A Semidigital Dual Delay-Locked Loop

Stefanos Sidiropoulos, *Student Member, IEEE*, and Mark A. Horowitz, *Senior Member, IEEE*

***Abstract*—This paper describes a dual delay-locked loop architecture which achieves low jitter, unlimited (modulo 2π) phase shift, and large operating range. The architecture employs a core loop to generate coarsely spaced clocks, which are then used by a peripheral loop to generate the main system clock through phase interpolation. The design of an experimental prototype in a 0.8-μm CMOS technology is described. The prototype achieves an operating range of 80 kHz–400 MHz. At 250 MHz, its peak-to-peak jitter with quiescent supply is 68 ps, and its jitter supply sensitivity is 0.4 ps/mV.**

***Index Terms*—Clock synchronization, delay-locked loops, phase interpolation, phase-locked loops.**

I. Introduction

PHASE-LOCKED loops (PLL's) and delay-locked loops (DLL's) are routinely employed in microprocessor and memory IC's in order to cancel the on-chip clock amplification and buffering delays and improve the I/O timing margins. However, the increasing clock speeds and integration levels of digital circuits create a hostile operating environment for these phase alignment circuits. The supply and substrate noise resulting from the switching of digital circuits affects the PLL or DLL operation and results in output clock jitter which subtracts from the I/O timing margins.

In applications where no clock synthesis is required, DLL's offer an attractive alternative to PLL's due to their better jitter performance, inherent stability, and simpler design. The main disadvantage of conventional DLL's, however, is their limited phase capture range. This paper presents a dual DLL architecture which combines several techniques to achieve unlimited phase capture range, low jitter and static-phase error, and four orders of magnitude operating frequency range. This architecture is based on a cascade of two loops. The core loop generates six clocks evenly spaced by 30° which are then used by the peripheral loop to generate the output clock, under the control of a digital finite state machine (FSM). By using phase interpolation, the dual loop can provide unlimited phase shift without the use of a voltage controlled oscillator (VCO). Using an FSM for phase control offers the advantage of enabling the flexible implementation of complicated phase capture algorithms in the digital domain. Finally, by utilizing self-biased techniques, the loop achieves large operating range and low jitter.

This paper begins with a brief overview of conventional DLL design. After outlining some of the disadvantages of conventional approaches, Section II presents the dual interpolating DLL architecture. Section III discusses circuit design issues that arose in the prototype implementation of the architecture in a 0.8-μm CMOS technology. Section IV discusses the experimental results, and concluding remarks follow in Section V.

Manuscript received April 10, 1997; revised June 5, 1997. This work was supported by ARPA under contract DABT63-94-C-0054.

The authors are with the Computer Systems Laboratory, Stanford University, Stanford, CA 94305 USA and with Rambus Inc., Mountain View, CA 94040 USA.

Publisher Item Identifier S 0018-9200(97)08033-5.

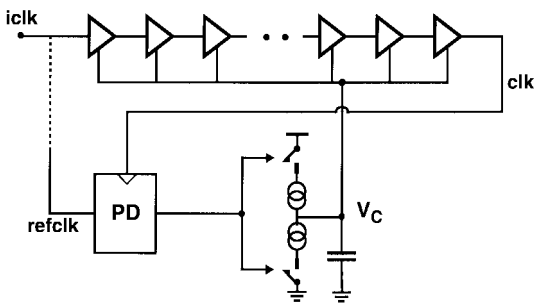


Fig. 1. Block diagram of a conventional DLL.

II. Architecture

A. Conventional DLL's

A simplified block diagram of a conventional DLL [1] is outlined in Fig. 1. The components are a voltage controlled delay line (VCDL), a phase detector, a charge pump, and a first-order loop filter. The input reference clock drives the delay line which comprises a number of cascaded variable delay buffers. The output clock *clk* drives the loop phase detector (depicted in this example as a conventional flip-flop). The output of the phase detector is integrated by the charge pump and the loop filter capacitor to generate the loop control voltage V_C. The loop negative feedback drives the control voltage to a value that forces a zero phase error between the output clock and the reference clock.

This simple design offers many advantages compared to VCO-based PLL's. Due to frequency acquisition constraints, PLL's usually resort to a specific type of phase detector, the state-machine-based phase frequency detector (PFD). In contrast, DLL's can be easily implemented by using "bang–bang" control—i.e., the control signal of the loop, rather than being proportional to the phase error magnitude, can simply be a binary "up" or "down" indication. Thus, in a "bang–bang" DLL the phase detector can be a replica of the input data receiver resulting in an optimal placement of the sampling clock in the center of the input receiver's sampling uncertainty window. Additionally, since DLL's do not use a VCO, phase errors induced by supply or substrate noise do not accumulate over many clock cycles. This improved noise immunity is

