not only did he pick me up as his student when my first research project did not work out, but he also provided me very interesting and challenging thesis work to focus on. I feel very honored and privileged to have worked with him.

I would also like to thank my associate advisor Prof. Nick McKeown. Without his Tiny-Tera project, all my work would not have been possible. In addition, I thank him for teaching me how to give presentations. I would also like to thank Prof. Kunle Olukotun for his support during my early years at Stanford, before I transferred to Mark's group. Next, I thank Prof. David Miller for being my oral chairman, and I thank Prof. Bruce Wooley for being on both my oral committee and reading committee.

Without the help I received from several Stanford graduated/graduate students, my research would have been much more difficult than it was. First, I would like to thank Dr. Stefanos Sidiropoulos and Dr. Ken Yang, both of whom helped me a lot during my transition to Mark's group and taught me many things about circuit and chip design. Next, I would like to thank Bill Ellersick for all the technical and non-technical discussions that helped me learn how to work with people. I would like to thank Shang-Tse "Da" Chuang for writing amazingly long but elegant Verilog codes for both of my chips. From my work with both Bill and Da, I truly appreciate the importance and beauty of digital design. I also would like to thank Jeff Hsieh for his help with the layout and verification work on the one-channel test chip, and both Jaeha Kim and my long-time officemate Lance Hammond for assisting the layout work on my 32-channel test chip. I also thank Lance for many interesting discussions on a variety of topics and for his patience in proofreading my thesis draft and all of my previously published papers.

I would like to thank Charles Orgish for computer support, Darlene Hadding, Terry West, and Kersten Barney for administrative support.

I would also like to thank many friends I made prior to or during my stay at Stanford. They made my life at Stanford so much more enjoyable. For this, I would like to name Dr. Chung-Li Lu, Li-Yi Wei, James Pan, Dr. Heng-Chih Lin, Dr. Ricardo Gonzalez, Victor Lam, and Ming-Ju Lee. I also have to thank Ming-Ju for proofreading my thesis draft.

I would like to thank Dr. Martin Izzard and Ah-Lyan Yee from Texas Instruments for their support in taping out my two chips. I would like to thank Ed Priest from Abrizio for suggestions on board design. I would also like to thank Texas Instruments and Cisco Systems for their financial support of my thesis study.

Finally, I would like to thank my parents, my twin brother Dr. Tsung-Yung Jonathan Chang, and other members of my family for their unconditional support and encouragement. I especially feel grateful to them for all the support as I was switching research groups. No matter which state I was in, they were always with me, so this thesis is dedicated to my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Interest in the design of high-speed links for digital systems has greatly increased in recent years. In computer systems, the links usually appear in the processor-to-memory interfaces [31] and in multiprocessor interconnections [84][85][86]. In computer networks, the links commonly show up in the long-haul interconnect between systems that are physically far apart [87][88], and recently in the backplane interconnect within a switch or a router [82][83].

This thesis looks at optimizing high speed links for applications that require many links to be connected on one chip, to support devices with a large fanin. This situation most often arises when building a switch or a router for computer networks.

Computer networks are made up of packet switches interconnected by links, as shown in Figure 1.1, where the square boxes represent the packet switches and the arrows represent the links. In the early days of the Internet, the system capacity was limited by the data rate of the links: by the end of the 1980s the NSFNET was built around 64kb/s links. However, in recent years, the bottleneck has shifted towards the packet switches that form the core of the network. This change has been brought about by the rapid deployment of optical

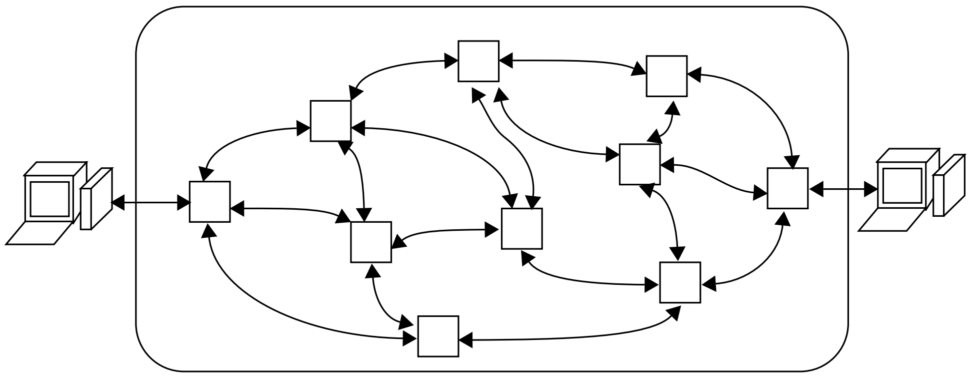


Figure 1.1: Computer Network

fibers and SONET, and it has recently been accelerated by the introduction of WDM (Wavelength Division Multiplexing). In commercial systems today, WDM allows up to forty OC48c channels (each operating at 2.48Gb/s) to be multiplexed onto a single fiber.

To provide increased switching capacity, the architecture of packet switches has been changed. Whereas the early packet switches were built around shared buses, and used shared central CPUs and memory, they are now typically built around a switch fabric that allows multiple packets to traverse the backplane simultaneously. Figure 1.2 shows an example of a packet switch built around a parallel crossbar switch fabric, which is at the center of the diagram. Around the switch fabric are the port cards that buffer data coming into and out of the switch. One side of the port cards interfaces with the long-haul interconnect in the network while the other side interfaces with the switch fabric. Many commercial high-performance routers have been built in this architecture: the *Cisco 12000 GSR* [82], the *Ascend GRF 400 and GRF 1600 IP Switch* [79], the *BBN MGR project* [80], the *Bay Network Accelar 1200 Routing Switch* [81], and many others [78].

The switch fabric forms one of the most important bottlenecks in the switch because all the data traffic has to go through it. The design of the switch fabric poses some interesting design challenges for link designers. For an efficient switch architecture, the switch needs to support a large number of ports, and for high bandwidth, each port should run the pins at high speeds. This implies that the switch fabric must contain a large number of serial links on one chip.

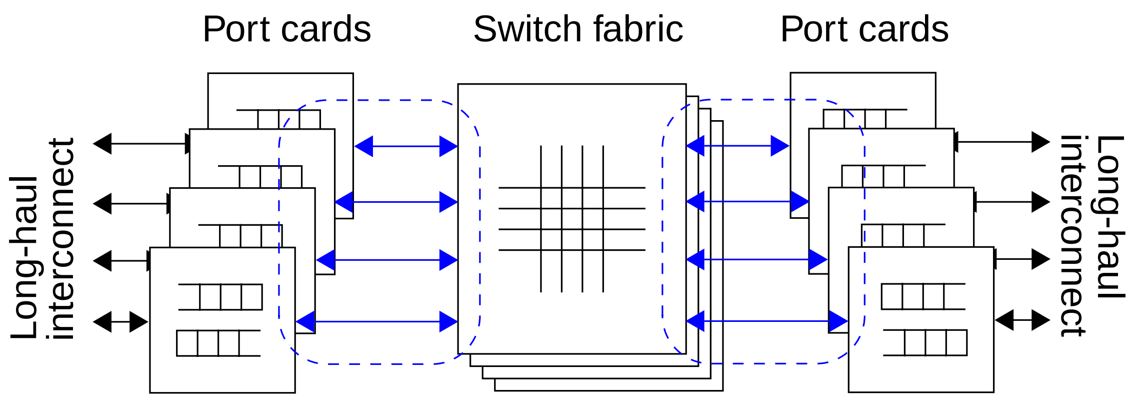


Figure 1.2: Switch Architecture

While the studies of a single-channel serial link [1][4]-[23] or of multi-channel parallel links [28]-[34] have been reported before, very little research has been reported on using CMOS serial links for a high fanin chip above Gb/s rate. Most of the very high-speed (over Gb/s/interface) crossbar switches reported thus far have not been implemented in CMOS, but in GaAs [69], Si-Bipolar [71] or other non-traditional processes [64][66][68]. The few CMOS crossbar switches that have been reported have serious limitations: their interface bandwidths are either well below Gb/s [58][59][62][70], or they cannot be scaled to support more interface ports [24][63][67] because the design would require high power or large pin count, which would make it too costly and impractical.

The research presented here focuses on the design of the high-speed serial links (called *Asymmetric Serial Links* [2][3]) for systems that have high-fanin chips. The study addresses the challenges posed by having many high-speed serial links on a highly-integrated CMOS chip–challenges that include both design feasibility and system cost–and then to propose a reasonable solution.

1.2 Overview

Chapter 2 describes the background behind this thesis research. It starts with an overview of a basic high-speed link and brings out issues in designing high-speed links. Then it presents the *Tiny Tera* switch and explains how serial links can be used to increase its total bandwidth. In a conventional design, there would be many PLLs at the switch core, which would cause problems with power consumption and noise coupling. To solve these problems, the Asymmetric Serial Link [2][3] is developed in this thesis study.

The Asymmetric Serial Link moves the PLL for each link from the crossbar switch core to the chips interfacing with it. Chapter 3 discusses the architecture and the clock recovery mechanism of the link. Chapter 4 presents the link's circuit implementation. Chapter 5 focuses on the design of a 32×32 crossbar chip using the Asymmetric Serial Links. The crossbar chip was designed for use in the Tiny Tera and hence follows the transmission protocols used in that system. The crossbar chip uses reverse routing tags [83] to facilitate multicast support. The tags are scheduled by a separate scheduler chip in the system.

Chapter 6 shows the experimental results from two chips designed for this thesis study. First, it presents a phase error injection method for measuring link timing margins and gives the results from a single-channel test chip implemented in a 0.25-μm CMOS process. The experiment showed that the serial link can run as fast as 2Gb/s in both differential and single-ended signalling. Next, the chapter shows results from an experimental 32×32 crossbar chip, designed and fabricated in a 0.28-μm CMOS process. One of the primary objectives of the crossbar-chip design was to demonstrate the robustness of the Asymmetric Serial Links within a practical noisy chip and system environment. The chip operated successfully, with each link running at 1.6 Gb/s. The measured bit-error-rate was less than 10^{-14} when all channels and the switch core were operating. The crossbar chip consumed 5W and provided a total bandwidth above 50 Gb/s.

Chapter 2

Background

To investigate the problems created by having many serial links on a single chip, one must understand the design of each individual serial link first. Hence, this chapter starts with an overview of a basic serial link in Section 2.1. One key challenge for any high speed link is to overcome the noise present in the transmission medium, the packages, and the chips themselves, and still able to transmit and receive data at a fast rate. The two main issues in the high-speed link design are signalling and clocking. The signalling issue is related to how to maximize the voltage margins of the interface and the clocking issue is related to when to transmit and receive the data to maximize the timing margins of the interface. These two issues and their impact on the link design are discussed in Section 2.2 and Section 2.3 respectively.

The timing recovery circuits play a key role in the clocking method of any serial link. Section 2.4 reviews the design of a dual-loop phase-locked loop (PLL) and delay-locked loop (DLL), proposed by Horowitz *et al.* [44] and Sidiropoulos *et al.* [56][91], respectively. This thesis work modified Sidiropoulos' dual-loop DLL architecture for use in a PLL to provide clock multiplying capability. Section 2.4 highlights the high-level architecture of the dual-loop PLL. The implementation details are discussed in Chapter 4.

Finally, Section 2.5 presents a fast network switch called the Tiny Tera, which uses high-speed serial links to increase its total bandwidth. The center switch fabric of the Tiny Tera is implemented by several parallel high-fanin crossbar chips. The section discusses the problems created by using high-speed serial links for these high-fanin crossbar chips and proposes a new serial link architecture–the Asymmetric Serial Link–to solve these problems.

2.1 Basic Link Design

Figure 2.1 shows the block diagram of a generic serial link. To prevent the digital logic at the chip core from limiting the link speed, all the data are parallelized before being processed by the digital logic. As shown in Figure 2.1, the N-bit datapath for the digital

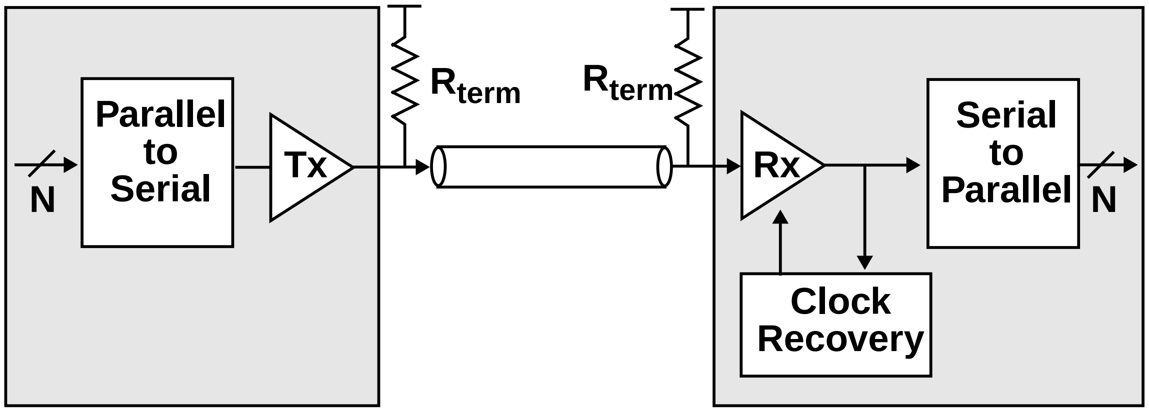
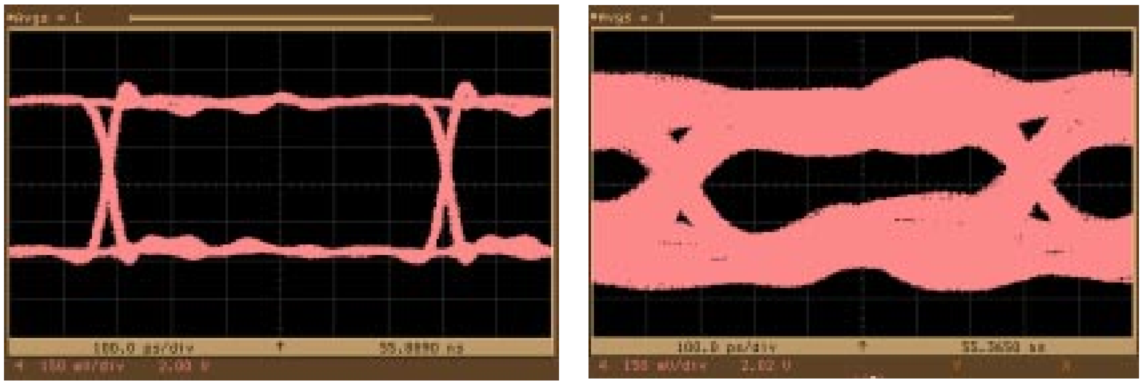


Figure 2.1: Block Diagram of a Generic Serial Link

logic is serialized by the parallel-to-serial converter. The transmitter sends the serialized bit stream to the channel, and the bit stream is received by the receiver. The termination resistors, which match the impedance of the channel, minimize signal reflection. The clock recovery circuit at the receiver end adjusts the receiver clock based on the received data to center the data sampling point in the data eye. The serial-to-parallel converter parallelizes the receiving bit stream into N parallel bits and resynchronizes them with the digital logic on the receiver chip.

To understand and analyze the design issues in high-speed links, one can use an eye diagram, which is an overlay of several signals on top of each other at a bit-time interval. Figure 2.2 shows two different eye diagrams, one measured from a very clean environment and the other from a very noisy environment. The horizontal axis is the time and the vertical axis is the signal amplitude. Since the waveforms are the overlay of many different data patterns, the opening in the center of the “eye” (the dark space) represents the place where the data can be received correctly. Therefore, the eye opening in the horizontal direction represents the timing margin at the measurement point whereas the eye opening in the vertical direction represents the voltage margin.

The left eye diagram, which represents the one from a clean environment, is measured from signals transmitted directly from a low-jitter function generator into an oscilloscope through a low-loss coaxial cable. It thus has a very wide-open eye. The right eye diagram, which represents the one from a noisy environment, is from signals transmitted from a noisy transmitter chip to a lossy trace on a printed-circuit board, and then to a receiver



Clean Eye *Noisy Eye*

Figure 2.2: Data Eye Diagrams

with on-chip termination. As a result, the right eye diagram appears to be much noisier and hence is much smaller in both the vertical (signal amplitude) and the horizontal (signal timing) direction than the left eye diagram.

Figure 2.2 points out the two main issues in the high-speed interface design: signalling and clocking. The signalling issue is related to how to maximize the voltage margins of the interface so that the receiver has enough voltage margin to acquire the data correctly. The clocking issue concerns the question of when to transmit and receive data to maximize the timing margins of the interface. The key design challenge to any high-speed link design is to transmit and receive data correctly at the maximum rate with the existence of the noise, which can affect both the signalling and clocking of the link. Section 2.2 and Section 2.3 discuss each of these issues and their impact on the link design respectively.

2.2 Signalling

The main sources of the signal distortion on any high speed link are signal reflection, crosstalk (or line coupling), self-induced power supply noise (dI/dt noise), and intersymbol interference. The noise resulting from these sources is proportional to signal amplitude and hence cannot be overcome by simply increasing the signal swing. The noise must be minimized or cancelled by careful design. There is also noise independent of signal amplitude, such as thermal noise or supply noise from the on-chip digital logic. This independent noise can be overcome by increasing signal swing and thus can be dealt with more easily than the proportional noise.

From the physical design standpoint, the different types of proportional noise are caused by package parasitics, connectors, and signal attenuation on the PCB traces and cables. The first two introduce impedance discontinuity that induces reflection noise. The capacitive and inductive coupling from the packages and connectors all are the major source of crosstalk. The inductive bonding wires on the packages cause dI/dt noise on the power supply for the I/O pins. The signal attenuation on the PCB traces and cables along with the package parasitics all contribute to intersymbol interference. To reduce the noise, the designers have to carefully design or select the packages, connectors, PCB, and cables.

For applications where many links converge on a single chip, crosstalk poses a major design challenge. Having many high-speed links converging on about one square inch of area from all directions prevents one from using the common method of locally optimizing a single link by making it physically distant from other signal traces. The problem is even worse for packages, including both wire-bonded and flip-chip packages, because of the high density of high speed signal traces on the package substrate and bonding wires. To reduce crosstalk, designers can insert shielding package traces and bond-wires or use differential signalling.

Differential signalling provides better noise immunity than single-ended signalling at the expense of one more signal wire and pin for each link, i.e., 2-to-1 ratio in the number of signal pins between a differential and single-ended link. In some cases, at the package level, both differential and single-ended links need shielding traces to reduce crosstalk. As a result, each differential link may cost three pins (two signals and one ground) while each single-ended link costs two (one signal and one ground). In this case, the ratio in pin count between differential and single-ended links becomes 3-to-2 instead of 2-to-1 described before. Even though the differential links require more pins, they are much easier to design for noise rejection than single-ended links. Moreover, the single-ended links need a reference signal with the same response as their receiver input, which is difficult to achieve [91]. Likewise, from the connector standpoint, the advantage of single-ended links in fewer signal pins may also be offset by the need for more shielding ground pins to avoid signal coupling. In a network switch or router where the links are run between PCBs and connectors, the cost of additional PCB traces is insignificant. All these drove us to use differential signalling for the serial link developed in this thesis work.

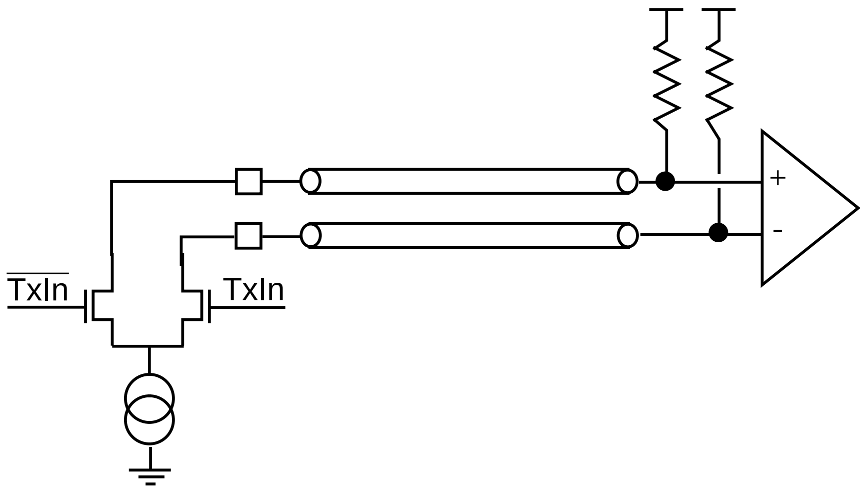


Figure 2.3: High-impedance Open-drain Differential pair Driver

Finally, while the dI/dt noise can be minimized by low inductance packages, its amplitude and effect can be further reduced by careful design of the output transmitter. For differential signalling, one can use a differential-pair with a constant current source, as shown in Figure 2.3. This type of transmitter is classified as a “high-impedance driver” by Sidiropoulos in [91] because of the high output impedance from its current source. This transmitter induces minimal dI/dt noise because it sinks constant current through the ground pin. In addition, because of the high output impedance of the current source, the ground noise is isolated from the transmitted signals. Therefore, the noise from the transmitter chip is separated from the receiver, hence improving the signal integrity of the link.

This section briefly reviewed several important noise sources and techniques to reduce their effect on high-speed signalling so that the voltage margin of signals can be maximized. The next section will discuss another important issue of high-speed link design–clocking, sometimes referred as synchronization. It is related to when to transmit and receive data so that the timing margin of the high-speed links is maximized.

2.3 Clocking

Figure 2.4 shows the eye diagram of the received data. The best place to sample the data is at the center of the eye, which maximizes the timing margin for data reception. To position the sampling point at the eye center for uncertain channel delay and still achieve a high

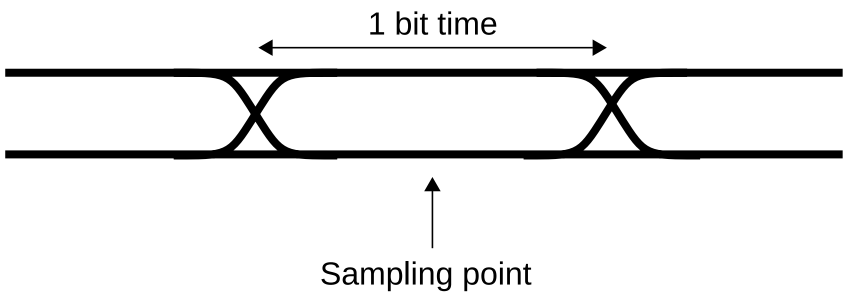


Figure 2.4: Received Data Eye

transmission bit rate, the source synchronous interface can be used [31][33][91]. The source synchronous interface sends the data and clock in-phase from the transmitter chip to the receiver chip, as shown in Figure 2.5. At the receiver end, the clocks are phase shifted by half a bit time to clock the received data. The phase shifting is done by the phase adjuster at the receiver end, which is commonly in the form of a PLL (Phase-

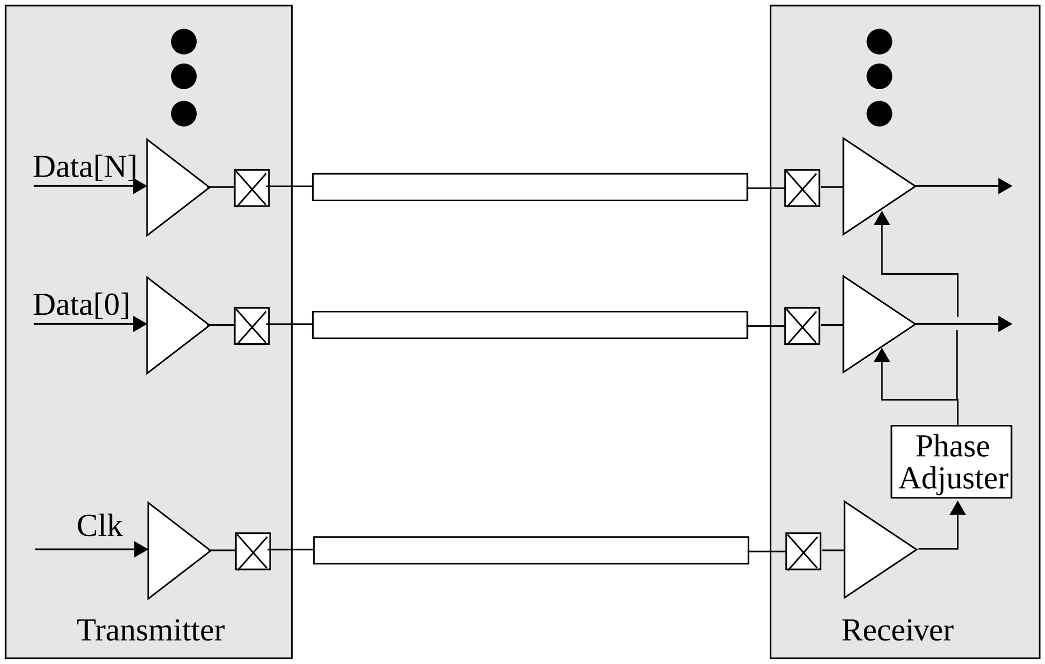


Figure 2.5: A Source Synchronous Interface

Locked Loop) [44][49][56]. Since the clock travels the same length as the data, the channel delay is cancelled out between the clock and data.

The explicit clock path of the source synchronous interface introduces additional cost for each interface. Therefore, this interface usually appears in parallel links where multiple links share the same clock path, such as processor-to-memory interfaces [31] and multiprocessor interconnects [84][85].

However, for high-fanin applications such as network switches and routers where each chip-to-chip interface on the high-fanin chip can only have a limited number of pins, the clock overhead of the source synchronous interface becomes very costly. For example, a 32×32 crossbar chip would need 64 clocks for the 32 inbound and 32 outbound links. To eliminate this clock overhead, serial links can be used.

Figure 2.6 shows the block diagram of a serial link, with no explicit clock path. The clock recovery circuit at the receiver end adjusts the receiver clock based on the received data. In general, the transmitter and receiver chips use different clock sources that have the same clock frequency. However, since the clock sources may come from two different crystals, there usually is a small deviation between the frequency on the transmitter chip and that on the receiver chip. As a result, the clock recovery circuit must perform both frequency and phase detection in order to lock both frequency and phase.

When frequency detection is necessary, the clock recovery circuit is usually designed to lock to a local reference clock that is very close to that on the transmitter chip in frequency. After the circuit locks the local reference clock, it switches to lock the phase of the received data. This type of frequency and phase detection for serial links has been

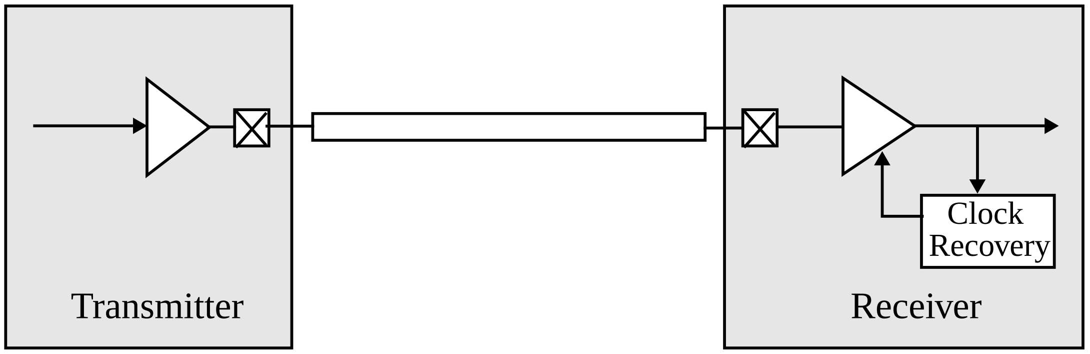


Figure 2.6: A General Serial Link

presented by Fiedler [9], Gu [10], Farjad-Rad [8], etc. In each design, the phase and frequency detection mechanism appears to be much more complicated than the phase detection mechanism of the source synchronous interface.

In the backplane interconnect for network systems, the designers sometimes can distribute a reference clock to every interface chip within the system so that the frequency can be identical for both the transmitter and receiver ends of the serial link. This greatly simplifies the clock recovery mechanism because only phase-locking is necessary.

Figure 2.7 shows the serial link that uses a distributed reference clock. The clock is distributed to both the transmitter and receiver chip. On the receiver chip, the phase adjuster uses the reference clock to generate the clock phase of the receiver clock. To perform clock recovery, the link has a phase detector in addition to the data receiver to adjust the clock. The phase detector is designed to be nearly identical to the data receiver so that their delay, setup-and-hold, and aperture time are approximately the same.

The phase adjuster in Figure 2.7 can be implemented by a DLL or a PLL. DLLs use a voltage-controlled delay line to provide adjustable clock phases whereas the PLLs use a

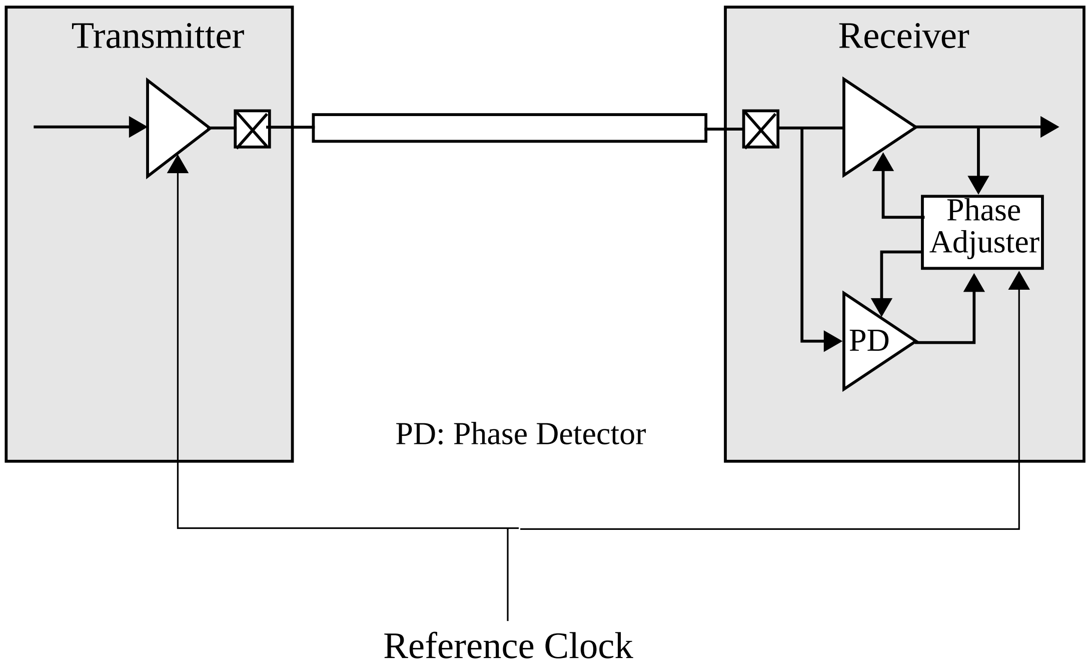


Figure 2.7: A Serial Link with a Distributed Reference Clock

voltage-controlled oscillator (VCO). The advantage of the DLL is that the control voltage changes the delay and hence the phase directly. In a VCO-based PLL, the control line changes the frequency so small errors are integrated and can lead to large accumulated phase errors. As a result, the DLL generally induces smaller clock jitter than the PLL. However, a standard delay line can only provide a finite adjustable range because it is limited by the maximum and minimum adjustable delay of the delay line. This may limit the operating frequency range of the DLL. To solve the problem, a dual-loop DLL proposed by Sidiropoulos and Horowitz can be used to provide an unlimited range of clock phases [56]. Another disadvantage of the DLL is that the input reference clock has to be at the same frequency as the operating frequency in the DLL. For systems where a very high clock rate is necessary, the designers may have to use PLLs, which can multiply the clock frequencies. Traditional single-loop PLLs phase lock their clocks to the reference clocks and thus cannot provide additional adjustable clock phases. To implement the phase adjuster in Figure 2.7 using a PLL, a dual-loop architecture is needed, which was first proposed by Horowitz, *et al.* [44]. Basically, in both dual-loop architectures, the inner loop is composed of either a delay line for a DLL or a VCO for a PLL and the outer loop provides adjustable clocks for the clock recovery of the serial link.

The dual-loop PLL has a large advantage when used in serial links. The inner loop does not need to be looked to the data stream. It can be looked to a local clock, and hence can use a conventional phase frequency detector. The outer loop is a simple first order loop, since it adjusts phase directly. Thus it can use a non-linear phase detector (e.g. an input receiver) without concern for loop stability. Figure 2.9, again, shows the eye diagram of the received data. The data bits are sent on both edges of the clock[1]. Therefore, there are two bits per clock cycle and each bit time is 180^o. The ideal data sampling point is at the center of the eye. To determine whether the data sampling point is at the correct spot, one can sample the data transition point, which is called the timing sampling point. The timing sampling point is 90^o from the data sampling point. The data receiver in Figure 2.7 receives the data samples while the phase detector receives the timing samples.

1. Unless specifically mentioned, all the links described in this thesis will assume that data is sent on both clock edges. However, the design techniques introduced later in the thesis are not limited to this transmission scheme.

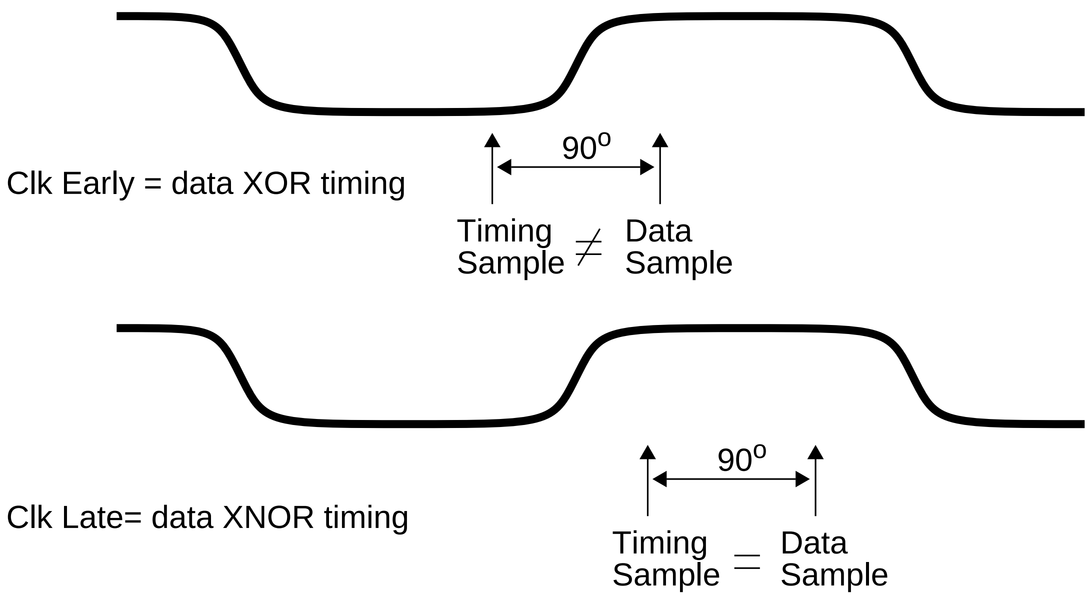

90°
Clk Early = data XOR timing
Timing Sample ≠ Data Sample
90°
Clk Late= data XNOR timing
Timing Sample = Data Sample

Figure 2.8: Phase Detection of Serial Links

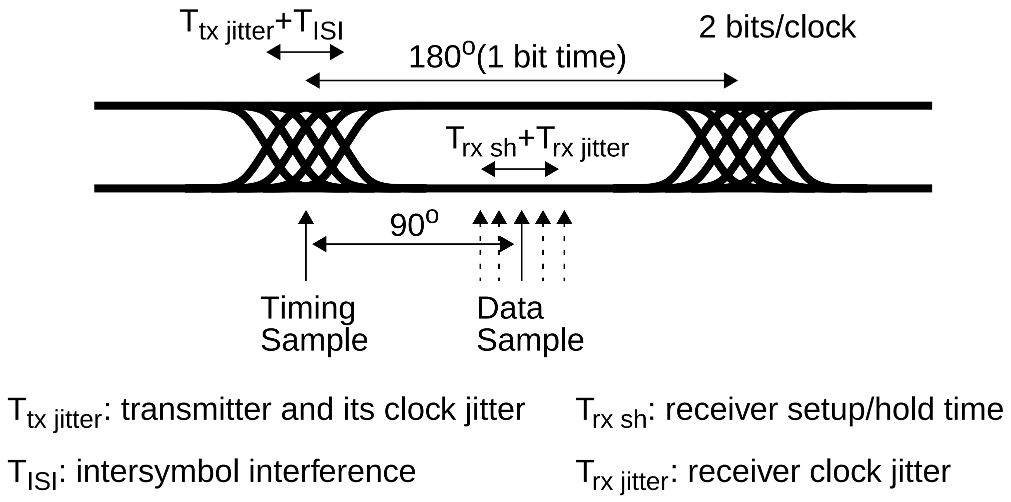

$T_{tx\ jitter}$+T_{ISI}
2 bits/clock
180°(1 bit time)
$T_{rx\ sh}$+$T_{rx\ jitter}$
90°
Timing Sample
Data Sample
$T_{tx\ jitter}$: transmitter and its clock jitter
$T_{rx\ sh}$: receiver setup/hold time
T_{ISI}: intersymbol interference
$T_{rx\ jitter}$: receiver clock jitter

Figure 2.9: Received Data Eye

Figure 2.8 shows a commonly used phase-detection mechanism for the serial link. When the timing and data samples are different, the data sampling point occurs earlier than the ideal eye center. Hence, the clock is early and has to be delayed by the phase adjuster in

Figure 2.7. On the other hand, when the timing and data samples are the same, the data sampling point occurs later than the ideal eye center and thus the clock is late.

The key challenge of clocking design is to position these sampling points in a non-ideal environment. The non-ideality can come from static timing offset and dynamic timing variation on the clock recovery circuit, the transmitter, and the receiver. The clock recovery circuit is especially important because it can be considered as a timing feedback block. Without careful design, the feedback might induce large timing uncertainty and degrade the link speed.

The static timing offset typically results from systematic clock skew or from an uncertain receiver setup-and-hold window. The dynamic timing variation usually results from clock jitter due to any power supply and substrate noise on the chip, or due to intersymbol interference from the package and channel. The static variation deviates the average positions of the sampling points from the center of the data eye. The dynamic variation from the transmitter and its clock causes the fuzziness around the data eye in Figure 2.9 while the dynamic variation from the receiver clock causes the timing uncertainty of the data sampling point, which is illustrated by the dash-arrows around the ideal data sampling point in Figure 2.9.

For integrated CMOS chips, supply and substrate noise resulting from the switching of digital logic or of output buffers is very significant. The noise introduces timing errors at the VCOs in PLLs or at the delay line in DLLs. The low-jitter distributed reference clock in Figure 2.7 provides a stable reference for the phase adjuster. For a DLL, the reference clock directly feeds its delay line. So the clean reference clock provides a low-jitter delay line. For a PLL the low-jitter reference clock allows a designer to adjust the PLL’s loop bandwidth to be as high as possible (but still keeps the loop stable) so that the feedback clock to the input of the phase detector can be aligned with the reference clock for any noise whose frequency is lower than the PLL loop bandwidth.

The serial link developed in this thesis study, which is targeted for 2Gb/s, uses a VCO based dual-loop PLL to allow a lower system reference clock (250MHz). The dual-loop PLL is based on Sidiropoulos’s design in [56] and is discussed in the next section.

