A 8x5 Gb/s Source-Synchronous Receiver with Clock Generator Phase Error Correction

Ankur Agrawal, Pavan Kumar Hanumolu* and Gu-Yeon Wei
Harvard University, Cambridge, MA 02138, *Oregon State University, Corvallis, OR 97331

Abstract— **This paper describes the design and implementation of a 8x5Gb/s source-synchronous receiver in a 0.13μm CMOS technology. The receiver employs a cascaded-DLL architecture that avoids filtering of the jitter on the received clock to enhance jitter tolerance bandwidth. A technique is proposed to correct phase spacing mimatch in DLLs that reduces the error standard deviations by more than 40% and improves receiver timing margins.**

I. Introduction

The need for high I/O bandwidth in multi-chip digital systems has led to the widespread use of parallel links. These links are generally source synchronous, with a clock sent along with the data signals for receiver timing recovery. As data rates increase, successful data recovery in the presence of jitter requires precise positioning of the sampling clock. Receivers need to perform per-pin skew compensation [1] while preserving the correlation in the jitter between the transmitted clock and data.

Source-synchronous receivers often use multi-phase clock generators to drive phase interpolators [2]. Multiple clock phases are also required when interleaved samplers are employed to easily accomodate high off-chip data-rates. Phase locked loops (PLL) or delay locked loops (DLL) can be used to generate multi-phase clocks. While the phase filtering action of a PLL reduces the jitter correlation between the incoming clock and data, DLLs are susceptible to systematic and random phase offsets and mismatch that can significantly reduce timing margins and degrade achievable data rates. If these phase errors can be corrected, DLLs are a better choice than PLLs for multi-phase clock generation in source-synchronous receivers.

This paper presents a cascaded-DLL architecture for receivers that avoids any phase filtering in the path of the received clock and incorporates techniques to correct for phase spacing errors in DLLs. It requires neither phase interpolators nor the distribution of multi-phase clocks over long on-chip wires. The next section describes the trade-offs between using DLLs and PLLs in source-synchronous receivers.

II. Source Synchronous Receiver Design Considerations

Fig. 1 shows the architecture of a general source-synchronous transceiver. The transmitter sends a parallel word of data along with a clock to the receiver. To save clock power and avoid jitter amplification, often the frequency of the transmitted clock is stepped down and a multiplying PLL or DLL is used in the receiver to step the frequency back up. This clock is then distributed to each of the receiver slices using either clock buffers or passive distribution [2], [3]. The receiver slices need to perform skew compensation to correct for flight time variations over the PCB traces. If multiple data bits are transmitted in each RxClk cycle, the receivers also need to generate multiple clock phases to sample the incoming data.

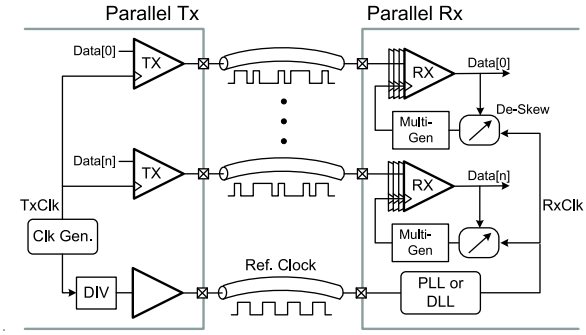

Fig. 1. General Source Synchronous Transceiver

A single PLL can be used for multi-phase clock generation and clock de-skew [4]. Alternatively, a combination of a DLL and a phase interpolator can be used to perform the two tasks independent of each other. PLLs have a low-pass transfer function from the phase of the reference clock to the output clock and, thus, filter out the middle and higher frequency jitter on the received clock. On the other hand, DLLs have a nearly all-pass phase transfer function, and are able to preserve the correlation between the jitter on the incoming data and received clock, resulting in good jitter tolerance over a wide frequency range. Recently reported designs [2], [5] have avoided the use of PLLs in the path of the received clock to achieve wide jitter tracking bandwidth.

