A 2 Gb/s Asymmetric Serial Link for High-Bandwidth Packet Switches

Ken K. -Y. Chang, William Ellersick, Shang-Tse Chuang, Stefanos Sidiropoulos,
Mark Horowitz, Nick McKeown: *Computer System Laboratory*, Stanford University
Martin Izzard: *DSP R&D Center*, Texas Instruments, Inc.

Abstract — **This paper describes the design of a novel CMOS 2 Gb/s asymmetric serial link. The serial link is designed for systems that use high speed chip-to-chip communications. In such designs, power dissipation is a common problem, particularly when multiple serial links are required on one chip. The power arises primarily from the phase adjustment circuitry used to align data with the clock. This circuitry is usually placed at the receiver, but in our asymmetric link design we take a different approach. We first assume that a link consists of two unidirectional connections — one for each direction of the link. We move the phase adjustment circuitry from one end of the link to the other, adjusting the phase of the transmitter rather than the receiver. Although this does not reduce overall system power, it allows us to choose the location of the phase adjustment circuitry, moving it from chips with a large number of links to chips with a smaller number. The link was designed for use in the Tiny Tera packet switch, which has a 32×32 crossbar switch at its center. Simulations suggests that the power dissipation of serial links on the crossbar switch can be reduced by a factor of 4.**

I. INTRODUCTION

There is growing interest in the use of chip-to-chip serial links for communication systems. The links typically consist of a fast driver at the transmitter, a fast receiver and clock recovery circuitry to align the phase of the incoming data to the receiver's clock. Operating in excess of 1 Gb/s, these links eliminate the need for wide, cumbersome buses, and can reduce system pin-count. Several high-speed serial links implemented in CMOS technology have been reported. Fiedler et al. [1] and Widmer et al. [6] presented serial links in which the receiver continuously recovered timing from data. But, although links have been tested at speeds up to 4 Gb/s [7], the fast clock rate leads to a very high power dissipation, making them unsuitable for chips with multiple links.

The aim of our work is to design a serial link operating at 2 Gb/s, that allows a large number of links to be placed on a single chip. It is therefore important to either reduce power, or move it to a chip that is less power sensitive. We take the latter approach. We are motivated by the design of the Tiny Tera packet switch [3], a physically small, high-bandwidth network switch comprised of 32 ports, each operating at over 16 Gb/s. At the center of the Tiny Tera is a parallel crossbar switch, and a scheduler for arbitrating access to the crossbar. A schematic of the switch is shown in Figure 1. The crossbar switch transfers fixed length packets between 32 identical switch ports. The switch ports provide buffering, and an interface to the outside world using variable length network packets.

The 32-way crossbar chips and scheduler[1] each require 32 serial links operating at 2 Gb/s to connect to the switch ports. Our motivation for using core chips with multiple serial links, is to reduce system pin-count. If instead we were to use a traditional parallel CMOS interfaces operating at 100Mbit/s without PLLs, we would require 1280 signal pins per core chip. Alternatively, we could reduce the number pins and use more core chips. This would inevitably require a huge rack of boards to build our switch. With fast skew-compensating serial links we can reduce the physical size of the switch. The Tiny Tera is designed to be approximately 6" tall.

However, because of the power generated by phase adjustment circuitry, it is inconceivable for us to use conventional serial links. With conventional links, we estimate that each crossbar chip would generate approximately 17W.

[1] For brevity, we will refer to the combination of crossbar chips and scheduler as the "core" chips.

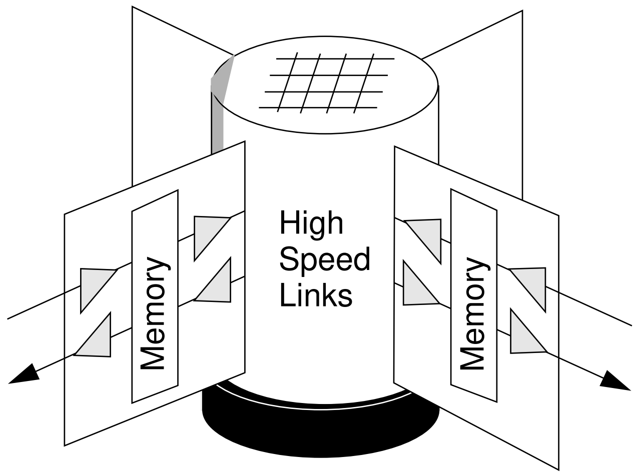

Figure 1 : The Tiny Tera Switch Core

For this reason, we have developed a new skew-compensating link based on the following two observations. First we observe that although the core chips require 32 serial links to connect to the switch ports, each port chip requires only one serial link to connect to the core. Our second observation is that the communication is bidirectional — the serial link consists of two unidirectional links, one for each direction. We can exploit this structure, and move some of the phase adjustment circuitry from the core chips to the port chips, where power generation is less of a concern, and cooling is simpler.