2.4 Clock Recovery Circuits using Dual-loop PLL

Figure 2.10 shows the block diagram of the dual-loop PLL, which is comprised of an inner analog loop and an outer digital phase control loop. The inner loop is a VCO-based third-order PLL with a low-jitter reference clock. A divide-by-four circuit in the feedback to the phase frequency detector makes the VCO run at four times the rate of the reference clock. The four-stage VCO uses differential elements [52] to generate eight clock phases, spaced 45^o apart. The same bias used for the VCO is also used for the charge pump circuit so that the loop bandwidth is scaled with the operating frequency, which widens the operating range of the PLL [51]. The outer loop uses the clock multiplexors to select two adjacent phases from the VCO. A digitally controlled interpolator mixes the two adjacent phases to generate any one of 17 finer phases. Changing the mux select signals to the input of the interpolator adds an extra fine phase. As a result, each VCO clock cycle can be subdivided into $8 \times 17 = 136$ phases, which results in a phase resolution of 7.4 ps for a 1GHz VCO

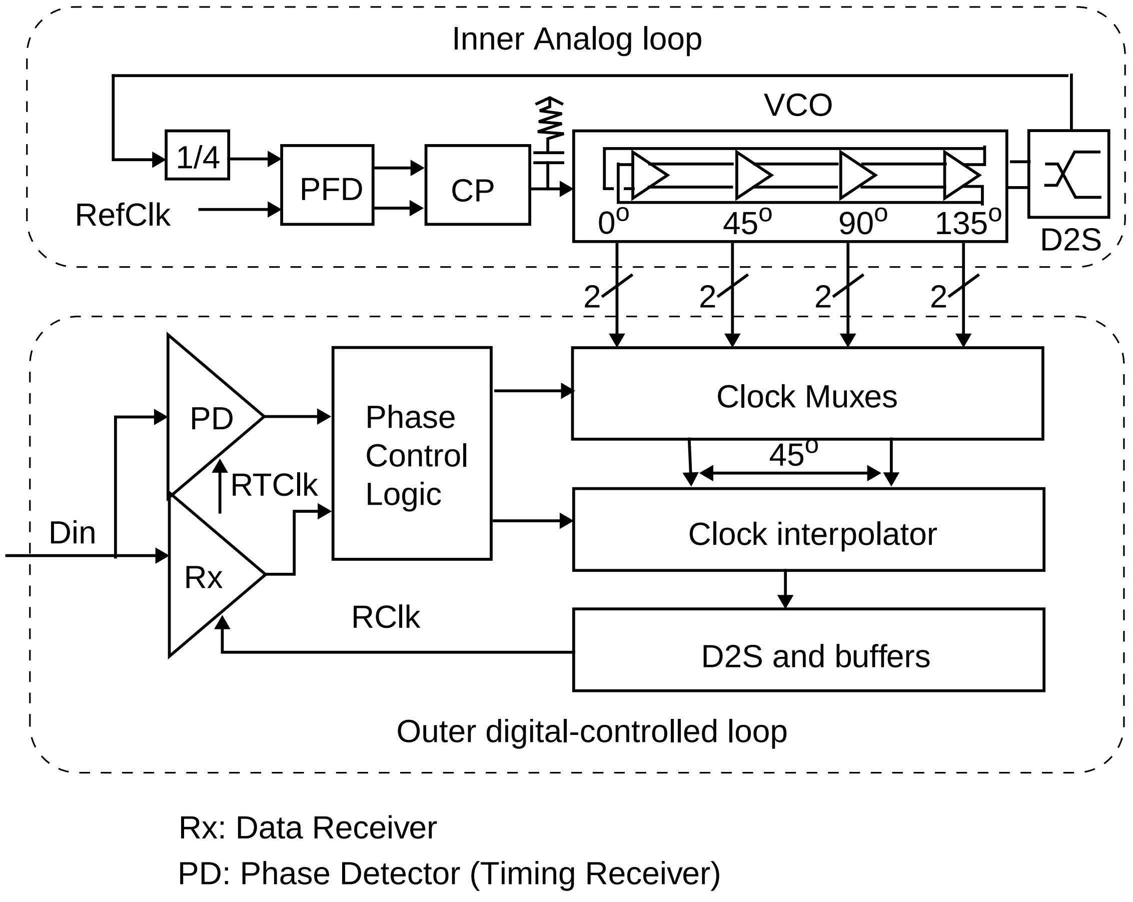


Figure 2.10: Block Diagram of a Dual-Loop PLL

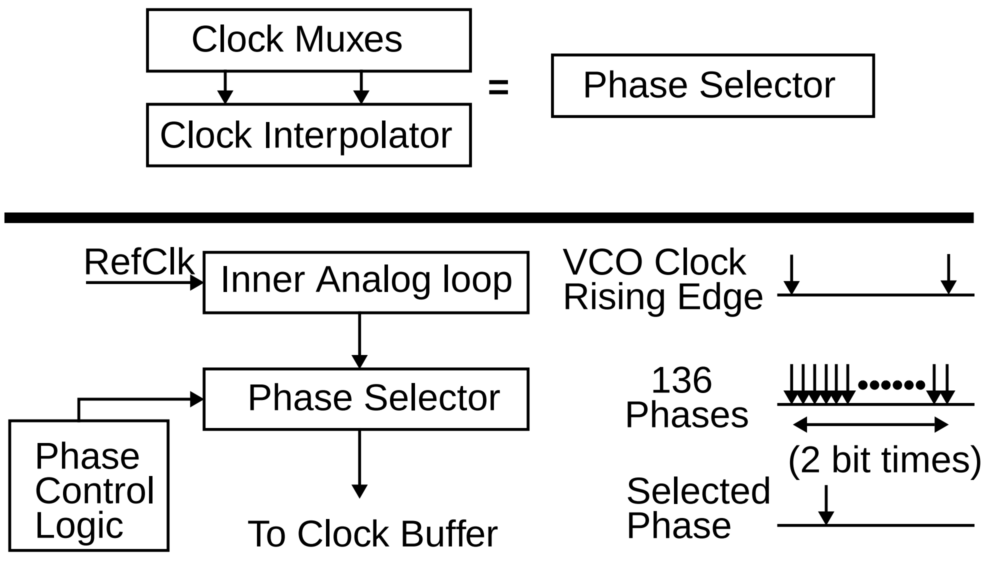


Figure 2.11: Clock Phase Selector

clock. This step size is much smaller than the peak-to-peak jitter for the overall clock circuits, which is about 50ps for a quiet supply and over 100ps for a noisy supply. Therefore, the phase offset due to the quantization error from the digital phase control loop is negligible.

Figure 2.11 recaps the function of the clock multiplexor and interpolator. For simplicity, these two blocks together are called the *phase selector* in the rest of the thesis. The phase selector divides a VCO cycle into 136 phases, in order to get very fine clock resolution. The digital phase control logic selects one of these very fine phases for the clock buffer.

The output of the phase selector is amplified and buffered by the differential-to-single-ended converter and clock buffers. The data receiver clock, RClk, is used by the receiver to sample the incoming data at the center of the data eye. The input data is also sent to the phase detector, which samples the data transition edge. The clock for the phase detector, RTClk, is one half bit time away from the data receiver clock. The phase detector clock (also called timing receiver clock) can be generated by another phase selector, with the phase to be one half bit time away from the data receiver clock, which is equal to two stages of the core ring delay. The results of both the phase detector and the data receiver are sent to the digital phase control logic to determine how to set the phase setting for the phase selector to keep the data sampling point at the center of the data eye. This closes the outer digital phase control loop. One can clearly see that this outer phase control loop is a first-order delay locked loop because it directly adjusts phase, in contrast to frequency in a second-order loop.

The digital-control property of the outer loop provides great flexibility in the design of the serial link's phase control mechanism. Designers can adjust the tracking bandwidth as well as the lock-in rate by implementing them in the digital phase control logic. In a noisy chip environment, any noise introduces timing variations on both the transmitter and receiver clocks as well as on the transmitter and receiver circuits. In addition, intersymbol interference on the channel also introduces timing noise into the data eye. As a result, the data on the channel has considerable jitter. To keep the data jitter from moving the receiver sampling point to an incorrect position, the clock recovery circuit needs to have a very low bandwidth to filter out the effects of any jitter. This can be achieved by lowering the update rate of the phase control logic to decrease the tracking bandwidth of the outer loop.

In addition to the advantage of flexibility in the phase control mechanism, the dual-loop architecture can also provide multiple adjustable clocks at low incremental cost. This can be accomplished by adding phase selectors to the outer loop of the PLL while sharing the same inner VCO loop, as shown in Figure 2.12. The stable VCO clocks are distributed to multiple phase selectors. Each phase selector is controlled by a different set of phase

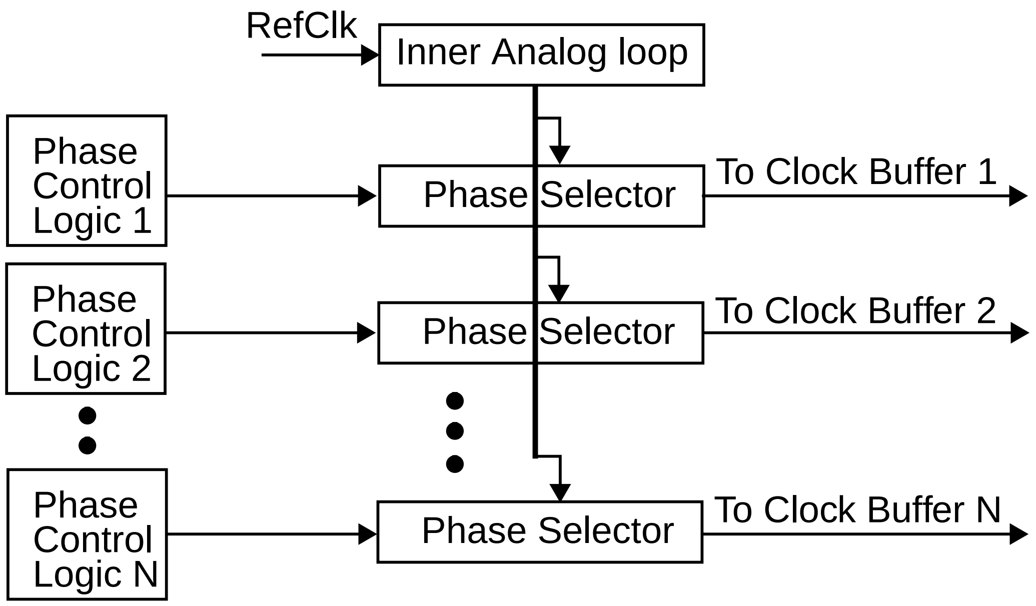


Figure 2.12: A Dual-Loop Architecture with Multiple Clocks

control logic and drives a different clock buffer. In other words, the same dual-loop PLL can provide multiple independent clocks with different, arbitrary clock phases. Sharing the inner VCO loop reduces the overall cost.

This ability to have multiple independent clocks from the same dual-loop PLL is quite useful. For a serial link pair, one PLL can generate both adjustable transmitter and receiver clocks. This thesis work takes advantage of this flexibility to move the clock recovery circuits of serial links for high-fanin chips around based on system constraints.

Next, Section 2.5 introduces a fast network switch, called the Tiny Tera, which utilizes high-speed serial links to increase its data bandwidth. The section then discusses the problems of using serial links for high-fanin crossbar chips and then presents a solution, which was inspired by the dual-loop PLL architecture.

2.5 The Tiny Tera

The design of the Asymmetric Serial Link was originally motivated by the design of the Tiny Tera packet switch [83], which is an example of a fast switch that uses high-speed serial links to increase its total bandwidth. The Tiny Tera, shown in Figure 2.13, is a small but high-bandwidth network switch, composed of 32 ports, each operating at 16 Gb/s. The main purpose of the Tiny Tera is to increase the raw interconnect bandwidth, which is crucial in building any high-speed network system.

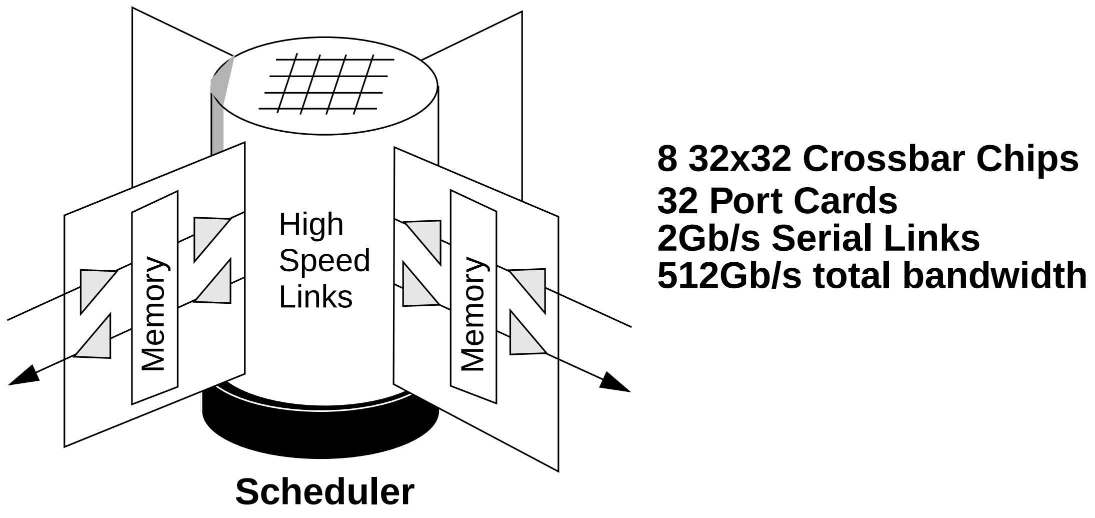


Figure 2.13: The Tiny Tera

The core of the Tiny Tera switch consists of eight synchronous crossbar chips. The interfaces between the ports and the crossbar switch core, and between the ports and the scheduler are implemented by high-speed serial links, with each link targeted at 2Gb/s. The use of high-speed links not only provides high bandwidth for the switch but also reduces the total pin count of the chips and connectors in the system.

The Tiny Tera transmits data on both edges of the *half-bit-rate clock*, which runs at 1GHz. At this rate, only very few simple logic gates can be fit between two flip-flops, even for a state-of-the-art 0.25µm CMOS technology. Therefore, the digital logic in the Tiny Tera runs at a slower *byte clock*, which is one-fourth of the half-bit-rate clock. To provide a synchronous timing reference to all the chips, a global reference clock at the byte clock rate is distributed to all the chips. As will be seen later in the thesis, this feature facilitates some aspects of the serial link design in the Tiny Tera.

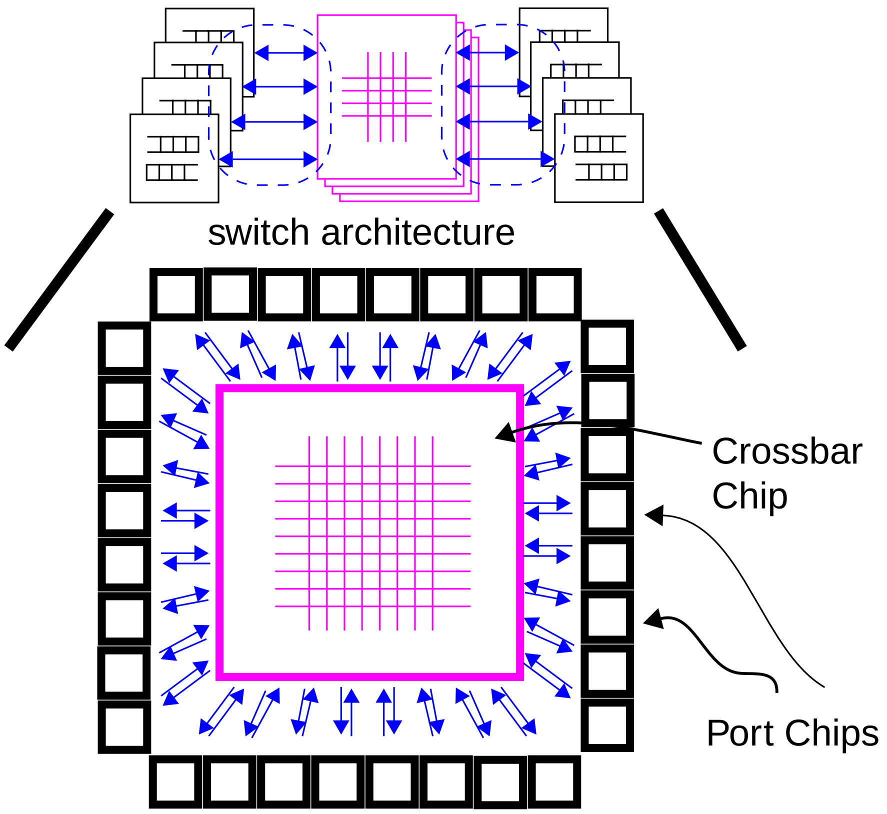


Figure 2.14: A System with High-fanin Chips

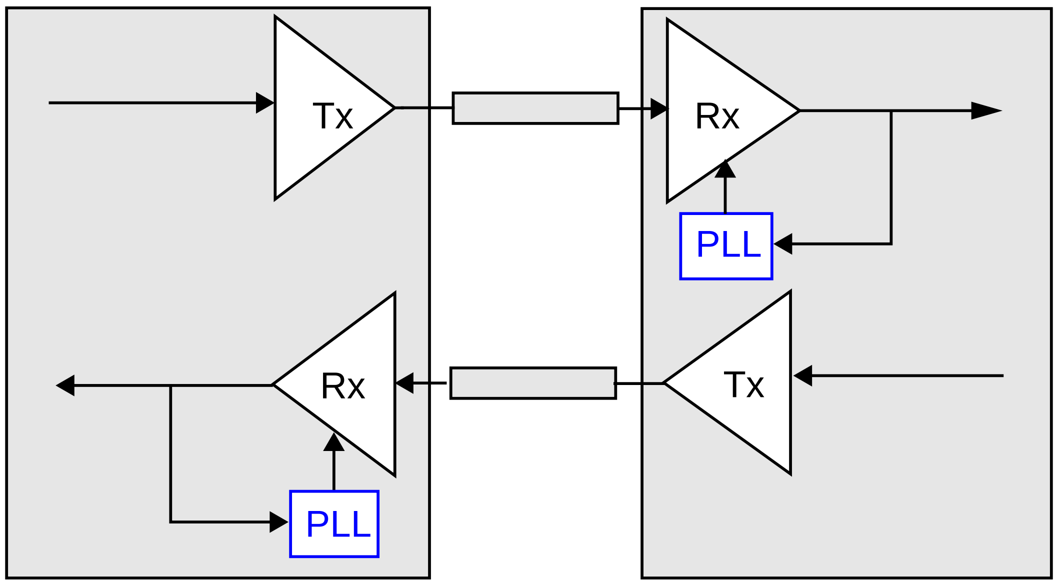


Figure 2.15: A Conventional Serial Link Pair

The Tiny Tera uses the switch architecture shown in Figure 1.2, which can be represented as in the bottom diagram of Figure 2.14. At the center is a 32×32 crossbar chip in the switch fabric, surrounded by 32 chips on the port cards. The arrows represent the high-speed serial links that form the interconnect between the crossbar chip and the port chips. In this design, the crossbar chip has a very high fanin because 32 serial links terminate on it.

Conventional serial links have clock recovery circuits, usually implemented with PLLs at the receiver ends, as shown in Figure 2.15. In the figure, a pair of serial links are transferring data in opposite directions. Each link’s PLL is placed at the end of the link to adjust the receiver clock. As a result, the two links are symmetric.

Figure 2.16 shows a high-fanin chip that uses conventional serial links to interface with its port chips. Each square represents a PLL. With the conventional design, each inbound link to the high-fanin chip needs to have a PLL on the chip to recover the timing. As a result, there are 32 PLLs on a single 32×32 crossbar chip in Figure 2.16.

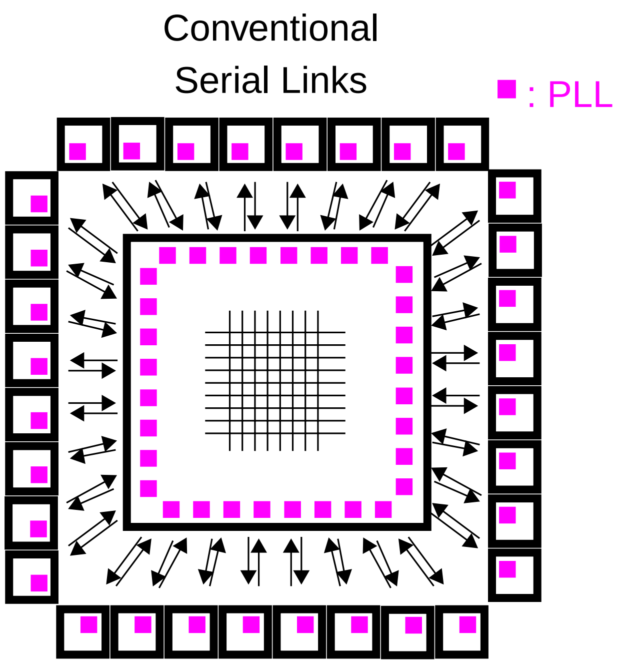


Figure 2.16: A High-fanin Chip and its Port Chips with Conventional Serial Links

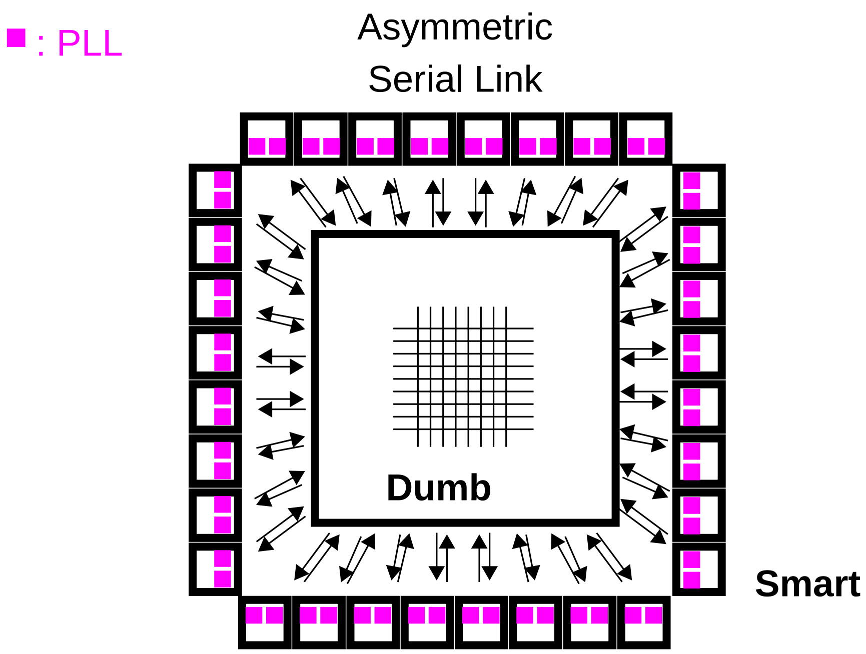


Figure 2.17: A High-fanin Chip and its Port Chips with Asymmetric Serial Links

Having many PLLs on a single chip may lead to problems in terms of power, noise coupling, and latency introduced through the high-fanin chip. To solve these problems, one can move the PLLs away from the crossbar chip to the port chips, which interface with the high-fanin crossbar chip, as shown in Figure 2.17. With this configuration, the timing of the inbound links to the crossbar chip is adjusted at their transmitter end, instead of at the receiver end, and the timing of the crossbar chip is fixed. Fittingly, the crossbar chip is named the *dumb end* and the port chips that adjust the clocks are named the *smart end*. Because the two links of the same pair are no longer symmetric, this type of serial link is called an Asymmetric Serial Link, which is discussed in detail in Chapter 3. The circuit implementation of the link is presented in Chapter 4.

2.5 The Tiny Tera

Chapter 3

Architecture and Synchronization

The Asymmetric Serial Link moves the timing circuits to the transmitter end of the inbound links coming into the high-fanin chips in the network switches to avoid many timing circuits clumped on the high-fanin chips. This chapter first introduces the link architecture in Section 3.1. The asymmetric nature of the link requires us to use explicit calibration packets to calibrate the link periodically. However, periodic link calibration provides very limited tracking bandwidth. Therefore, the asymmetric link needs to use a local PLL and a delay-replica clock generation scheme to track dynamic noise on the chip. The use of the calibration packets allows the link to use the same receiver for both data reception and phase detection. This minimizes the timing offset from the phase detection, which further improves the performance of the link. Section 3.2 discusses the detail of the clock synchronization, including link calibration, the PLL and delay-replica clock generation scheme, and phase detection. Finally, Section 3.3 goes over the synchronization procedure, including the bootup sequence and the periodic calibration.

3.1 Link Architecture

Figure 3.1 shows the architecture of the Asymmetric Serial Link. On the left is the *s*mart end, which is on the port chips of the Tiny Tera switch. On the right is the dumb end, which is on the crossbar chip. In contrast to a conventional serial link pair, which has a PLL at each receiver end (see Figure 2.15), the PLLs of both links are on the same end, the smart end, and there is no PLL at the dumb end. In other words, both the transmitter and receiver clock are fixed on one side.

For the dumb-to-smart link on the bottom of Figure 3.1, the PLL is at the receiver end to adjust the receiver clock. Hence, the link works in the same way as conventional links. For the smart-to-dumb link on the top of Figure 3.1, the PLL is at the transmitter end. Therefore, instead of adjusting the clock at the receiver as the conventional links do, the smart-to-dumb link adjusts the transmitter clock so that the data launched from the *s*mart transmitter can be received by the dumb receiver correctly.

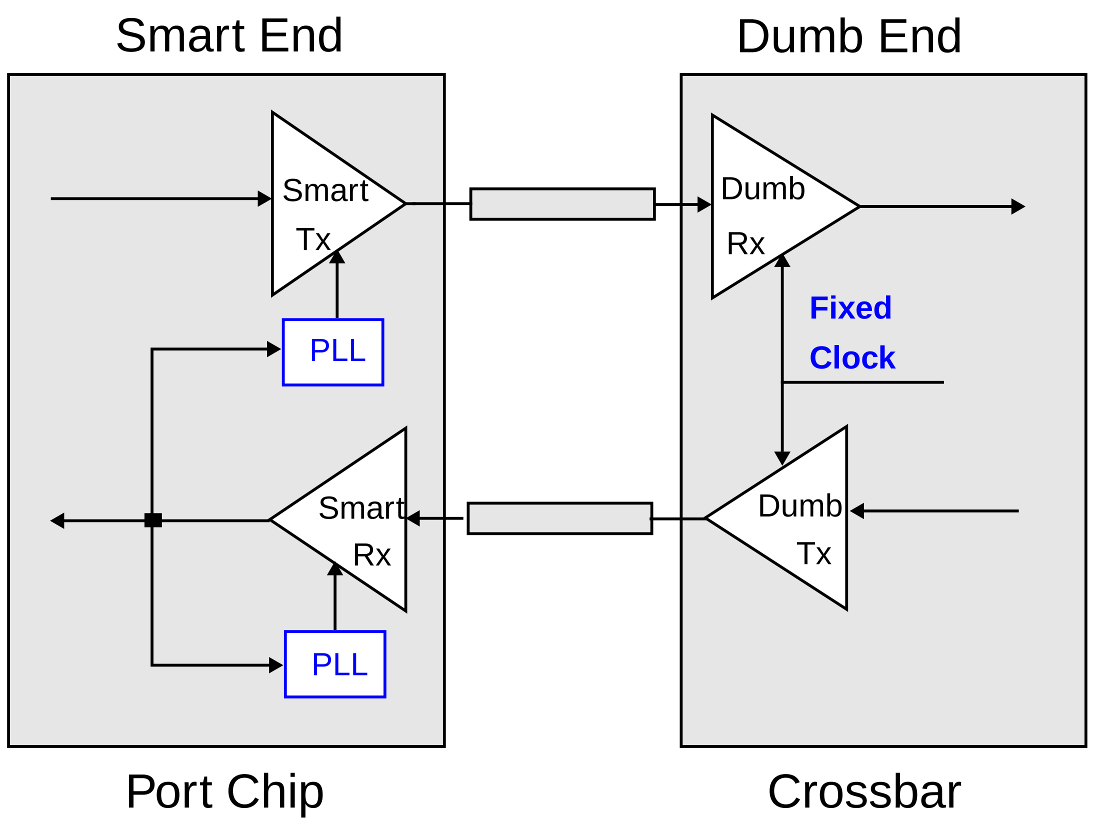


Figure 3.1: Link Architecture of the Asymmetric Serial Link

The key question for the smart-to-dumb link is how to adjust the transmitter clock. For any serial link, the timing information for clock adjustment can be acquired only at the receiver end, as explained in Section 2.3. Therefore, to adjust the transmitter clock, the timing information must be sent from the dumb end to the smart end. To avoid the need for any explicit feedback channel, the timing information can be sent back through the dumb-to-smart link.

Figure 3.2 shows a high-level view of the timing recovery of the Asymmetric Serial Link. To send back the timing information for the smart-to-dumb end correctly, the dumb-to-smart link must be synchronized first. However, both links also have to send their own regular data. As a result, the smart-to-dumb link can be calibrated only periodically, meaning that an explicit calibration packet is necessary to synchronize the asymmetric link. Figure 3.2 shows that both data and the calibration packets are multiplexed into each link.

The use of the calibration packets greatly affects the overall system design because it can provide only limited tracking bandwidth. Other means are needed to keep the link running

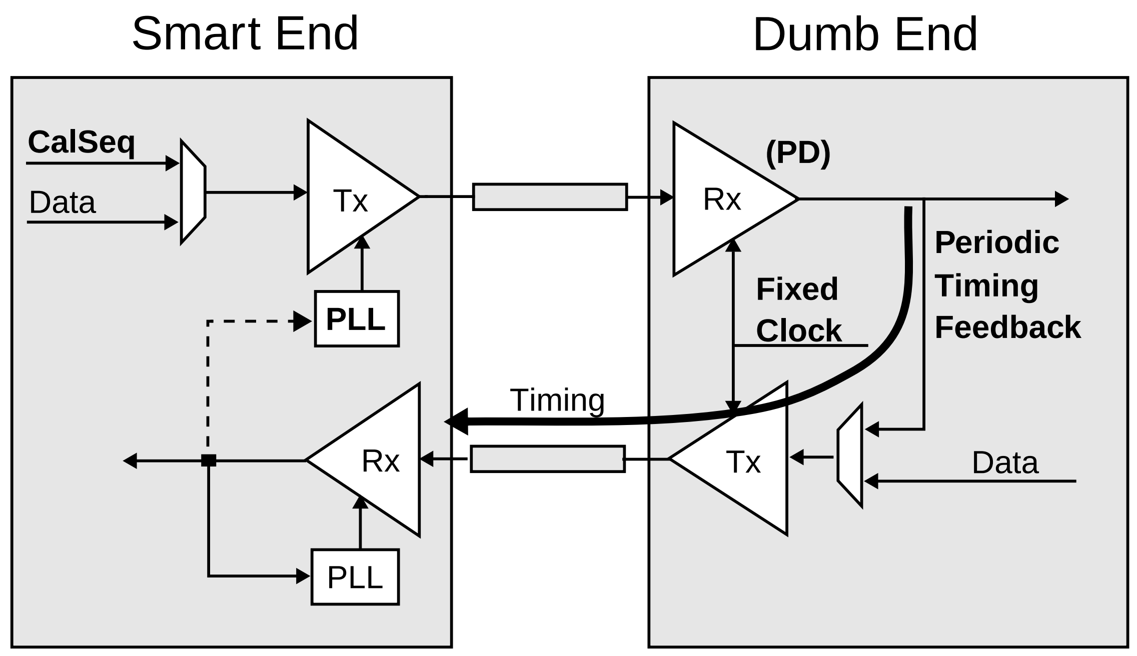


Figure 3.2: Timing Recovery of the Asymmetric Serial Link

reliably within a noisy chip environment. The next section presents in detail the synchronization mechanisms of the link, which not only provide a reasonable tracking bandwidth, but also eliminate the timing offset from the phase detection by taking advantage of the use of the calibration packets.

3.2 Synchronization Mechanisms

The Asymmetric Serial Link was intended to be used in the Tiny Tera so the link follows its data format, which requires different levels of synchronization. Figure 3.3 illustrates the high-level data format of the Tiny Tera, where the data is processed on a frame-by-frame basis. In the Tiny Tera, each frame is nine bytes long, including one header byte and eight data bytes. Section 5.2 will explain the data format in detail. With this data format, the link performs three levels of synchronization: bit, byte, and frame synchronization. The bit synchronization aligns the data sampling point with the center of the data eye, maximizing the timing margins for receiving data bits correctly. The byte synchronization groups data into bytes for the byte-wide datapath of the digital logic. The frame synchronization aligns the frame data.

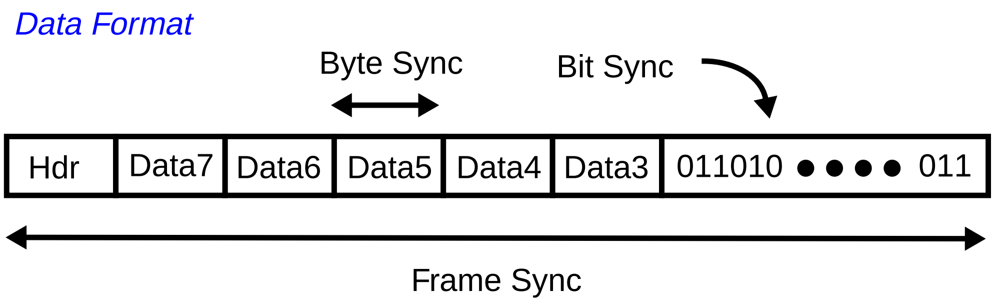


Figure 3.3: Bit, Byte, and Frame Synchronization

As explained in Section 3.1, the asymmetric nature of the link forces it to use calibration packets to calibrate the link. Section 3.2.1 discusses the link calibration mechanism and presents the calibration packet used for the link calibration. The use of explicit calibration can adjust the clock phase to compensate for the static channel delay but its effective tracking bandwidth is quite low. To prevent the noisy CMOS chip environment from injecting dynamic timing jitter to the clocks, the link calibration works in conjunction with a local PLL to track out local delay changes. Section 3.2.2 describes the PLL and delay-replica clock generation circuit on both the smart and dumb ends to reduce system jitter. To further improve the performance of the link, the phase detection of the Asymmetric Serial Link takes advantage of the use of the calibration packets to use the same receiver for both data reception and phase detection. This is presented in Section 3.2.3. Section 3.2.4 presents the implementation of the clock generation circuit on the smart end.

3.2.1 Link Calibration

The calibration packet, shown in Figure 3.4, is used in both the bootup and the periodic calibration. The first byte of the packet is a *control byte*, which contains the calibrate bit and the synchronization status of the other link. The calibrate bit is set to one for calibration and to zero for real data transmission. The synchronization status includes the state of byte synchronization, frame synchronization, and bit synchronization (clock early or late). The remaining three bits of the control byte are reserved for system use. The second and third bytes of the calibration packet are for byte and frame searching and checking. The 1s and 0s are in pairs to avoid aliasing of the byte/frame synchronization pattern even with clock uncertainty as high as one bit time. Therefore, the pattern can be

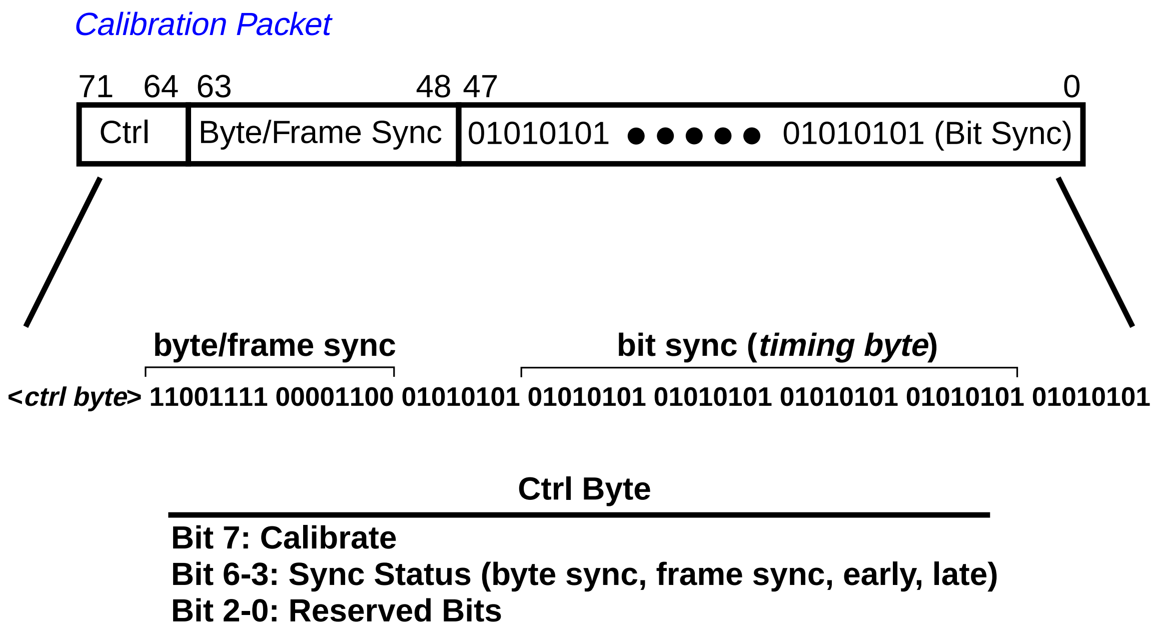

Calibration Packet
71
64
63
48
47
0
Ctrl
Byte/Frame Sync
01010101 ● ● ● ● ● 01010101 (Bit Sync)
byte/frame sync
bit sync (timing byte)
<ctrl byte> 11001111 00001100 01010101 01010101 01010101 01010101 01010101 01010101
Ctrl Byte
Bit 7: Calibrate
Bit 6-3: Sync Status (byte sync, frame sync, early, late)
Bit 2-0: Reserved Bits

Figure 3.4: Calibration Packet

used to check byte synchronization even *before* the bits are synchronized. The frame synchronization checks whether or not the second and third bytes of the frame match the second and third bytes in the calibration frame. The rest of the packet alternates between 1 and 0 and is used for bit synchronization. The Asymmetric Serial Link takes a majority vote of the 32 edges from the 5th to the 8th bytes and makes one clock early or late decision to filter out the effect of any high-frequency noise. The 4th and 9th bytes are not used by the phase-control logic, as will be explained in Section 3.2.3.

The slew rate of the link calibration indicates how fast the link's periodic calibration can correct any timing variation on the link. The calibration slew rate is defined to be the rate needed to track the maximum slew rate of timing variation throughout a variation cycle. Equation (3.1) shows the calibration slew rate based on this definition.

$$\frac{1}{T_{frame} \times (2\pi T_{jitter} / T_{rs})} \times Cal_{occupancy} \tag{3.1}$$

T_{frame} is the frame time, which is 72 bit times and 36ns at 2Gb/s in the Tiny Tera. T_{jitter} is the peak-to-peak jitter tracked by the calibration loop. 2π is added to account for the factor that the loop has to track the maximum slew rate of timing variation. T_{rs} is the phase resolution of the digitally controlled phase selector. The resolution is 7.4ps at 2Gb/s as described in Section 2.4. $\text{Cal}_{\text{occupancy}}$ is the percentage of the calibration frames in the data flow. If 1% of the frames are for calibration, i.e., $\text{Cal}_{\text{occupancy}}$ is 1%, the calibration slew rate from Equation (3.1) is about *650Hz* for tracking 500ps peak-to-peak jitter and is about *6.5kHz* for 50ps jitter. The slew rate can be increased by raising $\text{Cal}_{\text{occupancy}}$, i.e., by calibrating the link more often. However, from the system point of view, the link should be optimized by increasing the time for real data transmission. Therefore, 1% of $\text{Cal}_{\text{occupancy}}$ should be our goal.

As a result, using calibration packets allows us to lock the loop, but will really track only very low frequency changes. For a noisy system to have good performance, we need to design the smart and dumb ends so that the only phase noise that needs to be corrected is very low frequency (below kHz range). This objective can be accomplished by carefully design the clocks on both the smart and dumb chips. The next section explains this design detail.

3.2.2 PLL and Delay-Replica Clock Generation

As introduced in Section 2.5, in the Tiny Tera, a synchronous low jitter reference clock running at the same frequency as the on-chip byte clock is distributed to all the chips so there is no phase slipping. The Asymmetric Serial Link takes advantage of these low-jitter reference clocks and uses one PLL on each chip to phase-lock the timing at the pins of each link. The arrangement is to have one PLL for all the dumb ends on the dumb chip and to have one PLL for each smart end on the smart chip. Since the timing at the pins of the dumb chip is phase-locked to a stable reference, all the slow periodic calibration has to compensate for is the cable delay and the reference clock skew, which are nearly static. This greatly reduces the required calibration rate.

Figure 3.5 shows the delay-replica clock generation on the dumb end. One PLL is shared by all the links on the entire dumb end. The PLL drives the clock tree that provides clocks for all the transmitters and receivers on the dumb end. In the figure, D is a dummy buffer that has the same delay as the transmitter (from the transmitter clock to the transmitter output). The transmitter clocks are preskewed by this dummy buffer delay relative to the external reference clock whereas the receiver clocks are directly tapped from the end of the clock distribution tree. In the current-integrating receiver, which is used by the Asymmetric Serial Link and described in Section 4.2, the setup time is determined by the PMOS sampling switch and is thus very small provided the receiver clock has a fast edge

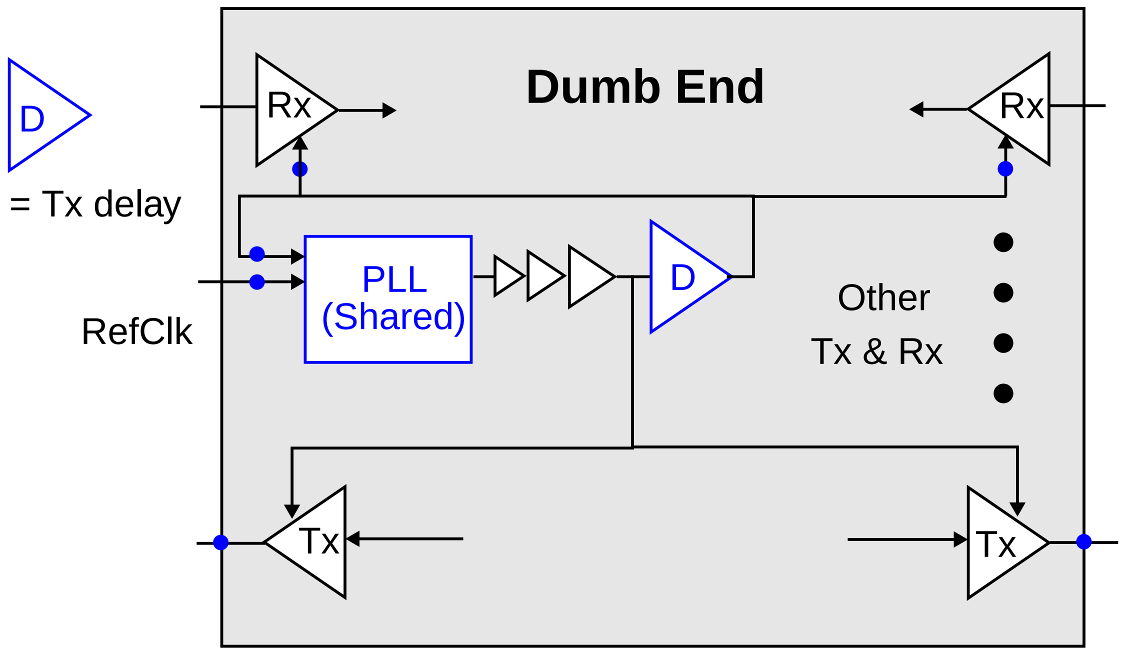


Figure 3.5: Delay-Replica Clock Generation on the Dumb End

rate. This makes the timing at the receiver clocks very close to the timing at the receiver pins. Therefore, with the delay-replica clock generation just described, all the timing at the pins on the dumb end are phase-locked to the external reference clock. The tracking bandwidth of this clock generation loop is determined by the shared PLL.