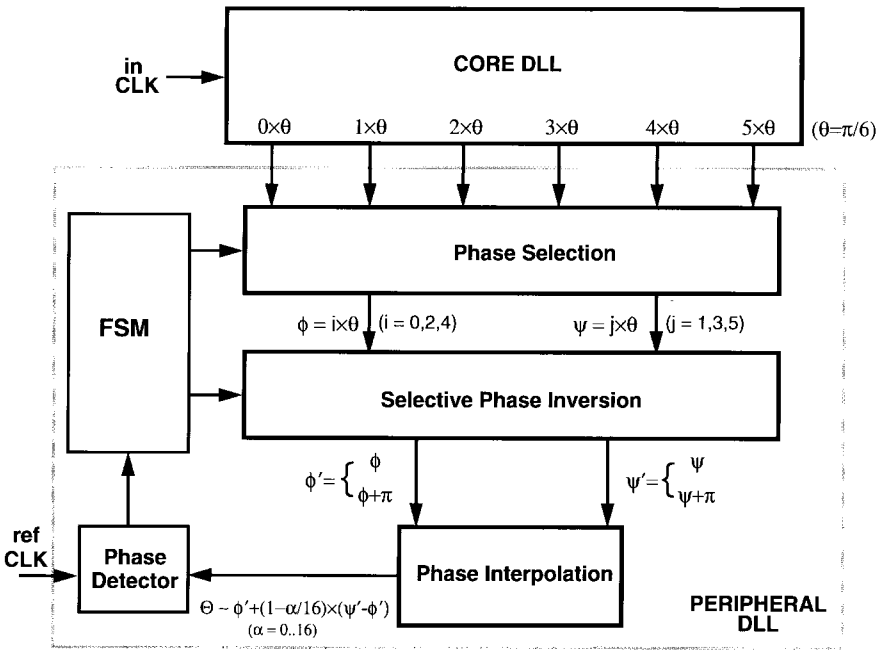


Fig. 2. Dual interpolating DLL architecture.

the main reason for the increased adoption of DLL's in applications that do not require clock synthesis.

The conventional DLL architecture of Fig. 1 suffers from two important disadvantages: clock jitter propagation and limited phase capture range. Since the VCDL simply delays the reference clock by a single clock cycle, the reference clock jitter directly propagates to the output clock. This all-pass filter behavior with respect to the frequency of the jitter of the reference clock results in reduced I/O timing margins, especially in "source-synchronous" interfaces where the reference clock emanates from another noisy digital chip. To overcome this problem, a separate low-jitter differential clock can be used as the input to the delay line. This way the on-chip common-mode noise and the reference clock jitter do not affect the I/O timing margins.

A more important problem is that a VCDL does not have the cycle slipping capability of a VCO. Therefore, at a given operating clock frequency, the DLL can delay its input clock by an amount bounded by a minimum and a maximum delay. As a consequence, extra care must be taken by the designer so that the loop will not enter in a state in which it tries to lock toward a delay which is outside these two limits. A compromising solution is to extend the VCDL range and use an FSM that controls the loop start-up. However, DLL's relying on quadrature phase mixing [2], [3] completely eliminate this problem. This approach is based on the fact that quadrature clocks can be easily generated, given a clock of the correct frequency. The quadrature clocks are then fed to a phase mixer which can produce a clock whose phase can span the complete 0–360° phase interval. This approach eliminates the limited phase range problem of conventional DLL's since it can essentially rotate the output clock phase infinite times providing seamless switching at the quadrant boundaries. The main disadvantage of quadrature mixing is that the output of the phase mixer is a clock with a slew rate inherently limited by $4 \times V_{\rm sw}/T$, where $V_{\rm sw}$ is the output swing of the phase mixer and T the period of the clock. This slow clock exhibits increased dynamic noise sensitivity, thus degrading the jitter performance of quadrature mixing DLL's.

The approach presented here overcomes this limitation of quadrature mixing DLL's since it generates the output clock by interpolating between smaller 30° phase intervals [5]. Simultaneously, by avoiding the use of a VCO it eliminates the phase error accumulation problem of similar approaches [4].

B. Dual Interpolating DLL

Fig, 2 shows a high-level block diagram of the proposed architecture. This architecture is based on cascading two loops. A conventional first-order core DLL is locked at 180° phase shift. Assuming that the delay line of the core DLL comprises six buffers, their outputs are six clocks which are evenly spaced by 30°. The peripheral digital loop selects a pair of clocks, ϕ and ψ, to interpolate between. Clocks ϕ and ψ can be potentially inverted in order to cover the full 0–360° phase range. The resulting clocks, ϕ' and ψ', drive a digitally controlled interpolator which generates the main clock Θ. The phase of this clock can be any of the N quantized phase steps between the phases of clocks ϕ' and ψ', where $0 \cdots N$ is the interpolation controlling word range.

The output clock Θ of the interpolator drives the phase detector which compares it to the reference clock. The output of the phase detector is used by the FSM to control the phase selection, the selective phase inversion, and the interpolator phase mixing weight. The FSM moves the phase of the clock Θ according to the phase detector output. In the more common case this means just changing the interpolation mixing weight by one. If, however, the interpolator controlling word has reached its minimum or maximum limit, the FSM must change

the phase of clock ϕ or ψ to the next appropriate selection. This phase selection change might also involve an inversion of the corresponding clock if the current interpolation interval is adjacent to the 0° or 180° boundary. Since these phase selection changes happen only when the corresponding phase mixing weight is zero, no glitches occur on the output clock. The digital "bang–bang" nature of the control results in dithering around the zero phase error point in the lock condition. The dither amplitude is determined by the interpolator phase step and the delay through the peripheral loop.

In this architecture the output clock phase can be rotated, so no hard limits exist in the loop phase capture range: the loop provides unlimited (modulo 2π) phase shift capability. This property eliminates boundary conditions and phase relationship constraints, common in conventional DLL's. The only requirement is that the DLL input clock and the reference clock are plesiochronous (i.e., their frequency difference is bounded), making this architecture suitable for clock recovery applications. Since the system does not use a VCO, it does not suffer from the phase error accumulation problem of conventional PLL's. Moreover, the input clocks of the phase interpolator are spaced by just 30°, so the output of the phase interpolator does not exhibit the noise sensitivity of the quadrature mixing approach. Finally, the fact that the capture algorithm can be completely implemented in the digital domain gives great flexibility in its implementation as will be discussed in Section III. Although the prototype described in this paper is implemented with an analog core loop, possible implementations of the architecture can use digital control in both loops, further enhancing the system versatility. Moreover, the architecture can be easily extended to use a clock recirculating scheme in the core loop, so that the output clock frequency is a multiple of the input clock [7].

