Transactions Briefs

A 15 MHz to 600 MHz, 20 mW, 0.38 mm^2 Split-Control, Fast Coarse Locking Digital DLL in 0.13 μm CMOS

Sebastian Hoyos, Cheongyuen W. Tsang, Johan Vanderhaegen, Yun Chiu, Yasutoshi Aibara, Haideh Khorramabadi, and Borivoje Nikolic

***Abstract*—A digital delay-locked loop (DLL) suitable for generation of multiphase clocks in applications such as time-interleaved and pipelined analog-to-digital converters (ADCs) locks in a very wide (40×) frequency range. The DLL provides 12 uniformly delayed phases, free of false harmonic locking. A two-stage digital split-control loop is implemented: a fast-locking coarse acquisition is achieved in four cycles using binary search; a fine linear loop achieves low jitter (9 ps rms @ 600 MHz) and tracks process, voltage, and temperature (PVT) variations. The false harmonic locking detector, the frequency range and the jitter performance among other design considerations are analyzed in detail. The DLL consumes 20 mW and occupies a 470 μm × 800 μm in 0.13 μm CMOS.**

***Index Terms*—All-digital delay-locked loop (DLL), DLL, split control loop.**

I. Introduction

Synchronization is crucial for the operation of digital functional blocks and mixed-signal interfaces. As an example, time-interleaved and pipelined analog-to-digital converters (ADCs) require generation of multiple clock phases in a very wide operating frequency range with challenging jitter requirements in the upper frequency range. Delay-locked loops (DLLs) are often used in these applications and their design faces tradeoffs between the requirements for low jitter, fast locking, wide frequency-range, and low power consumption. Low voltage headroom, associated with supply voltages in scaled technologies presents a challenge for analog control loops in a DLL to achieve a very wide locking range. This limitation is solved by using digital control loops that ideally can use longer wordlengths to extend the dynamic locking range [1]–[5]. Jitter in digital DLLs is determined by the size of the digital-to-analog converter's (DAC's) least significant bit (LSB) that controls the delay line. However, the interaction of the wide dynamic range control with the delay line dramatically impacts other performance metrics such as the locking time, jitter, power consumption, and silicon area.

There have been several recent notable advances in nanometer CMOS DLL architectures. Duty-cycle correcting techniques have been demonstrated in [5] and analog and digital solutions to the false detection problem are developed in [1]. DLL-based clock generation with frequency scaling is presented in [3]. Low-jitter open-loop architecture is proposed in [4].

This DLL achieves 40× locking range, coupled with fast locking and low steady-state jitter at high frequencies. This locking range enables a wide set of operating modes and is invaluable in system testability. A 10-bit digital control is used to limit the jitter and to enhance the locking range. It adjusts the delay of the current-starved-inverter-based delay line, where the four most significant bits (MSBs) coarsely select the frequency range (15–600 MHz) using a fast binary search coupled to a binary-weighted DAC replicated at each delay cell. The six LSBs linearly control the delay elements for a low jitter in steady state. The unit-element LSB DAC is shared among all delay cells. This split-control architecture enables the adjustment of delay elements with low supply voltage in the desired operating range. This design also allows for low power consumption and a moderate silicon area for a DLL with 12 clock phases and 40× locking range.

Section II of this paper explains the DLL architecture and implementation, provides analysis of clock jitter performance and design derivations of the false locking detector and LSB and MSB DACs. Section III elaborates on the testing details and conclusions are provided in Section IV.

II. Architecture and Implementation

A. Wide Range DLLs

DLLs with a wide locking range are enabled by using a digital control loop. While the analog control relies on a charge pump, the digital control is based on a digital counter and a state machine. Reduced voltage headroom of few hundred millivolts in deeply scaled CMOS limits the operating range of conventional analog charge pumps. Digital control loops decouple the requirements for the wide locking range from the jitter magnitude by using more bits in the control loop; However, they still face several design tradeoffs. The most notable one is the steady-state error proportional to the LSB magnitude; this is in contrast to analog charge pumps that produce analog control voltage. Increasing the number of bits in the digital loop achieves the desired resolution and steady-state error. Increase in digital wordlengths presents trades off the desired locking range for the reduced speed of digital circuitry.