Similar to the dumb end, the smart end also has to phase-lock the signal timing at the pins. However, in contrast to the dumb end where the clocks are all phase aligned to the external reference clock, the smart end has to adjust its transmitter and receiver clock to compensate for the uncertain cable delay and the reference clock skew. Figure 3.6 shows the delay-replica clock generation on the smart end that fits both requirements. The PLL is implemented by the dual-loop PLL presented in Section 2.4. Each buffer chain in the figure represents a phase selector and a clock buffer. The transmitter and receiver clocks are generated by different phase selectors and phase control logic. The implementation detail is presented in Section 3.2.4.

As shown in the figure, there is a dummy feedback clock path back to the input of the PLL, which phase-locks with the external reference clock. Again, D is the dummy buffer that has the same delay as the transmitter. The clock path of the transmitter clock is again one dummy buffer shorter than the buffer chain of the dummy feedback. Therefore, after

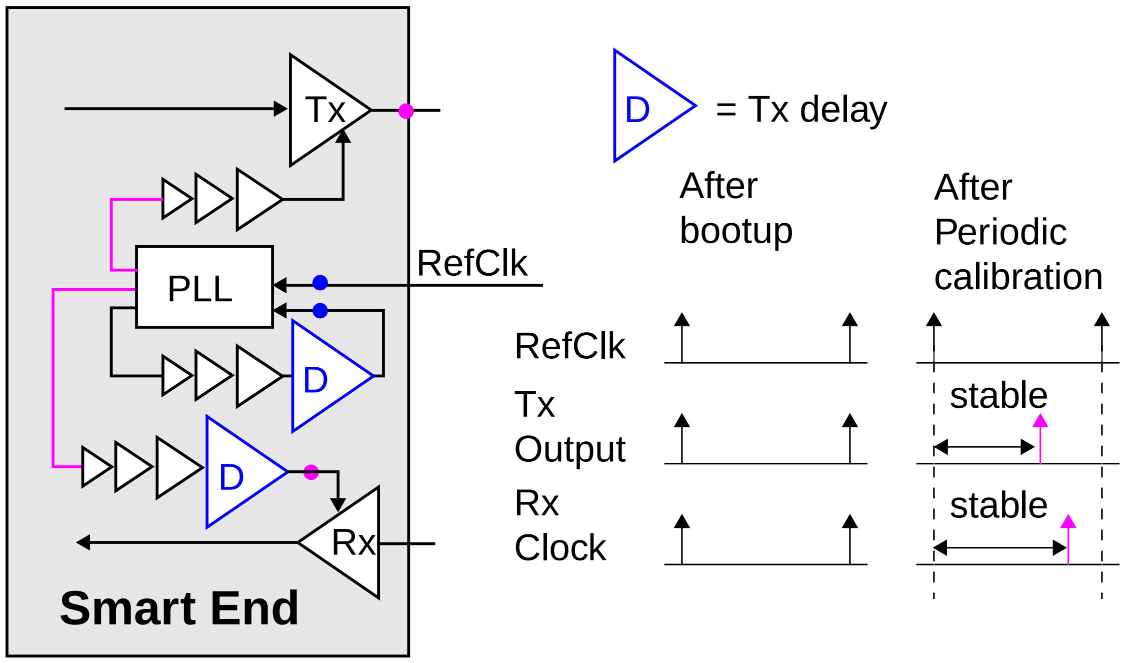


Figure 3.6: Delay-Replica Clock Generation on the Smart End

adding the transmitter delay, the transmitter output is phase-locked to the external reference clock. The receiver's clock path is an identical replica of the dummy feedback clock path so that the receiver clock is phase-locked to the external reference clock. The phase setting of the phase selectors in all the clock paths are set to be the same after bootup so that after bootup, both the transmitter output and receiver clock will be phase-aligned to the external reference clock. After the link calibration adjusts the clocks, their final positions maintain a fixed timing relationship relative to the reference clock even with the existence of supply and thermal noise.

Given the PLL and delay-replica clock generation scheme, any on-chip dynamic variations have been compensated for by the PLL loop. In other words, the tracking bandwidth of the asymmetric link becomes equivalent to the PLL loop bandwidth instead of the slow calibration bandwidth. The tracking bandwidth of most of the conventional serial links is also equal to their PLL loop bandwidth. Therefore, from the tracking bandwidth perspective, the asymmetric link is comparable to any conventional link.

So far, both the interconnect delay and the system clock skew has been assumed to be near DC. While this is true for the interconnect delay, which is the delay on the package, board traces, and connectors, the reference clock skew depends on the accuracy of the clock buffer chips. Most of the ECL or PECL clock buffer chips have very low sensitivity with respect to supply and temperature variations. Therefore, the system clock skew is also almost static over time and can be tracked by a very slow calibration loop.

In reality, the delay matching in the delay-replica clock generation cannot be perfect. Therefore, the timing at the transmitter and receiver pins cannot be perfectly stable. Nevertheless, the dynamic variation comes only from the difference in the delay matching paths. For instance, for a 1ns long clock path, which is typical on the dumb end, 10% of path mismatch is equal to 100 ps delay mismatch. The dynamic variation only affects this 100ps mismatch. 10% supply and substrate noise will result in about 10% variation of the 100ps, i.e., 10ps, for an inverter chain. This should be small enough for 500ps bit time at 2Gb/s and thus can be neglected.

3.2.3 Phase Detection

The serial links recover timing information directly from the received data stream. As explained in Section 2.3, most serial links rely on both data and timing samples to center the data sampling point in a data eye. The distance between the data sampling point and the timing sampling point is half a bit time (90^o). If this distance is off from 90^o when the timing sampling point is at the data transition point, then the data sampling point will not be at the center of the data eye. This degrades the timing margins of the link.

This type of timing offset in the phase detection is very common in conventional schemes where the data receiver and the phase detector are separate circuits, even with the same circuit style (shown in Figure 3.7). In this design, the phase setting of the phase detector is provided by adding a 90^o offset on the phase setting for the data receiver. Difference in delays between the two clock paths, different receiver offsets will affect data centering.

Since the asymmetric link needs to use calibration packets, it does not receive data while it is looking at timing. Thus the link can use the same receiver for both data reception and phase detection, completely eliminating phase offset caused by receiver offset and clock buffer mismatches. The timing information for bit synchronization, which keeps the receiver sampling point at the center of a bit time, is obtained by shifting the receiver clock for the smart receiver and the transmitter clock for the smart transmitter by 90^o. Figure 3.8 shows the case for the smart receiver. The phase selector multiplexes its phase setting between the setting for data reception and that for phase detection. With this scheme, the

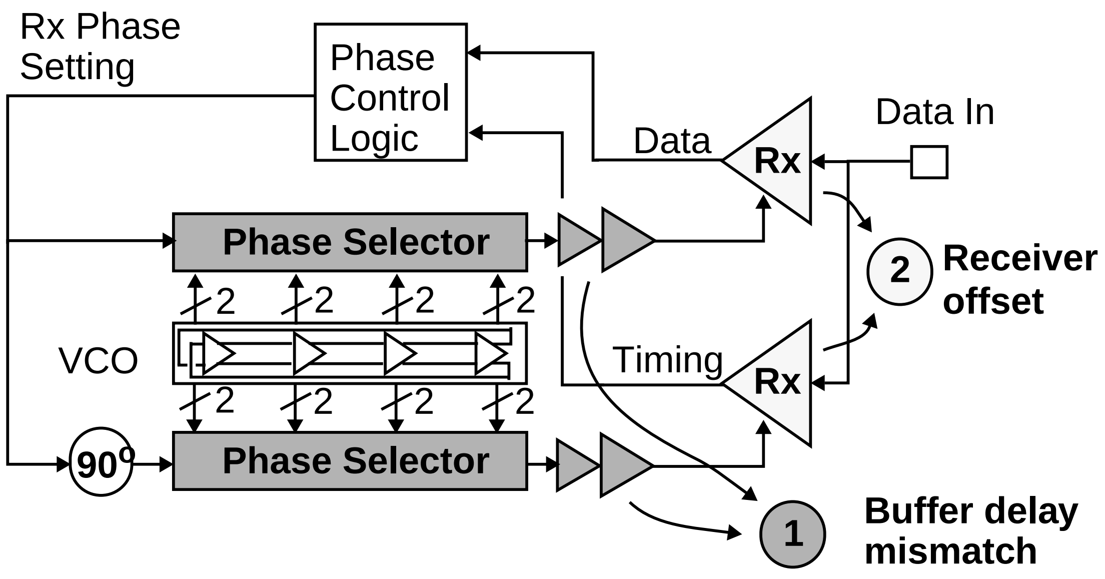


Figure 3.7: Phase Detection using Two Receivers

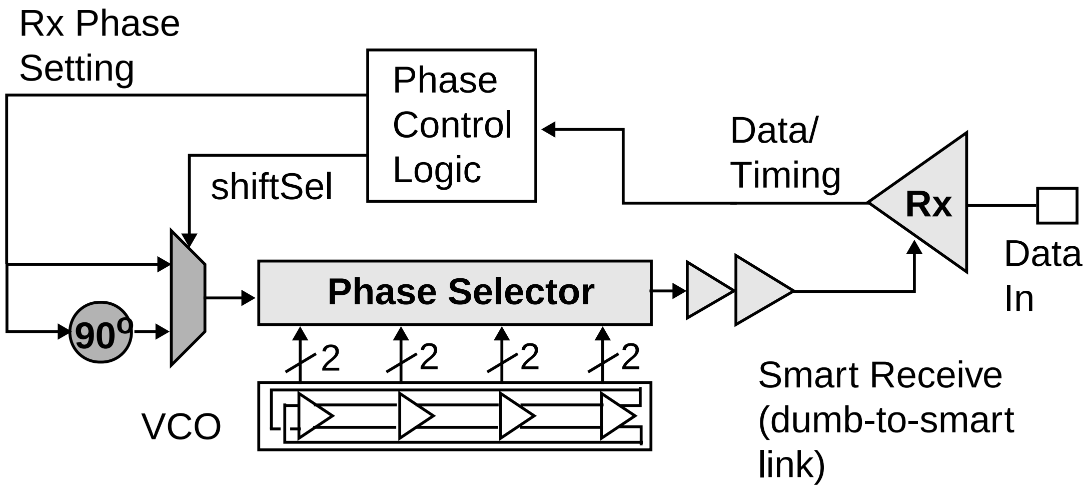
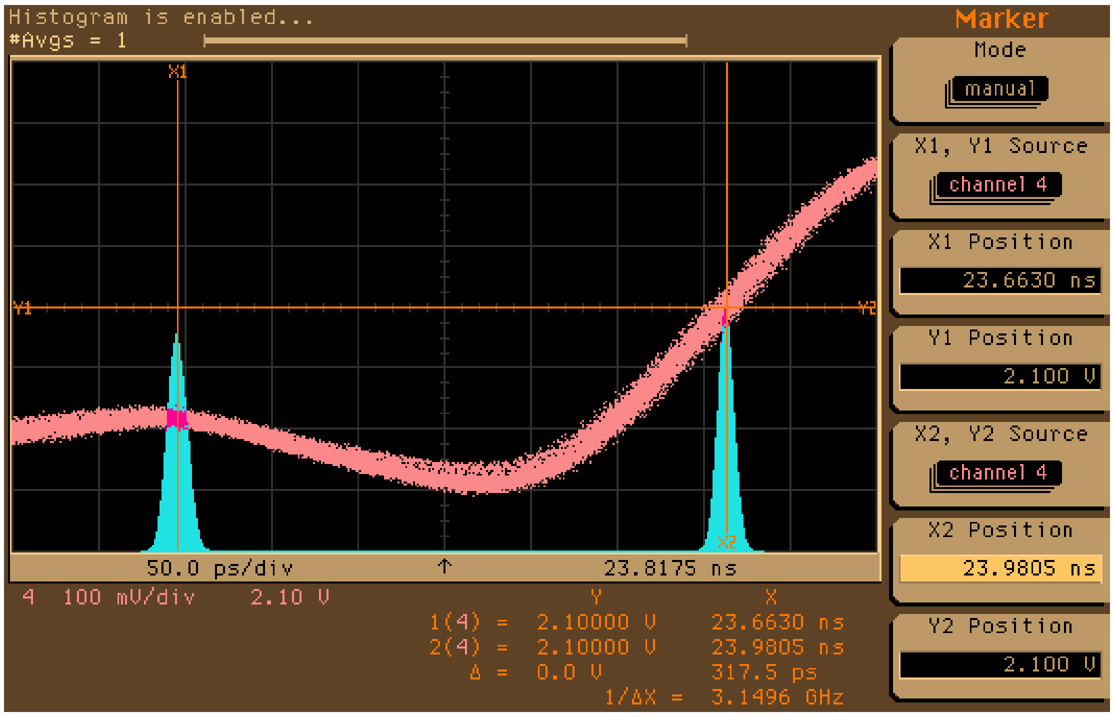


Figure 3.8: Phase Detection using a Single Receiver (for the Smart Receiver)

only timing offset in phase detection is from the clock shifting within the VCO, which can be minimized by careful layout such as the interleaving buffers shown in Figure 4.2.



```
Histogram is enabled...
#Avgs = 1
50.0 ps/div                23.8175 ns
4   100 mV/div   2.10 V
                     Y              X
           1(4) =  2.10000 V   23.6630 ns
           2(4) =  2.10000 V   23.9805 ns
              Δ =  0.0 V       317.5 ps
                       1/ΔX =  3.1496 GHz
X1, Y1 Source  channel 4
X1 Position    23.6630 ns
Y1 Position    2.100 V
X2, Y2 Source  channel 4
X2 Position    23.9805 ns
Y2 Position    2.100 V
```

Figure 3.9: 90^o Phase Shifting at 1.6Gb/s (625ps per bit time)

Figure 3.9 shows the histogram of the 90^o phase shifting measured at 1.6Gb/s. Each bit time at this rate is 625ps and hence half a bit time is 312.5ps. The waveform shows that the clock phase shifts by 317.5ps, i.e., there is only 1.6% deviation. This indicates that the VCO layout is well balanced and phase detection can have very low offset.

3.2.4 Implementation

The clock generation circuitry at the smart end of the links includes a dual-loop PLL with three digitally-controlled phase selectors, as shown in Figure 3.10. The circuit implementation of the dual-loop PLL and phase selectors is discussed in detail in Section 4.1. This section focuses on the loop architecture.

Figure 3.10 shows the architecture of the dual-loop PLL. The timing information from each link is processed by a logic block that controls the phase selectors. Three phase selectors respectively generate the digital divider clock, the transmitter clock, and the receiver clock. The phase setting for the digital divider clock is always fixed at the first phase of the 136 phases of the half-bit-rate clock. The phase settings for all other clocks are also set to the same first phase after the chip reset and are then adjusted later during calibration periods. During normal data transmission, both the transmitter and the receiver clock are setup in a way that the receiver sampling point is at the center of data eyes. During the bit synchronization, which takes place at calibration time using bytes 5-8 of the calibration packets, the transmitter clock phase and the receiver clock phase are shifted by half a bit time (90^o). The time required for the 90^o phase shift to take effect is about one bit time from the time when the phase setting is changed. The phase setting is controlled by digital logic running in the byte clock domain so the phase shifting can be certain only at the byte time level, and not at the bit time level. Therefore, in the calibration packet shown in Figure 3.4, the fourth and the last byte are not used. From the clock shifting standpoint, these two bytes are used as the guard band around the four timing bytes between them during the bit synchronization.

The clock generation circuit is carefully tuned so that all the output clock paths have approximately the same delay. The receiver clock path has an extra delay circuit to match

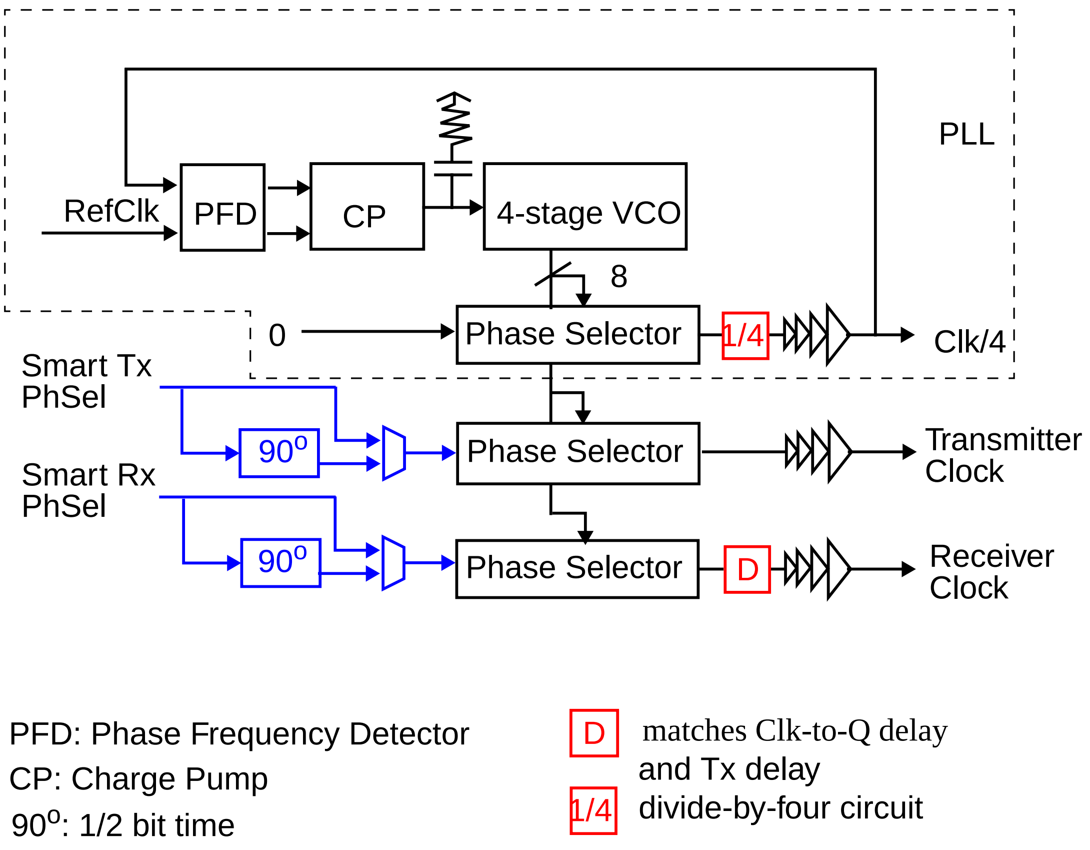


Figure 3.10: The Dual-Loop PLL for the Delay-Replica Clock Generation on the Smart End

the clock-to-output delay of the digital divider in the PLL. The digital divider is sized so that its clock-to-output delay matches that of the transmitter, so no extra delay circuit is needed in the transmitter clock path. This locks the transmitter output and the receiver sampling time to the reference clock, as needed by the delay-replica clock generation scheme.

3.3 Synchronization Procedure

The Asymmetric Serial Link has three stages of synchronization: PLL lock-in, bootup, and periodic calibration. In the first stage, the analog PLL locks to the external reference clocks. This provides stable clock phases to calibrate the link. During the bootup stage, the links are continuously calibrated to synchronize the phase-control loop for both links. And finally, during normal operation, the links periodically calibrate themselves.

3.3.1 Bootup

During bootup, both the smart-to-dumb and dumb-to-smart links continuously calibrate themselves to compensate for the cable delay and system clock skew until byte, frame and bit synchronization is achieved. The dumb-to-smart link is synchronized first; the smart-to-dumb link does not start synchronization until valid data are fed back over the dumb-to-smart link. To avoid any false-locking of the smart-to-dumb link, before the dumb-to-smart link is synchronized, the smart-to-dumb link sends a repeated special pattern[1] that is different from the second and third bytes of the calibration packet over the link so that the link cannot be falsely byte and frame synchronized. Other than this, both links follow a similar synchronization procedure and use the same calibration packet.

The interface of the serial links are byte wide and data are transferred each byte clock. By advancing the transmitter or the receiver clock, one can get the serial data to rotate a full byte clock cycle. Therefore, to synchronize the links, the Asymmetric Serial Link rotates the clock by one phase at a time in one predefined direction until the byte/frame pattern is detected at a byte boundary. Since the two bytes used for the byte synchronization occur only in the second and third bytes of each calibration packet, the frame and byte boundaries are found simultaneously, i.e., the link has to rotate the clock through one byte at most. After the bytes and frames are synchronized, the link starts synchronizing the bits.

1. The pattern is 11000001. The first bit is one to indicate that the link is in calibration mode.

To avoid false byte synchronization due to noise and jitter, a simple byte-match state machine is used. With this state machine, the link will be in byte synchronization only after four consecutive byte matches occur. Once the link is in byte synchronization, four consecutive mismatches will also be required to push it out of byte synchronization. This will keep a few bit errors from causing a loss of byte synchronization and further bit errors. Figure 3.11 shows the state machine that performs this function.

In addition, the long loopback delay decreases the phase margin of the link dramatically. To keep the phase-control loop stable, the new timing information for the next adjustment is taken only after the previous clock adjustment has taken effect.

After byte and frame synchronization are complete, the clocks are fine-tuned to achieve bit synchronization. The link uses a bang-bang phase-control loop controlled by the phase-control logic, which takes a majority vote over 32 edges in each calibration frame. At the beginning of bit synchronization, the clock phase is moved in the same direction as it is for

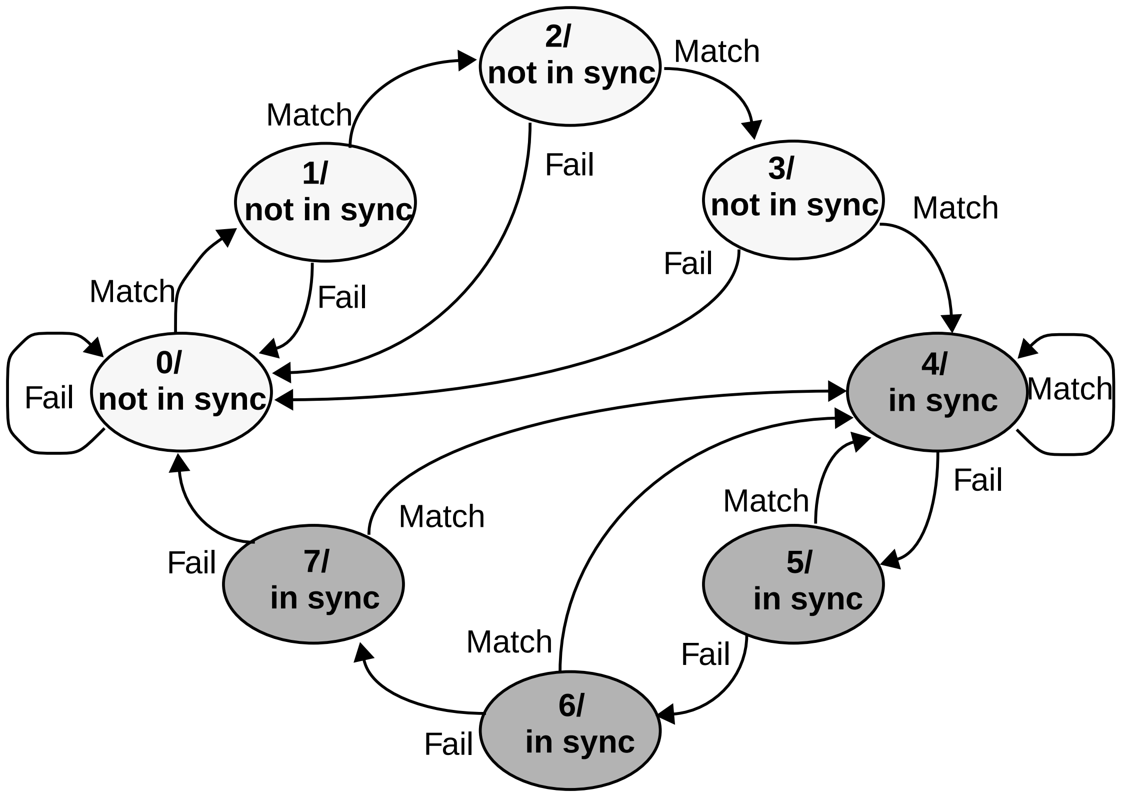


Figure 3.11: Check Byte State Machine

synchronizing bytes and frames until it is close to the center of the bit time. This is accomplished by moving the sampling point until the first time the bang-bang phase-control loop changes the direction of the clock movement. At this point, the link concludes that the edge has been located, completing the bit synchronization process.

When both the smart-to-dumb and dumb-to-smart links are byte, frame and bit synchronized, the links are ready for use. They are then calibrated periodically to track low frequency phase shifts.

In contrast to most of the conventional synchronization schemes, in which the bits are synchronized first and then the bytes and frames are aligned, the Asymmetric Serial Link first looks for the bytes and frames and then finally the bits. The reason this works is that all the clocks are synchronous because of the distributed low-jitter reference clocks in the Tiny Tera. The advantages of synchronizing bytes and frames before the bits are two-fold. First, no extra hardware rotator is necessary after the parallel-to-serial and serial-to-byte converters. This avoids extra time for the data to pass through the rotator and saves the power and area that would have been consumed by the hardware rotator. Second, the phase-control algorithm becomes very simple. The link needs only four bits: byte sync, frame sync, and clock early and late for both the smart-to-dumb and dumb-to-smart links. A simple bang-bang phase-control algorithm that moves clocks one phase at a time is sufficient for all bit, byte and frame synchronization.

3.3.2 Periodic Calibration

During each calibration frame, the status of byte/frame synchronization is first checked. If the links are out of byte/frame synchronization for four frames in a row, the Asymmetric Serial Link will change to the bootup mode and resynchronize the bytes, frames, and bits of the links. If the links remain byte/frame synchronized, the bang-bang phase-control logic moves the clock phase by one step to keep the clock at the center of the bit time.

3.3.3 Link Calibration–Putting Everything Together

Figure 3.12 shows the calibration of the smart-to-dumb link. During calibration, i.e., when the calibrate bit is set, the smart transmitter selects the calibration packet and sends it onto the link to the dumb receiver. When the packet gets to the timing bytes in the calibration packet, the transmitter clock is shifted by 90^o. This effectively shifts the 1010 data by 90^o, as shown on the top portion of Figure 3.12. Since the dumb receiver clock is never changed, the phase shifting of the data will place the data edges right on the receiver

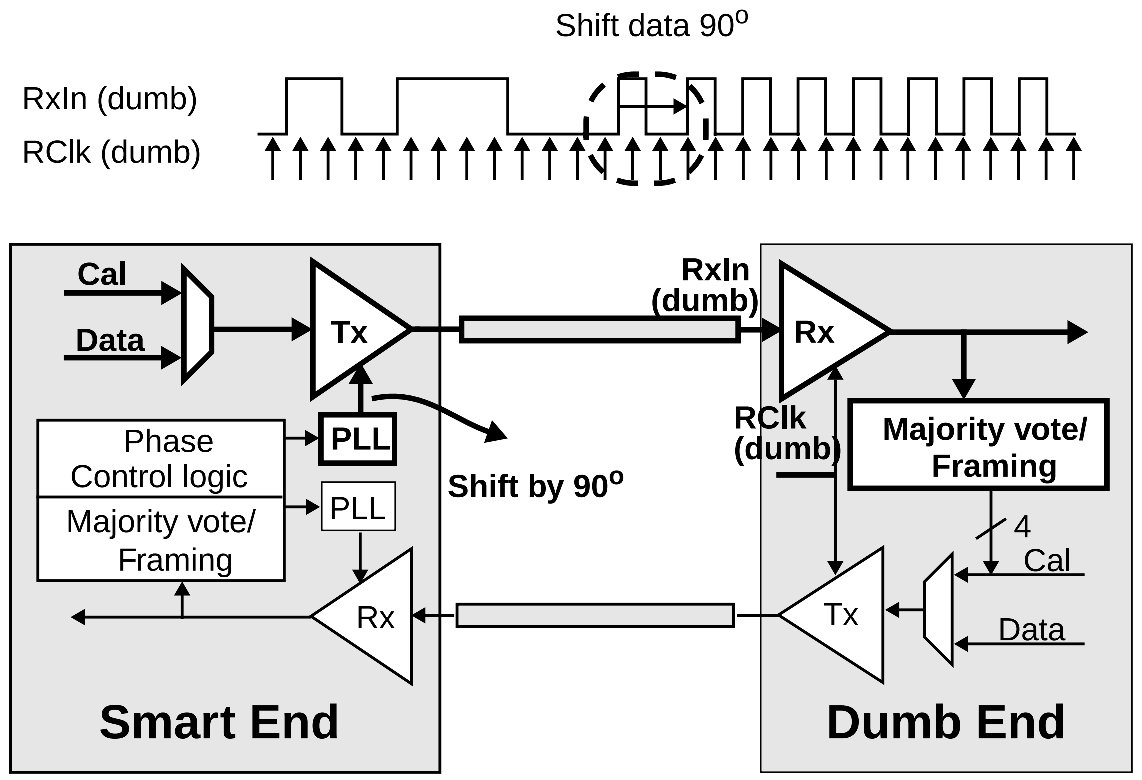


Figure 3.12: Link Calibration of the Smart-to-Dumb Link

sampling point for bit synchronization. The framing logic at the dumb end checks byte and frame synchronization, and the majority vote logic determines whether the clock is early or late. The result is stored on the dumb end until the calibration packet for the dumb-to-smart link arrives in the data stream.

Figure 3.13 shows the calibration of the dumb-to-smart link. When the calibration packet of the dumb-to-smart link arrives, it takes the synchronization status of the smart-to-dumb link from the majority vote and that from the framing block and merges these status bits into the control byte of the calibration packet. The calibration packet is then sent to the smart end. When the timing bytes arrive, the smart receiver clock is shifted by 90^o so that the clock is right on the data transition edges. Similar to what happens in the smart-to-dumb link, there is framing logic to check the byte and frame synchronization and there is the majority vote logic to determine whether the clock is early or late for the bit synchronization. The synchronization status bits of both links are sent to the phase-control logic to adjust the corresponding transmitter or receiver clocks. Before the bytes and

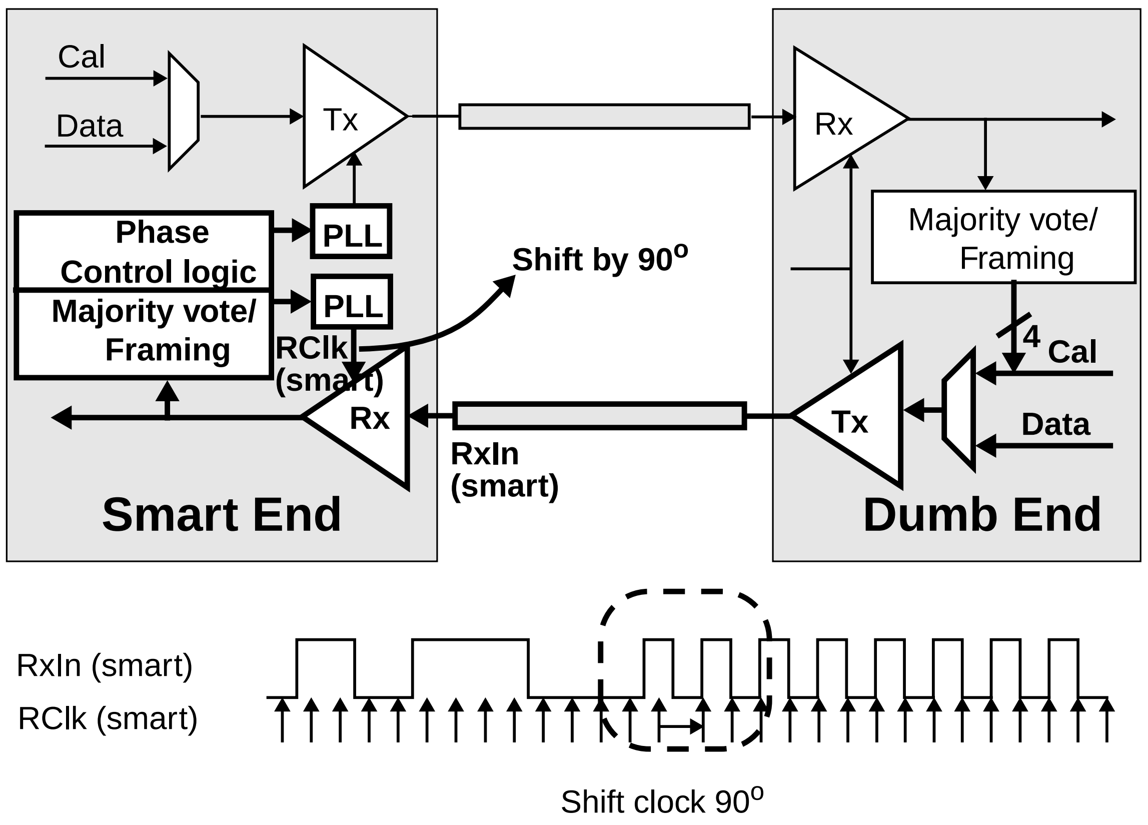


Figure 3.13: Link Calibration of the Dumb-to-Smart Link

frames are synchronized, the clock early or late from bit synchronization is overridden by the predefined direction for the byte and frame synchronization. After the bytes and frames are synchronized, the phase-control logic adjusts the clock based on the clock early or late for bit synchronization.

The above description of the link calibration shows that the smart-to-dumb and dumb-to-smart links are calibrated in a similar manner. The only asymmetry comes from the fact that the PLLs for calibrating the links are only on the smart end.

3.4 Summary

This chapter has presented the architecture and synchronization of the Asymmetric Serial Link. This link has the PLL and the phase-control logic that adjusts the clocks on the smart end while the clocks on the dumb end are fixed. This avoids having many PLLs on each high-fanin crossbar chip in the Tiny Tera.

The smart-to-dumb link is calibrated periodically when the dumb end sends timing information received by the dumb receiver through the dumb-to-smart link. This can compensate for cable delay and reference clock skew over the link. However, the tracking bandwidth of this calibration loop is too low to track timing variations from low frequency supply noise. To solve this problem, the asymmetric link adds a local PLL shared by all the transmitters and receivers on the dumb end. For both the smart and dumb end, the link uses the PLL and delay-replica clock generation to compensate for any timing variations slower than the PLL's loop bandwidth. This greatly reduces the required periodic calibration rate. The use of calibration packets allows the link to use the same receiver for both data reception and phase detection by shifting the clocks between the data and timing clock phases. This reduces the static phase error of the 90^{o} phase shift and thus improves the performance of the link. The only offset in 90^{o} phase shift comes from the clock shifting in the VCO, which can be optimized by very careful layout.

While the architecture and synchronization mechanisms of the Asymmetric Serial Link resolve the system-level problem created by having many links connected to a high-fanin chip, the performance of the link, in terms of speed and robustness within a noisy environment, is strongly affected by the detailed circuit implementation of each individual block of the link. Chapter 4 presents several key circuits used by the Asymmetric Serial Link.

Chapter 4

Circuit Building Blocks

This chapter examines each individual circuit block used in the Asymmetric Serial Link and discusses the blocks' circuit architecture and implementation details. Section 4.1 presents the circuits used in the dual-loop PLL. Its inner loop provides coarse but stable phases from the VCO. The outer loop uses digital-controlled phase selectors to provide high-resolution clock phases.

The signalling circuits use differential receivers and transmitters to reject common-mode noise from the on-chip power supply. Section 4.2 presents the receiver circuit, which is based on the current-integrating receiver designed by Sidiropoulos [34]. Section 4.3 presents the current-mode differential-pair transmitter, which isolates the effect of the supply noise in the transmitter chip from the receiver chip and to reduce the dI/dt noise on the transmitter chip.

Section 4.4 discusses the design of the parallel-to-serial and serial-to-parallel converters that handle the speed conversion between the half-bit-rate and byte clock domains. To match the real designed hardware, the parallel-to-serial and serial-to-parallel converters are called byte-to-serial and serial-to-byte converters in the thesis.

4.1 Dual-loop PLL with Digitally Controlled Phase Selectors

The dual-loop PLL is composed of an inner analog loop and one or more outer digital phase control loops. Section 4.1.1 covers the circuits used in the inner analog loop and the following subsection discusses the phase interpolator used in the outer loop. Section 4.1.3 explains design issues with having multiple phase selectors in the outer loop. Section 4.1.4 presents the replica bias circuits for both the inner and outer loops. Section 4.1.5 focuses on the differential-to-single-ended converter and its duty-cycle corrector. The latter is crucial because the data is transmitted on both edges of the half-bit-rate clock. Finally, Section 4.1.6 discusses how to start up the PLL properly. Most of the circuits have been

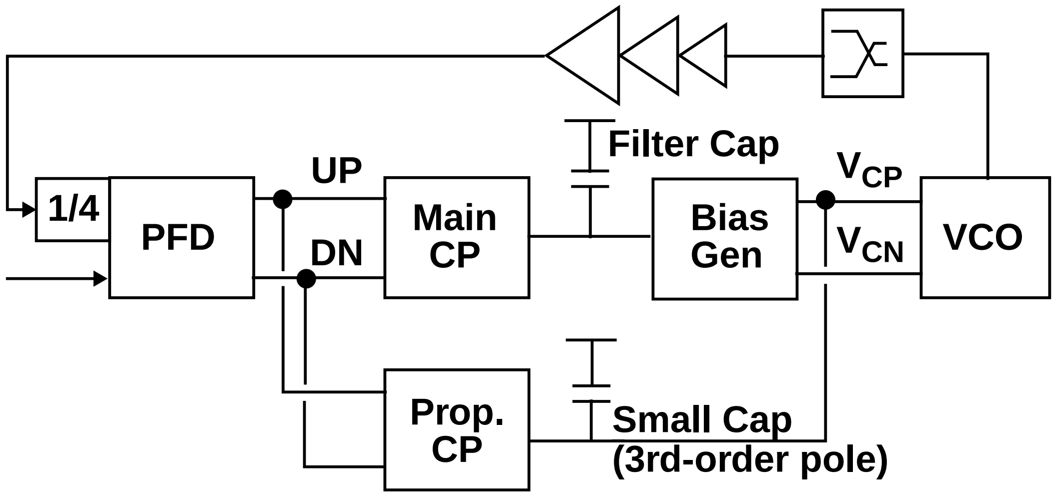


Figure 4.1: Block Diagram of the Inner Analog Loop

presented before. The intent here is not only to review the key circuit blocks but also to point out several important details of the circuits from the practical design standpoint.

4.1.1 Inner Analog Loop

Figure 4.1 shows the circuit block diagram of the inner analog loop. It is a third-order loop comprising a four-stage VCO, a phase frequency detector, a main charge pump filter, a proportional charge pump, and a replica bias generator for the VCO [51]. The proportional charge pump is used to generate the equivalent resistance for the feedforward zero from the load resistance of the bias generator [53]. This charge pump circuit causes ripples on V_{CP}, which introduces a duty-cycle error on the VCO clocks. To make things worse, the large ripples on V_{CP} may make the loop operate in a non-linear fashion and invalidate all the stability analysis for the PLLs, which assumes the entire loop operates linearly. To avoid both problems, a capacitance is added on the V_{CP} to minimize the disturbance. This capacitance, however, introduces a third-order pole in the loop transfer function, which may reduce the phase margin of the loop and hence cannot be arbitrarily large. In general, the capacitance is made as large as possible while the phase margin of the PLL is kept at over 60^o.