C. Dual-Loop Dynamics

Cascading two loops can compromise the overall system stability and lead to undesired jitter peaking effects. However, as the analysis in this section will show, this dual-loop architecture does not exhibit any jitter peaking irrespective of the dynamics of the two loops. The behavior of the DLL can be analyzed with respect to two types of perturbations: i) input or reference clock delay variations and ii) delay variations resulting from supply and substrate noise. The frequency response of the dual loop can be analyzed by making a continuous time approximation, in which the sampling operation of the phase detectors and the digital nature of the peripheral loop are ignored. This approximation is valid for core and peripheral loop bandwidths at least a decade below the operating frequency. This constraint needs to be satisfied anyway in a DLL in order to eliminate the effects of higher order poles resulting from the delays around loop.

Fig. 3 shows the dual loop linearized model including both the loop clocks $D_{\rm IN}(s)$ and $D_{\rm REF}(s)$, and delay errors introduced by supply or substrate noise $D_{\rm N}(s)$. Each of the two loops is modeled as a single pole system, in which the input, output, and error variables are delays, similar to the single-loop analysis published in [7]. For example, the output delay of the

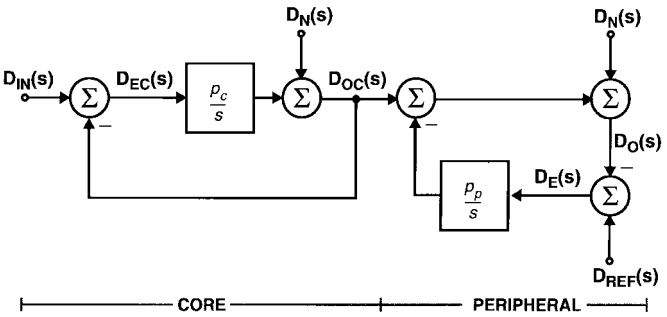


Fig. 3. Linearized dual DLL model.

core loop $D_{\rm OC}(s)$ (in seconds) is the delay established by the core loop delay line, while the input delay $D_{\rm IN}(s)$ is the delay for which the core loop phase detector and charge pump do not generate an error signal. Since the core loop VCDL spans half a clock cycle, $D_{\rm IN}(s)$ is equal to half an input clock period. By using these loop variables, the input-to-output transfer function of the core loop can be easily derived

$$\frac{D_{\rm OC}(s)}{D_{\rm IN}(s)} = \frac{1}{1+s/p_c} \tag{1}$$

where p_c (in rads/s) is the pole of the core loop as determined by the charge pump current, the phase detector and delay line gain, and the loop filter capacitor. Similarly, the noise-to-delay error transfer function of the core loop can be shown to be

$$-\frac{D_{\rm EC}(s)}{D_{\rm N}(s)} = \frac{s/p_c}{1+s/p_c} \tag{2}$$

where $D_{\rm N}(s)$ is the additional delay introduced in the core loop from supply or substrate noise, and $D_{\rm EC}(s)$ is the delay error seen by the core loop phase detector. This transfer function indicates that noise induced delay errors can be tracked up to the loop bandwidth and that the response of the loop to a supply step consists of an initial step followed by a decaying exponential with a time constant equal to $1/p_c$.

Before proceeding to analyze the response of the dual loop, it should be noted that the linearized model of Fig. 3 uses a simplifying assumption. The assumption is that the delay error $D_{\rm N}(s)$ introduced by supply or substrate variations is identical in both loops and does not depend on the state of the phase selection multiplexers. Since the supply and substrate sensitivity of the peripheral loop depends on the phase selection and will be typically higher due to the presence of the final CMOS system clock buffer, this assumption is not necessarily accurate. However, it does not affect the conclusions drawn below about the stability of the loop, since it only removes a modifying constant, which is equal to the ratio in the delay sensitivities of the two loops. This constant only affects the relative location of the poles and zeros of the resulting transfer function, and, as it will be shown below, the loop is unconditionally stable irrespective of the relation between the individual poles and zeros. Using the model of Fig. 3, it is straightforward to show that the transfer function $D_{\rm O}(s)/D_{\rm REF}(s)$ of the peripheral loop is identical in form to that of the core loop. This result agrees with intuition since in the dual loop system reference clock perturbations do not

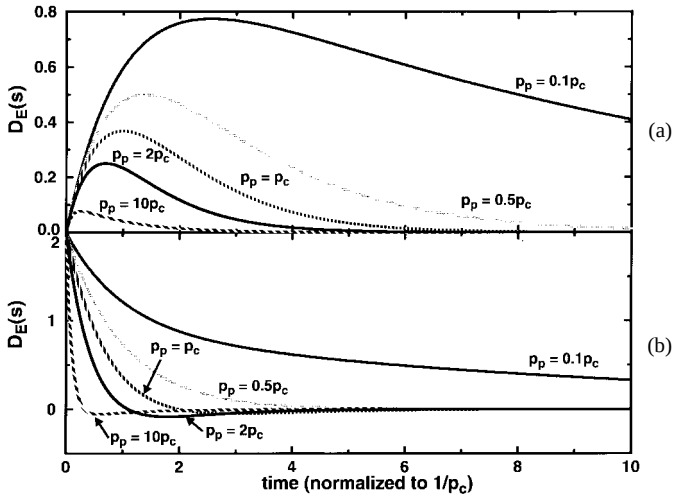


Fig. 4. Dual loop response to: (a) step change in input clock and (b) supply noise step.