Conventional serial links with receiver timing recovery would require 32 phase adjusters on the core chips. Our asymmetric serial link separates the two ends of the link into a “smart” end, and a “dumb” end, with all the phase adjustment circuit at the “smart” end. The core chips only implement the “dumb” end of the links, and do not perform any clock adjustment.

In the rest of this paper we describe the design of our serial link in detail, with particular emphasis on its application to the Tiny Tera. But we note that the same technique can be applied to many communication systems with a similar structure. In particular, those consisting of a central core with multiple bidirectional links to communicate with a number of peripheral components.

II. LINK ARCHITECTURE

A. Asymmetric Serial Link Architecture

Figure 2 shows a 16×16 crossbar chip illustrating the conceptual difference between conventional serial links and asymmetric serial links, with each phase adjuster shown as a black square. With conventional symmetric links, there are 16 phase adjusters; one at the receiver end of each link. With asymmetric serial links, all 16 of these phase adjusters are moved to the “smart” end of the links. The crossbar has only one PLL for all 16 interfaces which aligns the transmitter and receiver clocks to an external reference clock. Of course, the total power dissipation of the system remains approximately the same, but the dissipation of the crossbar chip is reduced.

Figure 3 shows the asymmetric serial link architecture when used to connect a port chip to the crossbar chip. On the dumb end, all 32 transmitters and receivers share the same clocks, synchronized to an external reference clock by a PLL. In contrast, on the smart end, a dual-loop PLL with digitally controlled phase adjusters [4] generates phase-locked transmitter and receiver clocks. The link transmits a fixed repeating frame structure, consisting of multiple bytes. Framing logic at the dumb end checks the status of bit, byte, and frame synchronization of the smart transmitter link, and sends it to the smart end through the smart receiver link. Similar framing logic in the smart receiver checks

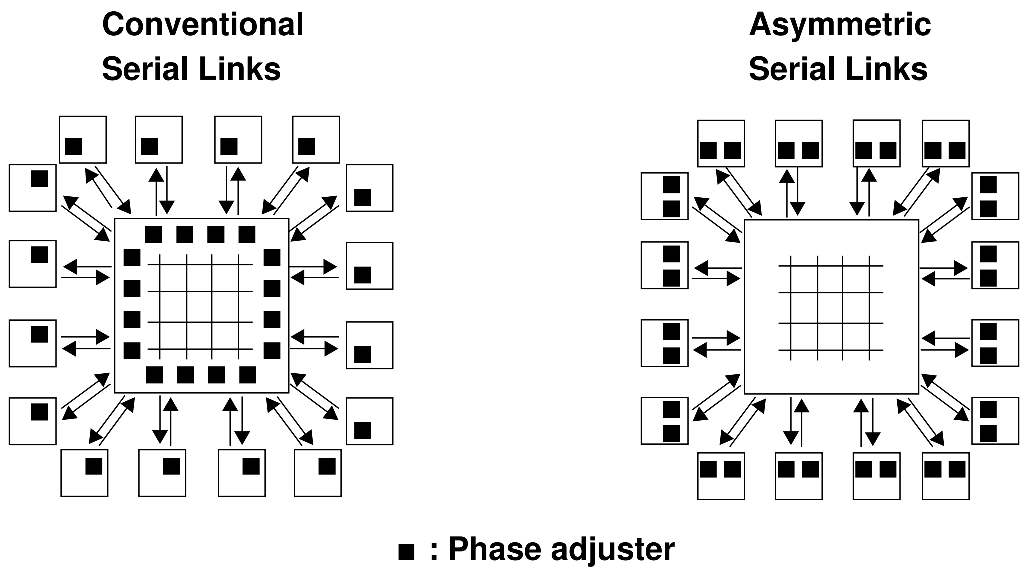

Figure 2 : Difference between the conventional and asymmetric serial links.

