Burst Mode Packet Receiver using a Second Order DLL

Haechang Lee, Chi Ho Yue, Samuel Palermo, Kenneth W. Mai, and Mark Horowitz

Center for Integrated Systems, Stanford University
420 Via Palou,
Stanford, CA 94305-4070, U.S.A

Abstract

This paper describes a CDR that can be used to receive optically switched packets. Rather than using fast phase acquisition to lock onto each packet, it uses a second order delay locked loop to acquire both the frequency and phase of each source to predict future bit transitions. A 0.25μm CMOS prototype can track frequency offsets of 100ppm to better than 0.1ppm and can retain lock on 10Kbit 3.125Gbps packets that occur once every 2.4Mbits.
Keywords: burst mode packet receiver, second order delay locked loop, clock and data recovery and CMOS

Introduction

An increasing number of applications are using optical switching to share the high optical bandwidth among a number of clients. One key application is the uplink in fiber-to-the-home (FTTH) systems. Here, the uplink is time division multiplexed (TDM) and each home transceiver is given a time slice to use. The receiver at the head end needs to recover the high-speed data from each home (Fig. 1). Similar situations arise in systems that use true optical routing[1]. In such systems, consecutive packets arrive from different transmitters (TX) each with different phase and frequency offsets with respect to the receiver (RX). Since the clock and data recovery (CDR) block must reacquire phase lock for each packet, the lock time becomes extremely important as it reduces the effective data rate.

Previous approaches to solve this problem attempted to build receivers with zero lock time. Yang accomplished this goal by building a receiver that roughly 3x oversamples the data[2]. After sampling, the system stores the samples and uses a non-causal filter to decide which sample has the most timing margin. While this approach works, the high sampling rate and significant digital processing required make this a higher power solution. In addition, the timing margins are set by the oversampling rate which results in a power/performance trade-off. Banu used a gated oscillator that is enabled with the arrival of each data edge[3]. Using a single data edge provides fast phase acquisition, but since this system performs no jitter filtering on the data and accumulates jitter over long run lengths, it has degraded timing margins.

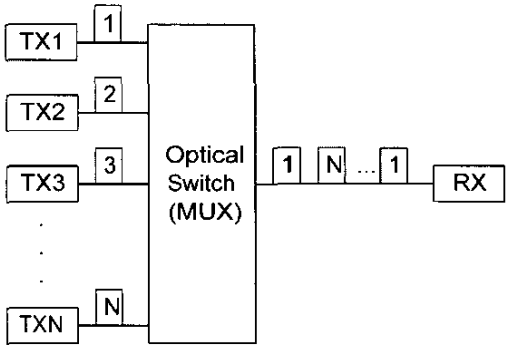

Fig. 1 An example application of this burst mode packet receiver: uplink of FTTH. Each TX is located in separate homes with its own clock source.

The CDR described in this paper uses past information to predict the phase position of future packets rather than starting from scratch each time a new packet arrives. More specifically, the receiver acquires the phase and frequency information of a given TX and uses it to estimate the timing of the next packet arriving from that TX. While this requires the receiver to know which source is transmitting a given packet, this information is usually available in these systems. Clearly, very accurate frequency estimation is needed if the time between two packets arriving at a RX from the same TX can be extremely long (hundreds of thousands of bits). This estimation is accomplished by using a second order DLL.

Second Order Delay Locked Loop

A second order dual loop DLL capable of tracking frequency offsets between the TX and RX with high precision is shown in Fig. 2. A dual loop architecture, consisting of a core loop (frequency synthesizer or multiphase generator) and a peripheral loop (CDR), allows the designer to set the bandwidth in the two loops independently so as to minimize jitter in the system. In general, the core loop has very wide bandwidth to correct noise in its VCO or VCDL while the CDR bandwidth is set low to filter jitter on the incoming data. Traditional dual loop DLL's are first order systems that only track phase[4]. For this reason, the low phase bandwidth in the CDR limits its ability to track frequency offsets. The second order dual loop DLL improves the traditional design by adding a frequency tracking loop. So far, published second order DLL's were designed for coarse frequency tracking with precision larger than 30ppm [5][6].