The four-stage VCO uses differential elements with replica-biased symmetric loads [52], as shown in Figure 4.2. To balance the load of each stage, the VCO buffers are laid out in an interleaved fashion. The symmetric load is biased at V_{CP}, the voltage on the filter

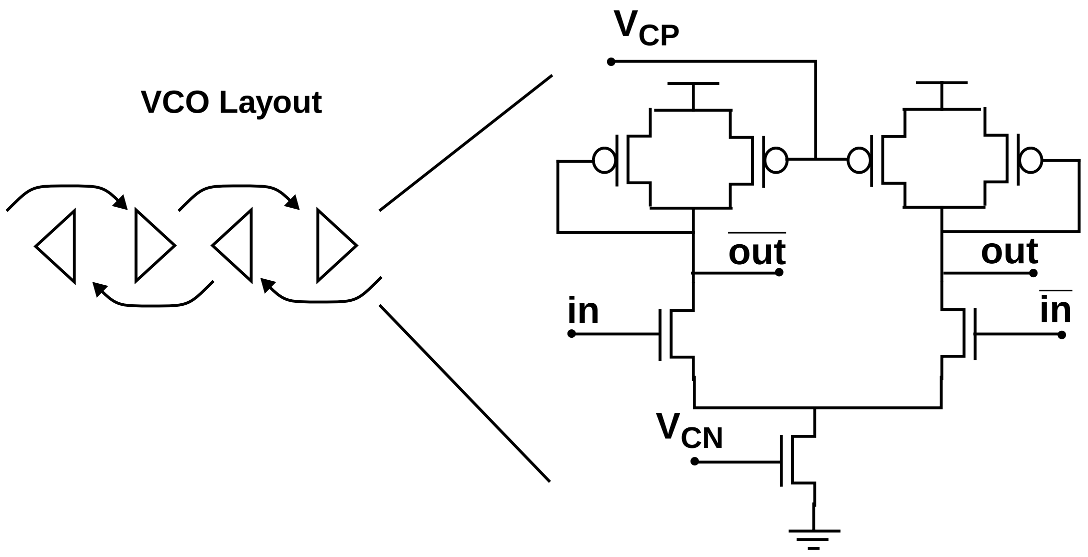


Figure 4.2: VCO Layout and its Differential Buffer with Symmetric Loads

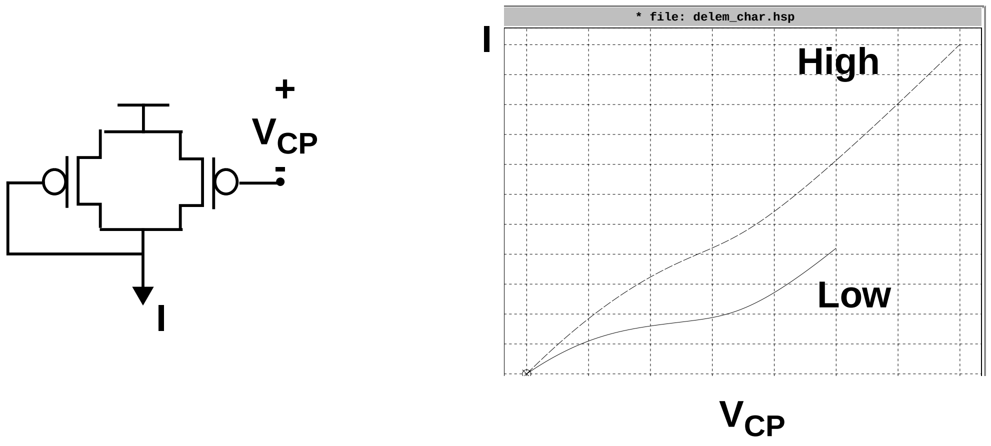


Figure 4.3: The Symmetric Load and its I-V Curve

capacitor. V_{CN} is generated from a replica-biased generator to bias the current source of the differential buffers so that the buffers have an output swing from V_{DD} to V_{DD}-V_{CP}.

Figure 4.3 shows the I-V curve of the symmetric load. At large V_{CP}, the curve is close to that for linear resistance. At low V_{CP}, it bends further away from the curve for linear resistance but it exhibits more symmetric behavior than the curve for high V_{CP}. Therefore, the dynamic supply rejection ratio of the symmetric load should not depend on V_{CP} too much. However, small V_{CP} means small gate overdrive (V_{GS} - V_{TH}). Although PMOS devices made in conventional nwell processes do not suffer from substrate noise, any coupling noise to V_{CP} will cause large variations on slope of the I-V curve of the symmetric load. This introduces large delay variations on the VCO buffers. Therefore, the PMOS symmetric load should not be oversized. Likewise, V_{CN} should also be biased to provide sufficient gate overdrive on the NMOS current source in the VCO buffers. In fact, because NMOS devices suffer from threshhold variations due to substrate noise, it is even more crucial not to oversize the NMOS current source. In general, the sizing of both the PMOS symmetric load and NMOS current source should be only large enough to support the speed in the slow process corner but small enough to provide sufficient gate overdrive in the fast process corner. The size of the differential pair in the VCO buffer also has effect on the operating frequency. Smaller size will provide higher operating frequency but it may push the NMOS current source out of saturation region in the slow corner because of the large gate overdrive of the differential pair itself. Increasing the size of the differential pair will slow down the buffer but will help to provide larger gate overdrive for the PMOS load and more saturation margin for the NMOS current source. However, large differential pair may introduce a side effect–the Miller effect. The Miller effect will cause the slew rate of the VCO signals to become smaller at the middle of the swing. This will introduce large jitter. In conclusion, the designer should be careful in sizing the VCO buffers. One should never blindly optimize the buffer for one extreme condition without considering others. The sizes should be chosen for running the VCO at the operating frequency while each transistor in the buffer still has sufficiently large margin for a wide frequency range over different operating conditions.

Figure 4.4 shows the self-biased charge pump, which has a circuit structure similar to that of the symmetric-load differential buffers and is biased by the same V_{CN} as the VCO differential buffers for its current sources. As a result, the current of the charge pump circuit is proportional to the current of the differential buffers in the VCO [51]. The advantage of the self-biased charge pump is that the loop bandwidth of the PLL is proportional to the operating frequency. This widens the operating range of the PLL. In the charge pump circuit, the delay of the DN to V_{CP} is equal to the current mirror delay from the left differential buffer to the middle. This delay is matched by the buffer delay of the rightmost buffer in Figure 4.4 to prevent the charge pump circuit from inducing large ripples on its output, V_{CP}.

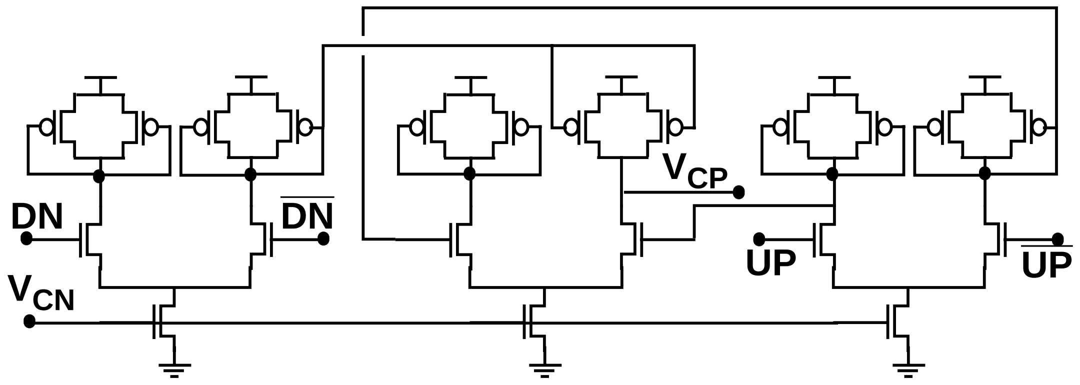


Figure 4.4: Delay Balanced Self-Biased Charge Pump Circuit

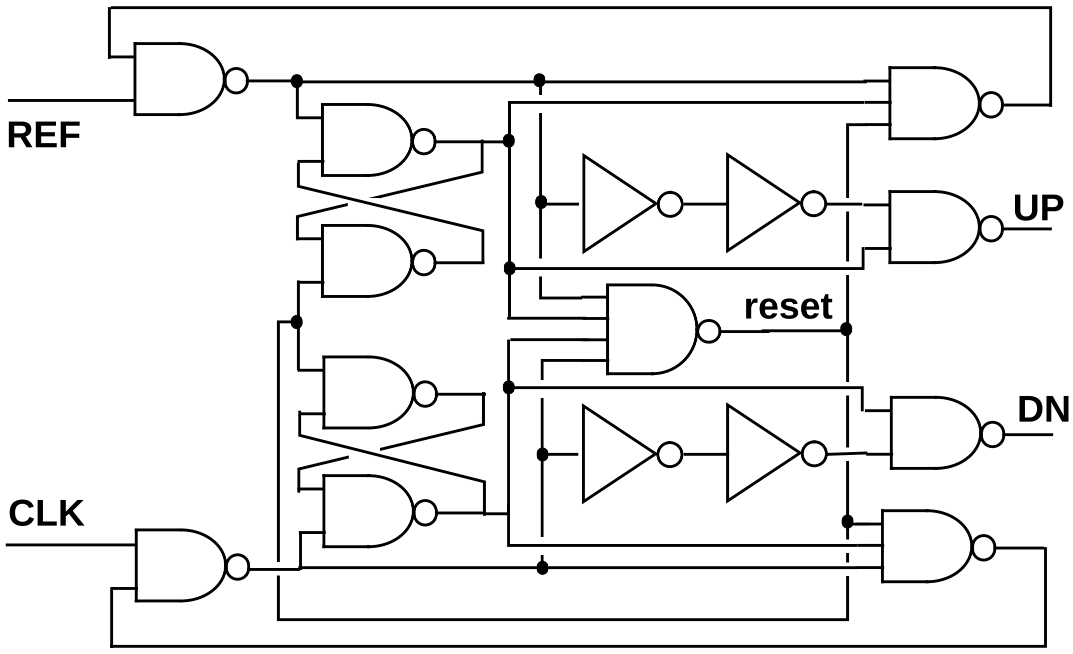


Figure 4.5: Phase Frequency Detector

Figure 4.5 shows the phase frequency detector used in the inner analog loop. The phase frequency detector is based on the design by Maneatis [51], which was adapted from the circuit presented by Jeong *et al.* [45] and Young *et al.* [57]. One of the key design considerations for the phase frequency detector is deadband avoidance for the charge pump filter: the voltage on the filter capacitor should always have a monotonic and one-to-one relationship with the difference between the phase of the input clock and that of the reference clock. To avoid deadband, equal and short duration of the UP and DN pulses are needed for the in-phase inputs. Conventionally, this is achieved by adding extra delay in the reset path [89], which, however, limits the maximum operating frequency of the phase frequency detector. The critical path of the phase frequency detector is from the 2-input NAND for the input clock, through the 4-input NAND for the reset signal and through any buffer for the reset (used in [89]), and finally through a 2-input NAND gate in the SR latch. Maneatis reconfigured the logic to generate the UP and DN signals from the duplication of the 3-input NAND gates on the far right with their reset input deleted [51]. The UP and DN signals are still reset by a slower path through the 4-input NAND and the SR latch. As a result, no extra buffers are necessary in the reset path and thus the maximum operating frequency is increased. Nevertheless, due to the extended reset path, the overlapping UP and DN pulses, which eliminate the deadband, become very long. This will cause large ripples on the control line if the up and down currents in the charge pump are not perfectly matched. The ideal overlapping pulses have to be only long enough to reach full swing (from V_{DD} to GND) for all the operating corners. In Figure 4.5, extra buffers are added in the forward path to delay the start of the overlapping UP and DN pulses so that the pulse width is minimized. Since this extra delay is not added in the critical path, the maximum operating frequency remains unchanged.

Figure 4.6 shows the relationship between the net charge pump current into the filter capacitor and the difference between the phases of the input clocks across different process corners. It shows that there is no deadband in any case. The difference in the zero crossing points among the curves are caused by the process-dependent static offset of the phase frequency detector and the charge pump circuit.

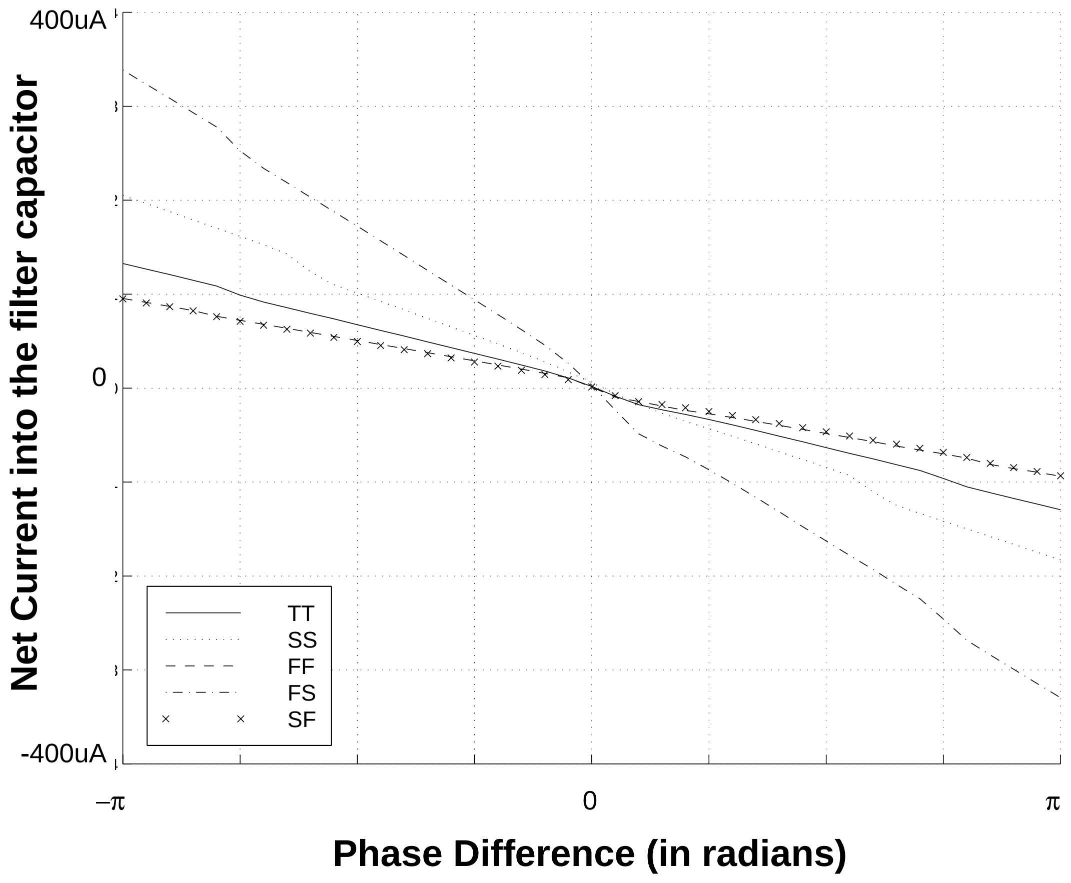


Figure 4.6: Current Output of the Phase Frequency Detector and the Charge Pump Circuit

4.1.2 Clock Phase Interpolator

The clock interpolator for the phase selectors in Figure 2.11 is shown in Figure 4.7. It is based on the design by Sidiropoulos [56][91]. The interpolator takes two clocks that are 45^o apart with sufficient overlap in their edges from a 4-stage VCO, and it generates an output clock that depends on the weighting of the two clock phases. If the weighting is completely on $\phi1$ in the figure, the output is a delayed version of $\phi1$, and the same relationship holds for $\phi2$. If the weighting is somewhere in between, the output is somewhere between the interpolator delay version of both $\phi1$ and $\phi2$. The position depends on the weighting factor.

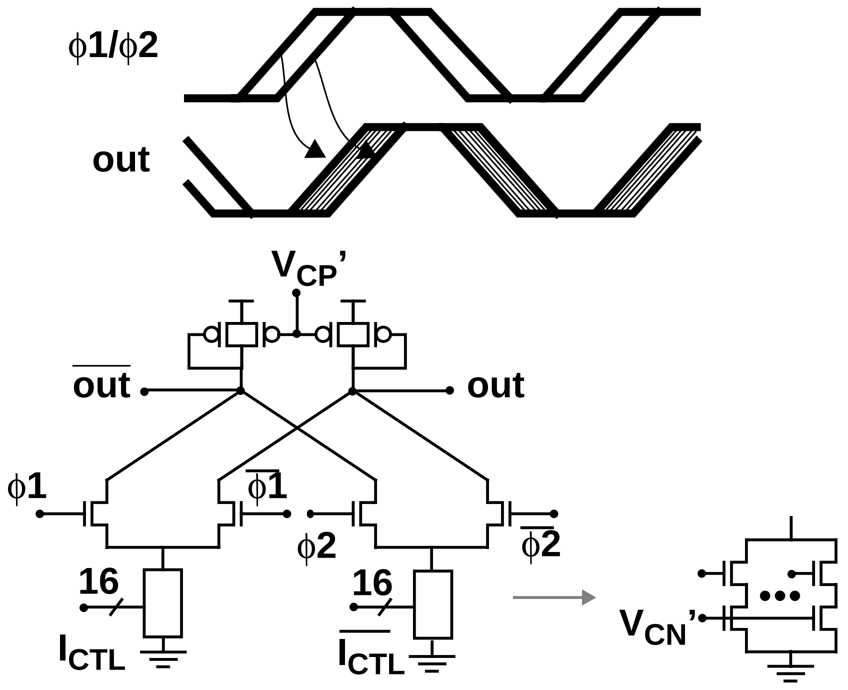


Figure 4.7: Clock Phase Interpolator

The load of the interpolator is biased by V_{CP}', and the multi-leg current sources are biased by V_{CN}'. V_{CP}' and V_{CN}' are generated by a replica-biased generator similar to that for the VCO except that, unlike the VCO, V_{CP}' is not connected to the proportional charge pump. The bias generation will be discussed in the next subsection.

To avoid glitches when the weighting is changed, the weighting factor uses thermometer coding so that the adjustment of the current sources will always be monotonically incremented or decremented by the same small amount. This precaution dramatically increases the hardware required since these thermometer codes have to be stored. The codes have 16 bits in the link. The number of bits needed is determined by the resolution necessary to minimize the dithering jitter. As described in Section 2.4, at 2Gb/s with a 1 GHz VCO clock, the resolution is 7.4ps for 8×17 = 136 phases for a VCO cycle.

The clock phase interpolators have two subtle design details. First, to prevent the change of clock inputs from affecting the interpolator output, the weight of the side that changes its input should be set to zero for the change to occur. Ideally this makes the input change transparent to the output. Second, even if the weight is zero, the change of the clock inputs will be still coupled to the output because of the gate overlap capacitance. This introduces another phase step that is larger than the normal step. To minimize the maximum step size, the change of the input phase while the weight remains unchanged is considered an additional step in the digital phase control loop. As a result, each 45^o of the VCO cycle has 17 steps instead of 16.

Figure 4.8 shows the phase steps of the clock phase interpolator measured at 1.6 Gb/s. This measurement is done across three interpolation boundaries where each boundary is 45^o from its nearest neighbor. The measurement shows that the curve is reasonably linear (maximum INL is about 2LSB) but it is not monotonic at the interpolation boundary. Such nonideality is tolerable in the close-loop phase control loop because the phase control loop will always find the phase closest to the ideal sampling point. The only effect of the non-monotonicity will be that more steps will be necessary to reach the final phase. The

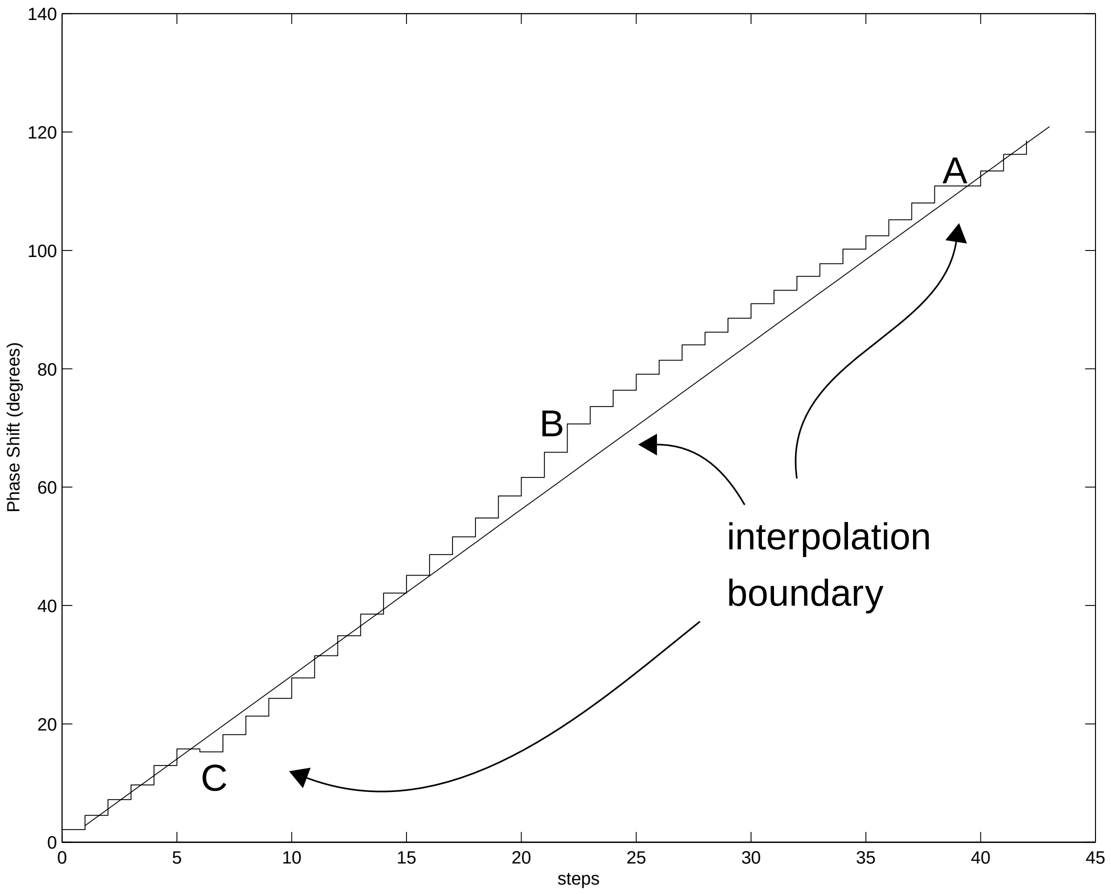


Figure 4.8: Phase Steps for the Clock Phase Interpolators at 1.6Gb/s

nonlinearity, too, is tolerable. It is less of a concern than the maximum step size, which affects the dithering jitter as mentioned above.

In Figure 4.8, the phase steps at the upper right (A) and lower left ends (C) of the interpolation boundary are close to zero while the phase step at the center (B) appears to be much larger. There are two reasons for this discrepancy. First, there is a design mistake in the digital phase control block. The larger step size at B occurs when both the clock inputs to the interpolators and the interpolator weight are changed simultaneously, instead of one at a time. Second, the total size of the two steps at both A and C is still smaller than the step size at B. This may be caused by the delay mismatch from the VCO to the two interpolator inputs, which cause different coupling effects at the interpolation boundary. These factors increase the maximum step size to 20ps, which will increase the clock jitter.

4.1.3 Multiple Phase Selectors

As explained in Section 2.4, each dual-loop PLL can have multiple phase selectors in the outer loop to provide independent clocks. Each set of phase control logic in Figure 2.12 can change its setting independently. However, changing the phase setting varies the loading of the individual clock multiplexor circuit in the phase selector. This is illustrated in Figure 4.9. When the phase select changes from sel1 to sel2, The state of input transistors of $\phi 1$ change from conducting current to off. This changes the capacitance seen by $\phi 1$ because transistors' capacitance varies with current. As a result, $\phi 1$ will be affected

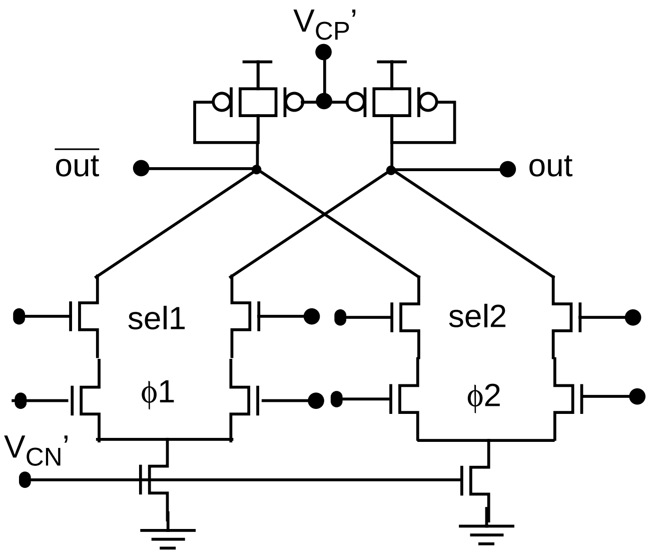


Figure 4.9: A 2-to-1 Clock Multiplexor

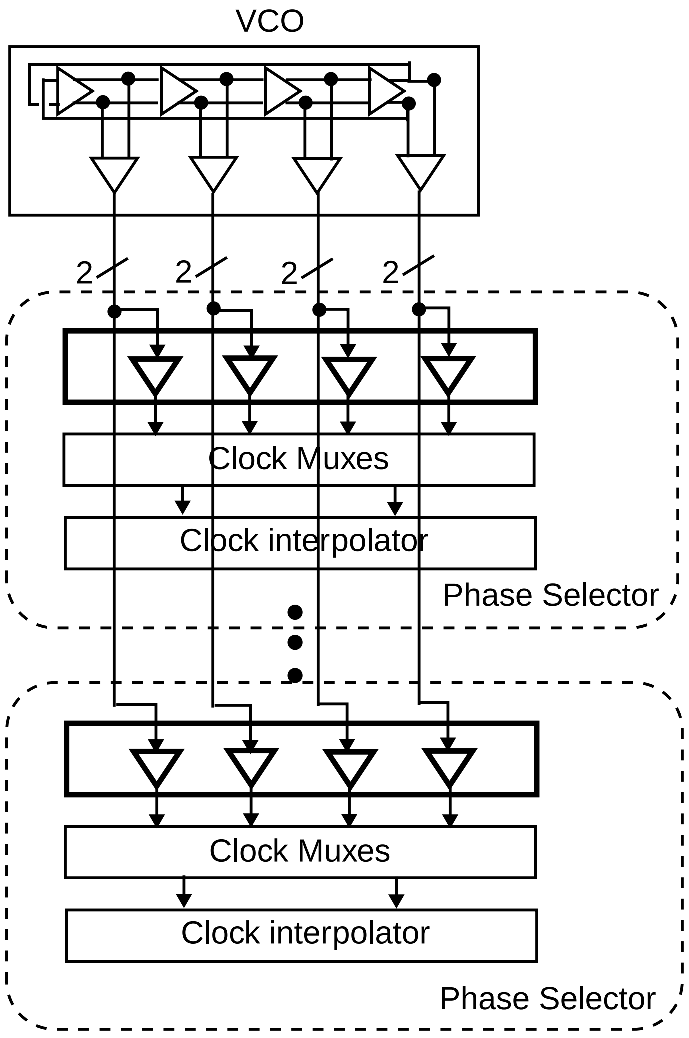


Figure 4.10: Buffering between the VCO and each Phase Selector

by the change of the phase setting. If $\phi 1$ also drives clock multiplexors in other phase selectors, then the change of the phase setting in one phase selector can affect the clocks in other phase selectors, too. To solve this problem, one should buffer each phase selector, as shown in Figure 4.10. The buffer is implemented using the same delay element in the VCO, but sized differently. This buffering isolates the effect of loading change from the clock multiplexors due to the change of phase setting and hence prevents each phase selector from affecting any others.

4.1.4 Bias Generators

The bias generators for both the inner and outer loops of the dual-loop PLL are based on the replica-feedback biasing generators from Maneatis [52][89], as shown in Figure 4.11. The biasing circuit continuously adjusts V_{CN} to keep the signal swing of the VCO fixed from V_{DD} to V_{DD}-V_{CP} even with the disturbance of the supply and substrate noise. The bandwidth of this biasing circuit determines the frequency spectrum under which the variations due to the supply and substrate noise can be corrected. The bias signal for the opamp is self-biased based on V_{CN}. Therefore, the bandwidth of the bias circuit is also proportional to the operating frequency (i.e., to the phase detecting rate) of the inner PLL loop. In general, the bandwidth of the bias circuit is designed to be much higher than the operating frequency of the digital logic since most of the supply and substrate noise is at this frequency. The NMOS capacitor at the output of the opamp is meant to compensate for the feedback, keeping the bias generator stable.

Figure 4.12 shows the block diagram of the bias generation for the dual-loop PLL. Each bias generator block is a replica-feedback biasing circuit of the type shown in Figure 4.11. As shown in Figure 4.12, there are two bias generators: one for the VCO and the other for

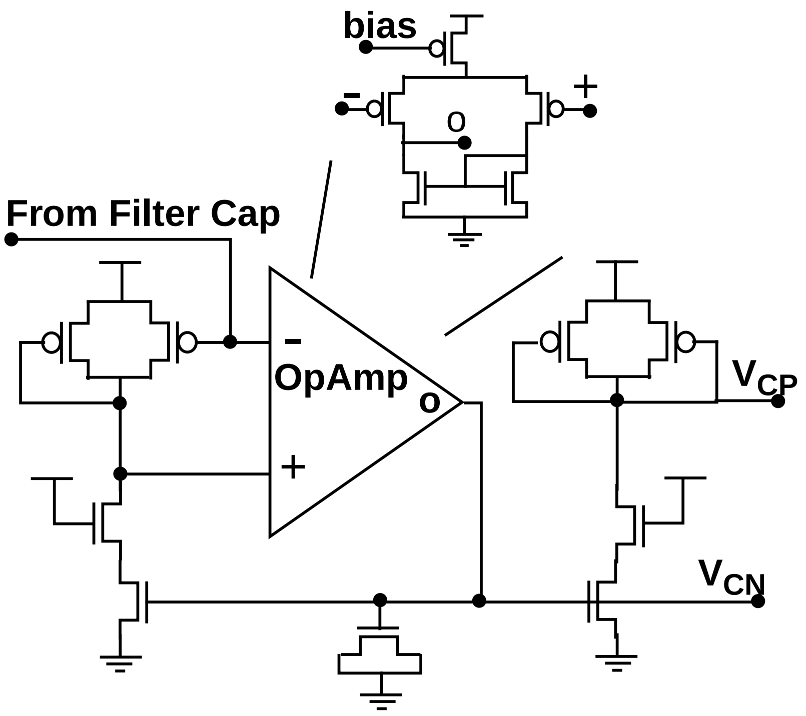


Figure 4.11: Replica-Feedback Biasing Generator

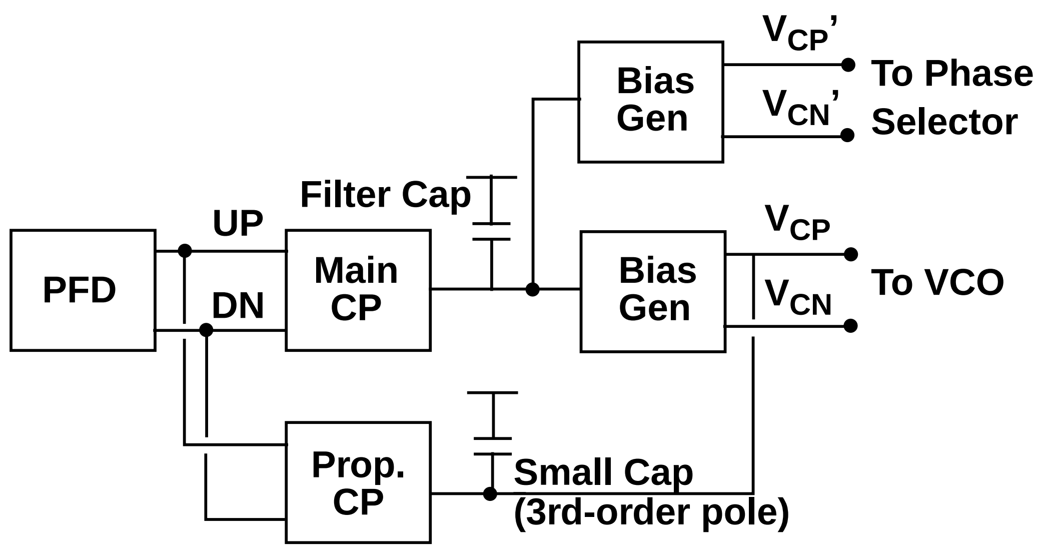


Figure 4.12: Bias Generation for the Dual-Loop PLL

the phase selector in the outer loop. The PMOS load of the half-replica on the right of Figure 4.11 is used by the bias generator for the VCO to provide the resistor for the feedforward zero in the inner PLL loop. This is the reason why the V_{CP} for this bias generator is connected to the proportional charge pump. The resistor is needed only for the inner loop, not the outer loop. Therefore, the bias generator for the phase selector in the outer loop is not connected to the proportional charge pump and hence there is no problem with ripples on its V_{CP}'.

4.1.5 Differential-to-Single-Ended Converter and its Duty-Cycle Corrector

The Asymmetric Serial Link transfers data on both edges of the clock. Hence, any duty-cycle error on the clock will result in loss of link timing margins. In the dual-loop PLL, a duty-cycle error can come from two main sources. First, the ripples on the control line (V_{CP}) can cause a duty-cycle error in the differential small-swing clocks in the VCO. Second, any mismatch in the differential-to-single-ended converter, which converts low-swing differential signals into large-swing single-ended signals, will result in a duty-cycle error as well. As described before, one can minimize the control-line ripples by the delay-balanced self-biased charge pump circuit shown in Figure 4.4. For the duty-cycle error from the differential-to-single-ended converter, a duty-cycle corrector is needed to minimize the error.

Figure 4.13 shows the differential-to-single-ended converter [51] and the duty-cycle corrector. The duty-cycle corrector is modified from the design by Sidiropoulos [34][91]. The differential-to-single-ended converter comprises two stages. The first stage has two NMOS differential-pair amplifiers with PMOS current mirror load and biased by V_{CN}. Each differential-pair amplifier has an extra NMOS differential pair to adjust the input offset of the amplifier. The offset voltage is generated by the charge pump circuit that effectively integrates the clock waveform. The second stage of the differential-to-single-ended converter is a PMOS common-source amplifier, which amplifies the signal to full-swing. Mismatch in the transistors of this PMOS common-source amplifier will cause unbalanced rise and fall delay and is the main source of duty-cycle error. To correct the

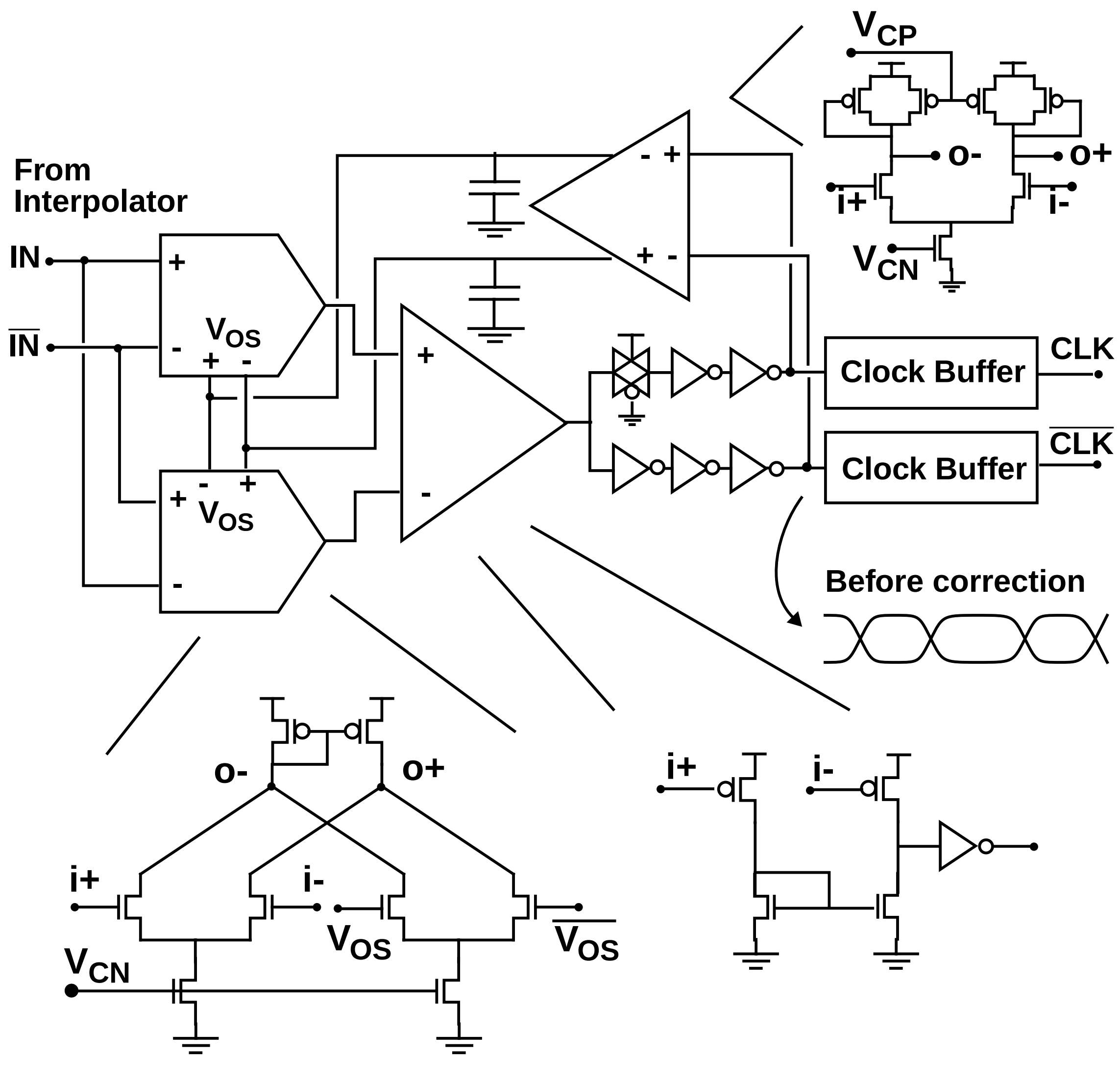


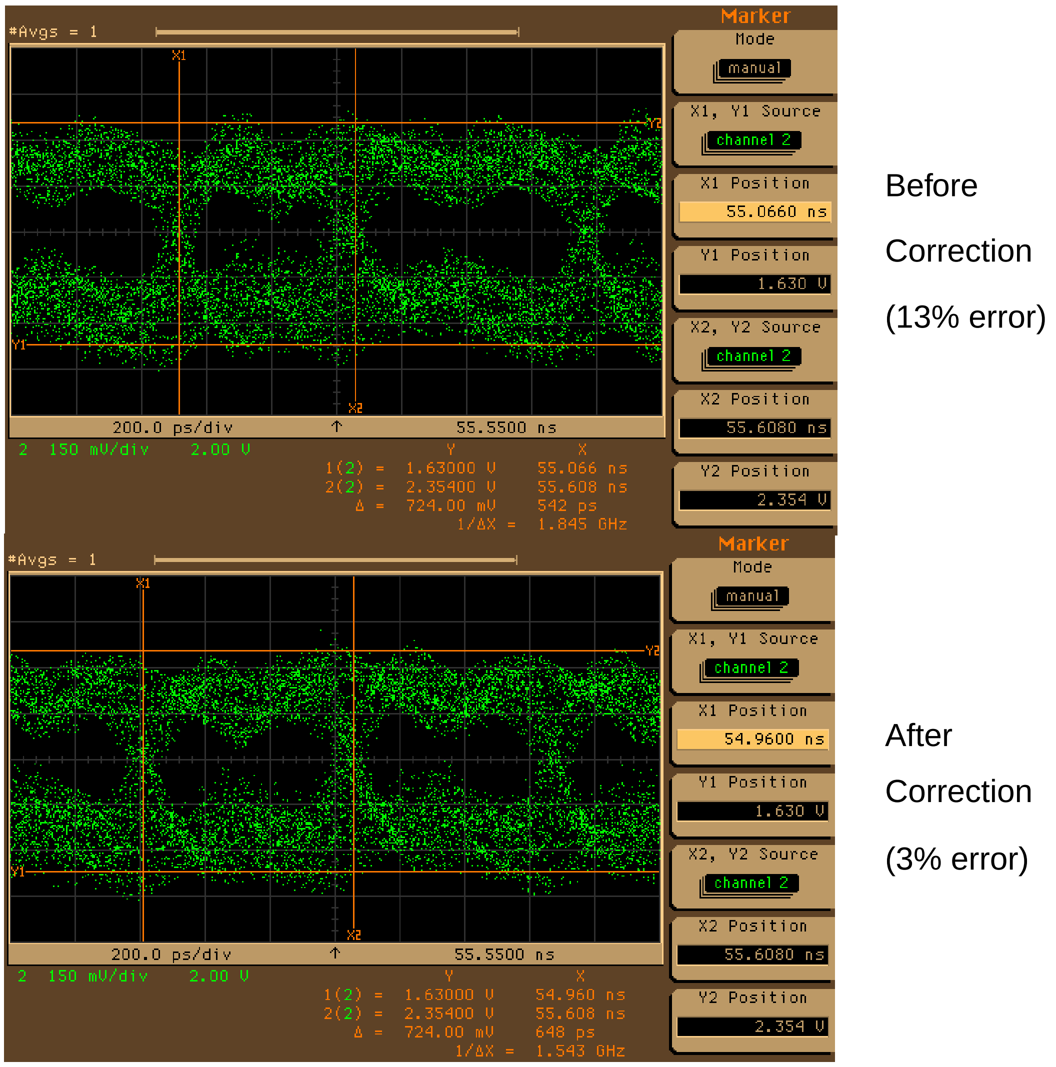
Figure 4.13: Differential-to-Single-Ended Converter, Duty-Cycle Corrector, and Clock Buffers

duty-cycle error, a delay-balanced 3-2 inverter block is used to convert the single-ended full-swing output to both true and complement clocks that are complementary to each other in waveform, as shown in Figure 4.13. The pass gate in the 3-2 inverter block slows down the 2-inverter path to match the 3-inverter path. A circuit implemented by a similar delay element for the VCO filters the complementary waveform and converts the duty-cycle error of the waveform into offset voltages and stores them on the two capacitors after the charge pump. These offset voltages are fed back to the differential-pair amplifier in the first stage of the differential-to-single-ended converter. This feedback loop will adjust the offset voltage continuously to correct any duty-cycle error at the output of the 3-2 inverter block. The clock buffers after the 3-2 inverter block consist of an even number of inverter stages to avoid the introduction of any duty-cycle error. Since the clock buffers are usually very large, the variations on the transistors are very small and should not affect the rise and fall delay too much. This is a crucial fact to note for clocks that have a large fanout clock tree, such as those of the crossbar design described in Chapter 5.