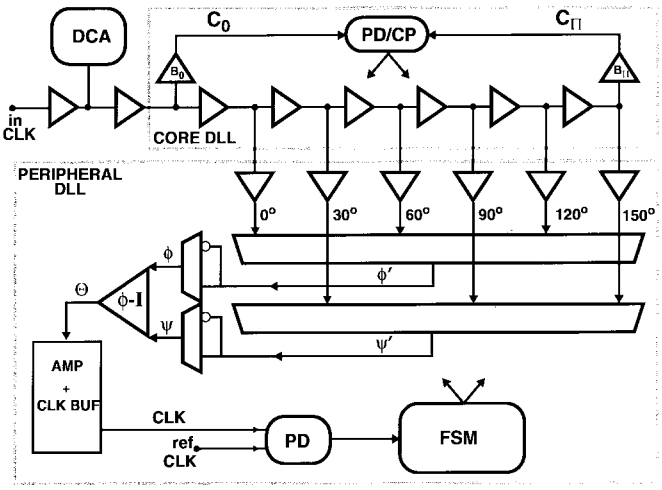


Fig. 5. Dual DLL detailed block diagram.

affect the core loop. More interesting is the transfer function of the input clock $D_{\mathrm{IN}}(s)$ to dual-loop error $D_{\mathrm{E}}(s)$ since changes in the period of the input clock will cause both the core and peripheral loop to react. Based on (1) and (2), this transfer function can be shown to be

$$-\frac{D_{\mathrm{E}}(s)}{D_{\mathrm{IN}}(s)} = \frac{s/p_p}{(1+s/p_c)\cdot(1+s/p_p)}. \tag{3}$$

This bandpass transfer function exhibits no peaking at any frequency regardless of the relative magnitudes of p_c and p_p. The step response of the system, shown in Fig. 4(a), reveals that unit-step changes in $D_{\mathrm{IN}}(s)$ (i.e., step changes in the input clock period) will initially peak at a less than unity value determined by the ratio of the two poles.[1] Moreover, as the magnitude of p_p increases, the disturbance on the output is reduced since the peripheral loop compensates quickly for disturbances at the output caused by changes of the input clock.

Finally, the transfer function from supply or substrate noise-induced delay errors $D_{\mathrm{N}}(s)$ to the delay error of the dual loop $D_{\mathrm{E}}(s)$ can be derived

$$-\frac{D_{\mathrm{E}}(s)}{D_{\mathrm{N}}(s)} = \frac{(1+2s/p_c)\cdot s/p_p}{(1+s/p_c)\cdot(1+s/p_p)}. \tag{4}$$

Equation (4) also exhibits no peaking at any frequency since the location of the last zero can never be above that of the poles. The step response of the system is plotted in Fig. 4(b) for various ratios of the core to peripheral pole frequencies. Under all conditions, the initial delay error is equal to twice the injected unity error $D_{\mathrm{N}}(s)$ since this error is added on both loops. When the peripheral loop bandwidth is less than half that of the main loop, there is no overshoot in the dual-loop step response. This result occurs because the core loop compensates for its delay error quickly, while the slower peripheral loop compensates for the output delay error later. When the pole frequencies of the two loops are very close, the system overshoots since the peripheral loop compensates for the output delay error at approximately the same rate as the peripheral loop. The worst case overshoot of approximately 4.5% of the initial disturbance occurs when the peripheral loop bandwidth is twice that of the core loop. As the peripheral loop bandwidth increases, the overshoot becomes progressively smaller since the peripheral loop corrects for both the peripheral and core delay errors. Subsequently, the influence of the slower core loop correction on the output delay error is compensated by the peripheral loop. Therefore, even in the worst case, the dual loop cascade exhibits only minor overshoot.

[1] It should be noted that in case *in-CLK* and *ref-CLK* are identical or correlated, the resulting transfer function exhibits a low-pass peaked behavior. Nevertheless the resulting peaking is small, exhibiting a maximum of 15% when $p_p = p_c$, while it is less than 5% as long as p_p and p_c are an order of magnitude apart in frequency.

III. Circuit Design

A. Overview

A more detailed block diagram of the dual loop is shown in Fig. 5. This design uses a separate local differential clock as the input to the delay line. Although the use of this clock is not inherent in the loop architecture, it minimizes the supply sensitivity in applications such as "source synchronous" interfaces. To minimize the effects of input clock duty cycle imperfections and common-mode mismatches, a duty cycle adjuster (DCA) [2] is employed after the first clock receiving buffer. The 50% duty cycle clock drives the core DLL. The core delay line consists of six differential buffers. An extra pair of buffers B_0, B_Π generate two clocks which drive the core loop 180° phase detector. The output of the phase detector controls the charge pump which forces clocks C_0 and C_Π to be 180° out of phase. Since all the buffers in the core delay line (including B_0 and B_Π) have the same size, all the core VCDL stages have the same fan-out and delay. Therefore, forcing C_0 and C_Π to be 180° out of phase will generate six evenly spaced by 30° clocks at the outputs of the core delay line.