The circuitry on the left-hand side of Fig. 2 is a conventional dual loop design. A VCO PLL is used to generate multiple phases that are coarsely spaced. These phases are input to a phase DAC that generates an output phase by interpolating between the two adjacent coarse

 0-7803-8287-0/04/ $20.00 © 2004 IEEE

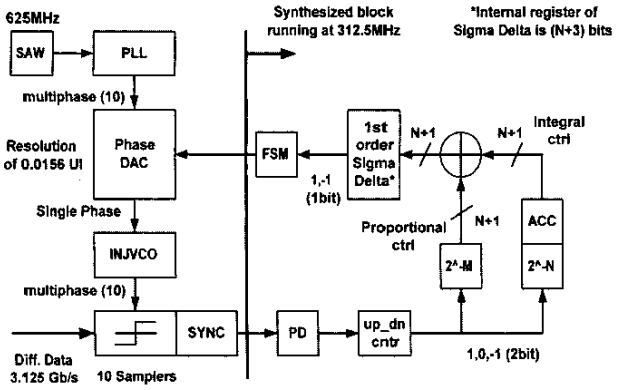

Fig. 2 Block Diagram of CDR with accurate frequency tracking.

phases closest to the desired phase. The digital input from the controller determines the position of the phase DAC output. This single output is used to injection lock another VCO to regenerate multiple phases (if they are desired) that clock the input data and timing samplers. A synchronizer aligns the output of the samplers after which a phase detector (PD) decides whether the recovered clock is early or late.

The right-hand side of Fig. 2 forms the second order filter. It uses the timing error information from PD to estimate both the phase of the input and the frequency offset between the input and the local reference. The phase is stored in the FSM while the frequency is stored in the accumulator (ACC). The proportional and integral gains are implemented respectively by 2^{-M} and 2^{-N}. The DLL's ability to accurately track phase and frequency is limited by quantization noise due to the finite precision of components, limit cycles resulting from the loop delay, jitter in both the multiphase clock that triggers the front-end samplers as well as in the data, and any non-linearity in the phase DAC.

A. Quantization Noise

The phase DAC resolution determines the minimum phase quantization noise of the system. Our prototype's resolution was a 64^{th} of a bit. This phase resolution also sets the maximum frequency drift that can be tracked, since the integral control at best can increment the phase once each sysClk cycle – one tenth the bit rate (312.5MHz) in our prototype. Thus we can rotate through one bit every 640 bits, which is a frequency offset of 1/640 or 1562.5 ppm. Since the frequency offset can be either positive or negative, at least 15 bits is required to achieve a frequency resolution better than 0.1ppm. Essentially, the accumulator contains the fractional phase DAC step that should be added each cycle to track phase drift due to a frequency offset.

We use a first order digital ΣΔ modulator to convert the higher resolution frequency information to the lower resolution input of the FSM. The modulator achieves this by encoding the (N+1) bit information into the duty cycle of its 1bit output.

B. Limit Cycles

The smallest limit cycle occurs when the loop dithers between two adjacent phases that straddle the actual phase position of the data. Any loop delay increases the limit cycle. Thus, the gains in the loop have been limited to exponents of 2 such that multiplication is simply a shift operation that incurs marginal delay. Still, the loop delay exceeds 8 cycles. This will increase both the phase and frequency wander of the CDR. One solution widely employed is a filter that prevents additional phase and frequency updates until the last update has taken effect. It was implemented using a 6 bit up-down counter. When the register reaches ±16, an appropriate up or dn command is issued and the register is reset to zero. With this filter, the dithering is limited to one step.

C. Jitter in TX and RX

Jitter at both the TX and RX add uncertainty to the phase and frequency estimates of the CDR since the input to the second order filter will have errors. Since these errors should be random, their effect on phase error can be minimized by decreasing the phase bandwidth (i.e. increasing M) of the CDR. Furthermore, increasing the resolution of the frequency accumulator (i.e. increasing N) ensures that the uncertainty only corrupts bits that are well below the 15^{th}. For this reason, the system was designed with at least 21bits frequency precision with programmability up to 24.

