A 900 Mbit/s CMOS Data Recovery DLL using Half-Frequency Clock

X.Maillard, Frédéric Devisch and Maarten Kuijk

Vrije Universiteit Brussel, ETRO/LAMI, Pleinlaan 2, 1050 Brussel, E-mail: xavier.maillard@vub.ac.be

A CMOS sub-circuit that is able to improve data communication is described. It removes jitter and hence improves the eye diagram of high-speed digital data signal. The circuit is based on a delay-locked loop and uses a half-frequency reference clock. The prototype circuit is fabricated in 2.5 V, 0.25-µm CMOS and occupies an area of only 270 x 50 mm^2. It is demonstrated that at 900 Mbit/s NRZ data, jitter is reduced from 118.2 ps down to 31.3 ps RMS jitter for a power consumption of only 3 mW.

I. Introduction

Resampling digital data removes noise, but requires clock-data synchronization. For this purpose, delay locked loops (DLLs) are generally used when a reference clock is available, while phase locked loops (PLLs) synthesizes an in-phase equally frequency clock out of the data. However, for this clock regeneration the required feedback is more complex, consumes considerable silicon real estate and power. Further more, phase errors introduced by power or substrate noise also accumulate over many clock cycles, especially in cases with few data transitions made by long streams of successive ones or zeroes. In applications with an available reference clock, DLLs offers an attractive alternative due to their better stability and faster lock speed. The drawbacks of DLLs are their clock input jitter propagation and their limited phase capture range as the delay of a variable delay line is restricted between a minimum and maximum delay.

In contrast with DLL receiver circuits, we only aim at jitter removal, and therefore have the choice between delaying the data input signal and delaying the clock input signal to achieve synchronization. Delaying the clock is superior to delaying the data, because NRZ data contains more important high frequency components than the half frequency reference clock. Also, in this way the data is delayed as little as possible, which can be appreciated in applications where multiple repeaters are need and total latency is considered important.

This paper presents a differential DLL architecture, which recovers NRZ data in order to decrease jitter. To simplify the voltage controlled delay line (VCDL) and restrict the number of high frequency nodes, a half frequency clock is used as in [1].

II. Architecture

A block diagram of our circuit is outlined in Fig. 1. By means of the sampler and hold circuit (SH), triggered on the synchronized clock (*Clk)*, the input data (*Data*) is sampled in order to reject jitter. *Clk* is extracted from a reference clock (*RefClk*) by a VCDL that is controlled by a feedback loop. The loop regulates the phase between *Clk* and *Data* close to zero with the following basic principle. The *Data* and *Clk* signals drive a phase detector PD, which output is low pass filtered by capacitor C_1, to generate a stabile loop control voltage *Vc*. If *Vc* is too high (low), the VCDL will produce a too long (short) delay and the rising *Clk* edge arrives too late (early) at the PD related to the *Data*

edge. This still low (already high) *Clk* level is conducted to *Vc* on every *Data* edge, what decreases (increases) *Vc* and the related phase error decreases.

As mention earlier, the DLL limited phase capture range creates difficulties in several circumstances. For example when the initial lock is already near the limit of the *Vc* range or when the *RefClk* frequency varies from the data frequency. In this case the loop will correct *Vc* too much in one direction and the DLL enters in a state where the VCDL cannot generate the asked delay. Although once the VCDL is able to generate one bit delay variation, it is always possible to create the correct *Clk* phase. In this design we propose a self-correcting (SC) block that is responsible for the out of range detection and the following *Vc* shift. When passing the maximum or minimum limit, the SC block decrements or increments *Vc* with a preset voltage step *dV*, corresponding with approximately 1 clock cycle, which generates the same *Clk* phase with an easier obtainable delay. [2] and [4] have also tackled the problem. An approach with quadrature phase mixing is given in [4] and one with an adapted phase detector in [2]. In contrast with these, this approach will not change the PD or jitter performance of the DLL and consumes negligible power.

The remaining phase error depends on the loop amplification ($A_{loop} = A_{VCDL} . A_{PD}$) made in the VCDL and the PD. The simulated A_{VCDL} is 1.8ns/V and the PD sensitivity A_{PD} is 5 V/ns phase error for small errors. With a VCDL that has the minimum required delay variation of one bit for the used *Vc* range, and a SC block that holds *Vc* in this range, the phase errors will be small, which offers an A_{loop} of 9. This results in a small remaining phase error of 0.1 ns at 900 Mbit/s, which corresponds with the measurements.

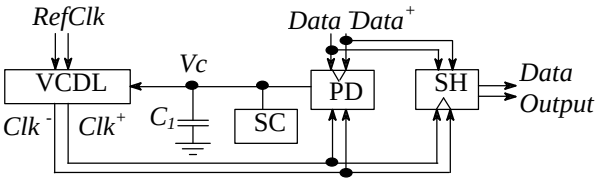

Fig. 1: Circuit architecture.