The phase selection and phase inversion multiplexers are differential elements controlled by the core loop control voltage. In order to eliminate jitter-sensitive slow clocks, all buffers in the clock path need to have approximately the same

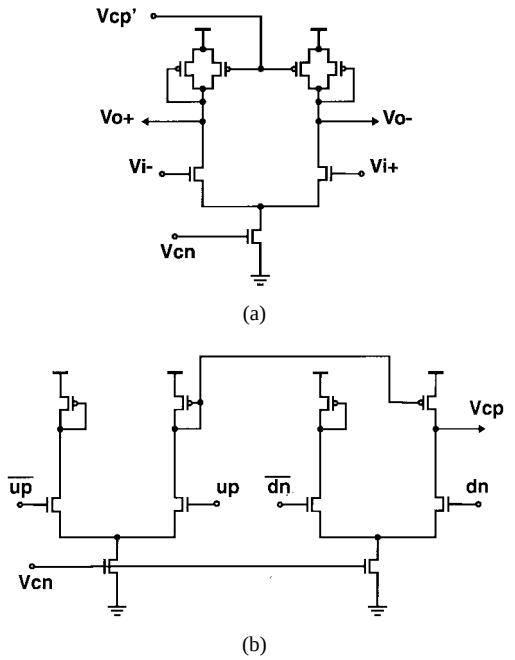


Fig. 6. (a) Core loop delay buffer and (b) charge pump.

bandwidth. For this reason, the phase selection in this design is implemented as a combination of a 3-to-1 and a 2-to-1 multiplexers, instead of a single 6-to-1 differential multiplexer with lower total power. Since the phase selection multiplexer can affect the phase shift of the core delay line through data-dependent loading, the six output clocks are buffered before driving the phase selection multiplexers. This way, changing the multiplexer select does not affect the core delay line phase shift.

The outputs ϕ, ψ of the phase inversion multiplexer drive the phase interpolator which generates the low swing differential clock Θ. This clock is then amplified and buffered through a conventional CMOS inverter chain generating the main clock (CLK). The peripheral loop phase detector [1] compares that clock to the reference clock, generating a binary phase error indication that is then fed to the FSM. The FSM based on the phase detector (PD) output selects phases ϕ, ψ and controls the phase interpolation.

B. Core Loop

To minimize the jitter supply sensitivity, all the delay buffers in the design, from the input clock (*in-CLK*) to the output of the phase interpolator (Θ), use differential elements with replica feedback biasing [6]. In order to linearize the loop gain and obtain large operating range, the core loop charge pump current is scaled along with the VCDL buffer current as illustrated in Fig. 6 [7]. Voltage $V_{\rm cn}$ is generated through the replica-feedback biasing circuit while $V'_{\rm cp}$ is a buffered version of the charge pump control voltage $V_{\rm cp}$. In addition to the core VCDL buffers, voltages $V'_{\rm cp}$ and $V_{\rm cn}$ control the differential buffer elements of the peripheral loop. This ensures that all the buffers in the design have approximately equal delays and that the edge rates of the interpolator input clocks (ϕ, ψ) scale with the operating frequency of the loop.

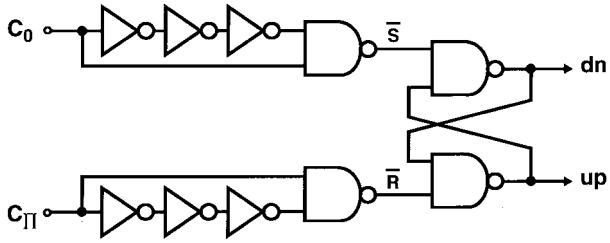


Fig. 7. Core loop phase detector.

The sensitivity of the dual-loop architecture to the core loop phase offset depends on the particular application. For the case that the dual DLL is used to just generate a clock whose phase is directly controlled by the phase detector output, the phase offset of the core loop does not affect the system phase offset. In this case, the loop operation will not be affected as long as the core loop phase offset is bounded. An absolute core loop offset less than 30° ensures monotonic switching at the 0° and 180° interpolation boundaries, so the interpolating loop functions correctly, albeit with a larger than nominal interpolation phase step. Core loop phase offsets larger than this amount will result in a hysteretic locking behavior at the quadrant boundaries, which will increase the dither jitter if the reference clock phase forces the dual loop to lock at this point.

The dual-loop operation becomes more sensitive to core loop phase offsets in case the designer wants to use this architecture to generate an additional clock that is offset by 90° relative to the reference clock. In such an application, the quadrature clock would be generated by using an extra pair of phase selection and inversion multiplexers whose selects would be offset by three relative to those generating the main clock. This would create a 90° interpolation interval offset, resulting in the required quadrature phase shift. In this case the core loop phase offset would impact the quadrature phase if the selects of the extra multiplexers happen to wrap around the 0° or 180° interpolation interval boundaries.

Even though the prototype does not implement quadrature phase generation, a low offset phase detector and careful matching of the layout were used to ensure uniform spacing of the six clocks. A self-biased DLL requires a linear phase detector. To avoid start-up problems that would result from the use of a conventional state machine PFD [7], the core loop uses the phase detector depicted in Fig. 7. This design comprises an S–R latch augmented with two input pulse generators. The absence of extra state storage in this design eliminates any start-up false locking conditions. Additionally, its symmetric structure and the use of pulse triggering minimize the core loop phase offset.