D. Phase DAC non-linearity

When the DLL is receiving data, differential non-linearity (DNL) of the phase DAC will add to the peak to peak (p2p) jitter distribution by an amount roughly equal to 2 times the maximum step size. Thus the worst-case step is generally chosen to be small compared to the expected jitter. In normal phase tracking loops, integral non-linearity (INL) is not an issue – the control loop never depends on the accuracy of the phase DAC since it is constantly measuring the actual phase error. The situation changes with packet reception. Between packets, there is no input from the TX and the loop estimates the phase with the assumption that the phase DAC is linear. For this reason, the error caused by the INL adds onto the jitter caused by the DNL. If the run length and frequency offset are large enough for the receiver to traverse the full range of the interpolator, the p2p jitter distribution will increase by the amount of maximum INL. The worst case occurs when the receiver finishes at the worst phase location (at maximum step size) when receiving data and in addition makes another estimate error of INL until the next data arrives. In terms of frequency accuracy, INL and DNL does not matter much since it appears as noise and corrupts the lower bits which we have already allocated to guard against noise.

Given the importance of linearity, the prototype contains several circuits to correct non-linearity. First, there is static phase offset correction for each of the ten phases from the PLL to minimize INL issues caused by core loop buffer delay mismatches. To minimize DNL, the interpolator was designed with four times more resolution than required and a lookup table was added to map FSM outputs to interpolator codes. Unfortunately as we show in the results, this

correction was not sufficient.

Burst Mode Packet Receiver

Until now, we have addressed how to build an accurate phase and frequency estimator that can retain lock in the absence of data for long periods of time. One problem not addressed is the acquisition of phase and frequency lock when the system starts. The RX must simultaneously acquire lock with all TX by gathering snippets of timing information from the packets. Unfortunately, if the phase drift between two packets from the same TX exceeds half a bit period, the CDR can false lock onto incorrect frequencies. The second order DLL in Fig. 2 will suffer from this problem.

To solve this, the first packet after reset is treated special and the CDR is modified (Fig. 3). Furthermore, we specify that until the DLL acquires frequency and phase lock with all TX, the optical switch is operating in round robin configuration such that the time between packets from a TX is fixed. During the first half of the first packet, the input to the integral branch is turned off (via MUX2) and the CDR acquires phase lock as a first order DLL. The packet length is chosen long enough to ensure that the first order DLL is phase locked by midpoint of the packet.

During the second half of the first packet, the input to the integral branch is enabled (via MUX2) while the integral control into the adder is zeroed (via MUX4). In addition to the previous condition on packet length, the packet length must also be an exponent of 2 number of system clock cycles (2^P). The input to the accumulator is scaled by 2^{-Q} (via MUX3). Q is equal to (M+P-1) and is closely related to the number of cycles in half a packet. As the first order DLL tracks the TX data, the phase updates accumulate. After half a packet, the accumulator value is equal to the number of phase updates divided by the observation time (i.e. frequency). Thus the first packet is used to obtain a coarse frequency estimate good enough so that the phase drift between the first two packets is less than half a bit. This coarse frequency estimate also suffers from quantization effects. The error in the estimate is bounded by $\pm 2^{-Q}$*1562.5ppm.

Packets and the intervals between them are emulated by turning on or off the output of the up-down counter (via MUX1). Matlab is used to simulate the acquisition behavior (Fig. 4). The settings used for the packet receiver are M=1, N=20, Q=10, packet length is 10kbits (i.e. P=10), and interval between packets is 320kbits. Fig. 4 (a) shows

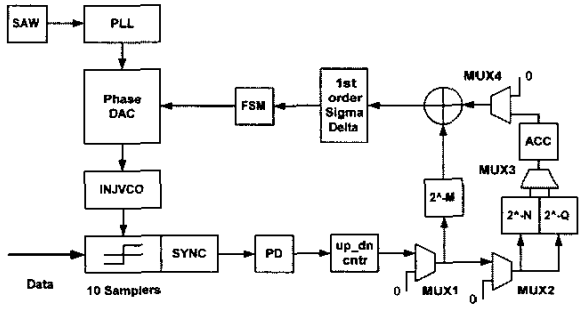

Fig. 3 Block diagram of packet receiver implemented in silicon.