Another important difference is the design of the phase detector (PD). The conventional analog DLL design requires PDs that modulate the output pulse width proportionally to the input clock phase difference. In this way, the charge pump adjusts dynamically the amount of capacitor charge in order to smoothly converge to a steady-state value. The closer the steady-state, the lower the capacitor charging/discharging current. On the other hand, digital DLLs only require a binary control signal from the PD, usually implemented just by a latch or a flip-flop. This binary signal determines the direction in which the counter moves: UP or DOWN. This simpler PD design still requires a careful delay balancing between the two input clocks that are to be locked. Any mismatch between the wires and buffers that drive

Manuscript received July 14, 2010; revised November 13, 2010; accepted January 05, 2011. Date of publication January 31, 2011; date of current version February 17, 2012.

S. Hoyos is with the Department of Electrical and Computer Engineering, Texas AM University, College Station, TX 77843-3128 USA (e-mail: hoyos@ece.tamu.edu).

C. W. Tsang is with the Agilent Technologies Inc., Santa Clara, CA 95051 USA

J. Vanderhaegen is with the Robert Bosch Corporation, Palo Alto, CA 94304 USA (e-mail: johan.vdhaegen@gmail.com).

Y. Chiu is with the Department of Electrical and Computer Engineering, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: chiuyun2004@gmail.com).

Y. Aibara is with the Renesas Technology Corporation, Mizuhara, Itami-shi, Hyogo 664-0005, Japan (e-mail: aibara.yasutoshi@renesas.com).

H. Khorramabadi and B. Nikolic are with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: haidehk@eecs.berkeley.edu; bora@eecs.berkeley.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2011.2106170

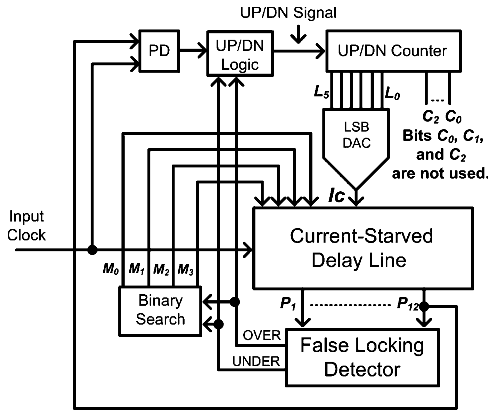

Fig. 1. Block diagram of the DLL.

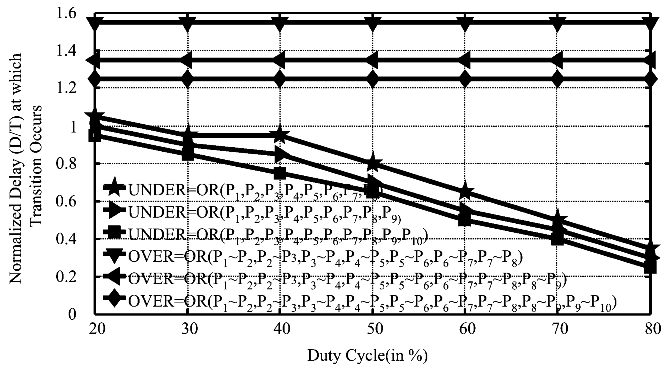

Fig. 3. False harmonic locking detector logic for generating the UNDER and OVER control signals.