The core of the phase detector is an S–R latch-based phase detector. The S–R latch ensures a 180° phase shift between the falling edges of its inputs only when the duty cycle of the two input clocks is identical. However, when the duty cycle of the two input clocks is different, this mismatch will propagate as a core loop phase locking offset. This happens because an unbalanced overlap of the two input clocks causes the output of the S–R latch to have a duty cycle deviating from 50%. To compensate for this effect, the S–R latch is

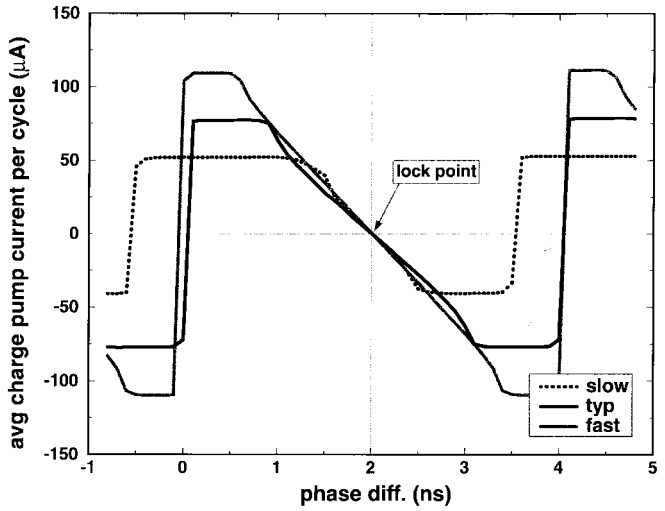


Fig. 8. Phase detector and charge pump simulated transfer function.

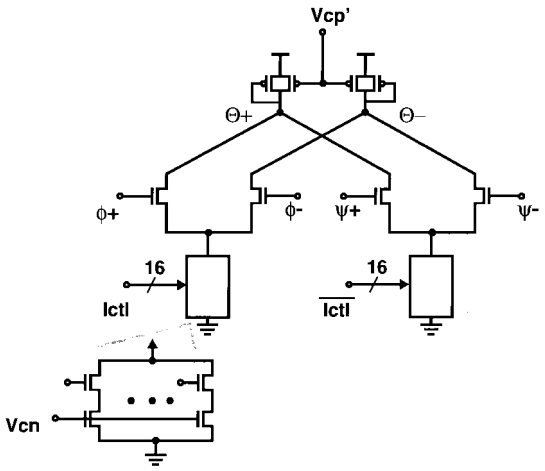


Fig. 9. Phase interpolator (type-I) schematic.

augmented with two pulse generators which propagate a low pulse on the positive edges of the input clocks. Since potential overlaps are minimized, the design can tolerate large duty cycle imperfections and still provide an accurate 180° lock in the core loop.

Fig. 8 shows the simulated transfer characteristics of the phase detector and charge pump over three extreme process and environment conditions. The cycle time of the two input clocks is set at 4 ns, while their duty cycles are mismatched by 0.5 ns such that the duty cycle of C_0 is 37.5% while the duty cycle of clock C_Π is 62.5%. It can be seen that the transfer function is linear and has no offset or dead-band around the 2-ns point where the loop actually locks. However, the combination of input pulsing and duty cycle imperfections results in nonlinear transfer function characteristics at the vicinity of the boundaries of the locking range (i.e., 0 and 4 ns). The only effect of this nonlinearity is that the core loop can exhibit an initial slew-rate limited reduction of its phase error, since the output of the phase detector and charge pump is constant. After the phase error has been reduced, such that the phase detector operates within its linear region, the core loop will exhibit a conventional single-pole response. Harmonic locking problems, common in PLL's using S–R phase detectors, are eliminated in this design since the core loop is reset to its minimum delay at system start-up.

C. Phase Interpolator Design

The most critical circuit in the design of the peripheral digital loop is the phase interpolator. The phase interpolator receives two clocks ϕ, ψ and generates the main clock Θ whose phase is the weighted sum of the two input phases. Essentially, the phase interpolator converts a digital weight code generated from the FSM to the phase of clock Θ. Linearity is not important in the design of this digital-to-phase converter since it is enclosed in the peripheral loop feedback. The important requirement is that the interpolation process is monotonic to ensure that no hysteresis exists in the loop locking characteristics. Additionally, the phase step must be minimized since it determines the loop dither amplitude. In this case, the interpolation step is 1/16 of the 30° interval resulting in approximately 2° peripheral loop nominal dither. Another important requirement is that the design should provide for seamless interpolation-boundary switching. This means that when the input code is such that the weight on one of the input clocks is zero, this clock should have no influence on the output.

Fig. 9 shows a schematic diagram of the interpolator used in the prototype chip. This design is a dual input differential buffer which uses the same symmetric loads as all the core VCDL buffers and peripheral loop multiplexers. The bias voltages $V'_{\rm cp}$ and $V_{\rm cn}$ are identical with those biasing the rest of the loop, ensuring that its total delay is approximately 30° of the clock period which is the same as the rest of the loop buffers. Therefore, the transition time of the interpolator input clocks is larger than the minimum delay through the interpolator, and the two input transitions overlap. This condition ensures that the interpolator outputs never settle at half of the swing range. The current sources of the two differential pairs are thermometer controlled elements. The thermometer codes are generated by a 16-b long up/down shift register which is controlled by the peripheral loop FSM. By changing the thermometer code, the FSM adjusts in a complementary fashion the currents of the two input differential pairs resulting in a mixing of the two input clock phases. This design (type-I) does not completely satisfy the seamless boundary-switching requirement. Even when the current through one of the differential pairs is zero, the input still influences the output of the interpolator. This influence is due to the capacitive coupling of the gate-to-drain capacitance of the differential pair input transistors.

Fig. 10 shows an alternative design which does not suffer from this problem. In this design (type-II), the interpolator differential pairs consist of unit cell differential pairs. Therefore, when one of the interpolation weight thermometer codes is zero, the corresponding input is completely cut off from the output, eliminating the gate-to-drain coupling capacitance.