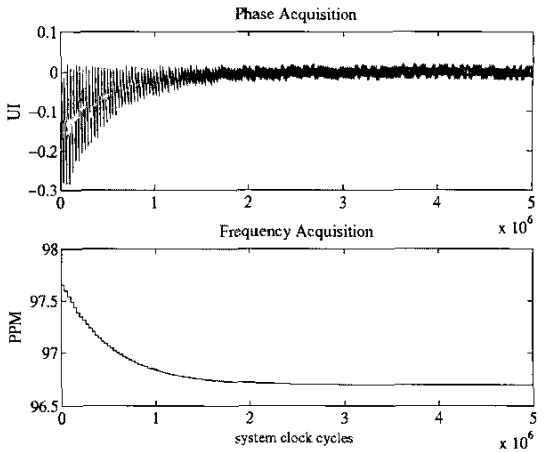

Fig. 4 Simulation of packet receiver (a) acquisition phase lock over sparse packets (b) acquisition of frequency lock

the phase error sampled at 10kbit intervals. Initially, the frequency estimate is not very good, so the accumulated phase error at the start of each packet is large. However, the phase error due to drift decreases over time as the frequency estimate converges to the final value of 96.7ppm. Fig. 4 (b) shows the CDR successfully acquiring frequency lock.

While we have shown how to build hardware that can estimate frequency to high precision, a key question that remains is whether commercial oscillators are stable enough to support long idle times between packets. Our preliminary results using an EPSON EG2102 clock source are promising. Fig. 5 plots the frequency offset between the TX and a RX that use this SAW oscillator. The data was gathered by observing the contents of the frequency accumulator over time. There is a quickly varying component on the order of 0.01ppm superimposed on a slow gradient. This slow gradient varies by only 0.03ppm over 4 minutes, and will be tracked by the loop, even in packet mode. The somewhat random 0.01ppm limits the frequency accuracy of our system since it cannot be tracked. This level of stability is more than sufficient for our system. With 0.01ppm error, 1Mbit of time can elapse between packets while degrading timing margin by only 0.01UI.

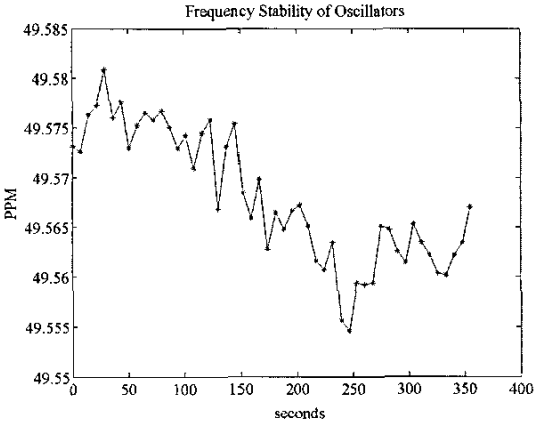

Fig. 5 Frequency stability of oscillator over time.

Experimental Results

The packet receiver was implemented in National's 0.25□m CMOS technology. Die size was 3mm x 2mm and included a TX, RX, and scan chain. The RX can be configured to function as either a normal CDR or a burst mode packet receiver. The jitter on the core loop PLL is 1.57ps rms, 11.1ps p2p. Unfortunately due to a design error, there was an uncorrectable phase gap of 34ps at the coarse phase boundary. As expected, this gap caused the transfer function of the interpolator to have a large DNL. All other INL errors were correctable and the resulting measured performance was DNL=31.4ps, INL=33.2ps. When receiving continuous PRBS (2^{10}-1) data, the jitter of the recovered clock is 9.2ps rms, and 61ps p2p (Fig. 6). As expected, the p2p jitter is about 2 times the maximum step size. Fig. 6(b) is the jitter histogram from Matlab using the measured transfer function and 2.4ps rms of gaussian jitter. It shows rough agreement with the measured results. As expected, p2p jitter increases by about 30ps (the INL) in packet reception. Fig. 7 shows the measured histogram with the same setting as Fig. 4 but with N=23. In addition, we were able to demonstrate that the packet receiver acquires lock even with intervals of 2.4Mbits.

Conclusions

In most semi-digital CDR loops, a very simple estimator for phase is used. But by tracking both phase and frequency we can build a much better phase estimator, one that can predict the phase of bits that are hundreds of thousands of cycles away. The key requirement is that the phase DAC must be linear to reduce estimation errors. With this improved phase estimator, a zero-overhead, packet-mode CDR can easily be built.