Figure 4.14 shows the effect of the duty-cycle correction. With the duty-cycle corrector off, the signal waveform shows 13% duty-cycle error. With the corrector on, the error is reduced to 3%. This indicates the robustness of the duty-cycle corrector.

4.1.6 PLL Startup

The self-biased charge pump circuit widens the operating range of the PLL but it also makes the inner loop very tricky to start up. Right after boot-up, the control voltage on the filter capacitor will start at V_{DD} because V_{CN} is originally at the ground voltage due to the self-biased nature of the replica-feedback biasing circuit (see Figure 4.11). If this is the case, the opamp in Figure 4.11 will push V_{CN} back to ground, thus preventing the PLL from starting up. To solve this problem, one needs to preset the control voltage of the filter capacitor to a voltage slightly lower than that of the other feedback input of the opamp in the replica-feedback biasing generator. In this way, the opamp will gradually increase V_{CN} and hence will draw some current into the charge pump circuit, which in turn will drive the control voltage on the filter capacitor down to further increase V_{CN}. From then on the charge pump circuit can pump some charge into the filter capacitor based on the up and down signal from the phase frequency detector. Therefore, the PLL loop starts working and gradually gets the loop into lock.



```
#Avgs = 1
200.0 ps/div        55.5500 ns
2  150 mV/div   2.00 V
                 Y              X
        1(2) = 1.63000 V    55.066 ns
        2(2) = 2.35400 V    55.608 ns
           Δ = 724.00 mV    542 ps
                  1/ΔX =    1.845 GHz

Marker
Mode: manual
X1, Y1 Source: channel 2
X1 Position: 55.0660 ns
Y1 Position: 1.630 V
X2, Y2 Source: channel 2
X2 Position: 55.6080 ns
Y2 Position: 2.354 V
```

```
#Avgs = 1
200.0 ps/div        55.5500 ns
2  150 mV/div   2.00 V
                 Y              X
        1(2) = 1.63000 V    54.960 ns
        2(2) = 2.35400 V    55.608 ns
           Δ = 724.00 mV    648 ps
                  1/ΔX =    1.543 GHz

Marker
Mode: manual
X1, Y1 Source: channel 2
X1 Position: 54.9600 ns
Y1 Position: 1.630 V
X2, Y2 Source: channel 2
X2 Position: 55.6080 ns
Y2 Position: 2.354 V
```

Figure 4.14: Experimental Results with and without Duty-Cycle Correction

4.2 Current-Integrating Receivers

For any high-speed interfaces on an integrated CMOS chip, the receiver must achieve a very low bit error rate on a low swing, high bandwidth signal in the presence of supply noise, reflection, and crosstalk. The Asymmetric Serial Link uses current-integrating receivers to filter out any effect from high-frequency supply noise at the receiver front end. The current-integrating receivers are especially robust for the single-ended links because they can filter out any high-frequency noise on the reference voltage.

The current-integrating receivers are largely based on the original design by Sidiropoulos [34][91]. The front-end integrator was modified to use an NMOS differential pair as opposed to the PMOS used in the original design. This allows the integrator to accept a signal level close to V_{DD}. This section will first review the basic operation of the integrating receiver and then present the circuit used in the link.

4.2.1 Basic Operation

The current-integrating receiver filters high-frequency noise by integrating the data for an entire bit time and determining whether data is one or zero based on the integration result. Therefore, any instantaneous spike of noise is unlikely to affect the result because of the

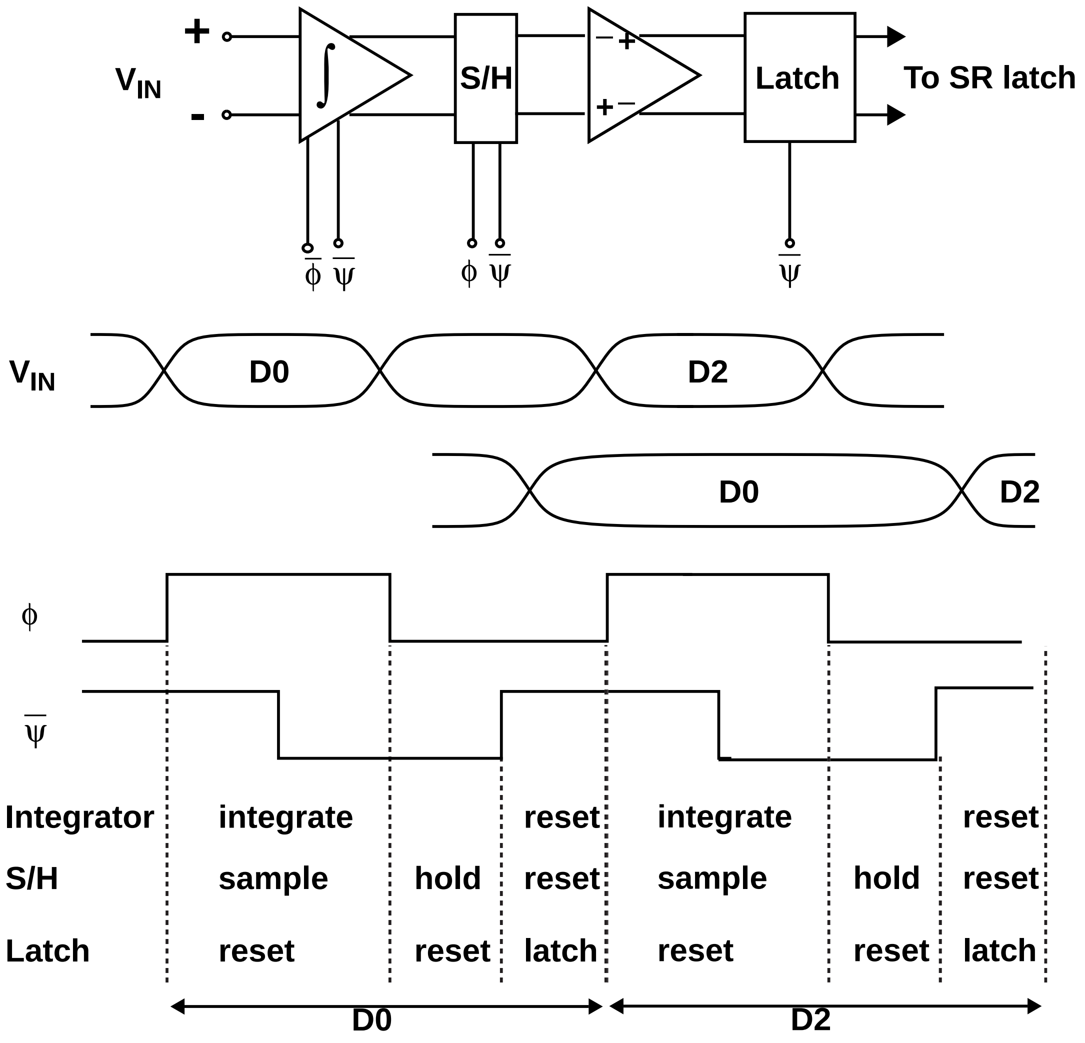


Figure 4.15: Block Diagram and Timing of the Current-Integrating Receiver (Excerpt from [91])

averaging effect from the integration. Figure 4.15 shows the block diagram of the current-integrating receiver. The input data stream is phase aligned with the clock, unlike in the conventional case, where the clock and data are offset by half a bit time. The front-end integrator integrates the input signal (differential or single-ended) for one bit time. During this time (ϕ high), both the amplifier and the latch are reset. The sample-and-hold circuit samples the integrating result at the end of the bit time and holds the result for a short period of time for the amplifier to amplify the sampled result. The length of the short period is determined by the ψ falling edge, which is two FO3 inverter delays after ϕ falls. At this time, the latch is triggered to pull the differential low-swing signals to full-swing while the front-end integrator and the sample-and-hold circuit are reset. The SR latch at the end then latches the full-swing signal for a full cycle (two bit times). This completes the data reception of the current-integrating receiver. Each Asymmetric Serial Link has two receivers: one clocked by ϕ and the other by $\bar{\phi}$ so that data are received on both edges of the clock.

4.2.2 Circuit Design

Figure 4.16 shows the circuit of the front-end integrator and the sample-and-hold stage. M1 and M2 form the differential pair that steers current from one side to the other based

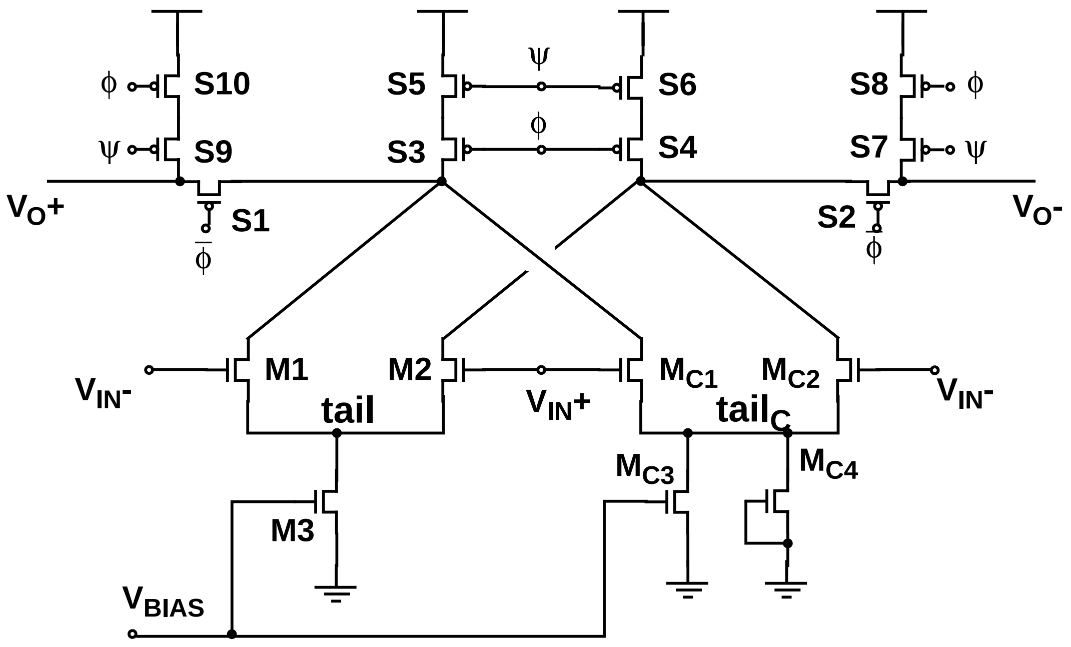


Figure 4.16: The front-end of the Current-Integrating Receiver

on the differential pair's input polarity. The integrating capacitor is formed by the parasitic drain capacitance of M1, M2 and of any other transistors connected to these nodes. M_{C1}, M_{C2}, M_{C3}, and M_{C4} are transistors that compensate for the voltage offset at the integrator output. The voltage offset is due to the input-to-output coupling through the gate overlap capacitance of M1 and M2, and due to voltage variations at the tail node of the differential pair. Note that M_{C1} and M_{C2}, which have the same size as M1 and M2, are connected opposite from how M1 and M2 are connected so that any injected charge from M1 and M2 will be pulled out by M_{C1} and M_{C2} and vice versa. In the design, M_{C3} is four times smaller than M3, while M_{C4} is three times larger than M_{C3} so that the tail node capacitances of tail and $tail_C$ are equal. With these relationships in size, the trade-off between the integrator's output swing and the offset voltage is optimized. Note that the equal amount of capacitance on tail and $tail_C$ cancels the tail charge injection error. This is crucial for single-ended signalling because otherwise the setup-and-hold window of this front-end integrator will be increased.

S1 to S10 form the sample-and-hold network. The sample-and-hold circuit uses stack transistors to qualify ϕ and ψ clocks so they generate a delayed pulse that resets the integrator. The use of the stack transistors enables a higher rate for the qualified clock so that the circuit is less limited by the speed of the ϕ and ψ clocks.

Figure 4.17 shows the characteristic of the integrator output relative to the skew between the data and the clock for different supply voltages and temperatures in the slow process corner[1]. The simulation data sequence includes various types of signal transitions (0101, 0011, etc) to stress the effect of the intersymbol interference in the front-end integrator and the sample-and-hold network. The spread of the zero crossing points of the waveforms in the figure represents the uncertainty in the setup-and-hold window of the receiver. The figure shows that the variation is smaller than 40ps for a bit rate of 2 Gb/s (500ps bit time), i.e., there is 8% uncertainty.

1. For brevity, only the slow corner is shown. The same simulation was done in the typical and fast corners which have smaller variations on the zero crossing points than the slow process corner has.

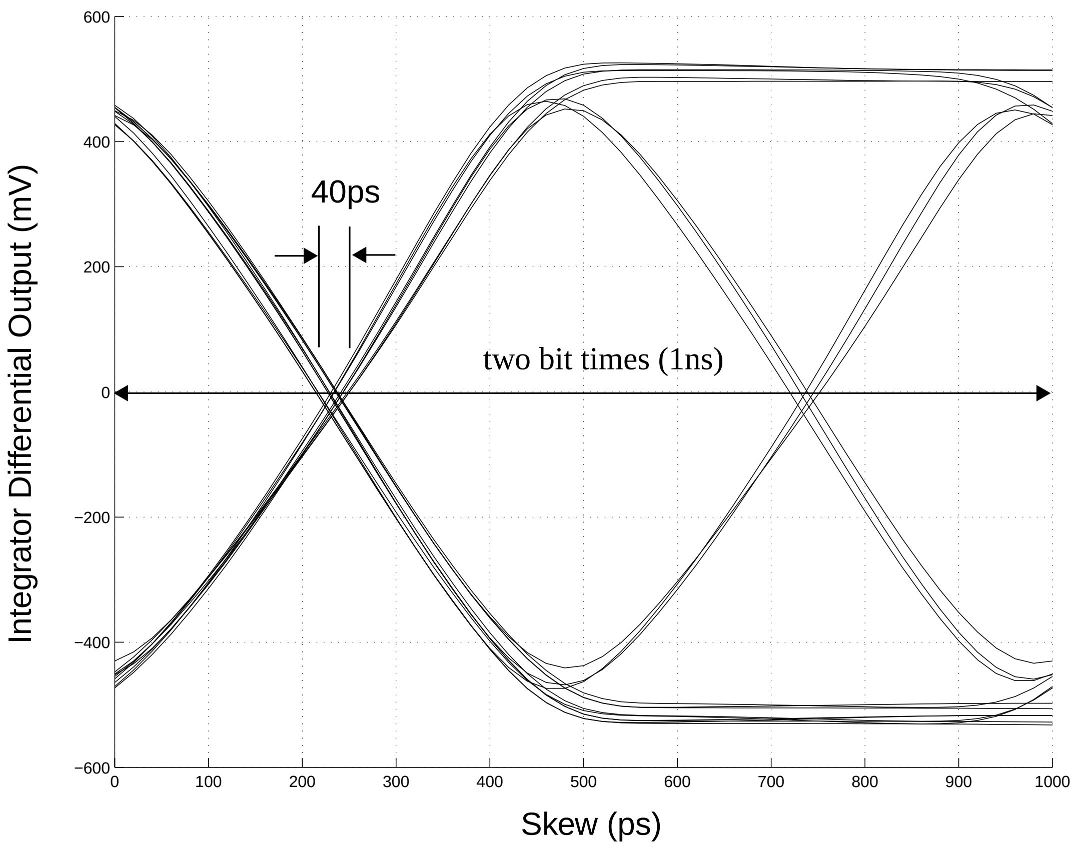


Figure 4.17: Phase Characteristic of the Front-End Integrator (SS Corner, 2.25V - 2.75V, 0C - 100C)

Figure 4.18 shows the amplifier and latch that convert the signal from small swing at the sample-and-hold output to full swing for use in digital logic. The primary function of the amplifier is to work as a buffer between the sample-and-hold network and the regenerative latch to isolate the kickback effect from the latch. In addition, the low offset and modest gain of the amplifier further improves the effective phase characteristic for the regenerative latch by steepening the slope of the curve shown in Figure 4.17. The amplifier utilizes a cross-coupled PMOS load to provide high differential and low common-mode output impedance [49][90][91]. The diode-connected PMOS load devices, which are in pair with the cross-coupled load, limit the swing of the amplifier so that it can have fast reset time and small kick-back to the sample-and-hold network. The reset switch further decreases the reset time of the amplifier. The regenerative latch is based on the design by Yukawa [77]. The output of the latch is fed to a NAND-gate-based SR latch to hold the data for a full cycle.

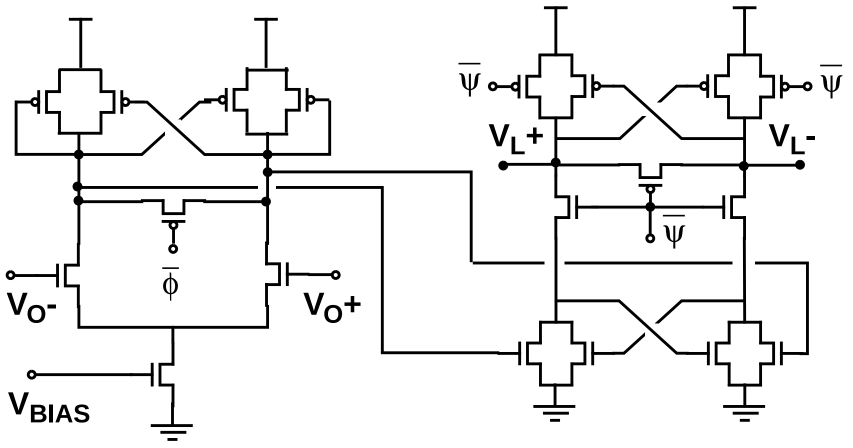


Figure 4.18: The Amplifier and Latch of the Current-Integrating Receiver

4.3 Current-Mode, Differential-Pair Transmitters

The Asymmetric Serial Link uses a straightforward current-mode, differential-pair transmitter, as shown in Figure 4.19. The true and complement data for both edges are clocked by the transmitter clocks (TCLK and $\overline{\text{TCLK}}$) before driving the high-skewed inverters. The high-skewed inverters ensure that the two inputs of the differential pair have a high-crossing point so that both NMOS transistors in the differential pair will not be turned off simultaneously and hence the tail node of the current source is kept from dropping. This ensures that the current source stays in the saturation region, which, in turn, keeps the transmitter in the current mode, thus isolating the effect of supply and ground noise of the transmitter chip from the receiver chip.

Another advantage of the differential-pair driver is that it does not cause any on-chip dI/dt noise because it sinks constant current. When the differential pair transmitter is used for sending single-ended signals, $\overline{\text{Txd}}$ can be connected to a V_{DD}. To prevent excessive noise on this V_{DD}, one can either connect it to a dedicated external pin directly or share it with other transmitters and then connect it to an external pin.

There is one more subtle but crucial design detail in the transmitter. As explained in Section 3.2.2 and Section 3.2.4, the transmitter delay has to match the clock-to-output delay of the divide-by-four circuit of the PLL in Figure 3.10. The transmitter delay is the

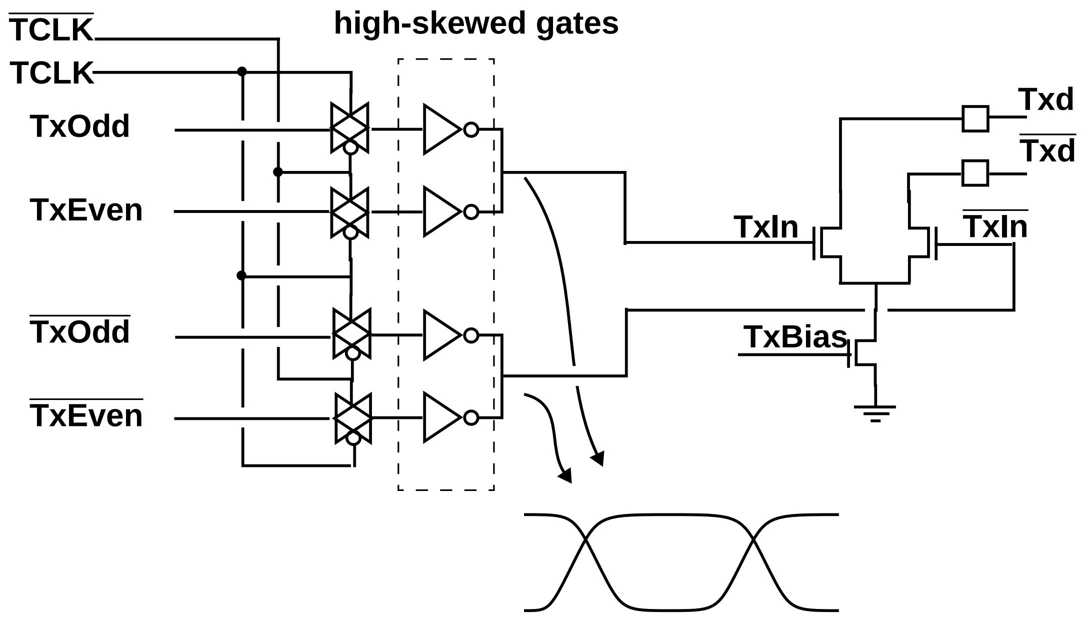


Figure 4.19: The Open-Drain Differential-Pair Transmitter and its 2x Multiplexors

delay from the transmitter clock (TCLK and $\overline{\text{TCLK}}$) to the transmitter's output (Txd and $\overline{\text{Txd}}$). This is very close to the circuit structure of the clock-to-output path of the divide-by-four circuit. The Asymmetric Serial Link uses pass-gate-transistor type flip-flops so the clock-to-output delay is from the clock driving a pass gate followed by an inverter, whereas the transmitter delay is also from the transmitter clock driving a pass gate followed by an inverter and the differential pair. The delay of the differential pair is very fast so can be considered to be negligible. Therefore, the transmitter delay can be very close to the clock-to-output delay of the divide-by-four circuit.

4.4 Serializer and Deserializer (Byte-to-Serial and Serial-to-Byte Converters)

In the applications of the Asymmetric Serial Link, such as the Tiny Tera, the core logic of any two chips that communicate using the serial links is operated synchronously with the *byte clock*, which runs at one eighth of the bit rate and is synchronized across the entire system. The synchronous byte clocks provide the serial link blocks with information about the position of the clock boundary between the half-bit-rate clock domain and the byte clock domain. Both the transmitter clock and the receiver clock are half-bit-rate clocks.

The interfaces of the core logic to the serial links are the byte-to-serial converter (serializer) at the transmitter end and the serial-to-byte converter (deserializer) at the receiver end. These two blocks absorb the skew between the half-bit-rate clock domain and the byte-clock domain.

Figure 4.20 shows the byte-to-serial converter and the transmitter. With the data transmitted at twice the rate of the transmitter clock, two serial streams are generated. First, each byte is separated into two groups, one with the odd bits and one with the even bits. Each byte is first stored in an 8-bit register clocked by ByteClk (the byte clock). It is then parallel loaded into two 4-bit shift registers enabled by TxLoad and clocked by TClk. These two shift-registers shift out bits into a latch and a flip-flop to align the odd bits for transmitting at the rising edges and the even bits for transmitting at the falling edges.

TxLoad controls the time to load and serialize a byte from the byte clock domain into the transmitter clock domain. Therefore, TxLoad is used to set the byte boundary in the serial

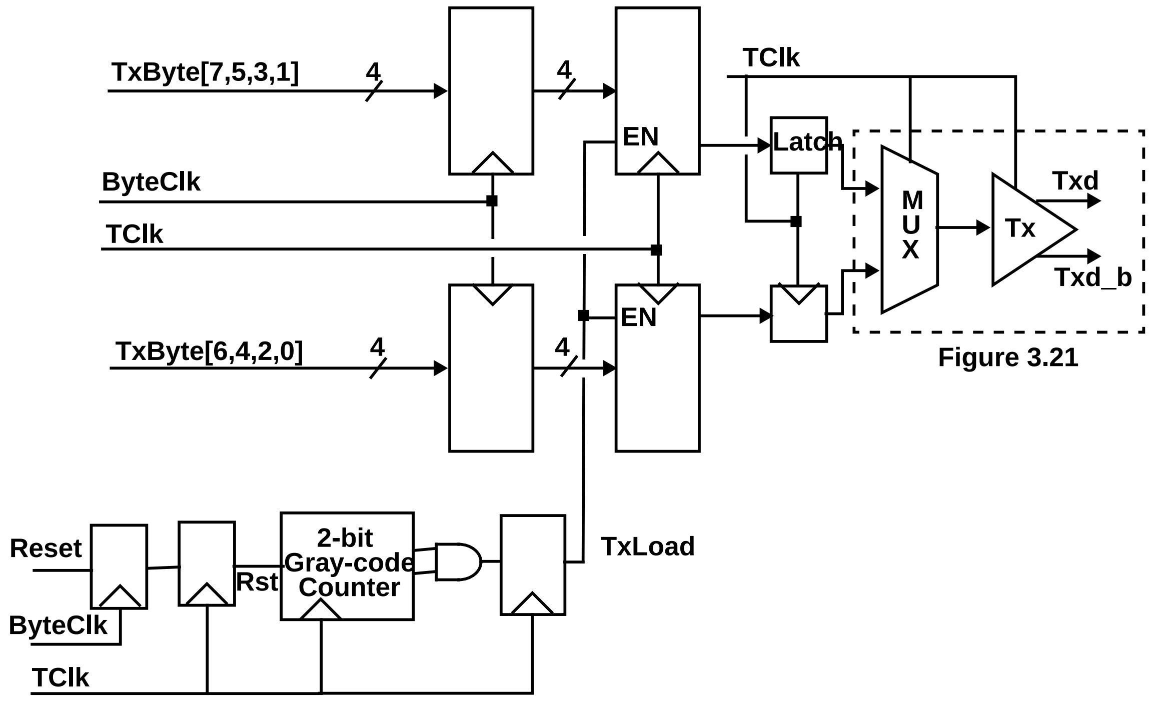


Figure 4.20: The Byte-to-Serial Converter and the Transmitter

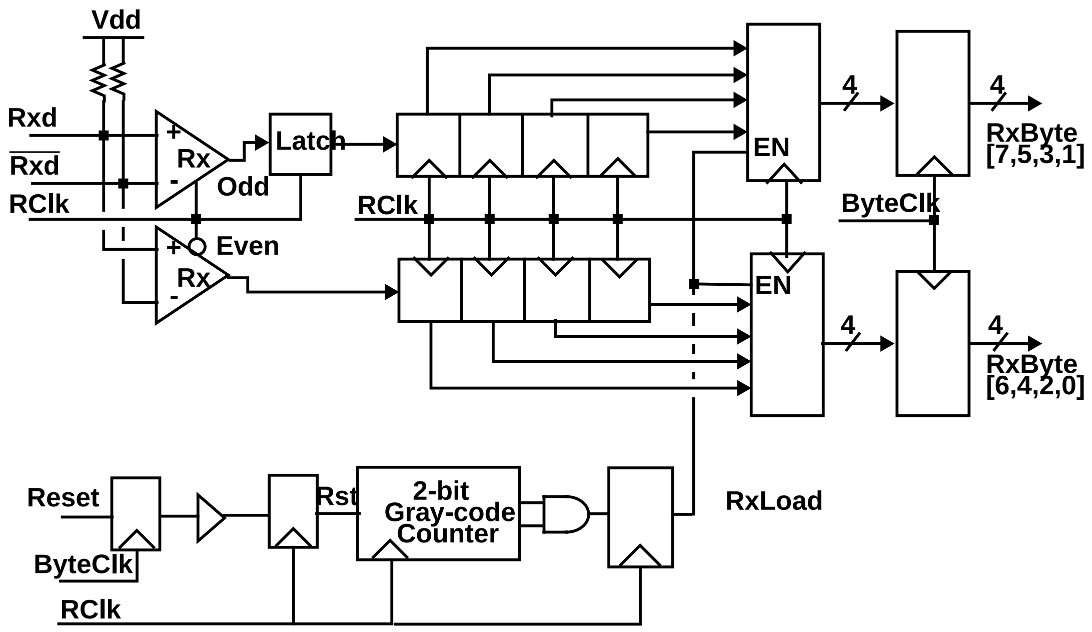


Figure 4.21: The Serial-to-Byte Converter and the Receivers

bit stream and thus is only asserted once every four TClk cycles. To generate TxLoad, a 2-bit resettable Gray-code counter clocked by TClk is used. The outputs of the counter are then decoded and registered to generate TxLoad. A Gray-code counter changes only one bit at a time, which minimizes logic complexity, so it can be implemented with only a single 2-input gate between its flip-flops and therefore is able to run at the half-bit-rate clock.

Figure 4.21 shows the serial-to-byte converter and the receivers. The data flow is essentially the reverse of that of the byte-to-serial converter and the transmitter. The differential high-speed signals are received by two integrating receivers, clocked at different edges of RClk. The odd bits, which are valid at the receiver output after the falling edge, are held by a latch until they are synchronous with the rising edge. The latch output and the even bits, which are both valid after the rising edge, are then sent to two 4-bit shift registers. Enabled by RxLoad, the outputs of these two shift registers are parallel loaded into two 4-bit registers each byte period. The 8-bit output of the two registers is then reclocked by ByteClk. The shift registers and the load counters are clocked by RClk.

Similar to TxLoad, RxLoad controls the time to parallelize an 8-bit data stream and load the data from the receiver clock domain into the byte clock domain. A 2-bit resettable

Gray-code counter synchronous with RClk is used while the counter outputs are decoded and registered to generate RxLoad.

Both TxLoad and RxLoad are synchronous with, and thus move with, the half-bit-rate transmitter and receiver clocks, respectively. For the Asymmetric Serial Link, the clocks at the dumb end are not moved and thus TxLoad and RxLoad are fixed at the dumb end. This means the byte boundaries at the dumb end are set by the reset position of TxLoad and RxLoad. Therefore, to synchronize the bytes at both ends, TxLoad and RxLoad at the smart end must be moved to compensate for the wire delay and for the byte clock skew between the transmitter and the receiver ends.

In the Asymmetric Serial Link, the load pulses at the dumb end are placed in such a way that the link can compensate for -2 to +6 bit time wire delay, where the negative delay is equivalent to the clock skew. For example, RxLoad at the dumb receiver is set to such a position that with zero wire delay, TxLoad at the smart transmitter is moved to serialize the bytes at the third quadrant of a byte clock cycle. Therefore, TxLoad moves toward the first quadrant with increased wire delay until it almost reaches the byte clock edge. On the other hand, TxLoad moves into the fourth quadrant with decreased wire delay (i.e., increased clock skew). As a result, the byte-to-serial and serial-to-byte converters can compensate for -2 to +6 bit time wire delay.

For both the byte-to-serial and serial-to-byte converters, the reset is first clocked by the byte clock and then reclocked by the half-bit-rate clock to reset the Gray-code counter. After reset, RClk is aligned to the byte clock edge. Therefore, the clock jitter of RClk and of the byte clock can affect the position of RxLoad by a one half-bit-rate clock cycle, introducing uncertainty. To solve this problem, a buffer is added after the byte clock flip-flop to increase the clock-to-output delay to approximately half of a bit time. TxLoad does not have this problem because TClk is preskewed by the transmitter delay to the byte clock, as explained in Section 3.2.2, which effectively increases the clock-to-output delay of the byte clock flip-flop.

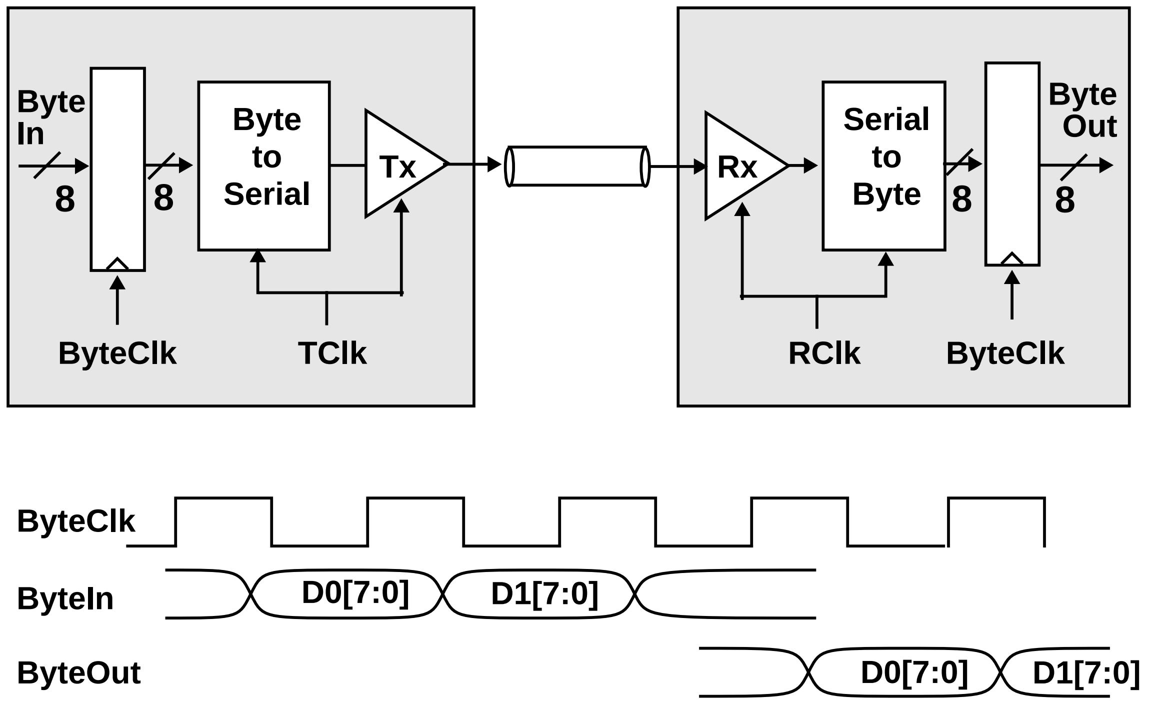


Figure 4.22: Equivalent Byte-to-Byte Datapath and Byte Timing for the Asymmetric Serial Link

In conclusion, the entire serial link, including the byte-to-serial and serial-to-byte converters, can be thought of as a byte-wide parallel link, as shown in Figure 4.22, with four byte time delays, when the wire delay is shorter than a byte time.

4.5 Summary

This chapter has provided an overview of the circuit blocks that are used in the Asymmetric Serial Link. As mentioned at the beginning of the chapter, the key circuit blocks for each serial link are the clock recovery circuit, the transmitter, the receiver, the serial-to-parallel converter, and the parallel-to-serial converter. The design details of each block have been presented and discussed.

The Asymmetric Serial Link uses a dual-loop PLL with digitally controlled phase selectors. The dual-loop property enables the link to provide stable clocks even with both noisy power supply and jittery data input. The effect of supply noise is overcome by making the inner loop high bandwidth, whereas the effect of the data jitter is dealt with by making the outer loop low bandwidth. The latter adjustment is controlled by digital phase-control logic, which controls the phase interpolators and hence has room for flexibility in adjusting the loop bandwidth.

To increase the robustness of the link within a noisy chip environment, differential circuits are used extensively for the PLL, the transmitter, and the receiver. The PLL uses differential buffers with symmetric loads and replica feedback biasing to decrease the PLL's supply sensitivity. The current-integrating receivers are used to filter the effect of high-frequency noise on signalling. The open-drain current-mode differential-pair transmitter is used to isolate the effect of supply noise in the transmitter chip from the receiver chip and to reduce the dI/dt noise.

To prevent the digital logic from limiting the link speed, the Asymmetric Serial Link parallelizes all the data into bytes before they are processed by the digital logic, which has byte-wide datapaths. The byte-to-serial converter converts byte-wide data that are synchronous with the on-chip byte clock into serial bits that are synchronous with the transmitter clock. The serial-to-byte converter converts the serial received data into parallel bytes and reclocks them with the on-chip byte clock. As a result, from the digital logic point of view, the serial link block presented in this thesis can be considered to be an 8-bit parallel interface with 4 byte-time latency. The serial link can absorb delay variations on the channel from -2 bit-time to +6 bit-time and can still maintain the same link delay.

Chapter 3 and this chapter together have presented the design of the Asymmetric Serial Link. The next chapter will present a high-bandwidth synchronous crossbar network switch designed for use in the Tiny Tera. The crossbar chip uses 32 asymmetric links and targets a total aggregate bandwidth of over 50Gb/s.

Chapter 5

Design of a 32 x 32 Crossbar Chip

Chapter 3 and Chapter 4 have presented the detail design of the Asymmetric Serial Link, which was intended for use in high-fanin crossbar chips. The link was designed to handle both the signalling and clocking issues for high-fanin chips. However, in reality, some of the noise is hard to quantify and emulate. For example, the crossbar chip in the Tiny Tera has a large digital crossbar core, which induces significant noise into both supply and substrate. In addition, the chip may have serious crosstalk effects at the package and board level. All these factors contribute to a noisy environment that may stress the design of the serial links on the high-fanin chips. Therefore, to demonstrate the robustness of the Asymmetric Serial Link within a practical environment, a 32×32 crossbar chip was implemented.

This chapter goes over the design of the crossbar chip in detail. Section 5.1 presents the architecture of the crossbar chip. Section 5.2 explains the data transmission protocol in the Tiny Tera, which affects the data flow in the switch core. Section 5.3 briefly reviews the synchronization of the asymmetric link in the crossbar chip, with focus on frame synchronization among links. Section 5.4 discusses the PLL and the delay-replica clock generation on the crossbar chip. Finally, Section 5.5 shows the digital byte-wide switch core, implemented by 32, 32-to-1 byte-wide multiplexors in static CMOS circuits.

5.1 Chip Architecture

Figure 5.1 shows the physical architecture of the 32×32 crossbar chip, which consists of 32 asymmetric links, a PLL, an 800MHz clock distribution tree (not shown), a 32×32 switch core, and a crossbar controller. The speed of the pin interface is 1.6 Gb/s[1]. The switch core and most of the digital logic on the chip runs at 200MHz. The speed conversion between these two clock domains is handled by the serial-to-byte and byte-to-serial converters inside the serial links, as described in Section 4.4.

1. The crossbar chip was designed for each interface running at 2Gb/s. However, the digital switch core can run only at 200MHz, a clock rate that can support only 1.6Gb/s at the interface. The measurement results show that each link alone can run at up to 2Gb/s.

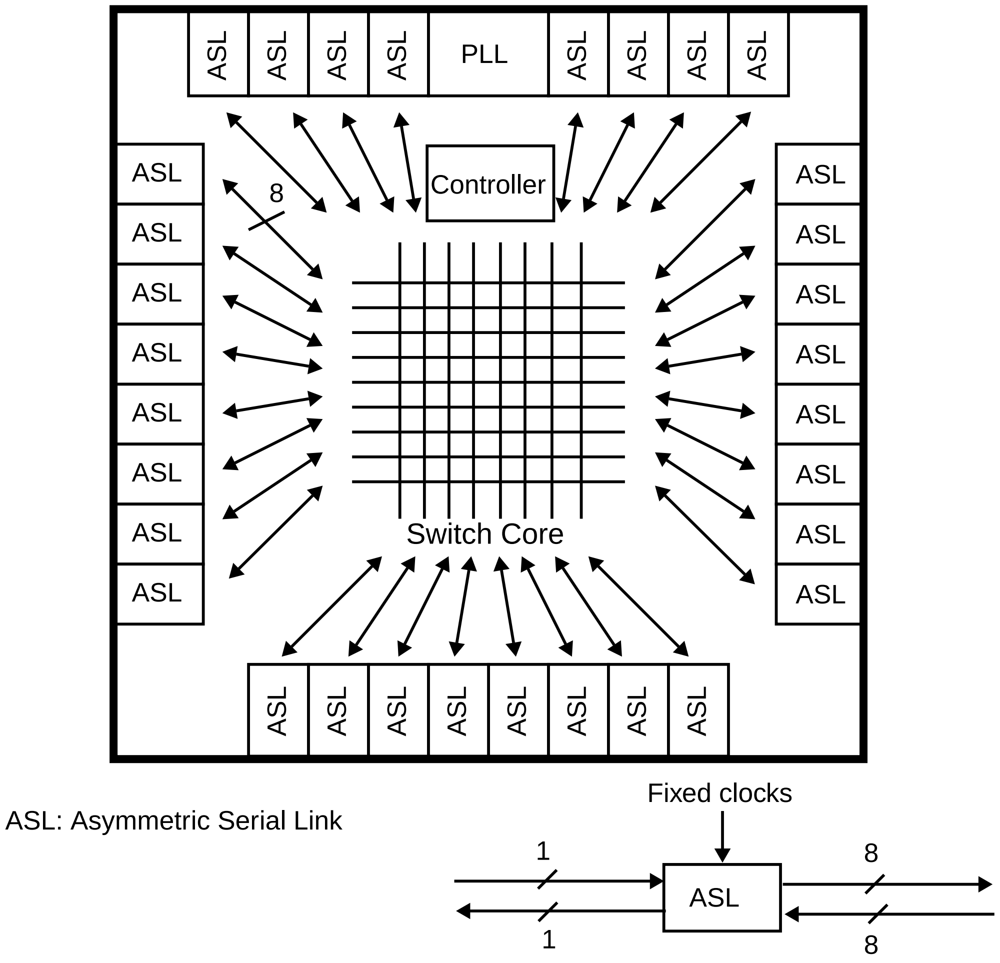


Figure 5.1: Chip Architecture of the 32 x 32 Crossbar Chip

A low-jitter 200MHz PECL (Pseudo-ECL) reference clock is driven onto the chip and into the PLL. The PLL serves two purposes. First, it multiplies the frequency of the clock by four to generate the 800MHz half-bit-rate clocks for all the transmitters and receivers. Second the PLL is also used to compensate for delay variations in the clock distribution due to low-frequency supply noise and thermal noise, as described in Section 5.4.