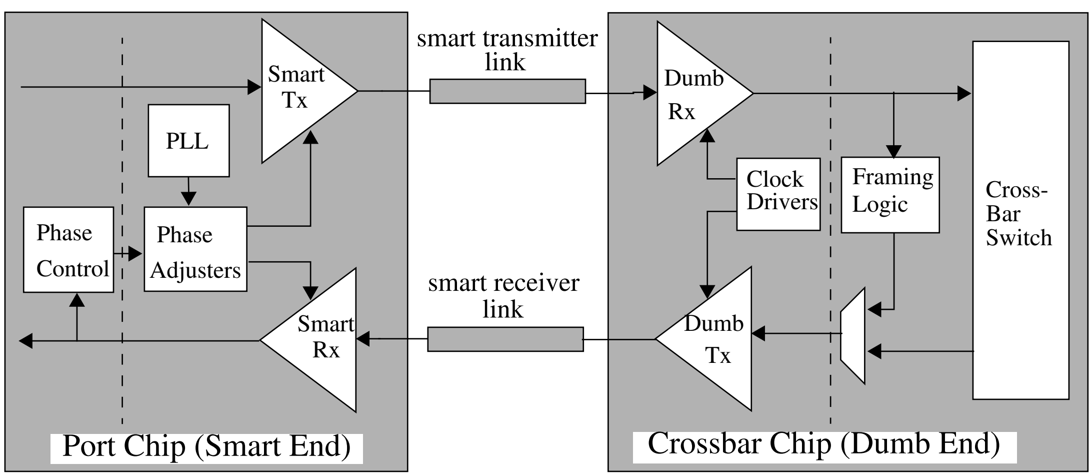

Figure 3 : Asymmetric Serial Link Architecture.

synchronization status of the smart receiver link. The smart receiver sends the synchronization status of both links to the phase control logic in the PLL.

The smart transmitter link can only feed back its synchronization status to the smart end while the smart receiver link is not transferring regular data. Therefore, the asymmetric link is only calibrated periodically. The periodic calibration compensates for the cable delay, and any offset in the system clock distribution. The timing drift resulting from temperature variations and low frequency supply noise is taken care of by a PLL that adjusts the internal clocks to the external reference. During calibration times, the smart and dumb transmitter transmit calibration frames containing special patterns for bit, byte, and frame synchronization. We will discuss this process in more detail in Section III-A. A bang-bang phase control loop is controlled by a majority vote over multiple edges in each calibration frame. The majority vote reduces phase noise caused by random noise and high frequency clock jitter.

Clock phases are only adjusted once per calibration period; each phase adjustment is approximately 1/78 of a bit time, or 7.5 ps at 2 Gb/s. When the phase control loop is in lock, the clock phase toggles between two adjacent phases. Since the loop delay is less than the calibration period, the phase control loop is stable. Because each phase step is a small fraction of a bit time and smaller than worst case high frequency jitter, the toggling does not degrade performance significantly.

Although the link uses a fixed framing structure, it imposes no requirement on the phase relationship of the frames between two links. When a “dumb” end receives a calibration frame, it stores the status until it is instructed to send its own calibration frame. At that time, it inserts its status into the calibration frame and sends it to the “smart” end.

Our design uses both edges of the *half-bit-rate clock*. At this rate, only very simple logic functions can be performed. Therefore, all the serial bits are converted into parallel bytes for complex digital logic, including phase control, framing, and chip core logic. The digital logic uses the *byte clock*, one fourth the *half-bit-rate clock* rate. Data is processed on a frame basis. Any reasonable frame size can be used; in our design for the Tiny Tera, we use a nine byte frame.

The dashed line in Figure 3 separates the clock domains of the *half-bit-rate clock* and the *byte clock.* The entire serial link, including the serial/parallel converters, can be thought of as a black box equivalent to a byte-wide parallel link.

B. Same Receiver for Timing Acquisition and Data Reception

For a serial link, the timing information is directly recovered from the received data stream. Figure 4 shows a

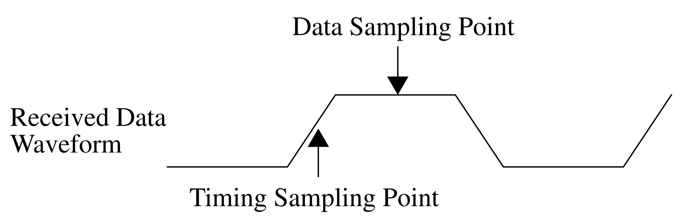

Figure 4 Data and Timing Sampling Points

received data waveform with ideal data and timing sampling points. The ideal sampling time is in the center of a bit. By fixing the distance between these two sampling points at half a bit time (90^o of a *half-bit-rate clock* period), we can determine whether the data is sampled earlier or later than the ideal point. For example, the receiver clock is early when the timing sample is zero for a zero-to-one transition.