Acknowledgements

The authors would like to thank Azita Emami-Neyestanak, Dean Liu, and Valentin Abramzon for assistance in layout. Thanks also go to Jaeha Kim, Elad Alon, and Ron Ho for helpful discussions. Chips were fabricated by National Semiconductor and funding was provided by GT MARCO, NSF, and DARPA.

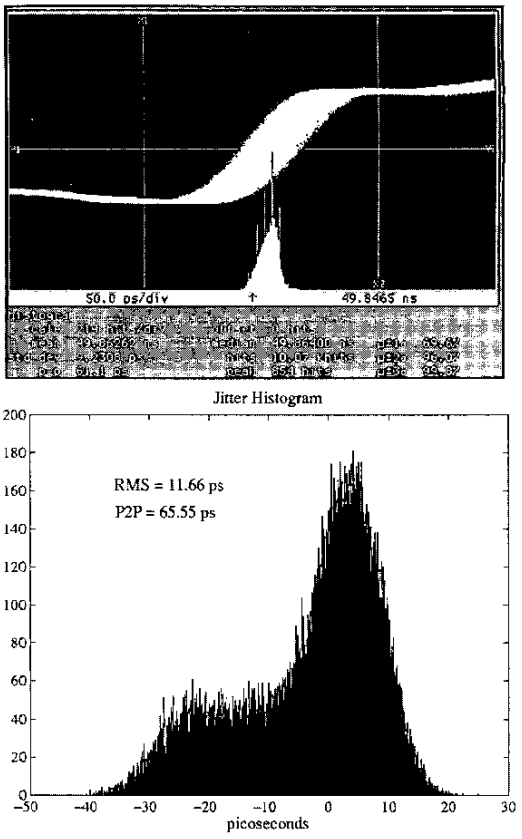

Fig. 6 (a) Measured Jitter (9.2ps rms, 61ps p2p) (b) simulated jitter (11.7ps rms, 65.6ps p2p)

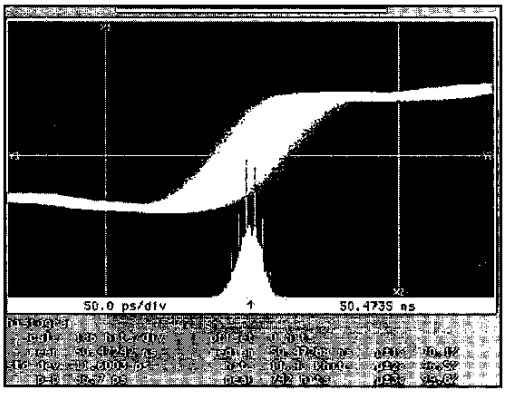

Fig. 7 Jitter histogram of packet receiver (11.6ps rms, 86.7ps p2p)

References

[1] I.M.White, M.S. Rogge, K. Shrikhande, and L.G. Kazovsky, "A summary of the HORNET project: a next-generation metropolitan area network", *IEEE Journal on Selected Areas in Communications*, Vol. 21, Issue: 9, pp. 1478 – 1494.

[2] C-K. K. Yang, R. Farjad-Rad, and M. Horowitz, "A 0.5-μm CMOS 4.0-Gbit/s serial link transceiver with data recovery using oversampling", *IEEE Journal of Solid-State Circuits*, Vol. 33, pp. 713 – 722, May 1998.

[3] M. Banu, and A. Dunlop, "A 660 Mb/s CMOS clock recovery circuit with instantaneous locking for NRZ data and burst-mode transmission", *Solid-State Circuits Conference, 1993. Digest of Technical Papers. 40th ISSCC*, pp. 102 –103.

[4] S. Sidiropoulos, and M. Horowitz, "A semidigital dual delay-locked loop", *IEEE Journal of Solid-State Circuits*, Vol. 32, pp. 1683 - 1692, Nov. 1997.

[5] M-J. E. Lee, et al, "A second-order semi-digital clock recovery circuit based on injection locking", *Solid-State Circuits Conference, 2003. Digest of Technical Paper. ISSCC. 2003.*

[6] M. Aoyama, et al, "3Gbps, 5000ppm spread spectrum serdes phy with frequency tracking phase interpolator for serial ata", *VLSI Circuits, 2003. Digest of Technical Papers*, pp. 107 –110