Our test chip, the details of which shall be discussed in the following sections, enables a direct comparison between the jitter tracking bandwidths of PLL and DLL based timing recovery. Fig. 2 plots the jitter tolerance curves for BER $< 10^{-9}$ for 2 different configurations of the test chip. The "DLL-only" case is the typical configuration of the test chip (as described in Section III), where a quarter rate clock is directly fed to the local recievers. In the "PLL/DLL" case, a sub-rate clock is first multiplied on-chip using a PLL to the desired frequency. The jitter tolerance bandwidth for the DLL-only case is >100 MHz and is limited by the difference in the on-chip path length between the data and clock signals. In

978-1-4244-2018-6/08/$25.00 ©2008 IEEE

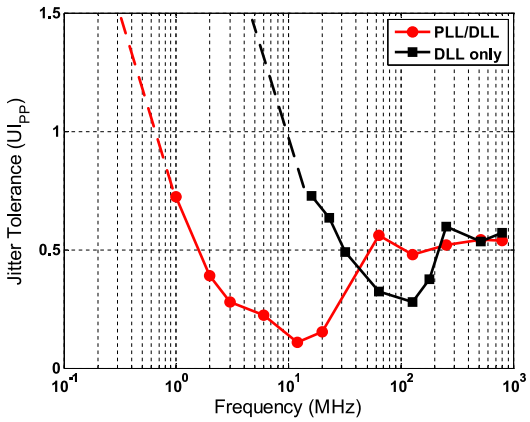

Fig. 2. Jitter Tolerance Plots for the DLL only case and PLL/DLL case measured at 3.2 Gb/s.

constrast, a jitter tolerance bandwidth of around 10 MHz is obtained for the PLL/DLL case. The PLL bandwidth limits the maximum achievable jitter tolerance bandwidth in this case. Due to test equipment limitations, a maximum jitter of 0.72 UI can be introduced. These measurements confirm that phase filtering should be avoided in the path of the received clock in source-synchronous receivers to improve jitter tracking.

However, there are some caveats to using DLLs: (1) A small difference between the charge pump up and down currents can result in static phase spacing error at the end of the delay line in a DLL, while only causing a slight increase in clock jitter in a PLL. (2) DLLs that lock the delay of the delay line to half a reference clock cycle are very sensitive to the duty cycle of the reference clock signal. (3) The shape of the reference clock entering the delay line can exacerbate phase spacing mismatches in DLLs. By carefully addressing each of these issues, DLL-based timing recovery can be made better suited to high-speed source synchronous receivers.

The first two issues listed above are well known and solutions have been proposed in literature [6]. The last one is more subtle. The shape of the clock, characterized by its voltage swing and the slope of its edges, entering the first delay cell can be different from those entering subsequent stages. This results in adjacent delay stages having different delays. Preshaping the clock signal by adding identical delay cells before the delay line helps, but does not eliminate the problem. The optimal clock shape is one that would be produced by a voltage-controlled oscillator composed of the same delay cells and having the same control voltage as the delay line. Fig. 3 shows the simulated nonlinearity in the phase spacings for a DLL consisting of a 4 stage differential delay line after preshaping using an identical delay line. It also shows the simulated non-linearity achieved after a technique to shape the reference clock signal is used. This technique, which we call phase spacing error correction (PSEC), is described in greater detail in Section IV. In the next section the receiver architecture is described that is able to easily incorporate techniques to correct for the phase spacing errors mentioned above.

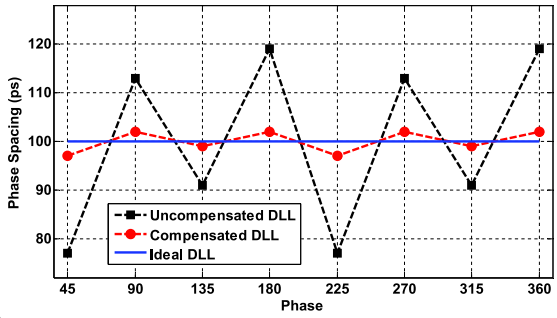

Fig. 3. Phase-spacing error caused by the shape of reference clock.

III. System Architecture

Fig. 4 presents details of the receiver architecture. A quarter rate clock is boosted to on-chip voltage levels and distributed to individual receiver slices using simple clock buffers. Each receiver slice consists of 2 DLLs, samplers, skew-compensation logic and phase spacing error correction circuits. The global RxClk first goes through a duty cycle corrector (DCC) to compensate for any distortion from the clock distribution buffers. The first DLL, called the phase-deskew DLL, adds a variable amount of delay to the clock path such that its output clock is phase aligned to the incoming data. The second DLL produces the 8 clock phases that drive 8 interleaved edge and data samplers. The phase detector generates early/late information that is forwarded to a digital filter. The digital filter controls a DAC that adds offsets between the up and down currents of the charge pump of the phase-deskew DLL that results in skewing in output clock. The tuning range of the de-skew circuit is greater than $\pm$0.5 UI. Also shown in the figure is the phase spacing error correction loop that balances the delays of the delay cells by adjusting the slope of the clock edges. Additional details of the receiver slice can be found in [7].