During normal operations, all 32 serial link blocks work as the dumb ends of the asymmetric links, which do not adjust their own transmitter and receiver clocks. For testing purposes, two serial link blocks on the chip were designed as smart ends with PLLs to adjust their clocks. These two smart ends may also be configured to operate as dumb ends.

The crossbar controller controls the frame synchronization among the 32 serial links. Before a link is properly synchronized, the controller disables the link so that it cannot be used by the crossbar chip. Any disconnected serial link will never be synchronized, so it will always be disabled by the controller without the need for any explicit logic.

The digital crossbar switch core handles the data switching of the crossbar chip. It is implemented by 32 32-to-1 byte-wide multiplexors. Its design is discussed in detail in Section 5.5. The data transmission protocol of the switch core is based on the design of the Tiny Tera and is discussed in the next section.

5.2 Data Transmission Protocol

The framing of all 32 ports is synchronized in the system. Hence, the switch core is a synchronous crossbar switch that is configured once for all 32 ports during every frame time. The scheduling of the crossbar chip is performed by a separate scheduler chip, which schedules all packets during every frame time and then sends the routing tags to the crossbar chip through all of the port chips interfacing with the crossbar. The routing tag is embedded in each data frame. Figure 5.2 shows the data format used by the crossbar chip. The data are grouped into nine-byte frames. The first byte is the header and the other eight bytes are the payload. In the header byte, the first bit is called the idle bit. It indicates whether the frame contains useful data. When the idle bit is set, the frame is used for calibrating the serial links. The scheduler periodically sets the idle bit to perform periodic calibration for the Asymmetric Serial Link. The next five bits in the header are the reverse

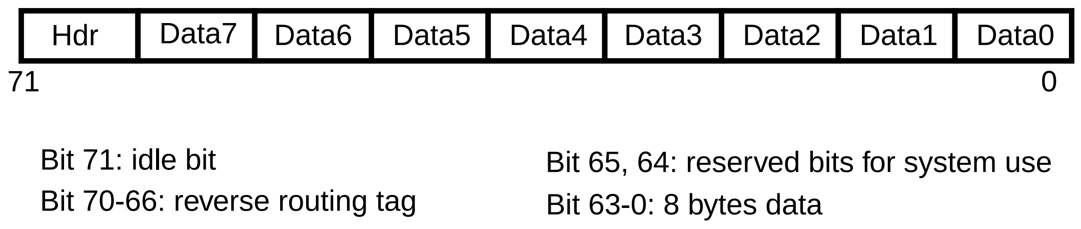


Figure 5.2: Data Format in the Crossbar Switch

routing tag, which indicates the port number of the input port (not the output port) that is going to send data to the port that carries the header in the same frame time [83].

5.2.1 Reverse Routing Tags

To understand the reason why reverse routing tags are used, one must know how conventional routing tags work first. Conventional routing tags carry destination of the current data packet. Figure 5.3 shows an example of the conventional routing tags with 5-bit binary coding for a 32×32 crossbar chip. In the example, the tag from input port 7 is 00011(3) so its data goes to output port 3. The binary coding can allow only one destination port for each tag to reduce the tag overhead. Therefore, the port 10 cannot have

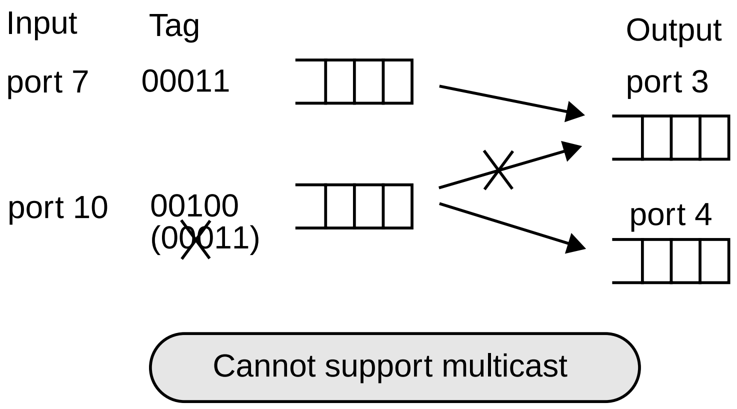


Figure 5.3: Conventional Routing Tags with binary coding

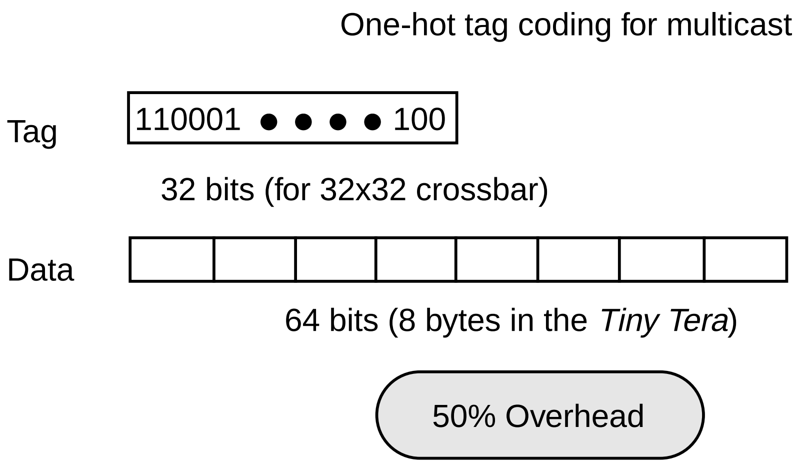

One-hot tag coding for multicast
Tag
110001 ● ● ● ● 100
32 bits (for 32x32 crossbar)
Data
64 bits (8 bytes in the *Tiny Tera*)
50% Overhead

Figure 5.4: Conventional Routing Tags with one-hot coding for multicast

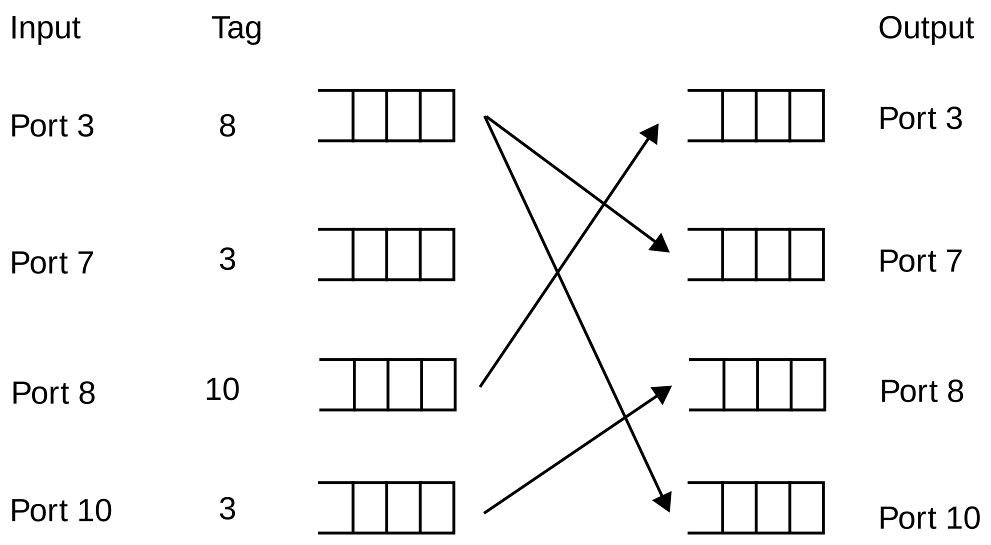


Figure 5.5: Reverse Routing Tag

port 3 as its tag, or there will be a data collision. This means that this scheme cannot support multicasting, since each input port can only specify one output port. To support multicasting, the conventional scheme needs to use one-hot tag coding with one bit for each output port (see Figure 5.4). For 32 ports, it needs 32 bits. If the tag comes with the data stream, which has 8 byte payload in the Tiny Tera, there is a 50% overhead incurred by transmitting the tag. To avoid this overhead, the crossbar chip needs explicit pins to send the routing tags. This increases the pin count of the system and hence is undesirable.

Reverse routing tags solve these problems by associating the tags with the ports rather than the data. The tags carry the source that will send data to the port that receives the tag. Figure 5.5 illustrates how reverse routing tags work. In the example, the tags of ports 3, 7, 8, and 10 are 8, 3, 10, and 3 respectively. Therefore, output ports 3, 7, 8, and 10 will receive data from input port 8, 3, 10, and 3. Both port 7 and 10 request data from port 3 so their tags are both 3. In this manner, one data packet can be multicast to many ports in the crossbar chip. Also, only one packet may be sent to each output port per frame time, so collisions cannot occur.

5.3 Synchronization

The synchronization of the 32×32 crossbar chip includes the bit, byte, and frame synchronization of each serial link alone and the frame synchronization among all the 32 links. The synchronization of each individual link has been discussed in Chapter 3. This section describes how to achieve the frame synchronization among all links.

To align the data frames from all the links, the frame counters of all the links are hardset to be the same on the crossbar chip. The frame counters on the port chips are adjusted by framing logic to synchronize with the counters on the crossbar chip[2]. In the asymmetric links, the dumb-to-smart link is first synchronized and hence the frame counter on the smart receiver is first adjusted to fit the data framing from the dumb transmitter. After the dumb-to-smart link is synchronized, the smart-to-dumb link starts its synchronization and adjusts its frame counter. The synchronization status in the control byte notifies each serial link whether or not the other link in the pair is synchronized. When the links of both directions are synchronized, the serial link pair is ready for use. The crossbar controller reads the ready signal from the synchronized port and enables the link of that port.

5.4 PLL and Clock Distribution

Figure 5.6 shows the clock generation on the crossbar chip. As explained in Section 3.2.2, the main goal of this clock generation circuit is to match all the clock paths so that the timing at all the transmitter and receiver pins are phase locked to a stable reference clock.

The PLL uses a 4-stage VCO with symmetric loads and replica feedback biasing to reduce its supply sensitivity [51][52]. The input reference clock is a 200 MHz PECL clock, which is amplified by a PECL-to-CMOS converter to full swing for the phase frequency detector. A delay matching circuit is inserted in the PLL feedback loop to compensate for any delay variation in the converter, which has 450ps delay in the typical process corner.

The 800MHz VCO output is multiplexed with the scan clock of the scan chain on the chip before driving the balanced clock distribution tree. Since these two clocks are multiplexed, the scan chains on the chip do not suffer from hold-time violations because the scan clock is distributed in a balanced fashion by the 800MHz clock distribution tree.

Each tail of the 800MHz clock tree drives local clock generators for the serial link blocks, the switch core, and the crossbar controller. The local clock drivers for the serial links generate byte clocks (200MHz), and transmitter and receiver clocks (800MHz). The delay

2. In fact, in the *Tiny Tera*, the first frame counter is unified at the scheduler chip. All the frame counters in the system are adjusted to that frame counter. This means the frame counter on the crossbar chip must be adjusted as well. This is a design detail of the *Tiny Tera* switch and therefore is not discussed in this thesis.

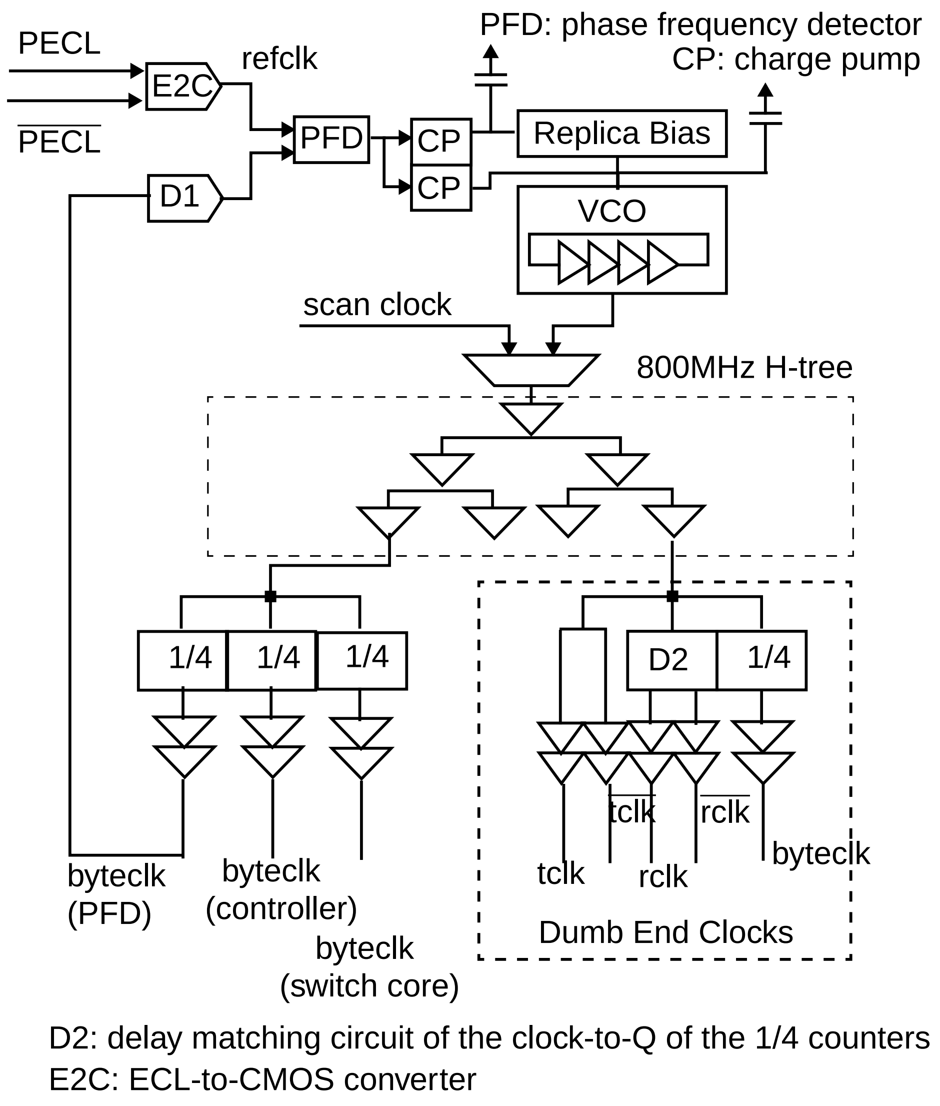


Figure 5.6: Clock Generation on the 32x32 Crossbar Chip

matching circuit in the receiver clock path matches the delay of the clock-to-Q of the divide-by-four counter and the transmitter output. The clock buffers for all three clocks are designed with approximately the same delay. Other local clock drivers, such as the ones for the switch core and the crossbar controller, generate only the byte clocks, which have the same delay in their clock buffers as the dumb ends. One of these byte clocks is fed back to the PLL. With this effort, the timing of transmitter output, receiver sampling time, and all the byte clocks are phase-locked to the external reference clock, achieving the goal stated at the beginning of the section.

Figure 5.7 shows the PECL-to-CMOS converter and the delay matching circuit shown in Figure 5.6. The converter includes a differential-to-single-ended converter, followed by two inverters. The first inverter is skewed in size to generate a 50% duty-cycle full-swing clock. The second inverter drives the signal to the input of the phase frequency detector. The delay matching circuit takes the feedback full-swing byte clock from the clock distribution tree to drive a pass gate and an inverter that generate complementary clock signals. These clock signals then drive a differential-to-single-ended converter that has the same design as in the PECL-to-CMOS converter. The inverter and pass gate are sized small, so the differential-to-single-ended converter is slowed and therefore the total delays of the two circuits are almost matched. Simulations showed that the maximum mismatch is 44ps across 400mV variations on the input common mode with the same supply voltage.

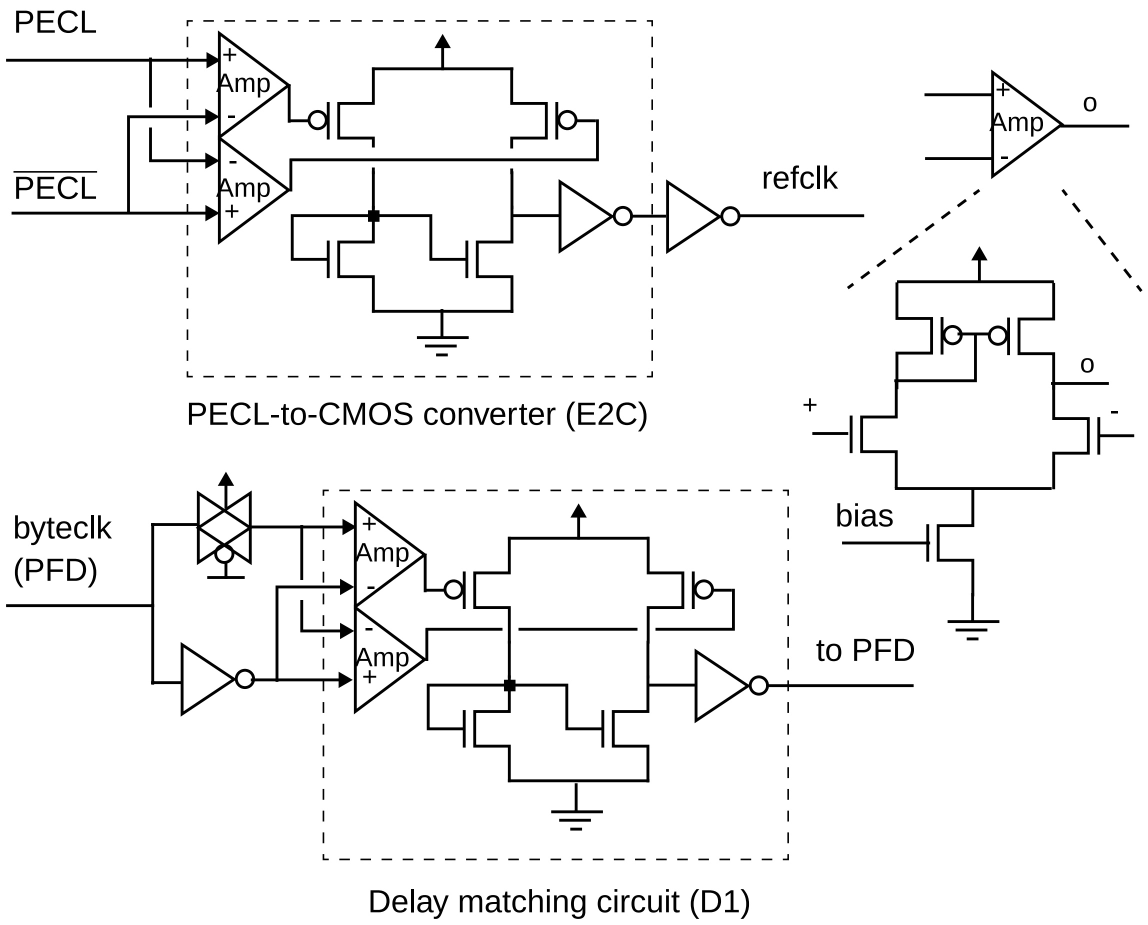


Figure 5.7: PECL-to-CMOS Converter and its Delay Matching Circuit

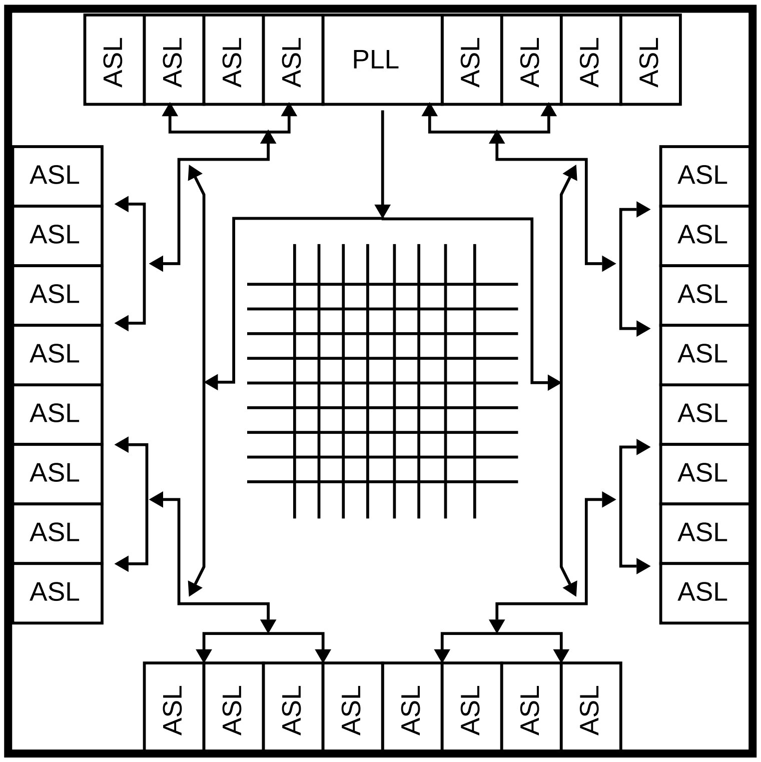


Figure 5.8: 800MHz H-Tree Clock Distribution

Figure 5.8 illustrates the physical implementation of the 800MHz clock tree, which provides clocks to all the 32 serial link blocks on the chip periphery. As will be explained in Section 5.5, the switch core uses static CMOS circuitry and is only clocked at its boundary, so no clock is needed near the center of the chip. To keep the layout of the switch core intact, the clock distribution tree is routed around it. As a result, an H-tree topology, as shown in the figure, is used to ensure that the arrival times of the clocks to each tail point of the clock tree are approximately the same. The clock skew of the 800MHz clock tree affects the timing margin for synchronizing the divide-by-four counters, which are synchronized in a daisy-chain fashion by the nearest divide-by-four counter. The worst-case timing margin comes from the edge of each clock tree. To satisfy the setup time of the signal that synchronizes the two divide-by-4 counters at the adjacent tails, the clock skew must be smaller than a bit time (625 ps), i.e., half of the half-bit-rate cycle. This can be easily achieved by the H-tree topology.

5.5 Switch Core

The switch core, which runs at the byte clock rate, is implemented with 32 static CMOS 32-to-1 byte-wide multiplexors. Figure 5.9 shows the datapath of the switch core. The 8-bit data are first registered at the output of each port before being sent to the switch core. Then the data are driven to the input lines and the tag decoder of the same port. If the data is a header byte, the tag decoder changes its setting and reconfigures the multiplexors. The data bytes are multiplexed and registered at the output of the switch core. The registers are physically located at the end of the output lines. As a result, the latency of the switch core is one byte-clock cycle.

With reverse routing tags, the tag from input *port i* configures the multiplexors of output port i. Since each input line goes to the multiplexors of all the output ports, when multiple

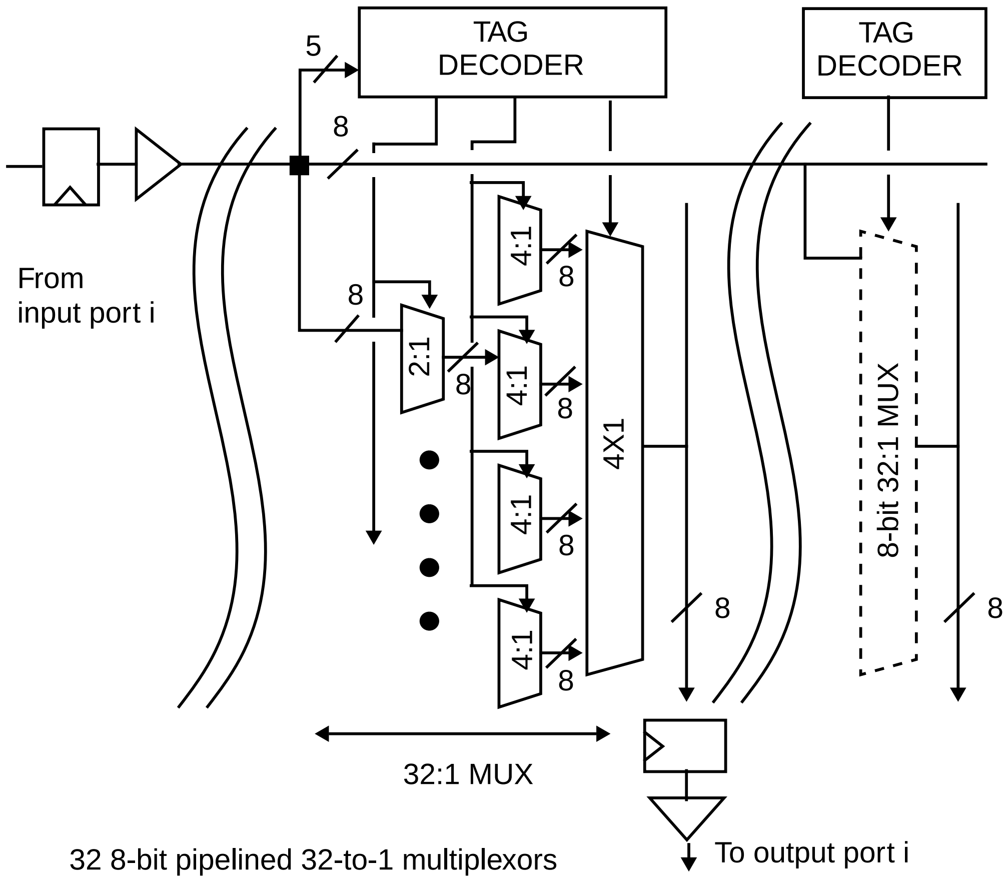


Figure 5.9: Switch Core Architecture and Reverse Routing Tag

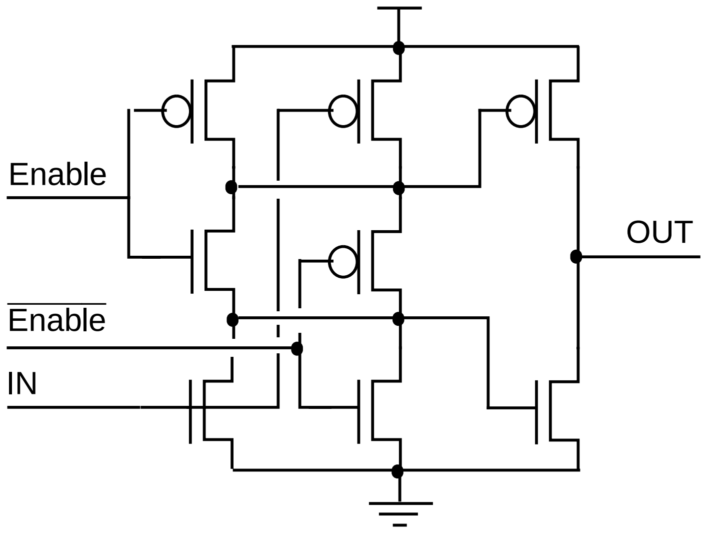


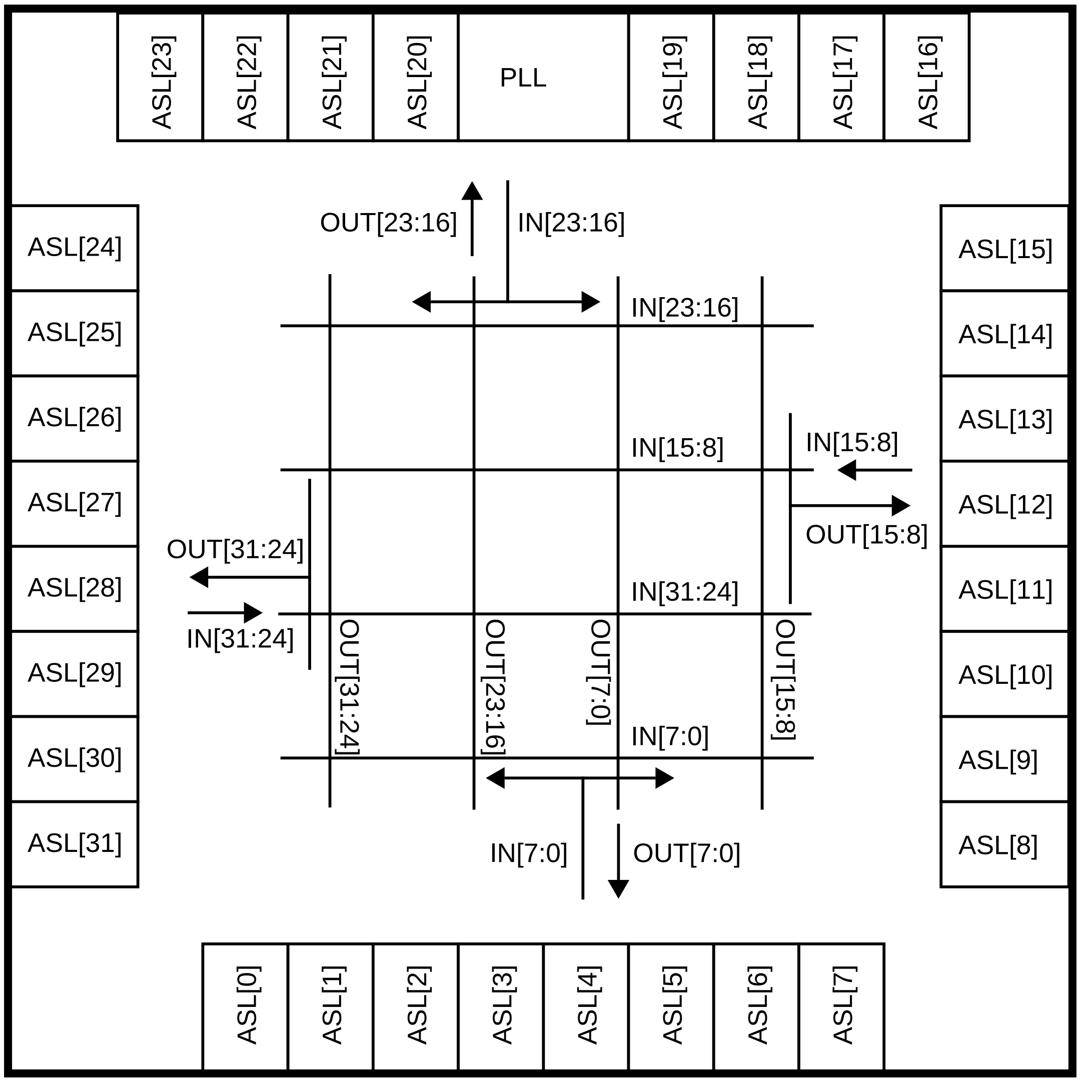
Figure 5.10: The Tri-State Buffer used in the Switch Core [95]

multiplexors for different output ports are turned on for the same input line, the data bytes are multicast to all of those ports.

Each 32-to-1 multiplexor is implemented by a series of three multiplexors: 2-to-1, 4-to-1, and 4-to-1. To minimize the wiring for both input and output, both sets of wires run straight through the entire switch core, with input wires running horizontally and output wires vertically, as also shown in Figure 5.9. The 2-to-1 multiplexors run between two adjacent 8-bit blocks to select between two input ports. The first four 4-to-1 multiplexors are placed across eight 8-bit blocks and a final 4-to-1 multiplexor is distributed across all 32 blocks in the vertical direction. Each of the output wires is 1.6mm long. The extreme length of these wires requires increasing the size of the multiplexors to reduce the large signal rise/fall time. However, larger multiplexors also have larger output loading. The problem is magnified because the “on” tri-state buffer has to drive a long wire and the output loading of all the “off” tri-state buffers. The multi-stage multiplexors relieve the output self-loading problem by reducing the total number of tri-state multiplexors on the same output wire to four.

To further minimize the output self-loading, the tri-state buffer shown in Figure 5.10 is used for the second and third stages of the multiplexors [95]. The advantage of this type of tri-state buffer is that its output transistors are not stacked. The true and complement enable signals are merged into its pre-driver.

The performance of the wire intensive switch core depends not only on the circuit design but also on its floorplan and the wiring to the peripheral serial link blocks. Figure 5.11 shows the global wiring and a high-level floorplan of the crossbar chip. The numbering of the serial link blocks starts with the lower left block on the bottom edge of the chip and increments counterclockwise. The center lines represent the wiring of the switch core. As explained before, to minimize the wire loading, both the (horizontal) input and (vertical) output wires run straight inside the switch core. Each center line represents 8 bits x 8 ports = 64 wires.



Each line represents 8 bits x 8 ports = 64 wires

Figure 5.11: Global Wiring of the Switch Core and Serial Links

While the wires within the switch core run in horizontal and vertical direction for the input and output wires, respectively, the wires from the serial link blocks can come from all directions, depending on where the blocks are located. For example, the input wires from ports 0 to 7 come vertically from the bottom edge and then make connections with the horizontal input wires in the switch core. To minimize the vertical length, the input wires for ports 0 to 7 are located at the bottom end of the switch core and the inputs for ports 16 to 23 are at the top, while the wires for the ports on the right and left sides are sandwiched in between. Layout of the output wires is similar, with the wires for the ports on the right side rightmost and the wires for the ports on the left side leftmost in the switch core.

There is one caveat regarding the design. In the switch core, the horizontal input and vertical output wires shown in Figure 5.9 (and Figure 5.11 as well) have very heavy capacitive loading due to the transistors and their own wire capacitance. This loading slows down the switch core and hence reduces the total crossbar bandwidth. In the layout, the wires are as long as 8mm. To reduce RC delay, the wire width should not be minimum width as specified by layout rules. However, in the real crossbar chip, due to a tapeout deadline, the author mistakenly used minimum width wires, which resulted in an additional 500ps delay and prevented the switch core from running at the target 250MHz rate.

5.6 Summary

This chapter discussed the design of a 32×32 crossbar chip containing 32 asymmetric links. The chip uses the data transmission protocols for the Tiny Tera and is a synchronous crossbar switch supporting multicasting. From the serial link standpoint, the chip provides a realistically noisy environment due to supply and substrate noise from the huge digital switch core and crosstalk at the package and board. This chip provides a vehicle for testing the robustness of Asymmetric Serial Links in a practical hostile environment. The next chapter will present experimental results obtained with the asymmetric link and this crossbar test chip.

Chapter 6

Experimental Results

We designed two test chips to demonstrate the feasibility and robustness of the Asymmetric Serial Links. In the summer of 1996, we implemented a one-channel test chip in Texas Instruments' 0.25μm CMOS technology. The one-channel chip can be configured as either a smart end or a dumb end. The chip was tested in the fall of 1997. The tests demonstrated that the asymmetric link worked, and the link ran at 2Gb/s with a bit-error rate of less than 10^{-14}. To further show the robustness of the link within a practical chip and system environment, we implemented a completely functional 32×32 crossbar chip in 1998 using Texas Instruments' similar 0.28μm CMOS technology. Tests on the crossbar chip in early 1999 showed that it works successfully with each link running at 1.6Gb/s while interfacing with the crossbar switch core.

6.1 Timing Margin Measurement

Traditionally, the timing margin of the links is measured using a tester to vary the timing between the clock and data channels and then to record the data output. From this information, designers can get a shmoo plot with one axis being the timing difference between the data and clock. The shmoo plot includes the effect of the receiver and clock buffers because the results are measured from the receiver output. This is crucial because the receiver and the clock buffer of the receiver also affect the timing margins, as explained in Section 3.2.3. However, this tester approach is applicable only for links with an explicit clock (called parallel links [28]-[34]) but not to serial links, which directly recover timing from data. For serial links, the timing margin is commonly observed only at the receiver input by means of the eye diagrams measured from oscilloscopes. However, this measurement cannot include the effect from the receiver and the clock buffer of the receiver. Therefore, a more accurate timing margin measurement from the receiver output for the serial links is needed.

Since the phase selection of the clock phase selectors is digitally controlled, this means that the phase settings for the data and timing sampling points can be controlled independently. In normal operation, their distance is half a bit time. However, designers can vary this distance arbitrarily for measurement purposes.

With this flexibility, one can measure the timing margin of the link at the receiver output accurately by keeping the phase setting for the timing sampling point unchanged while injecting phase errors to the phase setting for the data sampling point. The maximum phase errors that can be introduced before the link starts failing are the timing margins of the serial link. Interestingly, this measurement does not affect the timing recovery of the link. This allows designers to take measurements while the tracking loop of the link is continuously running.

Figure 6.1 illustrates this timing margin measurement. For this measurement, the link transmits PRBS (Pseudo-Random Bit Sequence) data and resets its PRBS verifier. The setting of the timing clocks (or phase detection clocks) is set by the phase-control loop so that the timing sampling edges are still right on the data transition edges. This keeps the tracking loop functional. The data clock moves in one direction by adding phase errors onto the phase setting of the data clock. This process continues until the PRBS testing fails. At this time, the link resets its PRBS and moves the data clock back to the original position, which should be the center of the data eye. Then the data clock moves in the other direction, again by adding phase errors until the PRBS testing fails again. Between the two failure points lies the timing margin of the serial link. A histogram of the failure points can be obtained by applying the measurements several times[1].

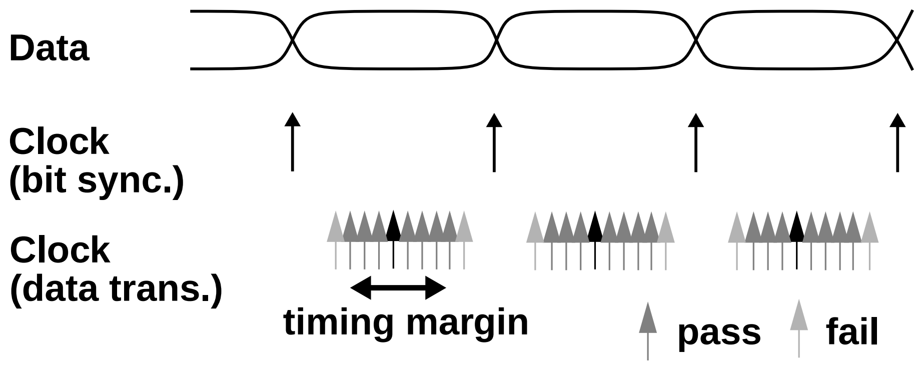


Figure 6.1: Phase Error Injection for the Timing Margin Measurement

1. Note that to measure the phase margins of the smart-to-dumb link, the transmitter data timing is moved because the phase errors are added to the transmitter clock.

6.2 One-Channel Test Chip for the Asymmetric Serial Link

We implemented the prototype one-channel test chip in a Texas Instruments 0.25µm CMOS process with five metal layers. The chip has an open-drain differential-pair transmitter, current-integrating receivers [34], a dual-loop PLL with digitally-controlled phase selectors [56], unsilicided polysilicon resistors (for on-chip termination resistors), and the serial-to-byte and byte-to-serial converters. The same chip can be configured to work as either a smart end or a dumb end. Figure 6.2 shows its die photo and its package. The chip is 3.9mm x 3.9mm in size and has about 50,000 transistors, packaged in a 160-pin CQFP package. There is a dedicated power supply pin for the PLL to isolate the supply noise from the digital logic and output drivers and a 32-bit parallel output bus for reading and writing the chip status. The wide bus can be also used to generate I/O switching noise on the chip for testing purposes.

The large synthesized digital block, running at the byte clock rate, includes the phase control logic for the smart end, a PRBS generator, a PRBS verifier, and some other logic for various testing functions. All the flip-flops running at the byte clock rate are scannable to increase the chip's testability. The PRBS used is a 7-bit sequence with X^7+X+1. The PRBS generator precalculates one byte of data at a time and sends the bytes to the byte-to-serial converter. The PRBS verifier takes the most recently received byte and uses it to generate the next PRBS byte using hardware identical to the PRBS generator. If the next PRBS byte and the next received byte do not match, this signifies PRBS failure.

In the one-channel test chip, we did not implement the scheme of using the same receiver for both data reception and phase detection. Thus this chip has two pairs of current-integrating receivers, one for data and the other for timing. Therefore, periodic calibration is only necessary on the smart-to-dumb link, whereas the dumb-to-smart link is continuously calibrated. This allows us to compare the performance of periodic calibration versus continuous calibration.