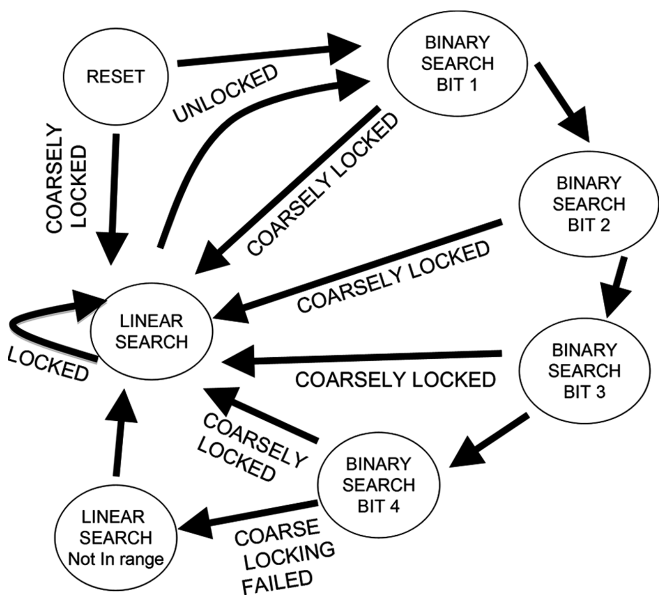

Fig. 2. State machine illustrating coarse fast binary search and the linear search.

the clocks to the input of the PD, or within the PD itself, leads to a steady-state delay error.

B. DLL Architecture and Implementation

Fig. 1 shows the block diagram of the proposed DLL and Fig. 2 illustrates the state machine with the two phases of the split-control loop operation. The split architecture implemented in this design allows for tracking of process, voltage, and temperature (PVT) variations, detection of frequency changes for automatic triggering of the locking mechanism and flexibility in the jitter specifications at different frequencies. The first control loop is formed by the binary search which brings the total delay D of the delay line within the locking range, $3T/4 < D < 5T/4$, where T is the clock period. Although $T/2 < D < 3T/2$ is the theoretical range, it leaves no margin for the D-flip-flop (DFF) PD to bring the loop to locked state. If there are mismatches or duty cycle variations and the binary search finishes just slightly outside the theoretical range, the linear loop will move in the wrong direction. The binary search is controlled by two signals: UNDER and OVER that are generated by the logic for detecting false harmonic locking adapted from [1], illustrated in Fig. 3. If $D < 3T/4$, the UNDER signal is activated and if $D > 5T/4$, the OVER signal is activated to prevent potential locking to a multiple of the fundamental

TABLE I
FALSE LOCKING DETECTION USING P1-P12 DELAY LINE PHASES

	0.6T	0.75 T	0.8T	T	1.22T	1.25 T	1.33 T	1.4T
P1	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0
P5	0	0	0	0	0	1	1	1
P7	0	0	0	0	1	1	1	0
P9	0	0	0	1	0	0	0	0
P10	0	0	1	1	0	0	0	1
P11	0	1	1	1	0	0	0	1
P12	1	1	1	0	0	0	0	1
	UNDER	IN LOCK RANGE					OVER	

period. The UNDER and OVER signals are determined based on the delay line phases P_1 to P_{12}. Table I shows the logic levels of the delay line phases at the rising edge of the input clock with 50% duty cycle. The simulated results in Table I can also be derived analytically as following. The delay of each delay cell can be expressed as

$$\Delta t = \frac{D}{N} \tag{1}$$

where Δt is the delay of each delay cell, D is the delay of the whole delay line, and $N = 12$ (the number of delay cells) for the current design. The nth phase has a delay $n\Delta t$; this delay can be made frequency independent by normalizing it to the clock period T.

Thus, the phases P_1 to P_{12} are equal to 1 or 0 according to the following conditions:

$$P_n = \begin{cases} 0, & \text{if } 0 < \frac{n\Delta t}{T} < 0.5 \\ 1, & \text{if } 0.5 \le \frac{n\Delta t}{T} \le 1 \end{cases} \tag{2}$$

where $n\Delta t < T$. This equation can also be extended for $n\Delta t > T$:

$$P_n = \begin{cases} 0, & \text{if } 0 < \mathrm{mod}(n\Delta t, T) < 0.5 \\ 1, & \text{if } 0.5 \le \mathrm{mod}(n\Delta t, T) \le 1 \end{cases} \tag{3}$$

where the modulus operation is defined as $\mathrm{mod}(x, y) = x - \mathrm{floor}(x/y)y$.