Fig. 11 shows the simulated transfer function of the interpolator alternative designs. This simulation includes random (<20 mV) threshold voltage offsets in the thermometer code

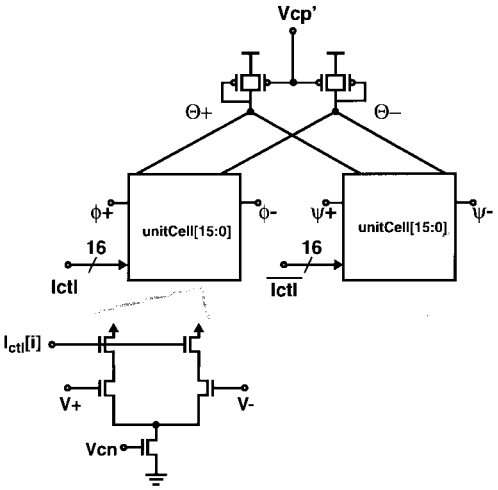


Fig. 10. Phase interpolator (type-II) schematic.

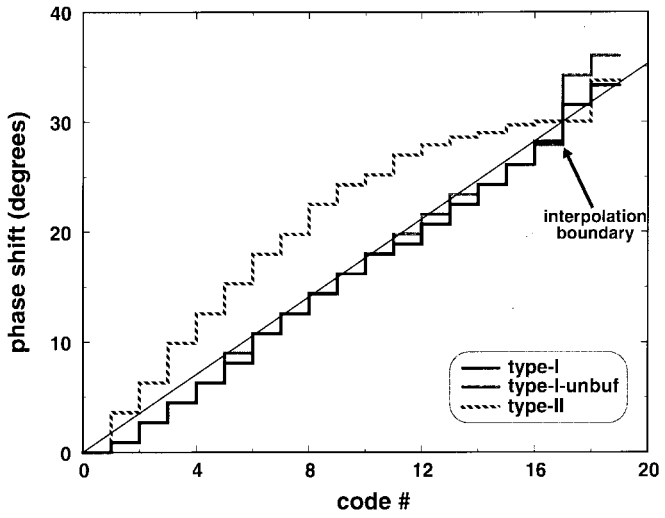
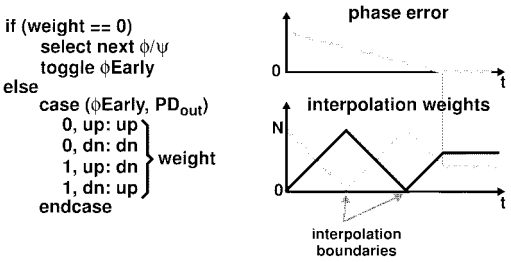


Fig. 11. Simulated phase interpolator transfer function.



(a) (b)

Fig. 12. (a) Simplified FSM algorithm and (b) resulting loop behavior.

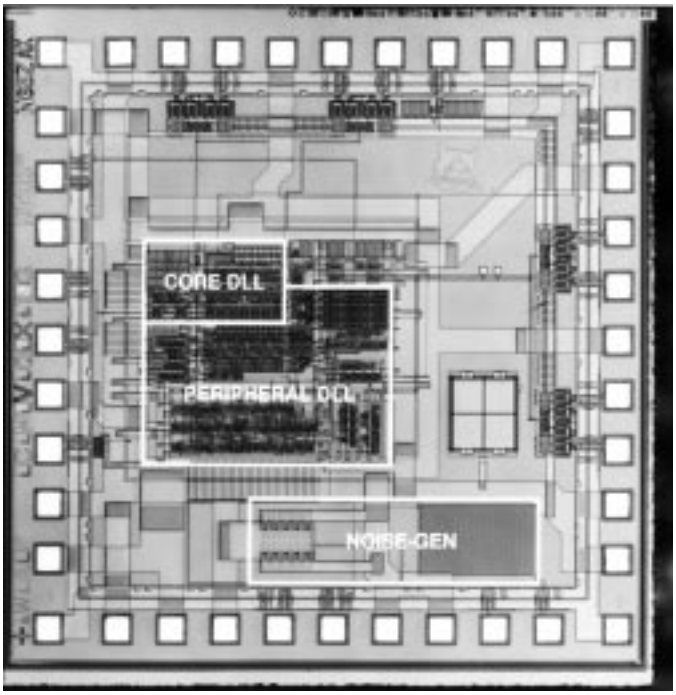


Fig. 13. Prototype chip microphotograph.

current sources. The type-I design exhibits a nominal step of approximately 2°. However, due to the gate-to-drain capacitive coupling effect, the maximum step of 3.8° occurs at the interpolation boundary when the input clock ϕ is switched to the next selection. In the lower power implementation where no buffering is used at the core delay line outputs (type-I-unbuf), the data-dependent loading on the previous stage results on a double phase step at the interpolation interval boundaries. Although the alternative design (type-II) does not exhibit a boundary phase step, it was not used since it occupies more layout area and exhibits more nonlinear characteristics due to data-dependent loading of the previous stage. So in the present implementation, worst-case dithering occurs at the interpolation interval boundaries and has an approximate magnitude of 3.8°.

D. Finite State Machine

A simplified version of the peripheral loop FSM algorithm is outlined in Fig. 12(a). The single state ϕEarly of the FSM indicates the relationship of the two interpolator input clocks. On every cycle of its operation, the FSM might undertake two actions.