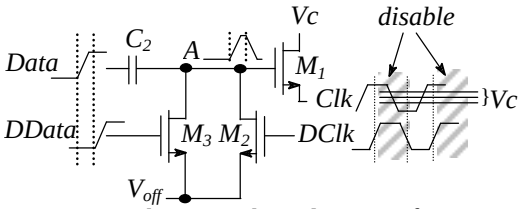

Fig. 2: PD sub circuit that adjusts *Vc* for a rising edge on *Data* and *Clk*.

III. Phase detector

The DLL adjusts *Vc* to synchronize the clock with the incoming data stream. Because of the random nature of the data, there is not necessarily a data transition at every clock edge. The loop has to handle sequences of consecutive logic zeroes or ones in the data stream.

Traditional PDs steer the loop control voltage *Vc* by high frequency “up” and “down” signals. In our design we use a pass-gate construction to adjust *Vc* at every data edge. The adjustment to *Vc* is proportional to the phase error. Without data edge, *Vc* is not adjusted.

The PD consists of 4 equal adjustment circuits (ACs) for adjusting *Vc*. Depending on whether the *Data* has a rising or falling edge and on whether the half frequency *Clk* has a rising or falling edge, the appropriate AC is selected. Fig. 2 shows such AC that operates on a rising *Data* and *Clk* edge. Through capacitor C_2 (10 fF), a pulse is constructed on node *A* when *Data* goes high. When delayed data (*DData*) signal rises, node *A* goes back to V_{off} by transistor *M3*. During the pulse on node *A*, transistor *M1* conducts. In this way, it looks as if the *Clk* signal is sampled, however transistor *M1* conducts only poorly inducing limited change on *Vc*. Futher, when delayed clock (*DClk*) signal is high, no pulse on node *A* is generated.

When the pulse is rather positioned at the beginning of the *Clk* edge, then *Vc* will be adjusted downwards by a small amount proportional to the time deviation between the center of the pulse and the *Clk*-*Vc* crossing. When the center of the pulse is after the *Clk*-*Vc* crossing, *Vc* is adjusted proportionally upwards. When the center of the pulse appears at the moment of the *Clk*-*Vc* crossing, the adjustment to *Vc* is equally down- as upwards.

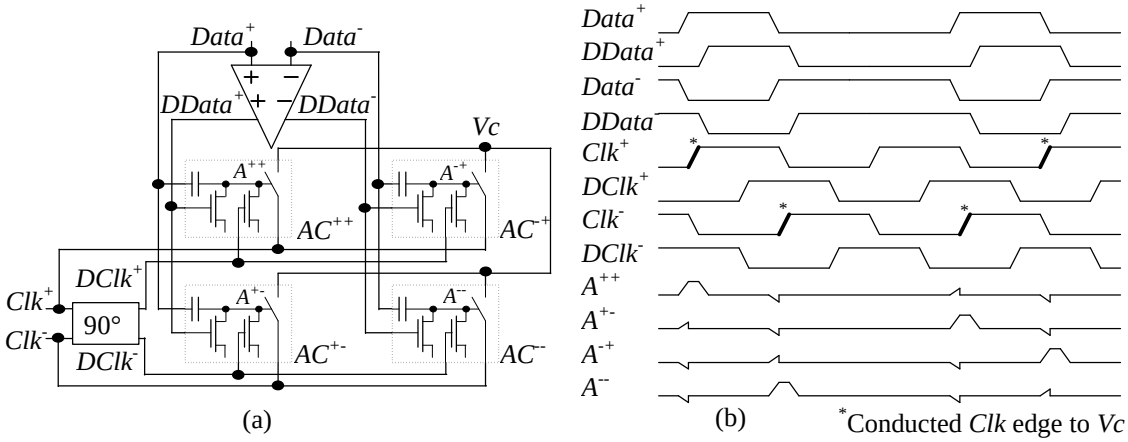

Fig. 3: (a) Symmetric phase detector. (b) Operation of the PD.

Fig. 3(a) shows the symmetric PD with 4 ACs including further a circuit that generates a 90° shifted phase clock *DClk*, and a delay block that creates the *DData* signals.

The adjustment circuits $AC^{+\pm}$ are only triggered on rising *Data* edges. The complementary adjustment circuits $AC^{-\pm}$, connected to the same *Clk* lines as $AC^{+\pm}$, use the inverse *Data* and are hence triggered on falling *Data* edges.

As mentioned earlier, the clock frequency is halved such that two data bits are allowed in one clock period. To make a symmetric PD that detects phase on both *Clk* edges, $AC^{\pm-}$ are implemented complementary to $AC^{\pm+}$.

Fig. 3(b) illustrates the working principle. At every data transition, there is always exactly one *M1* that connects *Vc* with a rising *Clk* edge.