The presented analysis assumes a 50% duty-cycle clock. For an arbitrary duty-cycle (DC), the (3) modifies

$$P_n = \begin{cases} 0, & \text{if } 0 < \mathrm{mod}(n\Delta t, T) < \left(1 - \frac{DC}{100\%}\right) \\ 1, & \text{if } \left(1 - \frac{DC}{100\%}\right) \le \mathrm{mod}(n\Delta t, T) \le 1. \end{cases} \tag{4}$$

In (4), the P_n values are valid at the rising edge of the master input clock P_0.

The UNDER signal is high if all the phases from P_1 to P_9 are low. The OVER signal is asserted if a "...10..." pattern is found in any two consecutive phases from P_1 to P_8. Fig. 4 shows the normalized delay

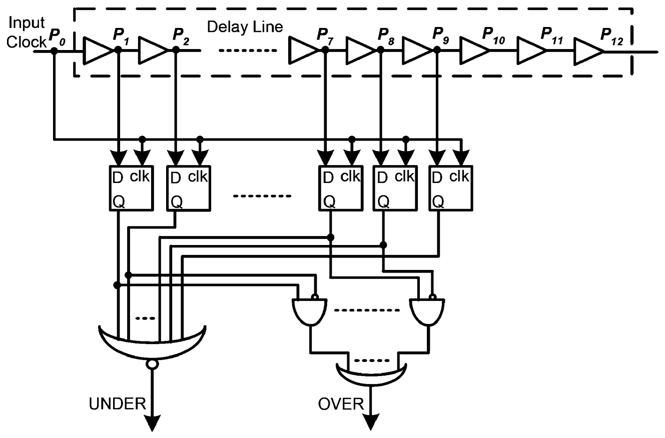

Fig. 4. Normalized delay (D/T) at which the transition 0 to 1 of the control signals UNDER and OVER occurs versus the duty cycle, parameterized by the number of phases used in the logic that computes them.

(D/T), for which the transition 0 to 1 of the control signals UNDER and OVER occurs, versus the DC, parameterized by the number of phases in the UNDER and OVER signal computation. This figure illustrates that the OVER signal is not affected by the changes in the duty cycle from 20% to 80% of the input clock, making this DLL immune to a wide range of duty cycle variations and free of false harmonic locking. Note that the UNDER signal is not involved in the false-harmonic locking detection; Instead, it is used to bring the delay D above $3T/4$. Since the phase detector is just a DFF that compares the rising edges of the input clock (P_0) and the output of the delay line (P_{12}), the UNDER signal 0–1 transition that triggers the linear loop operation should satisfy

$$\frac{DC}{100\%} > 1 - \frac{D}{T} \tag{5}$$

for a reliable operation of the PD. Fig. 4 shows that this condition is satisfied for any value of the duty cycle, which makes the UNDER signal also insensitive of variations on the duty cycle. Moreover, when desired to lock the falling edge of the clock it is also easy to show that the condition

$$\frac{D}{T} > \frac{DC}{100\%} \tag{6}$$

needs to be satisfied, as illustrated in Fig. 5. In this case, plot for $\mathrm{UNDER} = \mathrm{OR}(P_1, P_2, \ldots, P_8)$ signal in Fig. 4, shows that the present design is limited to $DC < 60\%$, when used for locking of the falling edges. If the DLL is to be used in an application that requires duty cycles larger than 60%, smaller number of phases in the OR logic operation for the UNDER signal should be used to extend the DC range.

In the event of either the UNDER or OVER signals remaining active after the binary search finishes, the UP/DN logic would disable the PD output, to avoid false locking and uses the UNDER or OVER signals to bring the linear loop into a locking range. This state is indicated in the state machine (Fig. 2—state "LINEAR SEARCH Not in range").