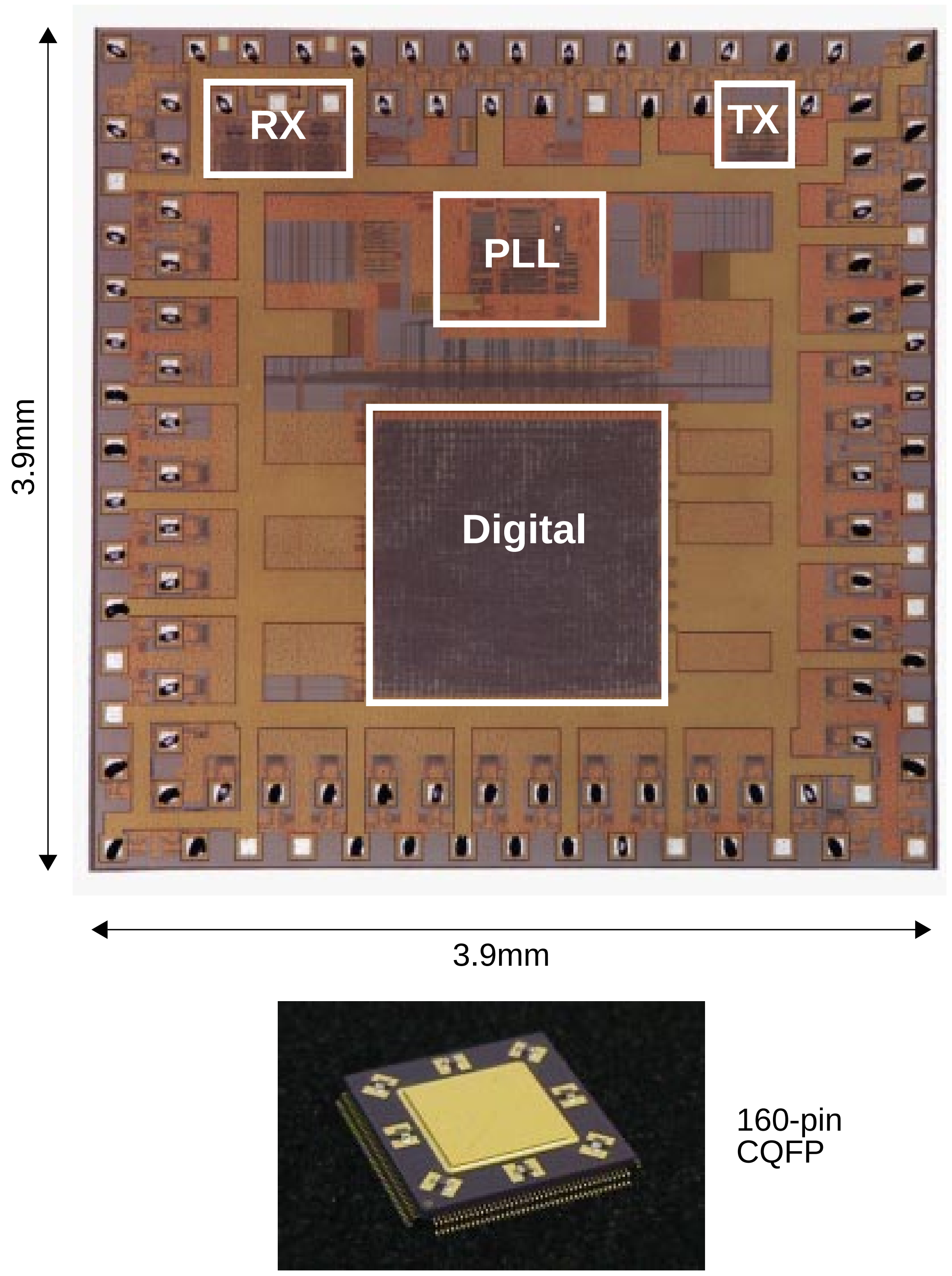


Figure 6.2: Die Photo and Package of the One-Channel Test Chip

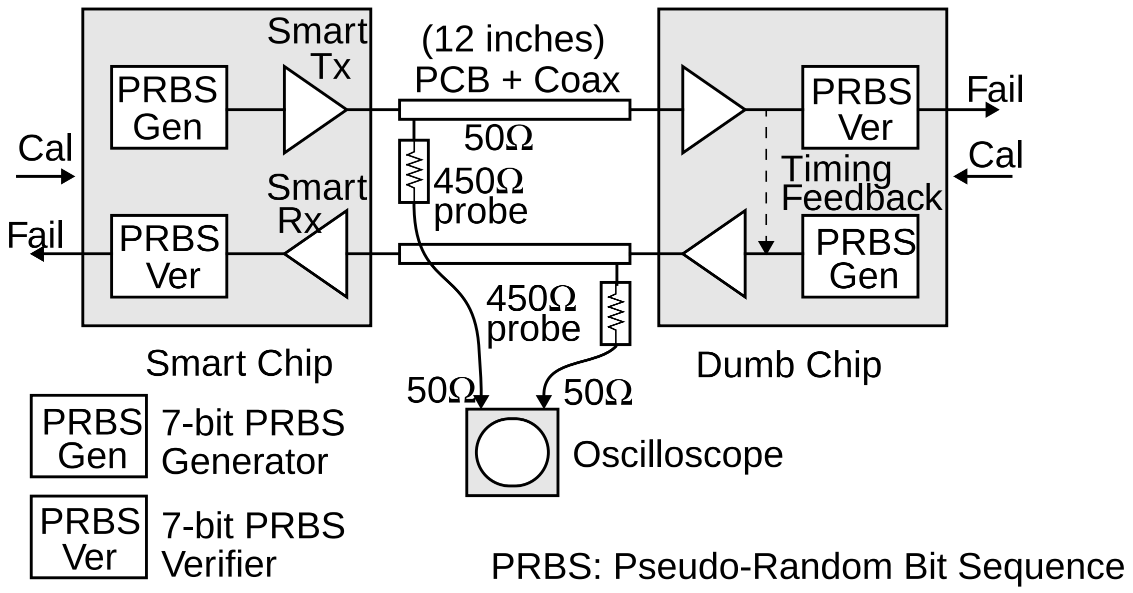


Figure 6.3: Test Setup for One-Channel Test Chip

Figure 6.3 shows the test setup of the one-channel Asymmetric Serial Link. One chip is configured as the smart end and the other as the dumb end. For each link, there is a 7-bit PRBS generator that generates PRBS data to the link. The PRBS verifier at the receiver verifies the data. In this setup, the testing of the two links is completely decoupled, so each link's performance should be independent of the other's. A PAL (Programmable Array Logic) generates the calibration signal at various frequencies, and the minimum required calibration rate can be learned from this. High-bandwidth 450Ω 10-to-1 coaxial probes are used to probe the signals on the link with an oscilloscope. The link must be probed this way for testing because both the smart-to-dumb and dumb-to-smart links have to be running for the phase-control loop to be functional. The wire impedance of the PCB and the coaxial cables is 50Ω.

Figure 6.4 shows a photo of the test boards. Each board has a one-channel test chip. The board on the lower right is for the dumb chip. The board on the upper left is for the smart chip. The 10-to-1 coaxial probes that probe signals on the dumb-to-smart link can be clearly seen near the center of the photo.

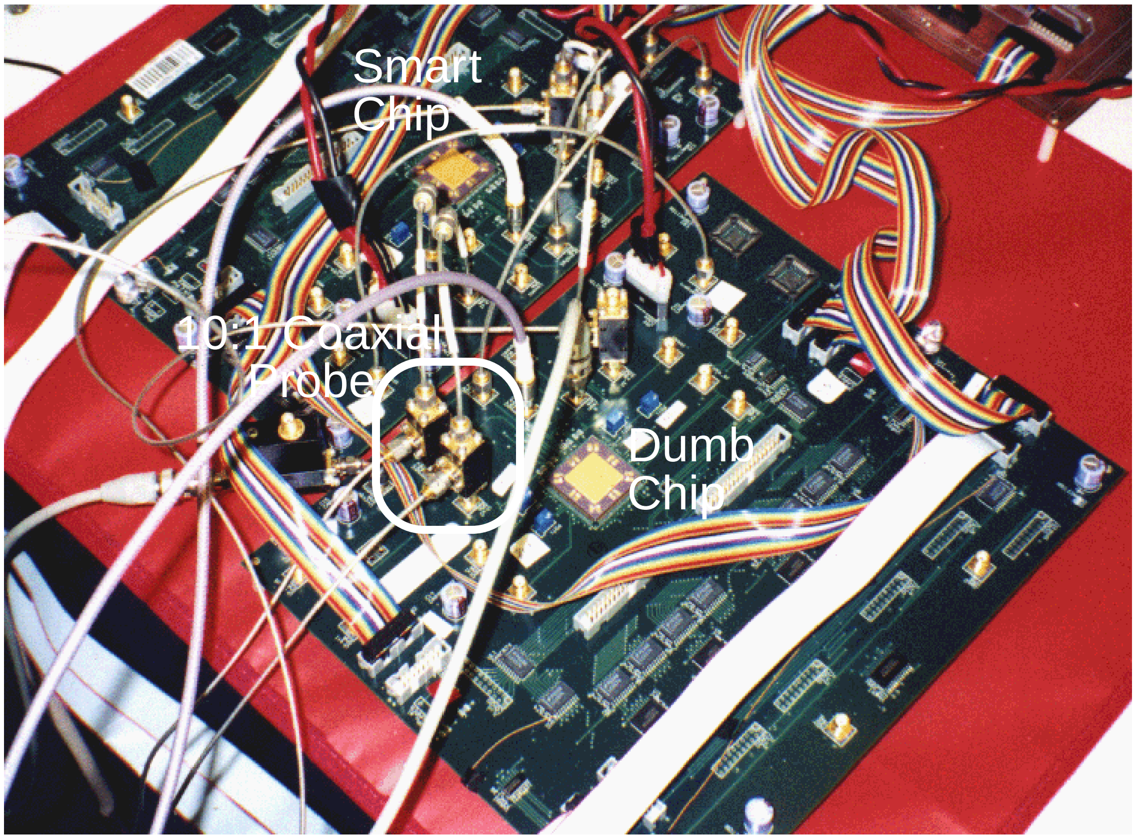


Figure 6.4: The Test Boards for One-Channel Test Chip

Figure 6.5 shows the eye diagram measured at the transmitter output. The top waveform is measured from the smart-to-dumb link, with periodic calibration. The bottom waveform is measured from the dumb-to-smart link with continuous calibration. The eye opening is about the same. Both links run at 2Gb/s with 650mV signal amplitude. This test indicates that the asymmetric link runs as well as the conventional links. This is expected because the tracking bandwidth of both links are determined by the PLL's loop bandwidth.

To compare the performance difference at different calibration rate, we used a PAL (programmable array logic) to vary the calibration rate from calibrating every 1000 data frames, every 10^6 data frames, to completely turning off calibration after the link is synchronized. The measurements showed that the link performed the same for all the three cases. In other words, the calibration rate does not affect the link's speed. For the extreme case where the calibration was completely turned off after the link acquired lock, the link stayed in lock even after we spreaded freezers or varied the voltage on the power supply.

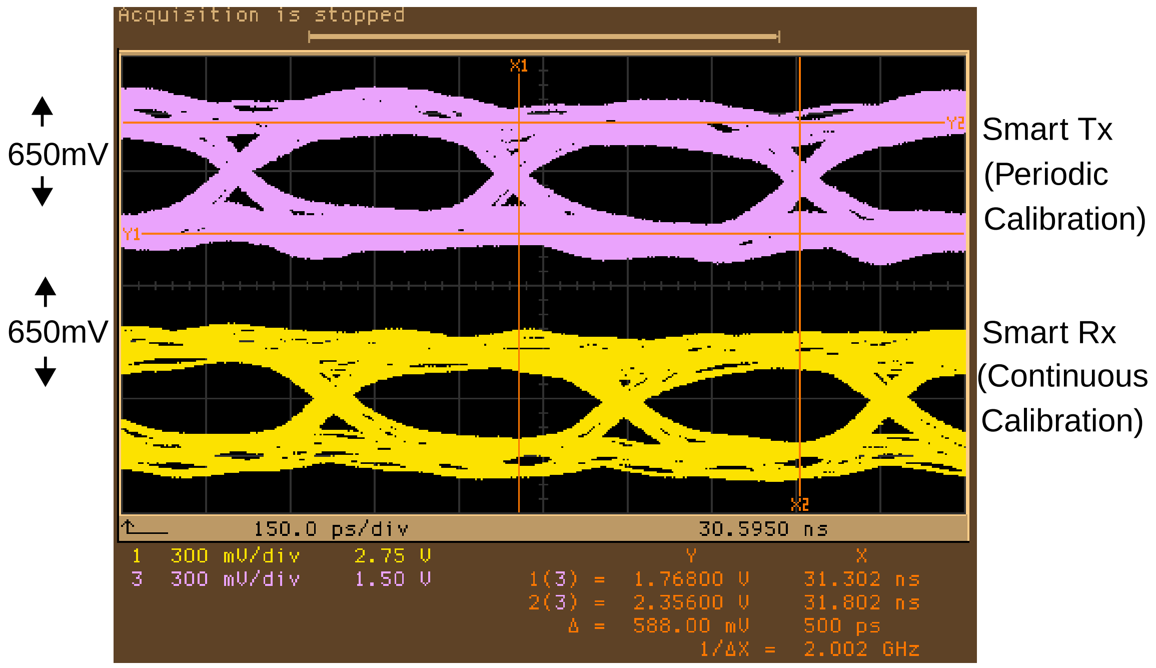


Figure 6.5: Eye Diagram of the Transmitter Output

This indicates that the PLL and delay-replica clock generation, described in Section 3.2.2, work very well in tracking any dynamic noise.

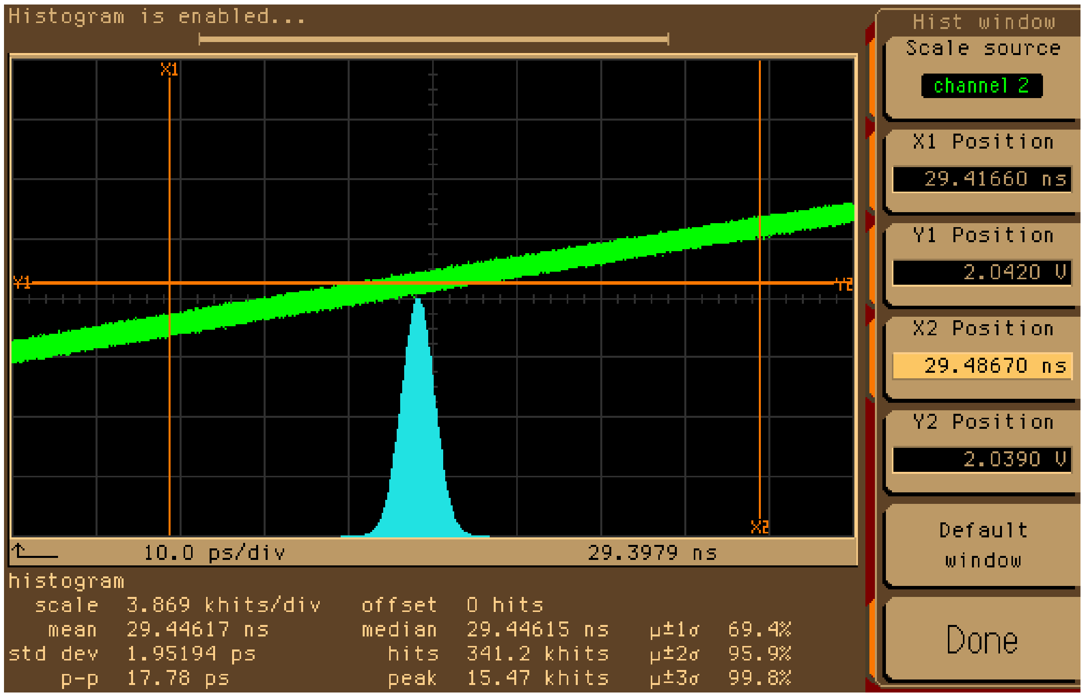
The eye diagram in Figure 6.5 shows a small amount of signal reflection. The reflection may result from two sources: the unsilicided polysilicon resistor and the package bond wires. The measured resistance for the unsilicided polysilicon resistor is 40Ω, which is 20% from 50Ω resistance. The bond wires in the CQFP are as long as 5mm due to the large cavity size of the package, which introduces about 5nH inductance before the receiver inputs. Both the non-50Ω on-chip termination resistor and the highly inductive bond wires introduce impedance discontinuity on the channel and thus cause signal reflection.

Table 6.1 summarizes the performance of the one-channel test chip. The link runs as fast as 2Gb/s for a bit-error rate of less than 10^{-14} with 150mV minimum signal amplitude for differential signalling and 650mV minimum signal amplitude for single-ended signalling. We measured the single-ended signalling by using an external reference voltage connected into the negative differential input, which suffered from significant packaging coupling. This is why the amplitude of the single-ended signalling needs to be much larger than that of the differential signalling. On the other hand, the fact that the single-ended link worked at approximately the same speed as the differential link indicates that the filtering effect from the current-integrating receivers works very well.

Supply Voltage	**2.5V**
Technology	**TI 0.25μm CMOS**
PLL Range (VCO rate)	**200MHz-1.44GHz**
Power Consumption - PLL	**175mW @2Gb/s**
Jitter (@1 GHz, quiet supply)	**18ps pk-pk/2ps RMS**
Supply Sensitivity (pk-pk)	**0.8ps/mV**
Max Data Rate	**2.2Gb/s**
Min Input Amplitude (differential signalling)	**150mV (10^{-14} BER)**
Min Input Amplitude (single-ended signalling)	**650mV (10^{-14} BER)**

Table 6.1: Performance of the One-Channel Test Chip

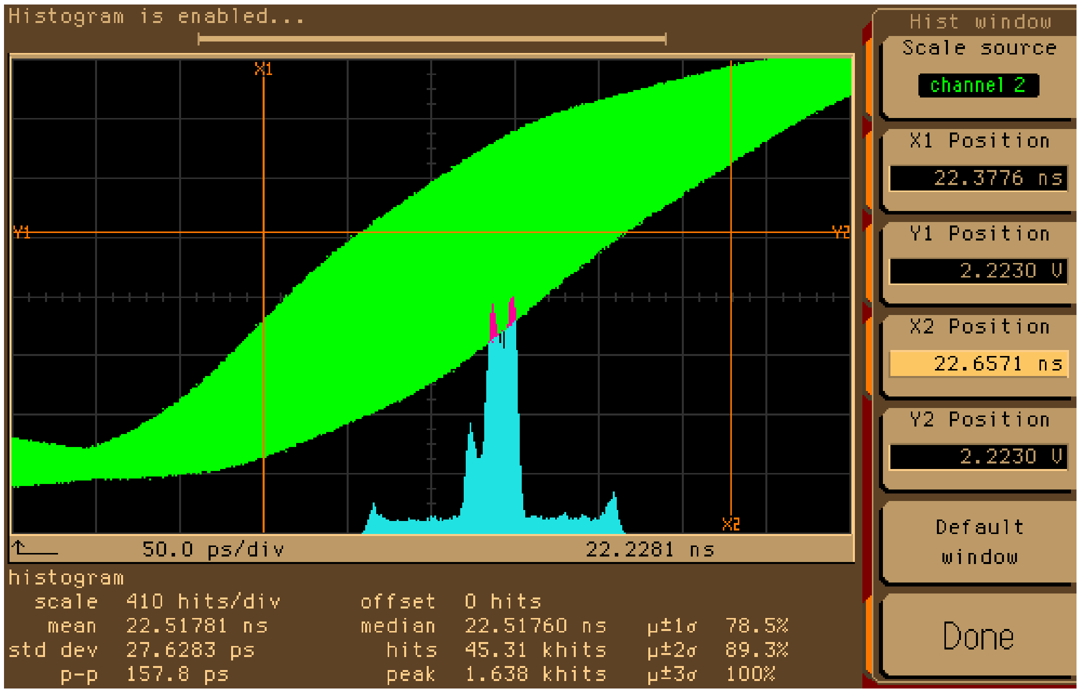
Figure 6.6 shows the clock jitter without supply noise on the one-channel test chip. We measured the jitter at the transmitter output by feeding a clock pattern to the transmitter. The measured waveform was triggered by the input reference clock of the PLL. The result shows that the PLL has very small jitter. However, in the real system, the PLL suffers from supply and substrate noise. To measure the supply sensitivity of the PLL, we applied a 10MHz, 200mV square wave noise to the PLL's power supply. Figure 6.7 shows the jitter histogram of this noisy clock. The measured supply sensitivity is 0.8ps/mV. This number is larger than 0.3ps/mV reported by Yang [26][92]. This is because the clock path, which includes clock multiplexors and interpolators, is longer in the Asymmetric Serial Link than in Yang's design.



```
Scale source   channel 2
X1 Position    29.41660 ns
Y1 Position    2.0420 V
X2 Position    29.48670 ns
Y2 Position    2.0390 V

10.0 ps/div                29.3979 ns
histogram
  scale  3.869 khits/div   offset  0 hits
   mean  29.44617 ns       median  29.44615 ns   μ±1σ  69.4%
std dev  1.95194 ps          hits  341.2 khits   μ±2σ  95.9%
    p-p  17.78 ps            peak  15.47 khits   μ±3σ  99.8%
```

Figure 6.6: Clock Jitter Histogram at 1GHz with Quiet Supply for the One-Channel Test Chip



```
Scale source   channel 2
X1 Position    22.3776 ns
Y1 Position    2.2230 V
X2 Position    22.6571 ns
Y2 Position    2.2230 V

50.0 ps/div                22.2281 ns
histogram
  scale  410 hits/div      offset  0 hits
   mean  22.51781 ns       median  22.51760 ns   μ±1σ  78.5%
std dev  27.6283 ps          hits  45.31 khits   μ±2σ  89.3%
    p-p  157.8 ps            peak  1.638 khits   μ±3σ  100%
```

Figure 6.7: Clock Jitter Histogram at 1GHz with 200mV Supply Noise for the One-Channel Test Chip

6.3 The 32 x 32 Crossbar Chip

The prototype 32×32 crossbar chip was implemented in a Texas Instruments 0.28μm CMOS process, which is similar to the 0.25μm CMOS process used for the one-channel test chip except that the channel is slightly longer to limit the leakage current. The wire capacitance and the transistor parasitics of the two processes are approximately the same. The simulated FO4 gate delay (140ps) is 14% slower than that (120ps) in the 0.25μm CMOS process[2]. Figure 6.8 shows the die photo and package of the chip. The chip is 8.5mm x 8.5mm in size and has nearly half a million transistors, 140,000 of which are in the digital switch core at the center of the chip. The die is wire-bonded to a 352-pin cavity-down PBGA with 50Ω package traces. The cavity-down package allows the chip to use a heat sink on top of the package to cool the chip. As shown in Figure 6.8, of the 32 Asymmetric Serial Links, two are the smart ends (SMART in the photo)), two are the dumb ends (DUMB), and 28 are the “true” dumb ends (T-DUMB). The smart ends and the dumb ends have test circuits for testing the serial links and the crossbar chip. These four blocks all have 7-bit PRBS generators and verifiers, which are the same as those in the one-channel test chip, for the bit-error-rate test of the serial link. To save silicon area, power, and pins, the “true” dumb ends do not have any testing circuits since they are duplicated many times on the chip. All 32 links use the same receiver for data reception and phase detection and thus use calibration packets for both the smart-to-dumb and dumb-to-smart links as described in Section 3.2.3.

One of the primary objectives of the crossbar chip is to test the robustness of the asymmetric links within a realistic noisy environment. The digital crossbar core, which is implemented with the large buffers and multiplexors as described in Section 5.5, provides an ideal noise source on the chip. Likewise, the 352-pin wire-bonded PBGA and the test board provides a practical package and system environment that induces crosstalk and signal reflection.

2. The numbers are for the slow process, low power supply (10%), and high temperature (125^{o}C).

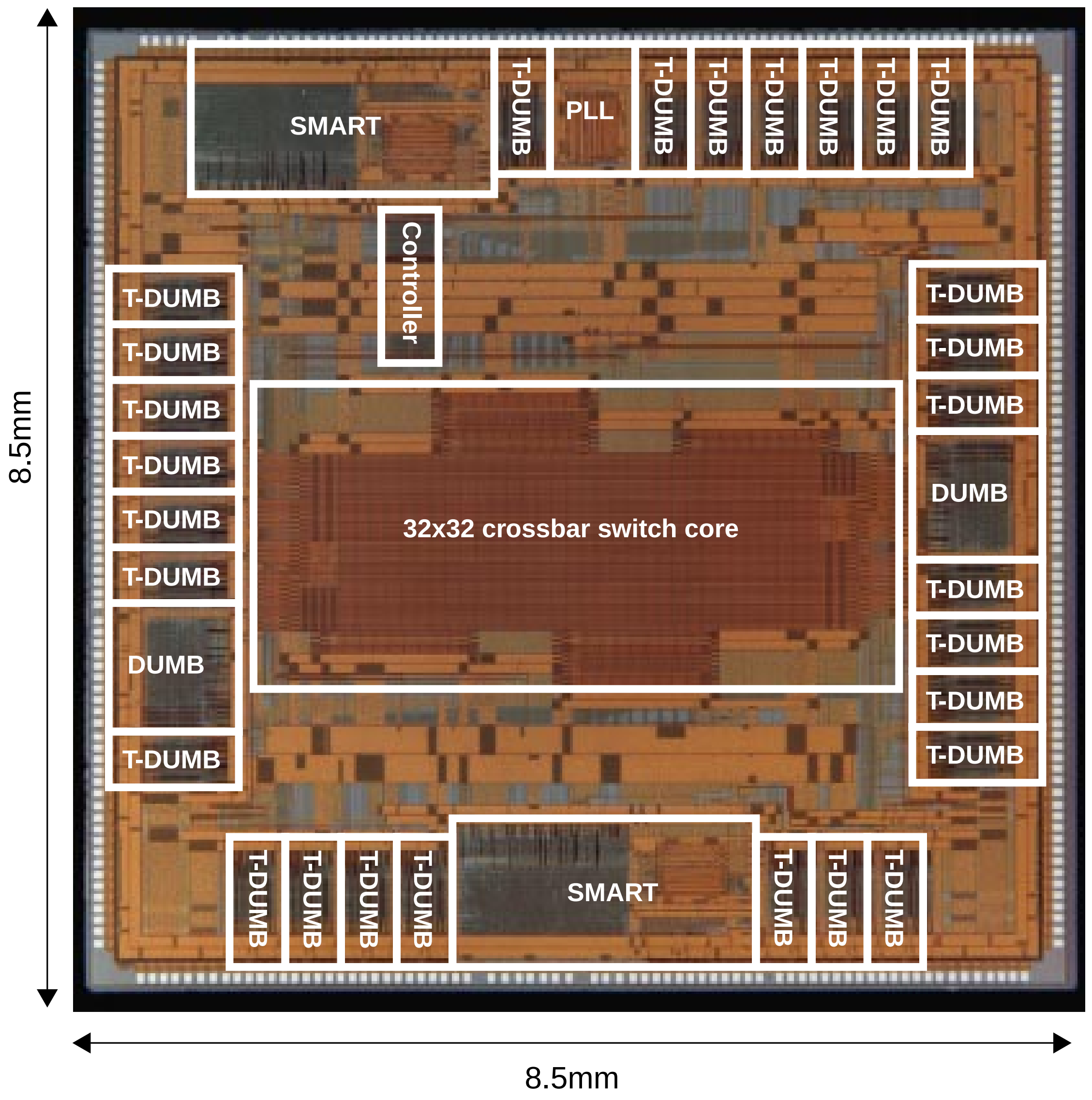


Figure 6.8: Die Photo and Package of the 32 x 32 crossbar chip

However, to deal with these different types of noise, not only did we design circuits that are highly noise-tolerant, but also we put effort into containing the magnitude of the noise so that the noise will not become arbitrarily large and beyond the noise rejection capability of the Asymmetric Serial Link. On the crossbar chip, a total of nearly 9nF of large bypass capacitors were added to minimize the supply variation due to the switching of the digital logic. To minimize the packaging crosstalk, all the bond wires and package traces for the high-speed links are isolated from each other by ground signals. To avoid signal reflection, instead of using unsilicided polysilicon resistors, which were not available for the process, we implemented the termination resistors with PMOS devices whose gates were tied to a bias line that was biased close to ground so that the PMOS devices were in the deep linear region and had their equivalent resistances close to 50Ω.

Figure 6.9 shows the test setup for the 32×32 crossbar chip. There are two types of tests. One is to test the bit-error rate of the asymmetric links alone (without the links interacting with the switch core). The other is to test the interaction between the serial links and the switch core to test the cell switching of the switch core at speed. In both cases, the switch core is actively switching to generate noise on the power supply and substrate.

Figure 6.10 shows a photo of the test boards. Two chips are used in the test setup shown in Figure 6.9. One chip is configured as the smart chip, which has two smart ends. The other is configured as the dumb chip, on which all the links are configured as dumb ends. Only

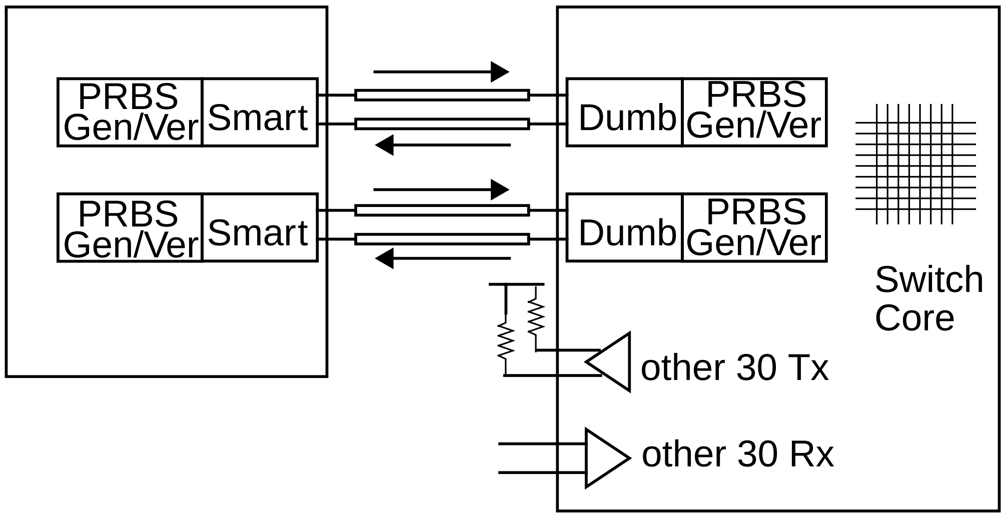


Figure 6.9: Test Setup for the 32 x 32 crossbar chip

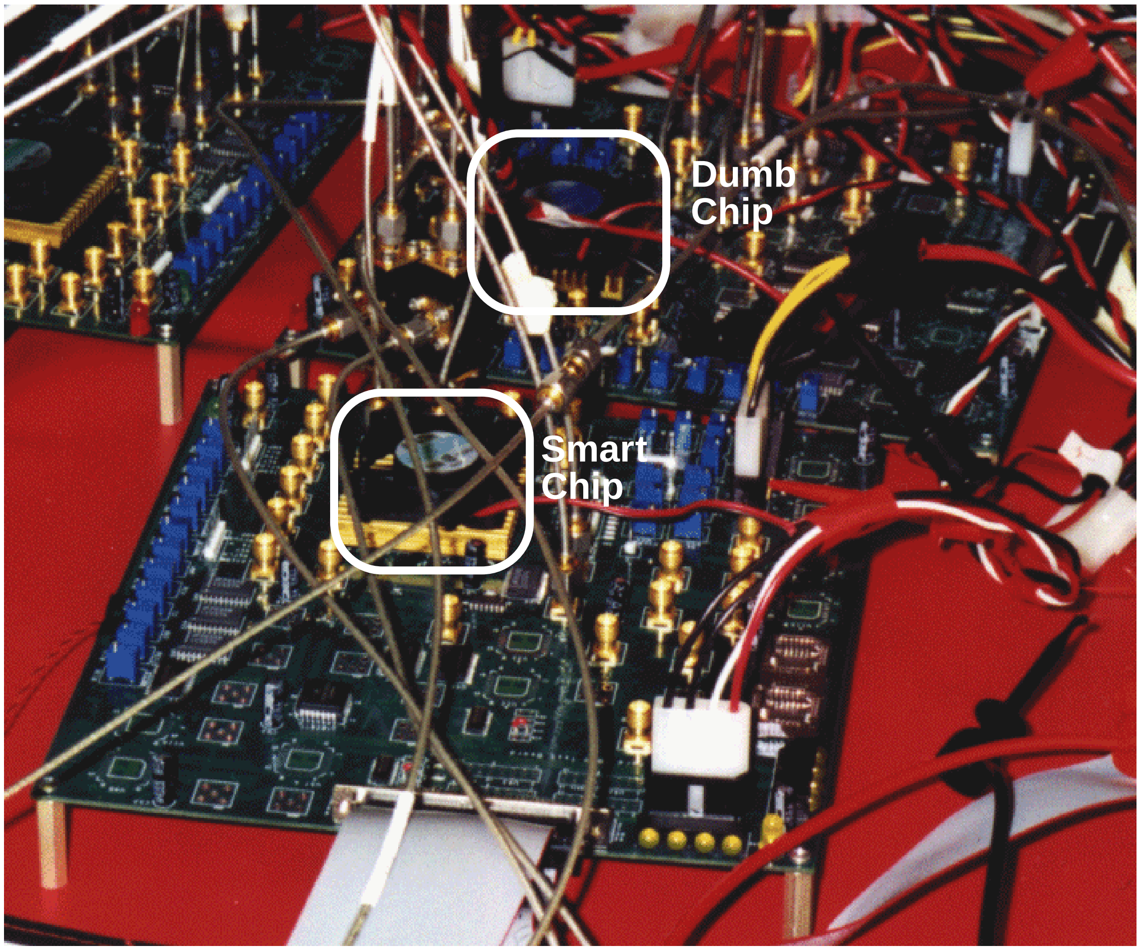
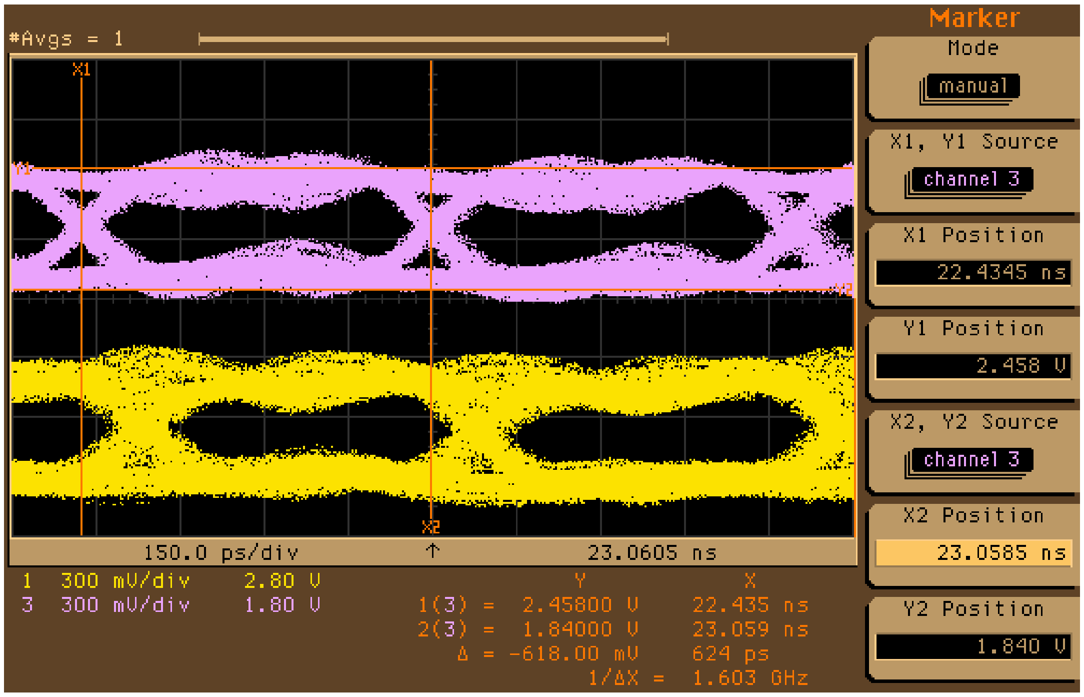


Figure 6.10: Test Boards for the 32 x 32 crossbar chip

the two dumb ends that have PRBS generators and verifiers are connected to the smart chip. The links from the other 28 dumb ends (T-DUMB) are terminated on the board.

To test the Asymmetric Serial Links (the first test), two links are running PRBS testing while all the other 30 links are running calibration patterns to the boards, which are similar to pseudo-random signals. The switch core broadcasts one of the received PRBS to generate switching noise. Figure 6.11 shows the eye diagram at the transmitter output for one of the smart-to-dumb and dumb-to-smart links measured at 1.6Gb/s. The signal amplitude is 600mV with differential signalling. The waveform shows significant reflection noise. This is due to a design mistake on the boards. The power and ground plane underneath the SMA connectors were left unvoided, making the SMA connectors very capacitive. The TDR measurement on the SMA connectors shows that the impedance is 25Ω for a period of about 60ps.



```
#Avgs = 1                                            Marker
                                                      Mode
                                                      manual
                                                  X1, Y1 Source
                                                      channel 3
                                                  X1 Position
                                                      22.4345 ns
                                                  Y1 Position
                                                      2.458 V
                                                  X2, Y2 Source
                                                      channel 3
                                                  X2 Position
        150.0 ps/div      ↑       23.0605 ns          23.0585 ns
1  300 mV/div    2.80 V            Y             X
3  300 mV/div    1.80 V   1(3) =  2.45800 V   22.435 ns   Y2 Position
                          2(3) =  1.84000 V   23.059 ns       1.840 V
                             Δ = -618.00 mV   624 ps
                                    1/ΔX =    1.603 GHz
```

Figure 6.11: Eye Diagram of the Asymmetric Serial Link on the 32 x 32 crossbar chip (1.6Gb/s)

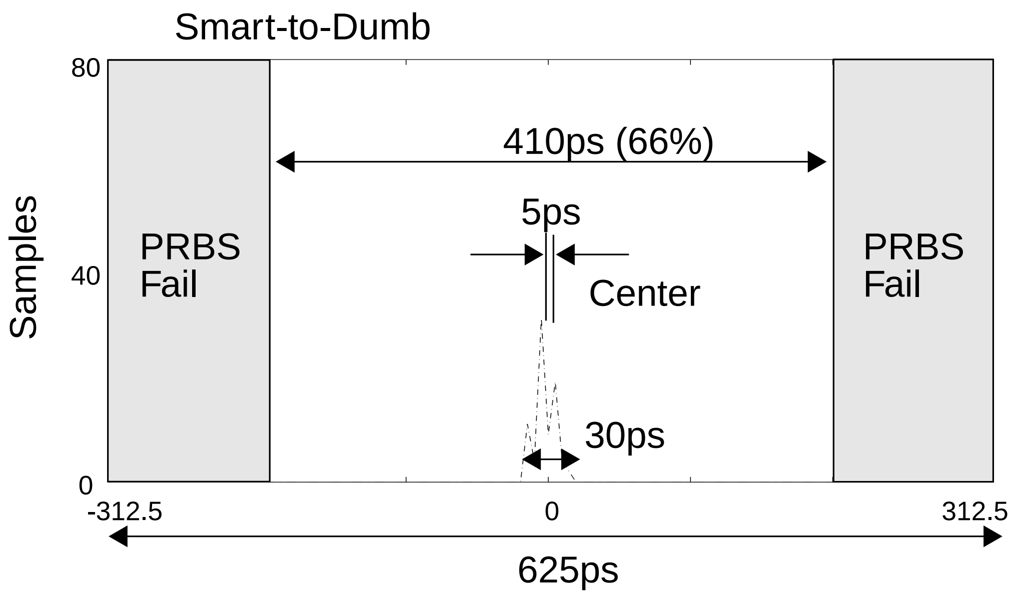


Figure 6.12: Timing Margin at 1.6Gb/s (the Smart-to-Dumb Link)

Figure 6.12 shows the timing margin of the smart-to-dumb link at the receiver output at 1.6Gb/s. The speed was limited by the digital measurement circuit instead of by the serial link itself. The measurement was based on the phase error injection method explained in Section 6.1 and [2]. The histogram was based on 80 measurement samples from the phase error injection. The two sides of the diagram show the failing points for the 80 measurements. The figure shows that the timing margin is about 66% of a bit time and that the average receiver sampling points are only 5ps from at the center of the data eye. This indicates that using the same receiver for both data and timing recovery compensates for any systematic offsets effectively.

To test the switch core with all of the serial links transmitting data at full speed, each smart end is assigned a port number. The transmitted data are associated with the port number. During the test, each port uses a second 7-bit PRBS generator to generate pseudo-random routing tags and sends them along with the encoded data. Since each tag carries the data source for the port, by extracting the port number from the incoming data, one can identify whether the data are sent correctly through the switch core and the serial links. For these tests, our experiments showed that each link can run reliably at a speed of 1.6 Gb/s, with the switch core running at a 200MHz clock rate. As a result, the crossbar chip can provide a raw data bandwidth above 50 Gb/s with all of the 32 channels running.