IV. VOLTAGE-CONTROLLED DELAY LINE

The choice of a delay stage is a critical task in PLL and DLL designs. Although the same delay stages could be used in a VCDL and a VCO, the basic requirements are different. The maximum VCO operating frequency is obtained when the VCO could reach the frequency with one control voltage *Vc* out of the *Vc* range, while the VCDL bandwidth has to be guaranteed for every *Vc*. This limits the *Vc* sensitivity of the individual VCDL stages, which restricts the variable stage delay. Semi digital approaches could provide wide delay variation by choosing or combining different delay stages but this requires a lot of logic and the delay resolution is finite. Analog switching between a fast and a slow path is possible but we use a single stage amplifier. One stage provides a theoretical maximum propagation phase delay of 90°. A stage has also a minimum propagation delay, depending on the used technology and topology. This results in a small delay variation at the technology limit and halving the clock frequency is useful.

We use 9 of the in [3] proposed stages, which are differential and hence good against power and substrate noise and provide the differential *Clk* signals needed in PD and SH.

The number of stages is chosen by considering the following trade-off. Fewer stages consume less power, but more stages make lower operating frequencies possible for which the required delay variation of 1 PD period is guaranteed.

More stages increment also A_{loop} and decrement the remaining phase error, but using the whole PD output range is less noise sensitive as the same Vc noise variation creates a smaller delay variation.

VII. EXPERIMENTAL RESULTS

To verify the DLL architecture, a chip has been fabricated in 2.5 V, 0.25-µm UMC CMOS process. The total area measures approximately 270 x 50 μm^2.

The DLL has been tested with 900 Mbit/s disturbed NRZ data. Channel 2 of Fig. 8 shows this data with 668 ps p-p or 118.2 ps RMS jitter (histograms not shown on Fig. 4).

Channel 4 of Fig. 4a demonstrates the DLL performance with *equal* clock and data frequency with 31.3 ps RMS or 244 ps p-p output jitter. A second measurement with a 50Hz *frequency difference* between *RefClk* and *Data* is given in Fig. 4b. This frequency difference reduces the eye diagram, as prospected, by 0.1ns p-p down to 59.5 ps RMS or 372.0 ps p-p jitter.

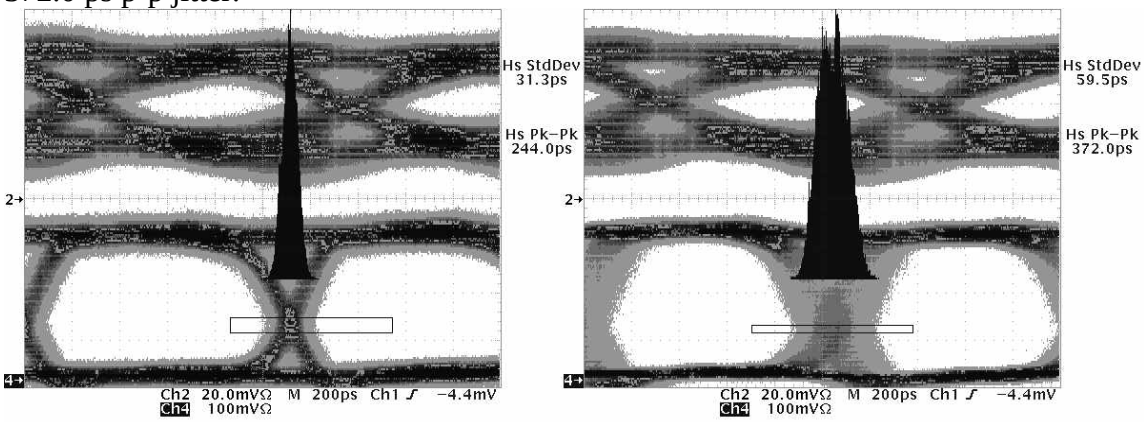

Fig. 4: Measurement of eye diagrams of NRZ Data in[ch2] and out[ch4] at (a) *RefClk* frequency (b) 50Hz frequency difference between *Refclk* and *Data*.

VIII. CONCLUSION

This paper shows the operation of a repeater circuit based on a data recovering DLL with half frequency clock. Fabricated in a 0.25-µm CMOS, an operating bit rate of 900Mbit/s of NRZ data with 31.3 ps of RMS jitter is achieved. The limited phase capture range present in conventional DLLs has been solved. The power consumption is low and silicon area is small allowing the use of one or multiple repeaters on a single chip possibly together with other functions.

References

[1] M. Rau, T. Oberst, R. Lares, A. Rothermel, R. Schweer, and N. Menoux, "Clock/Data Recovery PLL Using half frequency clock," *IEEE J. Solid-State Circuits*, vol.32, pp. 1156-1159, July 1997.

[2] D. J. Foley, and M. P. Flynn, "CMOS DLL-Bases 2-V 3.2-ps Jitter 1GHz Clock Synthesizer and Temperature-Compensated Tunable Oscillator," *IEEE J. Solid-State Circuits*, vol. 36, pp. 417-422, March 2001.

[3] J. J. Kim, S. Lee, T. Jung, C. Kim, S. cho, and B.Kim, "A Low Jitter Mixed-Mode DLL for High Speed DRAM Applications," *IEEE J. Solid-State Circuits*, vol. 35, pp. 1430-1436, Oct. 2000.

[4] S. Sidiropoulos, and M. A. Horowitz, "A Semidigital Dual Delay-Locked Loop," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1683-1692, Nov. 1997.