The binary search machine is triggered by either an external reset signal or by a change in either the UNDER or OVER signals. This feature allows this DLL to track frequency changes in its entire range of operation and makes it suitable for broadband applications. When the binary search completes, the 6-bit LSB linear loop, whose counter is initialized at mid-range, makes the final fine adjustments to bring the total delay D within one LSB of the desired input clock period T. Only the top 6 bits of the 9-bit counter are used to drive the unit element DAC; truncation of the three LSBs provide some kind of amplitude selection

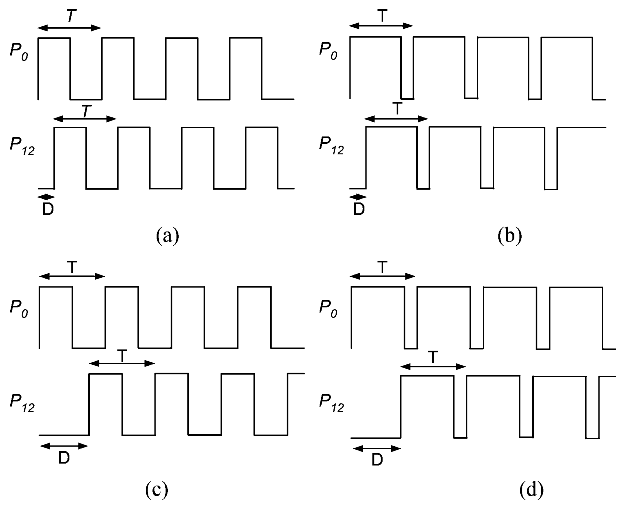

Fig. 5. Illustration of critical conditions between the phases $\boldsymbol{P_0}$ and $\boldsymbol{P_{12}}$ that need to be guaranteed for locking at the (a) rising edge and (c) falling edge. Cases (b) and (d) illustrate the conditions when the phase relationship does not guarantee locking (a) $DC/100\% < 1 - D/T$; (b) $DC/100\% > 1 - D/T$; (c) $DC/100\% < D/T$; (d) $DC/100\% > D/T$.

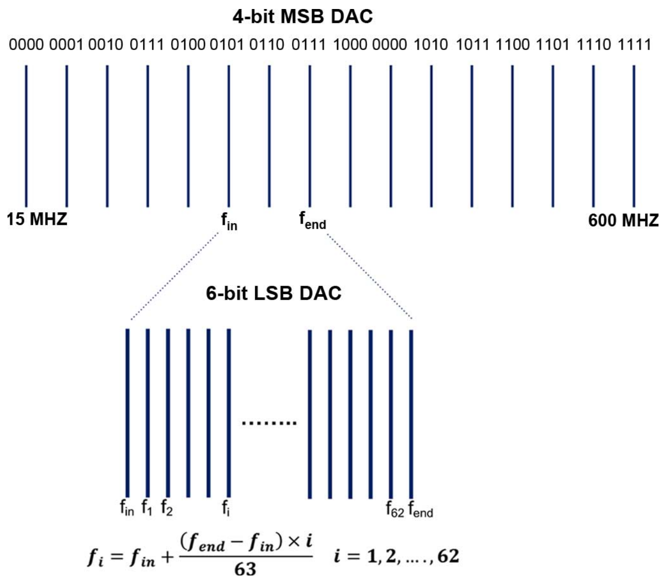

Fig. 6. MSB and LSB DACs for digital split-control.

filtering. Discarding the three LSBs also lowers jitter in steady state because it disregards random UP/DN signals due to noise. The linear search stays on during operation to provide compensation for voltage, temperature variations, and aging.

C. Design of LSB and MSB DACs

The four MSB bits expand the entire frequency range at uniform intervals determined by 15 MHz $+n \times (600 - 15)/15$ MHz for $n = 0, 1, 2, \ldots, 15$. The remaining six LSB bits provide a finer frequency resolution of $k \times 2 \times (600 - 15)/(63 \times 15)$ MHz between the MSB intervals. Fig. 6 illustrates the DACs digital split-control. This results in a frequency step of $2 \times (600 - 15)/(63 \times 15)$ MHz $= 1.24$ MHz which corresponds to a peak-to-peak jitter lower than 6.9 ps at 600 MHz. Note that the LSB DAC overlaps with two LSB steps of the MSB DAC in order to avoid frequency gaps due to mismatches in the MSB DAC. Therefore, the dynamic nonlinearity (DNL) of the MSB DAC must be better than one LSB which can be easily accomplished in practice.