Two identical receivers driven by two different clocks with 90^o phase shift can be used for separately sampling data and timing [1]. However, because of phase errors resulting from receiver and clock buffer mismatches, it is difficult to ensure that the data and timing sampling points are exactly 90^o apart. The mismatch means that the data sampling point would not be in the center of the bit time.

To counter this problem we use the same receiver for both data and timing acquisition, completely eliminating phase errors due to receiver 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 smart transmitter and the smart receiver clock during calibration. In particular, the phase of the smart transmitter is shifted by 90^o of a *half-bit-rate-clock* cycle (i.e. half a bit period).

III. LINK SYNCHRONIZATION

There are three stages for the synchronization of the asymmetric serial links: PLL lock-in, bootup, and periodic calibration. In the first stage, the analog PLL locks to the external reference clocks, as discussed in Section IV-A. 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 are periodically calibrated.

In this section, we first present the calibration sequence used during the bootup stage, and in normal operation. We will then discuss the bootup sequence and periodic calibration in more detail.

A. Calibration Sequence

The calibration sequence, shown in Figure 5, is used in both bootup and periodic calibration. The first byte of the

byte/frame sync **bit sync (*timing byte*)**

<*ctrl byte*> 11001111 00001100 01010101 01010101 01010101 01010101 01010101 01010101

calibrate, sync status, reserved **clocks shifted by 90^{o}**

Figure 5 Calibration Sequence

sequence 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 zero for real data transmission. The synchronization status includes the state of bit, byte, and frame synchronization, the clock early or late information, and some reserved bits for system use. The second and third bytes of the calibration sequence 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 reasonable clock uncertainty. The rest of the sequence alternates between 1 and 0 and is used for finely adjusting the clock phase to the center of the data eye. The clock is shifted forward by 90^{o} during the fourth byte to transmit or receive timing information and shifted back during the last byte. Each clock shift corrupts one byte of data so only four bytes, called *timing bytes*, are useful.

B. Bootup

During bootup, both the smart receiver and the smart transmitter links continuously calibrate the links to compensate for the cable delay and system clock skew until byte, frame and bit synchronization is achieved. The smart receiver link is synchronized first; the smart transmitter link does not start synchronization until valid data are fed back over the smart receiver link. Both links follow a similar synchronization procedure and use the same calibration sequence.

The byte boundaries of the serial/parallel converters move with the *half-bit-rate-clock* at the smart end. Therefore, rotating the *half-bit-rate clock* by small steps over several full cycles can move the byte boundary several bit times. Thus, to synchronize the links, we rotate the clock by one phase at a time in one direction until the byte/frame pattern is detected at a byte boundary. Since the two bytes used for byte synchronization only occur in the second and third bytes of each calibration frame, the frame and byte boundaries are found simultaneously, i.e., we only have to rotate the clock through one byte at most. After the bytes and frames are synchronized, we start synchronizing the bits.

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 only be in byte synchronization after four consecutive byte matches occur. Once in byte synchronization, it will also take four consecutive mismatches to push the link out of byte synchronization, to prevent a few bit errors from causing a loss of byte synchronization and further bit errors.

After byte synchronization and frame synchronization are complete, the clocks are fine-tuned to achieve bit synchronization. This ensures that the data sampling point is at the center of the bit time. During bit synchronization, the clock phase moves in the same direction until it is close to the center of the bit time. The sampling point is moved until the first time that the bang-bang phase control loop changes the direction of clock movement. At this point, we conclude that the edge has been located, completing the bit synchronization process.