Supply Voltage	**2.5V**
Technology	**TI 0.28μm CMOS**
PLL Range (VCO rate)	**200MHz-1.44GHz**
Power @ 1.6Gb/s per link	**5W (full crossbar)**
Jitter (@800MHz)	**75ps pk-pk/12ps RMS**
Supply Sensitivity (pk-pk)	**1.4ps/mV (@800MHz)**
Max Data Rate/Link w/o Crossbar Core	**1.92Gb/s (10^{-14} BER)**
Max Data Rate/Link with Crossbar Core	**1.6Gb/s (10^{-14} BER)**
Max Crossbar Bandwidth	**51.2Gb/s**

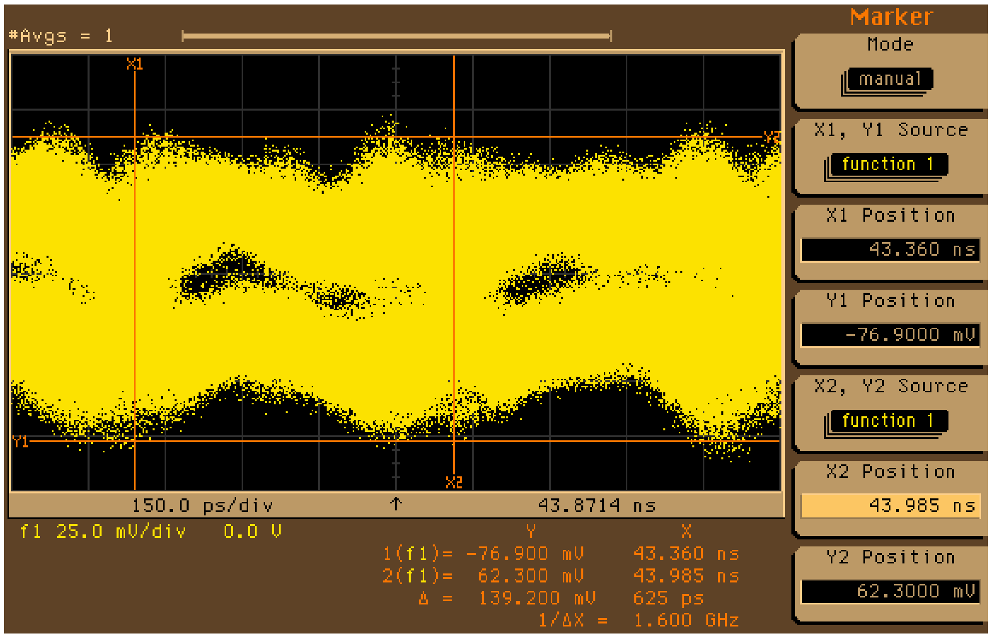
Table 6.2: Performance of the 32 x 32 Crossbar Chip

Table 6.2 summarizes the performance of the 32×32 crossbar chip. The chip operates successfully with links running at 1.6 Gb/s. The measured bit-error-rate is less than 10^{-14} when all channels and the switch core are operating. The crossbar chip consumes 5W and provides a total bandwidth of above 50 Gb/s. The clock jitter and supply sensitivity are larger than those shown in Table 6.1 for the one-channel test chip because the measurement for the crossbar chip includes the effect of jitter from the 800MHz clock distribution tree.

6.4 Differential Signalling and Single-Ended Signalling

Figure 6.13 shows the eye diagram of a differential input with minimum signal amplitude, created by subtracting one input from the other. According to the eye diagram, the minimum differential signal amplitude is 140mV and hence is 70mV peak-to-peak for each input. At this amplitude, the link can be reset, calibrated, synchronized, and can achieve a bit-error-rate less of than 10^{-11} (about 10 minutes without an error). Figure 6.14 shows the eye diagram of a single-ended input with minimum signal amplitude with the same bit-error rate as for Figure 6.13. It shows that the minimum amplitude is 360mV, much larger than the 70mV for the differential signalling. This difference in the minimum signal amplitude may come from three reasons. First, the pull-up termination resistor for the single-ended signalling measurement is off-chip[3] and thus the received signal is referenced to the off-chip power supply on the board. However, the reference voltage is generated on the chip and referenced to the on-chip power supply. The variations between the on-chip and off-chip power supply degrade the noise margin of the single-ended link. Second, even if the termination resistor is on-chip, the frequency response of the receiver input and the reference voltage is still unlikely to be identical for the single-ended link. This introduces differential noise between the receiver input and the reference voltage. Third, the common mode voltage of the single-ended input and the reference voltage varies with the input signal. Even though the receiver uses differential circuits at the front end, it still has finite common-mode rejection ratio. Hence the receiver can support smaller signal amplitude for differential input than single-ended input.

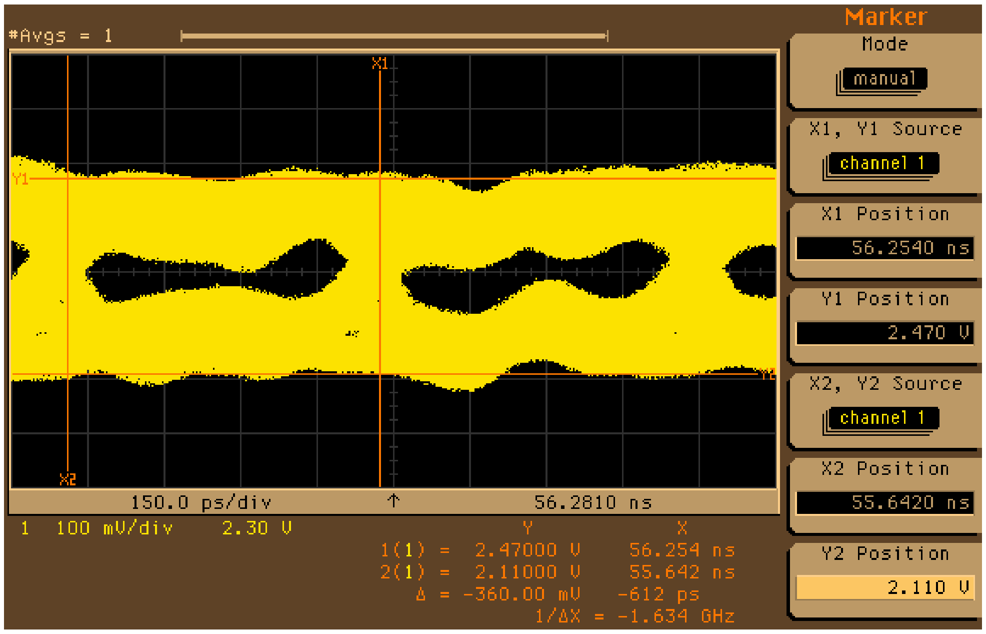
3. In fact, on-chip termination resistors using PMOS transistors operating in the linear region were implemented. However, the resistance value is controlled by analog gate voltage of the PMOS transistors. In the single-ended links, the coupling noise from the received signals greatly affects the accuracy of the effective resistance of the PMOS transistors. As a result, the PMOS transistors were turned off and external termination resistors were used for the single-ended links. For the differential links, the coupling noise from the complementary inputs cancelled each other so the on-chip PMOS termination still works for the differential links.



```
#Avgs = 1
150.0 ps/div          43.8714 ns
f1 25.0 mV/div    0.0 V
                    Y              X
         1(f1)= -76.900 mV    43.360 ns
         2(f1)=  62.300 mV    43.985 ns
             Δ =  139.200 mV   625 ps
                    1/ΔX =    1.600 GHz
```

```
Marker
Mode
manual
X1, Y1 Source
function 1
X1 Position
43.360 ns
Y1 Position
-76.9000 mV
X2, Y2 Source
function 1
X2 Position
43.985 ns
Y2 Position
62.3000 mV
```

Figure 6.13: Eye Diagram with the Minimum Signal Amplitude (Differential Signalling)



```
#Avgs = 1
150.0 ps/div          56.2810 ns
1   100 mV/div     2.30 V
                    Y              X
         1(1)  =  2.47000 V    56.254 ns
         2(1)  =  2.11000 V    55.642 ns
             Δ = -360.00 mV   -612 ps
                    1/ΔX = -1.634 GHz
```

```
Marker
Mode
manual
X1, Y1 Source
channel 1
X1 Position
56.2540 ns
Y1 Position
2.470 V
X2, Y2 Source
channel 1
X2 Position
55.6420 ns
Y2 Position
2.110 V
```

Figure 6.14: Eye Diagram with the Minimum Signal Amplitude (Single-Ended Signalling)

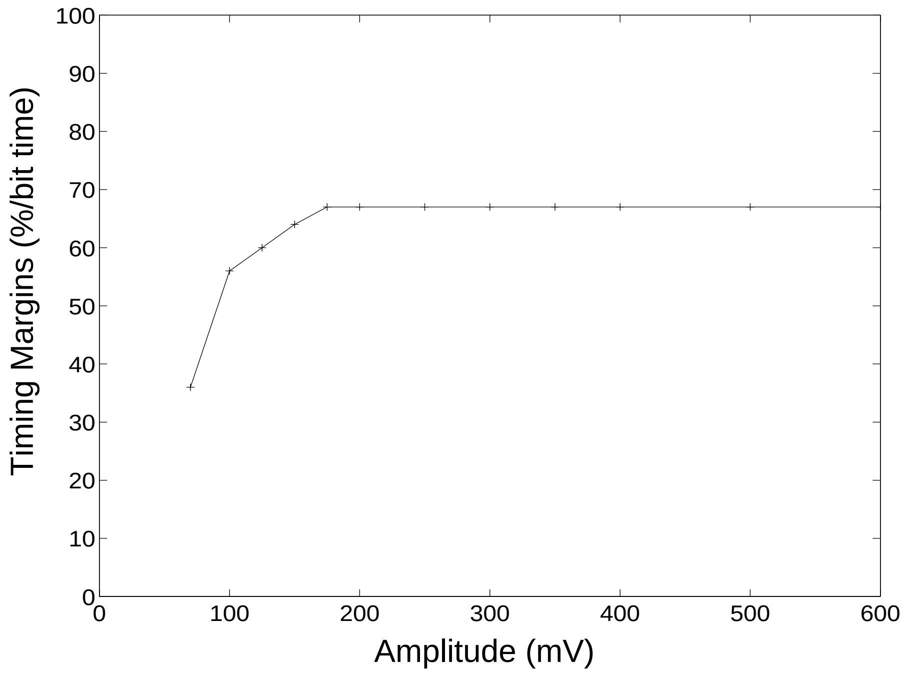


Figure 6.15: Relationship between the Timing Margin and the Signal Amplitude (Differential Signalling)

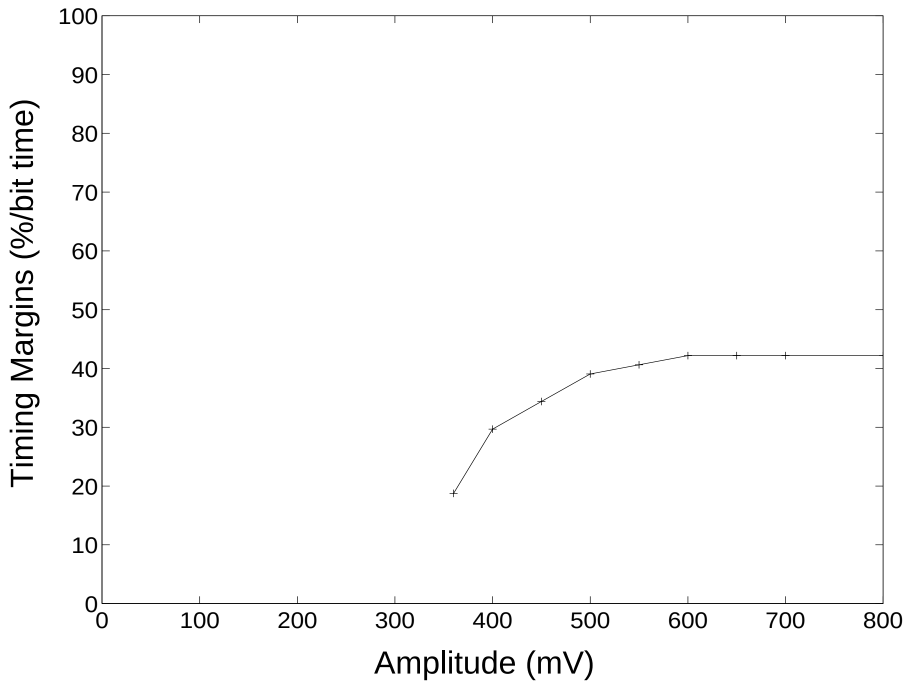


Figure 6.16: Relationship between the Timing Margin and the Signal Amplitude (Single-Ended Signalling)

Figure 6.15 shows the relationship between the timing margin of the link and the signal amplitude for differential signalling while Figure 6.16 shows the relationship for the single-ended link. The two figures show that the timing margins of the single-ended links are 60% smaller than those of the differential links. The three reasons described above for the difference in minimum signal amplitude also contribute to this difference in the timing margins.

Both figures indicate that the timing margin remains unchanged for signal amplitudes down to a certain voltage value. After that, the timing margin drops steeply, and stops working at and below the minimum signal amplitude.

6.5 Summary

This chapter has presented the experimental results from a one-channel test chip for the Asymmetric Serial Link and from the 32×32 crossbar chip that uses Asymmetric Serial Links.

The experiments demonstrated several interesting aspects of the asymmetric link. First, as expected, the link works as well as conventional serial links, due to the fact that the link relies on PLLs to track noise. Second, only DC calibration is needed. After the link is booted up, the PLLs on both the smart and dumb ends can track the dynamic timing variations so that no further calibration is needed during link operation. Finally, the links operating in the differential signalling mode are very robust against amplitude noise and can work with very small signal amplitude, down to 70mV, while still achieving a bit-error-rate of less than 10^{-11}.

Furthermore, the experiment also showed that an on-chip timing margin tester can be built with the help of the digitally-controlled phase selectors. The measurement results can provide useful information such as the position of the receiver sampling point in a data eye, which provides link designers a direct way to evaluate how well a timing recovery mechanism works. With the on-chip timing margin tester, our measurement results demonstrated that the receiver sampling points are very close to the ideal eye center. This indicates that the scheme of using the same receiver for both data reception and phase detection minimizes the timing offset of the phase detection.

The 32×32 crossbar chip demonstrated the robustness of the asymmetric link within a noisy environment. The chip operates successfully with links running at 1.6 Gb/s. The measured bit-error-rate is less than 10^{-14} when all channels and the switch core are operating. The crossbar chip consumes 5W and provides a total bandwidth of above 50 Gb/s.

Chapter 7

Conclusions

This dissertation presented the design of a CMOS Asymmetric Serial Link, which provides high data bandwidth for network crossbar switches while helping avoid the creation of hot spots in the switches. The link moves the clock recovery circuit of the inbound links from the crossbar chip to the transmitter end to avoid having too many PLLs on a single crossbar chip. This relieves the problems of heat dissipation and noise coupling among PLLs.

The design of these links is not very difficult. The synchronization procedure for the smart-to-dumb and dumb-to-smart links are quite symmetric. The only asymmetry comes from the fact that the PLLs for calibrating the clocks are only on the smart end. More importantly, the performance of the asymmetric link is comparable to that of any conventional serial link that has the same design cost.

The Asymmetric Serial Link provides several potential advantages for high-fanin chips. It reduces the number of PLLs needed for each chip. The high-fanin crossbar dumb end needs only one PLL for the entire chip, instead of one PLL for each serial link. This link also makes the high fanin chip consume less power and have a smaller area, lowering design cost compared to the same chip built with conventional serial links. Furthermore, from a design standpoint, there are no retiming issues in the high-fanin chip, and this could reduce its latency.

Link calibration used in conjunction with a local PLL and a delay-replica clock generation scheme can track noise. The experiments demonstrated that only DC calibration is needed. This indicates that the use of the periodic calibration costs only a little in terms of data transmission bandwidth.

Interestingly, the use of calibration packets can provide better timing for the link because one can use the same receiver for both data reception and phase detection. This eliminates timing offset from the receiver and the clock buffers. The only offset is from the 90^{o} clock phase shift used during phase detection and it can be minimized by careful layout in the

VCO. The scheme of using calibration packets is applicable to any other serial links or even parallel links; it is not limited to the Asymmetric Serial Link.

The experiments also demonstrated that with sufficiently large signal amplitude, single-ended signalling can achieve speed comparable to that of differential signalling. However, the single-ended links require greater engineering effort in circuit design, coupling noise analysis, package design, and board layout. In contrast, the differential links are more robust in noise immunity and easier to design than the single-ended links.

This thesis study provides a complete example of the design of the high-speed links, which can be used in practical applications. Our 32×32 crossbar chip using 32 asymmetric links can be used in the Tiny Tera or can be applied to any similar network switches or routers. The chip demonstrated that with the use of the high-speed serial links, designers can greatly increase the total bandwidth of the backplane interconnect in a high-speed network switch at a reasonable design cost. The links provide a fitting solution to the high-fanin nature of the system.

Bibliography

Serial Links

[1] M. Banu and A. Dunlop, “A 660Mb/s Clock Recovery Circuit with Instantaneous Locking for NRZ Data and Burst-Mode Transmission,” *1993 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 102-103, Feb. 1993.

[2] K.Y. Chang, *et al.*, “A 2 Gb/s Asymmetric Link for High-Bandwidth Packet Switches,” *Hot Interconnects V Symposium Record*, pp. 171-9, Aug. 1997.

[3] K.Y. Chang *et. al.* “A 2 Gb/s/pin CMOS asymmetric serial link” 1*998 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 216-17, Jun. 1998.

[4] William Dally, *et al.* “Transmitter equalization for 4-Gbps signaling” *IEEE Micro*, vol.17, no.1, pp 48-56, Jan/Feb. 1997.

[5] William Ellersick, *et al.* “GAD: A 12-GS/s CMOS 4-bit A/D Converter for an Equalized Multi-Level Link,” *1999 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 49-52, Jun. 1999.

[6] J. Ewen, *et al.*, “Single-Chip 1062Mbaud CMOS Transceiver for Serial Data Communication,” *1995 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 32-33, Feb. 1995.

[7] Ramin Farjad-Rad, *et. al.* “A 0.4-µm CMOS 10-Gb/s 4-PAM pre-emphasis serial link transmitter” *1998 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 198-199, Jun. 1998.

[8] Ramin Farjad-Rad, *et at.* “A 0.3-µm CMOS 8-Gb/s 4-PAM Serial Link Transceiver,” *1999 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 41-44, Jun. 1999.

[9] A. Fiedler, *et al*, “A 1.0625Gb/s Transceiver with 2x-Oversampling and Transmit Signal Pre-Emphasis,” *1997 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 238-239, Feb. 1997.

[10] R. Gu, *et al.*, “A 0.5-3.5Gb/s Low-Power Low-Jitter Serial Data CMOS Transceiver,” *1999 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 352-353, Feb. 1999.

[11] Mark Horowitz, K. Yang, and S. Sidiropoulos, “High-Speed Electrical Signalling: Overview and Limitations”, *IEEE Micro*, vol.18, no.1, pp.12-24, Jan/Feb. 1998.

[12] T.H. Hu and Paul Gray, “A monolithic 480 Mb/s parallel AGC/decision/clock-recovery circuit in 1.2-µm CMOS,” *IEEE Journal of Solid-State Circuits*, vol.28, no.12, pp. 1314-1320, Dec. 1993.

[13] N. Ishihara, *et al.*, “3.5-Gb/s x 4-Ch Si bipolar LSI’s for Optical Interconnections,” *IEEE Journal of Solid-State Circuits*, vol.30, no.12, pp. 1493-1501, Dec. 1995.

[14] M. Izzard, *et al.*, “Analog versus Digital Control of a Clock Synchronizer for a 3Gb/s Data with 3.0V Differential ECL,” *Proceedings of the 1994 Symposium on VLSI Circuits*, pp. 39-40, June 1994.

[15] H.O. Johansson, J. Yuan, C. Svensson, “A 4-GSamp/s Line-Receiver in 0.8µm CMOS,” *1996 Symposium on VLSI Circuits. Digest of Technical Papers*, pp. 116-117, Jun. 1996.

[16] B. Kim, *et al.*, “A 30-MHz Hybrid Analog/Digital Clock Recovery Circuit in 2-µm CMOS,” *IEEE Journal of Solid-State Circuits*, vol.25, no.12, pp. 1385-1394, Dec. 1990.

[17] S. Kim, *et al.*, “An 800Mbps Multi-Channel CMOS Serial Link with 3x Oversampling,” *IEEE 1995 Custom Integrated Circuits Conference Proceedings*, pp. 451, Feb. 1995.

[18] K. Lee, *et al.*, “A CMOS serial link for fully duplexed data communication,” *IEEE Journal of Solid-State Circuits*, vol.30, no.4, pp. 353-64, Apr. 1995.

[19] S.I. Long, J.Q. Zhang, “Low Power GaAs Current-Mode 1.2Gb/s Interchip Interconnections,” *IEEE Journal of Solid-State Circuits*, vol.32, no.6, pp. 890-897, Jun. 1997.

[20] A. Pottbacker and U. Langmann, “An 8-GHz Silicon Bipolar Clock-Recovery and Data-Regenerator IC,” *1994 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 116-117, Feb. 1994

[21] J. Poulton, et al., “A Tracking Clock Recovery Receiver for 4-Gb/s Signaling,” *Hot Interconnects V Symposium Record*, Stanford, pp. 157-69, Aug. 1997.

[22] M. Soyuer and H. Ainspan, “A Monolithic 2.3Gb/s 100mW Clock and Data Recovery Circuit,” *IEEE Journal of Solid-State Circuits* vol.28, no.12, pp. 1310-1313, Dec. 1993.

[23] R.C. Walker, et al, “A 1.5-Gb/s link interface chipset for computer data transmission,” *IEEE Journal of Solid-State Circuits*, vol.26, no.6, pp. 698-703, Jun. 1991.

[24] A. Widmer, *et. al.*, “Single-Chip 4X500 Mbaud CMOS Transceiver,” *IEEE Journal of Solid-State Circuits*, vol. 31, no. 12, pp. 2004-2014, Dec. 1996.

[25] C.K. Yang, et al., “A 0.8-μm CMOS 2.5 Gb/s oversampling receiver and transmitter for serial links,” *IEEE Journal of Solid-State Circuits*, vol.31, no.12, pp. 2015-23, Dec. 1996.

[26] C.K. Yang, *et. al*, “A 0.5-μm CMOS 4Gb/s Serial Link Transceiver with Data Recovery using Oversampling,” *IEEE Journal of Solid-State Circuits*, vol.31, no.12, pp. 2015-2023, Dec. 1996.

[27] Ah-Lyan Yee, *et al*. “An Integratable 1-2.5Gbps Low Jitter CMOS Transceiver with Built in Self Test Capability,” *1999 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 45-46, Jun 1999.

Parallel Links

[28] K.S. Donnelly, *et al.*, “A 660-MB/s interface megacell portable circuit in 0.3-μm-0.7-μm CMOS ASIC,” *IEEE Journal of Solid-State Circuits*, vol.31, no.12 pp. 1995-2003, Dec. 1996.

[29] K. Gotoh, *et al.*, “A 2B Parallel 1.25Gb/s Interconnect I/O Interface with Self-Configurable Link and Plesiochronous Clocking,” *1999 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 180-181, Feb. 1999.

[30] M. Griffin, *et al.* "A Process Independent 800MB/s DRAM Bytewide Interface Featuring Command Interleaving and Concurrent Memory Operation," *1998 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 156-157, Feb. 1998.

[31] N. Kushiyama, *et al.* "A 500-Megabyte/s Data-Rate 4.5M DRAM", *IEEE Journal of Solid-State Circuits*, vol. 28, No. 4, pp. 490-498, April 1993

[32] E. Reese, *et al.*, "A Phase-Tolerant 3.8GB/s Data-Communication Router for a Multiprocessor Supercomputer Backplane," *1994 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 296-297, Feb. 1994.

[33] S. Sidiropoulos, *et. al*, "A CMOS 500Mbps/pin Synchronous Point to Point Link Interface," *Proceedings of 1994 IEEE Symposium on VLSI Circuits. Digest of Technical Papers*, pp. 43-44, Jun. 1994.

[34] S. Sidiropoulos, *et al.*, "A 700-Mb/s/pin CMOS signaling interface using current integrating receivers," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 681-690, May 1997.

Bi-directional Signalling

[35] L. Dennison, *et al.*, "High-Performance Bidirectional Signaling in VLSI Systems," *Proceedings of the Symposium on Integrated Systems, MIT Press*, pp. 300-319, March, 1993.

[36] M. Haycock, *et al.*, "A 2.5Gb/s Bi-directional Signaling Technology," *Hot Interconnects V Symposium Record*, pp. 149-156, Aug. 1997.

[37] K. Lam, *et al.* "Simultaneous Bidirectional Signalling for IC systems," *Proceedings of the 1990 IEEE International Conference on Computer Design*, pp. 430-433, Oct. 1990.

[38] K. Ishibashi, *et al.*, "SBTL (Simultaneous Bi-directional Transceiver Logic) for a 26.8GB/s Crossbar Switch," *Hot Interconnects VI Symposium Record*, pp. 73-76, Aug. 1998.

[39] R. Mooney, *et al.*, "A 900 Mb/s bidirectional signaling scheme," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 12, pp. 1538-1543, Dec. 1995.

[40] T. Takahashi, *et al.*, “A CMOS Gate Array with 600 Mb/s Simultaneous Bidirectional I/O Circuits,” *IEEE Journal of Solid State Circuits*, vol. 30, no. 12. pp. 1544-1546, Dec. 1995.

[41] T. Takahashi, *et al.*, “110GB/s Simultaneous Bi-Directional Transceiver Logic Synchronized with a System Clock,” *1999 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 176-177, Feb. 1999.

PLLs

[42] F.M. Gardner, “Charge-pump phase-lock loops,” *IEEE Transactions on Communications,* vol. 28, no. 11, pp. 1849-1858, Nov. 1980.

[43] F. Gardner, *Phase Lock Techniques,* Wiley 1979.

[44] M.A. Horowitz, *et al.*, “PLL Design for a 500MB/s Interface,” *1993 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 160-161, Feb. 1993.

[45] D.K. Jeong, *et al.*, “Design of PLL-based clock generation circuits,” *IEEE Journal of Solid-State Circuits*, vol. 22, no. 2, pp. 255-261, Apr. 1987.

[46] M.G. Johnson, *et al.*, “A variable delay line PLL for CPU-coprocessor synchronization,” *IEEE Journal of Solid-State Circuits*, vol.23, no.5, pp. 1218-1223, Oct. 1988.

[47] J. Lee, *et al.*, “A 250MHz Low Jitter Adaptive Bandwidth PLL”, *1999 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 346-347, Feb. 1999.

[48] P. Larsson, “A 2-1600MHz 1.2-2.5V CMOS Clock-Recovery PLL with Feedback Phase-Selection and Averaging Phase-Interpolation for Jitter Reduction,” *1999 International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 356-357, Feb. 1999.

[49] T.H. Lee, *et al.*, “A 2.5 V CMOS Delay-Locked Loop for 18 Mbit, 500 Megabyte/s DRAM,” *IEEE Journal of Solid-State Circuits*, vol. 29, no. 12, pp. 1491-1496, Dec. 1994.

[50] T.H. Lee, J. Bulzachelli, "A 155MHz Clock Recovery Delay-and Phase-Locked Loop," *IEEE Journal of Solid-State Circuits,* Dec. 1992, vol. 27, no. 12, pp. 1736, Dec. 1992.

[51] J.G. Maneatis, "Low-jitter process-independent DLL and PLL based on self-biased techniques" *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1723-1732, Nov. 1996.

[52] J.G. Maneatis, *et al.*, "Precise delay generation using coupled oscillators," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 12, pp. 1273-1282, Dec. 1993.

[53] D. Mijuskovic, *et al.*, "Cell-based Fully Integrated CMOS Frequency Synthesizers," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, pp. 271-279, Mar. 1994.

[54] I. Novof, *et al.*, "Fully Integrated CMOS Phase-Locked Loop with 15 to 240 MHz Locking Range and +/- 50ps Jitter," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1259-1266, Nov. 1995.

[55] B. Razavi, Editor, *Monolithic Phase Locked Loops and Clock Recovery Circuits*, IEEE Press, 1996.

[56] S. Sidiropoulos, M.A. Horowitz, "A Semidigital Dual Delay-Locked Loop," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1683-92, Nov. 1997.

[57] I.A. Young, *et al.*, "A PLL clock generator with 5 to 110 MHz lock range for microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, pp. 1599-607, Nov. 1992.

ATM and Crosspoint switches

[58] M. Akata, *et al.* "A 250Mb/s 32×32 CMOS Crosspoint LSI for ATM Switching Systems," *IEEE Journal of Solid-State Circuits*, vol. 25, No.6, Dec. 1990.

[59] F. Barber, *et al.* "A 64×17Non-Blocking Crosspoint Switch," *1988 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 116-117, Feb. 1988.

[60] S. Carpenter, *et al.* “A 146Mb/s Time Space Switch Chip,” *1988 IEEE International Solid-State Circuits Conference. Digest of Technical Papers,* pp. 112-113, Feb. 1988.

[61] K.Y. Chang, *et al.* “A 50 Gb/s 32×32 CMOS Crossbar Chip using Asymmetric Serial Links,” 1*999 Symposium on VLSI Circuits Digest of Technical Papers,* pp. 19-22, June 1999.

[62] H, Kondoh, *et al.* “A 622-Mb/s 8×8 ATM Switch Chip Set with Shared Multibuffer Architecture,” *IEEE Journal of Solid-State Circuits,* vol. 28, No. 7, pp. 808-815, July 1993.

[63] A. Mu. *et al.*, “A 285 MHz 6-port Plesiochronous Router Chip with non-blocking cross-bar switch,” *Proceedings of 1996 IEEE Symposium on VLSI Circuits. Digest of Technical Papers*, pp. 136-137, June 1996.

[64] G. S. La Rue, *et al.*, “Gigabit Complementary HFET Communication Circuits: 16:1 Multiplexer, 1:16 Demultiplexer and 16×16 Crosspoint Switch,” *1996 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 124-125, Feb. 1996.

[65] A. Le Fevre, R. Flett, “A 100Mb/s Multi-LAN Crosspoint Chip-Set for Cable Management,” *IEEE Journal of Solid-State Circuits*, vol.32, no.7, pp. 1115-1121, Jul. 1997.

[66] K. Lowe, *et al.* “A GaAs HBT 16×16 10-Gb/s/channel Crosspoint Switch,” *IEEE Journal of Solid-State Circuits*, vol. 32, No. 8, Aug. 1997.

[67] Y. Ohtomo, *et al.*, “A 40Gb/s 8×8 ATM Switch LSI using 0.25µm CMOS/ SIMOX,” 1997 *IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 154-155, Feb. 1997.

[68] J. Pulver, *et al.*, “A 16×16 Si/SiGe HBT Cross-Point Switch Architecture for High-Speed Switching Applications,” *Hot Interconnects VI Symposium Record*, pp. 97-108, Aug. 1998.

[69] R. Savara, *et al.* “A 2.5Gb/s 16×16 bit Crosspoint Switch with Fast Programming,” *IEEE GaAs IC Symposium*, pp. 47-48, Nov. 1995.

[70] H. Shin, *et al.* “A 250Mb/s CMOS Crosspoint Switch,” *1988 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 114-115, Feb. 1988.

[71] H. Shin, *et al.* “An Experimental 5-Gb/s 16×16 Si-Bipolar Crosspoint Switch,” *IEEE Journal of Solid-State Circuits*, vol. 27, No. 12, Dec. 1992.

[72] T. Yoneda, *et al.* “An ECL Compatible Full CMOS 210Mbps Crosspoint Switch,” *IEEE Custom Integrated Circuits Conference*, May 1989 pp.10.7/1-4.

Circuits

[73] M.J. Pelgrom, “Matching Properties of MOS Transistors,” *IEEE Journal of Solid State Circuits,* vol. 24, no. 10, pp. 1433-1439, Oct. 1989.

[74] C.L. Portmann, *et al.*, “Metastability in CMOS library elements in reduced supply and technology scaled applications,” *IEEE Journal of Solid State Circuits,* vol. 30, no. 1, pp. 39-46, Jan. 1995.

[75] M.Y. Hsiao, K.Y. Sih, “Serial-to-Parallel Transformation of Linear-Feedback Shift-Register Circuits,” *IEEE Transaction on Electronic Computers*, pp 738-740, Dec. 1968.

[76] R.G. Swartz, *et al.*, “A Burst mode, Packet Receiver with Precision Reset and Automatic Dark Level Compensation for Optical Bus Communications,” *IEEE Journal of Solid-State Circuits*, vol. 29, no. 2, pp. 325, Feb. 1994.

[77] A. Yukawa, *et al.*, “A CMOS 8-bit High Speed A/D Converter IC,” *IEEE Journal of Solid-State Circuits*, vol. 20, no. 3, pp. 775-779, Jun. 1985.

Network Systems

[78] High Performance Networking, http://infonet.aist-nara.ac.jp/member/nori-d/inet/hpn.html

[79] http://www.ascend.com/230.html.

[80] http://www.bbn.com.

[81] http://www.baynetworks.com/accelar/1200.shtml.

[82] Michael Laor, “The 12000GSR switch fabric,” *Hot Interconnects VI Symposium Record*, pp. 189-201, Aug. 1998

[83] Nick McKeown, *et. al*. “Tiny Tera: A Packet Switch Core,” *IEEE Micro*, vol. 17, no. 1, pp. 26-33, Jan/Feb. 1997.

Computer Systems

[84] M. Galles, *et al.*, “Spider: a high-speed network interconnect,” *IEEE Micro*, vol. 17, no. 1, pp. 34-139, Jan/Feb. 1997.

[85] J. Kuskin, et al., “The Stanford FLASH Multiprocessor,” *Proceedings of the 21st International Symposium on Computer Architecture*, Chicago, IL, pp. 302-13, Apr. 1994.

[86] H. Nakajo, *et al.*, “High Speed Serial Communication in a Future Parallel Computer Architecture,” *1998 IEEE Proceedings on Innovative Architecture for Future Generation High-Performance Processors and Systems*, pp 125-132.

SONET/Optical Fiber

[87] M. Cerisola, *et al.*, “CORD - A WDM Optical Network: Control Mechanism Using Subcarrier Multiplexing and Novel Synchronization Solutions,” *IEEE International Conference on Communication*, 1995, pp. 261-265.

[88] Y. Ota, R.G. Swartz, “Multichannel Parallel Data Link for Optical Communication,” *IEEE LTS*, May. 1991, vol. 2, no. 2, pp. 24-32.

Dissertations

[89] John Maneatis, *Precise Delay Generation Using Coupled Oscillators*, Ph.D. Dissertation, Stanford University, 1994.

[90] C. Portmann, *Characterization and Reduction of Metastability Errors in CMOS Interface Circuits*, Stanford Universtiy, 1995.

[91] Stefanos Sidiropoulos, *High-Performance Inter-Chip Signalling*, Ph.D. Dissertation, Stanford University, 1998.

[92] Chih-Kong Ken Yang, *Design of High-Speed Serial Links in CMOS*, Ph.D. Dissertation, Stanford University, 1998.

Reference Books

[93] H. B. Bakoglu, "*Circuits, Interconnections, and Packaging for VLSI*," Addison-Wesley Publication Company, 1990.

[94] David K. Cheng, "*Field and Wave Electromagnetics*," Addison-Wesley Publication Company, 1989.

[95] Paul Chow, "*The MIPS-X RISC Microprocessor*," Kluwer Academic Publishers, pp.45-47, 1989.

[96] William Dally and John Poulton, "*Digital Systems Engineering*," Cambridge, 1998.

[97] Rao R. Tummala and Eugene J. Rymaszewski, "*Microelectronics Packaging Handbook*," Van Nostrand Reinhold, 1989.

Others

[98] Widmer, *et al.*, "A DC-Balanced Partitioned-Block, 8B/10B Transmission Code," *IBM Journal of Research and Development*, vol. 27, pp. 440-451, Sep. 1983.

Appendix A

A Serial Interface for Real-Time Chip Testing

In the design of prototype chips, debuggability is vital to chip testing. Debuggability includes both controllability and observability, which are also necessary for production scan tests. The conventional scan test connects all the scannable flip-flops into one or several scan chains. Read and write of the stored values in the flip-flops rely on scan-out and scan-in data in the scan chains. One of the drawbacks in this type of testing is that it destroys the internal states during the scan and thus is undesirable for any testing that requires real-time read and write operation of the internal states.

One of the keys to real-time chip testing is that one must be able to update and read out the flip-flops' values without affecting the normal operations. If there is no concern about the pin count of the chip, one can connect all the states to the pins. This setup achieves maximum observability and controllability. However, the cost of this approach is its high pin count. To solve the problem, one can use a serial interface that is similar to the testing scan chain except that it comprises shift registers independent from all the flip-flops. To write values into the flip-flop, the serial interface first shifts the write values into the shift registers and then downloads them to the internal flip-flops. To read values from the flip-flops, one can download the flip-flops' values into the shift registers and serially shift the values out.

Figure A.1 shows the circuit block diagram of the serial interface. The serial interface uses only four signal pins: SerialIn, SerialOut, SerialClk, and SerialLd. To guarantee the interface is free from any hold-time violation, each shift register is made of a positive edge-triggered and a negative edge-triggered flip-flop. As shown in the upper left of the figure, each shift register can write a value out or read a value in, depending on the Read/ Write signal. To avoid an extra pin for this Read/Write signal, the serial interface is designed to do read after every write. The serial interface works as follows:

The serial interface will do a write operation first and then do read. To write the value, the interface places input data at SerialIn sequentially and toggles SerialClk to shift the write data into the shift registers. The write data includes both address and data. The address indicates the write or read address for this operation. Once all the write data are shifted in, the interface toggles SerialLd. The decoder will decode the address to find what action to take next. If the operation is only write, the data are downloaded into the destination flip-flops based on the address. If the operation is read only or read after write, the decoder will set up the output mux to load the read data into the shift registers. The next SerialClk cycle after SerialLd is always read, and thus the shift registers load in the read data from the output mux. The serial interface continues to toggle SerialClk to shift out the read values.

This serial interface was used in the 32×32 crossbar chip. Each smart end, dumb end, and the crossbar controller have this type of serial interface. The interfaces are all daisy-chained together with the serial output of one block being connected into the serial input of the next block. This provides the maximum possible controllability and observability on the large crossbar chip at a cost of only four pins. The interface was used successfully in testing the crossbar chip. Because the serial interface is not performance bound, all the circuits for the serial interface in the 32×32 crossbar chip were built in synthesized logic and laid out by an automatic place and route tool. This further saved crucial design time in the tapeout process.

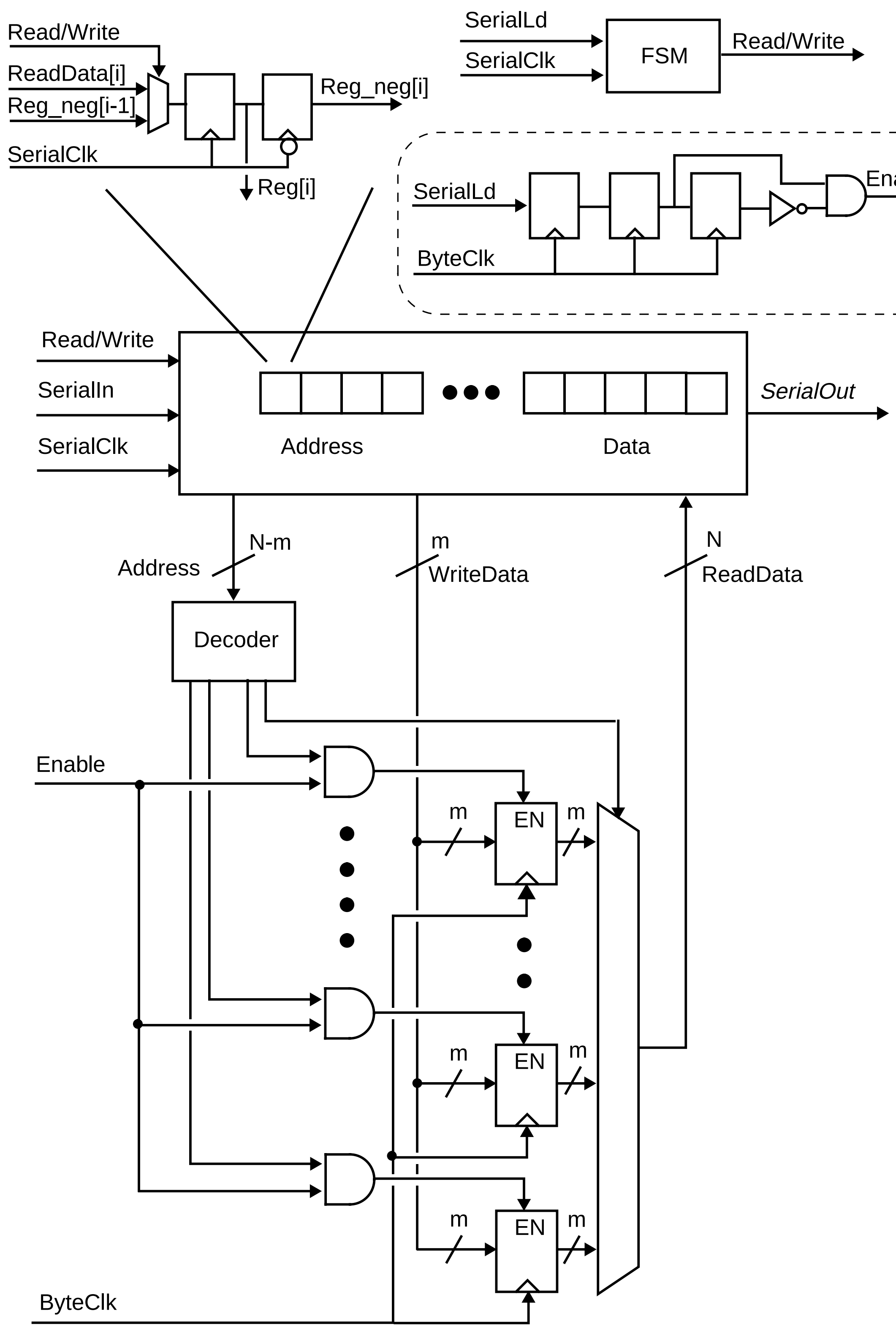


Figure A.1: Circuit Block Diagram of the Serial Interface