Moreover, the jitter at low frequencies can be much larger if the same frequency step is used. In order to achieve lower jitter at low frequencies, the LSB reference current can be reduced. In this way, the coarse locking information in the MSB codeword can be used to adaptively scale the LSB current reference to a desired level to obtain the clock jitter specification. This alternative would require a DLL redesign to avoid frequency gaps caused by the limited LSB DAC range. This can be compensated by introducing more nonuniformly spaced levels in the MSB DAC at low and mid-range frequencies.

The DLL presented in this paper is designed to generate the non-overlapping phases needed in a pipelined ADC. In this and most other applications of clock generation, the jitter specification tightens with increasing frequency; therefore low jitter is needed only at high frequencies and just 4 bits were sufficient in the MSB DAC. As a result, if the LSB reference current is not reduced for low-frequency operation, the peak-to-peak jitter can be as high as 8% of the clock period at 15 MHz and it still satisfies the intended application. The overlapping of the LSB DAC with two LSB steps of the MSB DAC showed to be sufficient to avoid frequency gaps in the targeted frequency range. The flexibility of the digital design of the control loop allows adding an arbitrary number of levels to the MSB DAC which in conjunction with the adaptive LSB DAC reference current can provide good jitter performance at low frequencies as well. In order to make the DLL lock up to $F = 600$ MHz, the current associated with the MSB codeword $M_0M_1M_2M_3 = 1111$ should satisfy $F = kI$, k being a constant that depends on the load capacitance of the current-starved delay cell and the number of cells that formed the delay line. Assuming low-to-high and high-to-low propagation delays linearly dependent on the capacitive load, C_L and Vdd, the proportionality constant is given by

$$k = \frac{1}{CL \times M \times Vdd}\left[(VF)^{-1}\right] \tag{7}$$

where M is the number of current-starved inverters.

D. Maximum Frequency Range

The MSB DAC determines the frequency range which, in principle, could be designed with as many bits as needed to obtain a desired range. There is a practical limit of the delay line's range, determined by the minimum current of the DAC, i.e., the minimum transistor size operating in saturation (to guarantee appropriate accuracy). The minimum delay is determined by the inverter connected rail to rail, which gives the intrinsic delay of the delay line $(0.69NC_L(R_{\text{eqn}}+R_{\text{eqp}})/2)$, where N is the number of inverters $(N = 24)$ and Reqn and Reqp depend on Vdd and I_{DSAT}. However, adding more bits in the MSB DAC will increase C_L which in turn will increase the minimum delay, reducing the maximum frequency range. There is an optimal, layout-dependent, point, determined by the DAC self-loading as it gets larger. The frequency range is determined by the minimum $(I_{\min})$ and maximum $(I_{\max})$ current drawn by the DACs that feed the current-starved inverters. These currents are determined by the digital words $M_0M_1M_2M_3$ and $L_0L_1L_2L_3L_4L_5$ (see Fig. 1). All these bits equal to 1 lead to $I_{\max}$ whereas for all of them equal to 0 lead to $I_{\min}$ which is determined by a fixed offset current I_{offset}. Then, the frequency range can be expressed as

$$kI_{\min} \leq F \leq kI_{\max} \tag{8}$$

where k is defined in (7) and $kI_{\min} = kI_{\text{offset}}$.

E. Clock Jitter Analysis

The analysis provided in this section shows that the jitter of this DLL topology is inversely proportional to the squared value of the operation frequency. This property is conveniently related to the jitter specification of Nyquist-rate ADCs where the SNR is inversely proportional to the product of the sampling clock jitter variance and the squared frequency. In order to derive this result, we first express the delay of the delay line as

TABLE II
RMS JITTER ACROSS THE OPERATING RANGE

Freq. (MHz)	600	500	400	300	200	150	50	15
rms off-chip jitter [ps]	9	9	9	9.5	10.2	20.5	100	116[1]
Expected rms on-chip jitter	5.7	5.7	5.7	6.4	7.4	19.3	99.7	115.7

[1] An LSB current that is lower than in the other measurements was used here to improve the jitter.