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

C. *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, they will change to the bootup mode and resynchronize the bytes, frames, and bits of the links. Otherwise, if the links remain byte/frame synchronized, the bang-bang phase control logic moves the clock phase to keep the clock at the center of the bit time. Each calibration frame only changes the clock by one phase based on a majority vote of the early/late signals generated from the *timing bytes.*

IV. IMPLEMENTATION

A. *Clock Generation*

The clock generation circuit is a dual-loop PLL with digitally controlled phase adjusters [4]. The inner loop is a four-stage VCO-based PLL. The VCO uses differential elements with symmetric loads and replica biasing [2] to generate eight clock phases, which are 45^{o} of a *half-bit-rate clock* cycle apart. Each phase adjuster selects two adjacent clock phases and uses a digitally-controlled interpolator to generate one of 17 finer phases between them. As a result, each *half-bit-rate clock* cycle encompasses $8 \times 17 = 136$ possible phases, for a phase resolution of 7.5 ps at 1 GHz.

One of the phase adjusters drives a divide-by-four circuit and generates the *byte clock*, which is then phase-locked to the reference clock. The remaining phase adjusters are for the transmitter, the receiver, and the serial/parallel converters.

The transmitter clocks are preskewed by the transmitter delay relative to the *byte clock* after reset and thus the transmitter output is phase locked to the reference clock. On the other hand, the receiver clock is directly aligned to the *byte clock* after reset and therefore the receiver sampling point is phase locked to the reference clock. As a result, even after the phase control loop adjusts the clock phases of the transmitter or receiver clock, the clock generation circuit still keeps the transmitter output and the receiver sampling point phase locked to the external reference. This alleviates the need to track clock jitter resulting from temperature variations and low-frequency supply noise.

B. *Signaling and Termination*

The high-speed serial links use differential signalling with swings from Vdd to Vdd-0.5V. Signal lines are on-chip terminated at the receiver end using unsilicided polysilicon pullup resistors tied to Vdd. Fully differential receivers and current-mode drivers are used for supply noise immunity. The low 50 Ω line impedance mitigates the effect of the large parasitic capacitances due to the package, the ESD protection devices, and the large transmitter transistors (receiver input capacitance is relatively small).

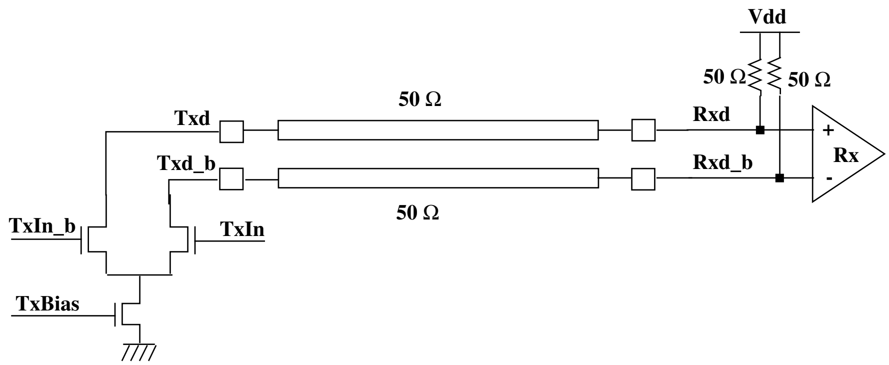

Figure 6 : Link Transmission Block Diagram

C. Transmitter

The Tiny Tera link transmitter must drive a low-impedance line and significant parasitic capacitances at GigaHertz rates. A current-mode driver with open-drain outputs accomplishes this. As shown in Figure 6, a differential pair switches current to either the true or inverted output. The current source in the differential pair keeps output current constant over process, temperature, and on-chip ground variations, but requires somewhat larger output transistors due to their reduced Vgs.

One drawback of current-mode transmitters is their limited output swing. With the output transistor gates driven to Vdd, the output can only drop to Vdd-Vt before the linear region is entered and the output resistance drops quickly. However, differential receivers have been shown to operate below 10^{-13} BER with as little as 200 mV of input [5], so larger swing is not needed.

D. Receivers

The receiver must achieve a very low bit error rate on a low swing, high bandwidth signal in the presence of supply noise and reflections. The use of fully differential circuitry provides excellent supply noise rejection, and the use of integrating receivers [5] rejects high frequency reflections and other noise. Two receivers operate on alternate bits in parallel, synchronized by the *half-bit-rate clock*.

V. SIMULATION RESULTS

A test chip has been designed and is being fabricated with Texas Instruments' 0.25μm CMOS technology. From HSPICE simulations, the peak-to-peak jitter of the transmitter output and the receiver clock is 150ps at slow process with a 250mV supply step. Since both data and clock edges should be centered at the mean of the clock phases, the data eye at 2 Gb/s with clock jitter is 200ps (500ps - 2X150ps), which is sufficiently large for receiver offset and the amplitude noise.