The cascaded DLL architecture is chosen for a number of reasons. It provides a simple mechanism to de-skew the global RxClk. No extra preshaping delay buffers are required in the clock path before the DLL used for multiphase clock generation. It also provides a convenient location for placing the duty cycle correction circuit. The duty cycle of the clock entering the second DLL is sensed and tuned to 50%, but the correction circuit is placed before the first DLL in order to not disturb the shape of the clock signal for the second DLL. The offsets in the phase-deskew DLL do not matter as they get absorbed into the overall delay of its delay line.

IV. Circuit Design

A. DLL

Fig. 5(a) shows the details of the DLL. The use of an active loop filter offers two advantages. It provides a 2^{nd}-order low pass transfer function from I_{CP} to Φ_{out} that is able to filter out the quantization noise in the control bits to the DAC. The feedback amplifier also biases the output node of the charge pump to V_{REF}, irrespective of the delay of the delay line. By

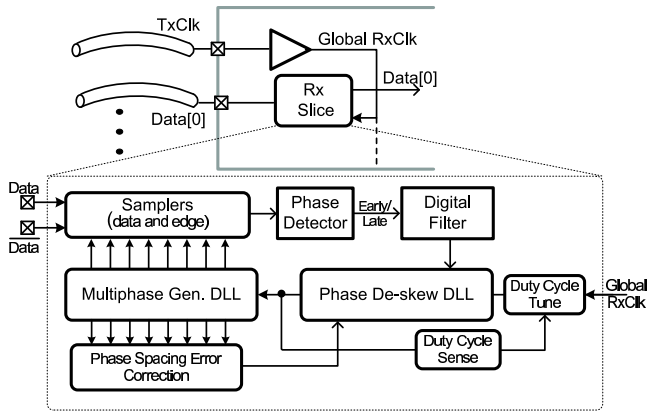

Fig. 4. Proposed receiver block diagram

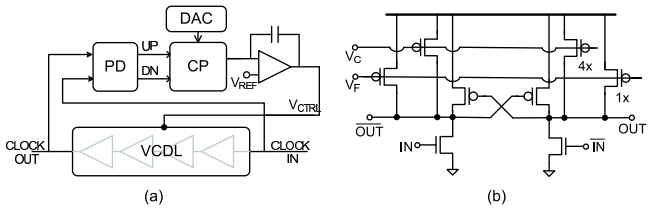

Fig. 5. (a) DLL block diagram, (b) Delay cell schematic

adjusting V_{REF}, small mismatches in the charge pump up and down currents can be compensated. The delay cell, that uses pseudo differential inverters with rail-to-rail swing, is shown in Fig. 5(b). An external coarse voltage (V_C) is used to bring the delay line to within the lock range of the DLL.

B. Phase Spacing Error Corrector (PSEC)

As described in Section II, the shape of the clock signal entering the DLL affects the delays through the delay cells. To correct for this effect, the rise and fall times of the input clock to the second DLL are adjusted by sinking or sourcing small amounts of current from the penultimate delay stage of the first DLL. A feed-back loop, shown in Fig. 6, consists of XOR based delay detectors, charge pump, loop filter and a V/I converter. As the error is such that all odd delay stages are either faster or slower than all even delay stages, the charge pump dumps or removes charge proportional to the difference between the odd and even delays. Multiple error detectors are used to average out the effects of random mismatches between the adjacent stages. The charge pump employs feedback biasing to ensure that static and dynamic mismatches in the up and down currents do not introduce offsets that can result in residual delay mismatches. The bandwidth of this loop is set such that there is no conflict with the de-skew loop.

Simulation results (Fig. 3) indicate that the RMS DNL reduces from 16.9 ps to 2.1 ps. The efficacy of this method is limited by random mismatches in the delay cells and XOR gates.