$$T_1 = \frac{1}{kI_1} \tag{9}$$

and the peak-to-peak jitter is

$$\Delta t_1 = \frac{1}{k(I_1 - \Delta I)} - \frac{1}{k(I_1 + \Delta I)} \tag{10}$$

where ΔI is the LSB current value. Similarly for a frequency that is larger by a factor N, i.e., $F_2 = NF_1 = kI_2 = kNI_1$, the associated jitter is, $\Delta t_2 = 1/k(I_2 - \Delta I) - 1/k(I_2 + \Delta I)$. Thus, the jitter drop-off when the frequency rises from F_1 to F_2 is given by

$$\frac{\Delta t_2}{\Delta t_1} = \frac{(I_1 - \Delta I)(I_1 + \Delta I)}{(I_2 - \Delta I)(I_2 + \Delta I)} = \left(\frac{I_1}{I_2}\right)^2 = \frac{1}{N^2}. \tag{11}$$

In the prototype chip, the MSB and LSB currents are programmable, which gives flexibility in testing.

III. MEASUREMENTS

The DLL has been implemented in a general-purpose, 0.13 μm 6M1P CMOS technology. The DLL occupies 470 μm $\times$ 800 μm area. Measured jitter performance is summarized in Table II and jitter measurement plots at 600 and 380 MHz can be found in [2]. The DLL clock is driven off-chip using LVDS pads, which worsens the jitter by up to 7 ps rms (up to 1 GHz) as specified by the pad's data sheet. The actual on-chip DLL jitter variance is expected to be up to $(7\text{ ps})^2$ better than the squared of the rms values in Table II. The 20 mW power consumption does not include all the buffers used to drive the clocks. The test chip integrates two DLLs and two buffers, in order to provide the non-overlapping clocks for two ADCs. The linear control loop has been left running during the measurements to absorb PVT variations in the locked state in a practical application. The steady-state jitter produced by the LSB toggling is reduced with increased clock frequency as indicated in (11). For frequencies larger than 300 MHz, the jitter produced by the LSB toggling is lower than the intrinsic jitter induced by the electronic noise. At low frequencies, however, the effect of the LSB toggling is higher, as the output will toggle between two phases as shown in Fig. 5 in [2]. In the worst case, when the locked edges of the input clock and the delayed clock, that the phase detector compares, are closer to each other than the peak-to-peak jitter of the clocks themselves, the LSB toggling will sometimes move the delayed clock to the right of the locked edge (when it is lagging the input clock) and some other times to the left of the locked edge (when is leading the input clock) creating a trimodal edge profile as shown in the measurements. As a result, the peak-to-peak jitter can be as large as two LSBs. This case is illustrated in the measured eye diagram of [2, Fig. 6].

IV. Conclusion

This paper presents an all digital implementation of a DLL with a 40$\times$ frequency locking range. A dual loop design, consisting in a coarse fast binary search combined with a linear search is proposed. This design achieves a large locking range with fast coarse locking while keeping the jitter and power consumption low. The chip occupies a 470 μm $\times$ 800 μm area and draws 20 mW at 600 MHz in 0.13 μm general-purpose CMOS. The 12 uniform phases of this DLL makes it suitable for providing the phases in applications such as time-interleaved and pipelined ADCs and broadband communications.

References

[1] Y. Aibara, E. Imaizumi, H. Takagishi, and T. Matsuura, "A novel false lock detection technique for a wide frequency range delay-locked loop," *IEICE Trans.*, vol. E89-A, no. 2, pp. 385–390, 2006.

[2] S. Hoyos, C. W. Tsang, J. Vanderhaegen, Y. Chiu, Y. Aibara, H. Khorramabadi, and B. Nikolic, "A 15 MHz–600 MHz, 20 mW, 0.38 mm^2, fast coarse locking digital DLL in 0.13 μm CMOS," presented at the IEEE ESSCC Dig. Tech. Papers, Edinburgh, Germany, Sep. 2008.