Table I shows the comparison of power consumption of the crossbar for conventional serial links and asymmetric

Table I: Power Consumption Comparison for Conventional and Asymmetric Links

	Conv (W)	Asym (W)
PLL	0.22X32	0.22
Phase Adjusters	0.05X32=1.6	0.05
Phase Control & Framing	0.125X32=4	0
Clock Drivers	1.6	1.2
Transmitters & Receivers	1.5	1.5
Crossbar	1.5	1.5
Total	17.2	4.5

serial links. The major power saving comes from the phase adjusters, the phase control and framing logic, and some from the clock drivers. They combine to save 12.7 W for the crossbar chip, reducing power by nearly a factor of 4.

The framing detection logic on the dumb end of the asymmetric link is only enabled during calibration frames to save power. Therefore, the framing logic consumes negligible power when the calibration frequency is reasonably low.

VI: SUMMARY

We have described a 2 Gb/s CMOS serial link for chip-to-chip communications. Our asymmetric link is designed for the following types of system:

- Systems in which a central "core" chip with multiple serial links is connected to a number of peripheral chips, each with a small number of links.
- Systems in which communications between chips is required for both directions.

For such systems, our design suggests up to 75% of the power can be moved away from the core chip(s), simplifying cooling, and reducing noise for other logic. This is achieved by moving phase adjusters from the core chips, to the peripheral chips. Periodic calibration is used to maintain synchronization at the bit- and byte-levels, as well as maintain a fixed repeating frame structure.

A number of techniques are used to maintain synchronization. A PLL-based clock generator with digitally-controlled phase adjusters tracks the timing drift resulting from temperature variations and low frequency supply noise. The phase is adjusted periodically during calibration, compensating for the cable delay and system clock skew. A bang-bang phase control loop moves the clock phase based on the result of a majority vote of many edges. This reduces phase errors introduced by high-frequency clock jitter. The same receiver is used for both data and timing acquisition while the clock is shifted by 90^o to acquire timing information. This eliminates any phase errors resulting from receiver and clock buffer mismatches.

To reduce amplitude noise, differential signalling open-drain drivers with receiver end termination are designed to achieve fast data transmission with low-swing and a fast edge rate without exciting large dI/dt noise. Furthermore, current integrating receivers are used to increase noise immunity.

Our link is designed for use in the Tiny Tera packet switch. We anticipate that our links will enable the system to operate at an aggregate bandwidth above 0.5 Terabit/sec, using small connectors and currently available CMOS technology.

Laboratory results are expected in September, 1997.

ACKNOWLEDGEMENT

We would like to thank Jeff Hsieh at Stanford for his help with simulation and layout of the test chip, and Ah-Lyan Lee at Texas Instruments for his help checking design rules. We also thank Ken Yang at Stanford for helpful discussions on PLL design.

REFERENCES

[1] A. Fiedler, *et al*, "A 1.0625Gb/s Transceiver with 2x-Oversampling and Transmit Signal Pre-Emphasis," ISSCC Digest of Technical Papers, pp.238-239, Feb. 1997.

[2] J. Maneatis, "Low-Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques," IEEE Journal of Solid State Circuits, vol. 31, no. 11, Nov. 1996.

[3] Nick McKeown, *et. al*. "Tiny Tera: A Packet Switch Core," IEEE Micro, Jan/Feb. 1997.

[4] S. Sidiropoulos, Mark Horowitz, "A Semi-Digital DLL with Unlimited Phase Shift Capability and 0.08-400MHz Operating Range," ISSCC Digest of Technical Papers, pp.332-333, Feb. 1997.

[5] S. Sidiropoulos, Mark Horowitz, "A 700 Mbps/pin CMOS Signalling Interface Using Current Integrating Receivers," IEEE Journal of Solid State Circuits, vol. 32, no. 5, May 1997.

[6] A. Widmer, *et. al*., "Single-Chip 4X500 Mbaud CMOS Transceiver," IEEE Journal of Solid State Circuits, vol. 31, no. 12, Dec. 1996

[7] K. Yang, *et. al*, "A 0.6µm CMOS 4Gb/s Transceiver with Data Recovery using Oversampling," IEEE Symposium on VLSI circuits, pp 71-72, June 1997.