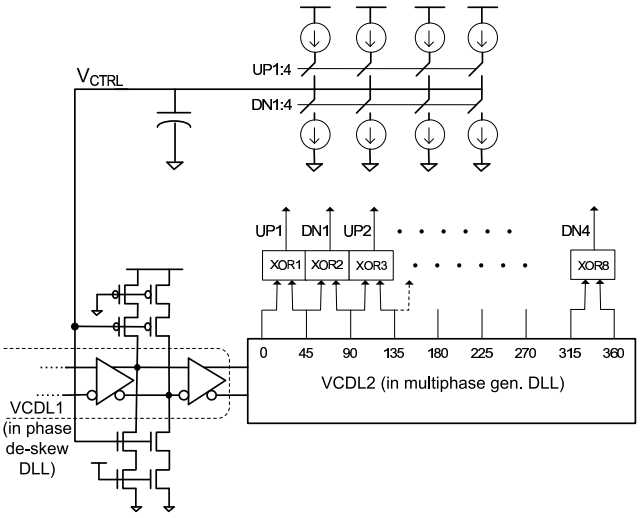

Fig. 6. Phase Spacing Error Corrector Loop

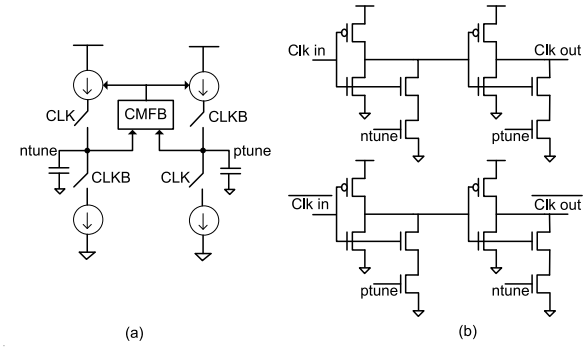

Fig. 7. (a) Duty-cycle sensor (DC_sense), (b) Duty-cycle corrector (DC_tune)

C. Duty Cycle Corrector

The duty cycle corrector (DCC) shown in Fig. 7, consists of a duty cycle sensing block (DC_sense) placed before the second DLL, and a duty cycle tuning block (DC_tune) placed before the first DLL.

The DC_sense circuit is a differential charge-pump that produces two voltages, $ntune$ and $ptune$ depending on the magnitude of duty cycle distortion in the clock signal. Common-mode feedback ensures $ntune$ and $ptune$ are differntial with a common-mode of $V_{DD}/2$. The DC_tune circuit, similar to that in [6], uses these voltages to slow down or speed up one of the clock edges. It consists of 2 tuning stages, where each stage consists of an inverter with weak pull-down and a parallel pull-down path whose strength depends on $ntune$ or $ptune$. Consequently, each stage can adjust the falling transition of the clock signal. Simulation results show that the circuit suppresses duty cycle distortion by more than 10x.

V. Experimental Results

A test chip fabricated in a 0.13μm CMOS logic process operates off a 1.2V supply voltage. To verify the efficacy of the peripheral correction loops, Fig. 8 plots the measured

phase spacings with the PSEC loop turned on and off for 2 representative DLLs in our parallel receiver. The reference clock frequency is 1.25GHz and the nominal delay of each stage is 100ps. A reduction in the differential nonlinearity (DNL) is observed. The residual error can be attributed to random delay cell mismatches that our correction loops cannot correct. In Fig. 9 we plot the rms value (standard deviation) of the DNL in phase spacings for 7 DLLs from 7 receiver slices, with the PSEC loop enabled and disabled. The 8th channel is inaccessible due to insufficient connector spacing on the board. On average, the RMS DNL reduces by 42%.

Fig. 10 presents the integral nonlinearity (INL) in the phase spacings with the DCC loop turned on and off. INL is defined as the deviation from the ideal location of the clock edge. We observe a reduction in the INL in phase spacings from 25.8ps to 7.8ps. The dotted line shows the theoretical INL for a 10% duty cycle error. Phase spacing measurements were made by sweeping a 1010... data pattern at 5 Gb/s across half a reference clock cycle and observing sampler outputs.

The 8x5Gb/s receiver consumes a total power of 258mW (including all output drivers), which translates to an efficiency better than 6.45mW/Gbps. Each receiver slice occupies an area of 350μm x 450μm. A die micrograph is shown in Fig. 11 with floorplan overlays.

VI. Conclusion

This paper describes a 8x5Gb/s source-synchronous receiver that avoids any phase filtering in the path of the received clock. Although source-synchronous receivers prefer DLLs over PLLs to enhance jitter tolerance bandwidth, DLL based receivers can suffer from timing margin degradation due to phase spacing mismatches. Techniques to reclaim the lost timing margin are proposed and experimentally verified.

VII. Acknowledgments

The authors would like to thank UMC for chip fabrication.