[3] J.-H. Kim, Y.-H. Kwak, S.-R. Yoon, M.-Y. Kim, S.-W. Kim, and C. Kim, "A CMOS DLL-based 120 MHz to 1.8 GHz clock generator for dynamic frequency scaling," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2005, pp. 516–517.

[4] B. Mesgarzadeh and A. Alvandpour, "A low-power digital DLL-based clock generator in open-loop mode," *IEEE J. Solid-State Circuits*, vol. 44, no. 7, pp. 1907–1913, Jul. 2009.

[5] M.-Y. Kim, D. Shin, H. Chae, and C. Kim, "A low-jitter open-loop all-digital clock generator with two-cycle lock-time," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 10, pp. 1461–1469, Oct. 2009.

High-Throughput Interpolator Architecture for Low-Complexity Chase Decoding of RS Codes

F. García-Herrero, M. J. Canet, J. Valls, and P. K. Meher

***Abstract*—In this paper, a high-throughput interpolator architecture for soft-decision decoding of Reed–Solomon (RS) codes based on low-complexity chase (LCC) decoding is presented. We have formulated a modified form of the Nielson's interpolation algorithm, using some typical features of LCC decoding. The proposed algorithm works with a different scheduling, takes care of the limited growth of the polynomials, and shares the common interpolation points, for reducing the latency of interpolation. Based on the proposed modified Nielson's algorithm we have derived a low-latency architecture to reduce the overall latency of the whole LCC decoder. An efficiency of at least 39%, in terms of area-delay product, has been achieved by an LCC decoder, by using the proposed interpolator architecture, over the best of the previously reported architectures for an RS(255,239) code with eight test vectors. We have implemented the proposed interpolator in a Virtex-II FPGA device, which provides 914 Mb/s of throughput using 806 slices.**

***Index Terms*—Algebraic soft-decision decoding, interpolation, low-complexity chase (LCC), low latency, Nielson's algorithm, Reed–Solomon (R-S) codes.**

I. Introduction

THE algebraic soft-decoding (ASD) of Reed–Solomon (RS) codes [1] provides significant coding gain over hard-decision decoding with polynomial complexity. The decoder in this case has three main functions: 1) multiplicity assignment; 2) interpolation; and 3) factorization of bivariate polynomials [2], being the interpolation the most computation-intensive one. Several architectures based on Nielson's algorithm [3]–[5] and Lee–O'Sullivan algorithm [6] are found in the literature for the VLSI implementation of interpolation stage. However, their hardware complexity is still high. The low-complexity chase (LCC) algorithm is proposed for reducing the complexity of interpolation, which interpolates over 2^η test vectors [7], being attractive for VLSI implementation [9]. The interpolation is simplified in LCC decoding by restricting the multiplicity to $m = 1$ and replacing the factorization step with Chien's search and Forney's algorithm. Furthermore, it is shown in [9] that LCC algorithm with eight test vectors (i.e., for $\eta = 3$) can achieve similar or higher coding gain than the Koetter-Vardy algorithm [1] with $m = 4$.

An interpolation architecture for LCC with $\eta = 3$, called backward interpolation, is proposed in [9], which could be considered as the best of the current approaches. Backward interpolator shares the computation of common points of the test vectors. These points are ordered in such way that a pair of adjacent vectors differ only at one point. Due to this feature, the backward interpolation architecture involves less area and provides higher speed than its prior ones.

Manuscript received June 03, 2010; revised September 04, 2010; accepted December 23, 2010. Date of publication February 04, 2011; date of current version February 17, 2012. This work was supported by FEDER and the Spanish Ministerio de Ciencia e Innovación, under Grant TEC2008-06787.

F. García-Herrero, M. J. Canet, and J. Valls are with the Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universidad Politécnica de Valencia, 46730 Gandia, Spain (e-mail: fragarh2@epsg.upv.es, macasu@eln.upv.es, jvalls@eln.upv.es).

P. K. Meher is with the Department of Embedded Systems, Institute for Infocomm Research, 138632, Singapore (e-mail: pkmeher@i2r.a-star.edu.sg).

Digital Object Identifier 10.1109/TVLSI.2010.2103961