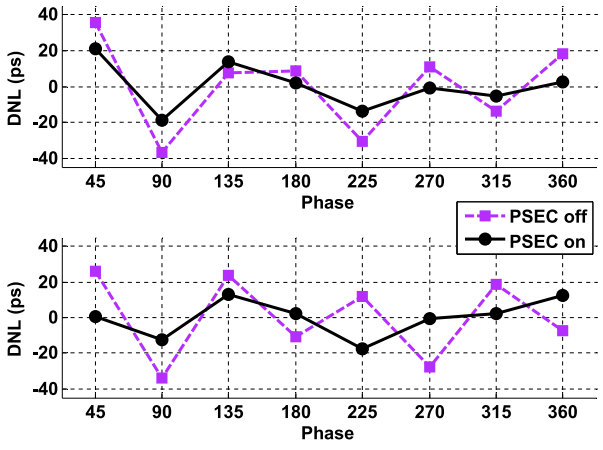

Fig. 8. Differential non-linearity in phase spacings with the PSEC loop disabled and enabled for 2 representative DLLs

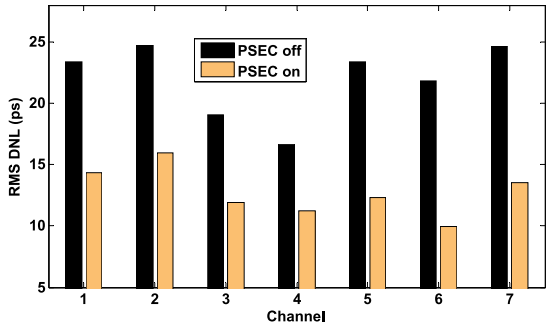

Fig. 9. Comparison of the RMS DNLs for 7 DLLs with the PSEC loop disabled and enabled

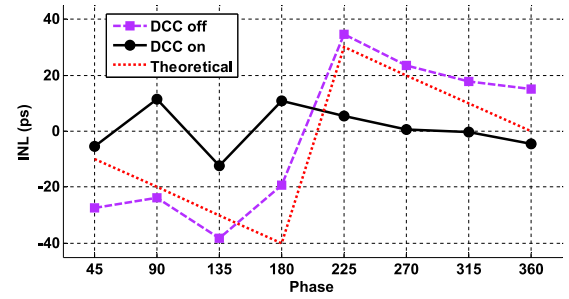

Fig. 10. Integral non-linearity for the phase spacings with the duty-cycle correction loop disabled and enabled

References

[1] E. Yeung and MA Horowitz, "A 2.4 Gb/s/pin simultaneous bidirectional parallel link with per-pin skew compensation," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 11, pp. 1619–1628, 2000.

[2] B. Casper et al., "A 20Gb/s Forward Clock Transceiver in 90nm CMOS," *ISSCC Dig. Tech. Papers*, pp. 90–1, 2006.

[3] K. Chang, S. Pamarti, K. Kaviani, E. Alon, X. Shi, TJ Chin, J. Shen, G. Yip, C. Madden, R. Schmitt, et al., "Clocking and circuit design for a parallel I/O on a first-generation CELL processor," *ISSCC Dig. Tech. Papers*, pp. 526–615, 2005.

[4] P. Larsson, "A 2-1600-MHz CMOS clock recovery PLL with low-Vdd capability," *Solid-State Circuits, IEEE Journal of*, vol. 34, no. 12, pp. 1951–1960, 1999.

[5] F. O'Mahony et al., "A 27Gb/s Forwarded-Clock I/O Receiver using an Injection-Locked in 90nm CMOS," *ISSCC Dig. Tech. Papers*, pp. 452–453, 2008.

[6] A.H. Tan and G-Y. Wei, "Phase Mismatch Detection and Compensation for PLL/DLL Based Multi-Phase Clock Generator," *Conference 2006, IEEE Custom Integrated Circuits*, pp. 417–420, 2006.

[7] A. Agrawal, P.K. Hanumolu, and G-Y. Wei, "A 8x3.2Gb/s Parallel Receiver with Collaborative Timing Recovery," *ISSCC Dig. Tech. Papers*, pp. 468–469, 2008.

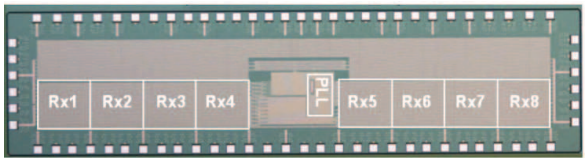

Fig. 11. Chip micrograph with floorplan overlays