- In the more frequent case of in-range interpolation (i.e., weight $\neq$ 0), the FSM simply increments or decrements the interpolation weight by shifting up or down the interpolator controlling shift register. The direction of the shift is decided based on the phase detector output and the current value of the state ϕEarly.
- If the peripheral loop has run out of range in the current interpolation interval, the FSM seamlessly slides the current interpolation interval by switching phase ϕ or ψ to the next selection. The fact that the interpolation has run out of range in the current interval is simply indicated by a combination of the current value of the state ϕEarly, the most or least significant bit of the thermometer register, and the output of the phase detector. In case the current selection of phase ϕ or ψ is adjacent to the 0 or 180° interpolation interval boundary, switching to the next selection involves toggling the select of the second-stage phase inversion multiplexer.

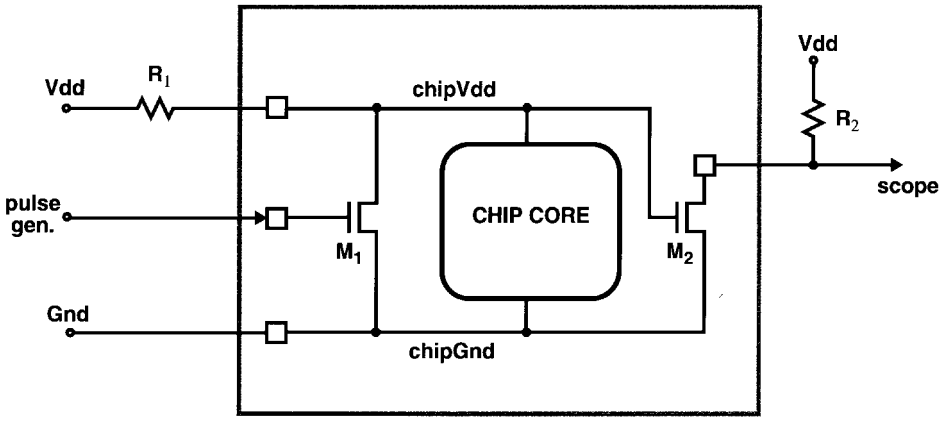


Fig. 14. Noise generation and monitoring circuits.

The loop phase capture behavior resulting from this simple algorithm is illustrated in Fig. 12(b). The phase error decreases at a linear rate until the system achieves lock. Subsequently, the loop dithers around the zero phase error point with a dither magnitude of one phase interpolation interval. This occurs because in this type of "bang–bang" system, the output of the phase detector is just a binary phase error without any indication of the magnitude of the phase error. The complementary interpolation weights slew linearly, changing direction at the interpolation interval boundaries. Once the system finds lock, they either dither by one or they stay constant if the dither point happens to lie on an interval boundary.

The magnitude of the peripheral loop phase dither is determined by the minimum interpolation step and the delay through the feedback loop. In conventional analog "bang–bang" DLL's, the loop delay is largely determined by the delay through the delay line and the clock distribution network. However, this digital implementation has a larger minimum loop delay. The underlying reason is that driving the FSM directly from the phase detector output might lead into metastability problems, especially since the whole loop operation is driving the phase detector to its metastable point of operation. For this reason, in this implementation the output of the phase detector is delayed by three metastability hardened flip-flops. This increases the mean time between failures (MTBF) of the system to a calculated worst case of approximately 100 years, but at the same time increases the peripheral loop delay by three cycles. To compensate for that delay and decrease the loop dither, the FSM logic implements a front-end filter which counts eight continuous phase detector "up" or "down" results before propagating this signal to the core FSM. This causes the FSM to delay its next decision until the results of its previous action have been propagated to the phase detector output and reduces the inherent peripheral loop dither to one phase interpolation interval.

The digital nature of the peripheral loop control enabled the implementation of the FSM to be done through synthesis of a behavioral verilog model followed by a simple standard cell place and route. The FSM behavioral model was verified by simulation in conjunction with a behavioral core loop model. The significance of this automated methodology is that other more complicated algorithms can be implemented requiring minimal effort from the designer. Faster phase acquisition can be obtained by disabling the front end counter/filter and changing the interpolation step by a larger amount while the loop is not in lock. The loop can also implement a periodic phase calibration algorithm. In this case, the FSM is activated initially to drive the loop to zero phase error. Then it is shut down to save power and it is periodically turned on to compensate for slow phase drifts. Since the FSM can run at a frequency slower than that of the system clock, the implementation of different algorithms is not in the system critical path.

IV. Experimental Results

To verify the dual DLL architecture, a chip has been fabricated through MOSIS in the HP CMOS26B process. This is a 1.0-μm drawn process with the channel lengths scaled to 0.8 μm. Although the gate oxide in this process is $\sim$170 Å allowing 5-V operation, the loop design and testing was done with a 3.3-V power supply voltage.

Fig. 13 is a micrograph of the chip. The chip integrates the dual DLL, along with noise injection and monitoring circuits and current-mode differential output buffers. The dual DLL occupies 0.8 mm^2 of silicon area, the majority ($\sim$60%) of which is devoted to the peripheral loop logic. This is mainly due to the relatively large standard cell size of the library used in this implementation.

The block labeled NOISE-GEN in Fig. 13 is used to inject and measure on-chip supply noise. Fig. 14 shows a schematic diagram of these circuits. The 1000-μm wide transistor M_1 shorts the on-chip supply rails creating a voltage drop across the off-chip 4-Ω resistor R_1. In order to monitor the droop on the on-chip supply, device M_2 and the external 5-Ω load resistor R_2 form a broadband attenuating buffer which drives the 50-Ω scope. The gain of the buffer is computed during an initial calibration step. The use of these circuits enables the injection and monitoring of fast ($<$1-ns rise time) steps on the on-chip supply.

The dither jitter of the loop with quiescent on-chip supply varies with the input phase. This occurs because the offset of the interpolator and the phase selection multiplexers change according to the point of lock. Fig. 15 shows the worst-

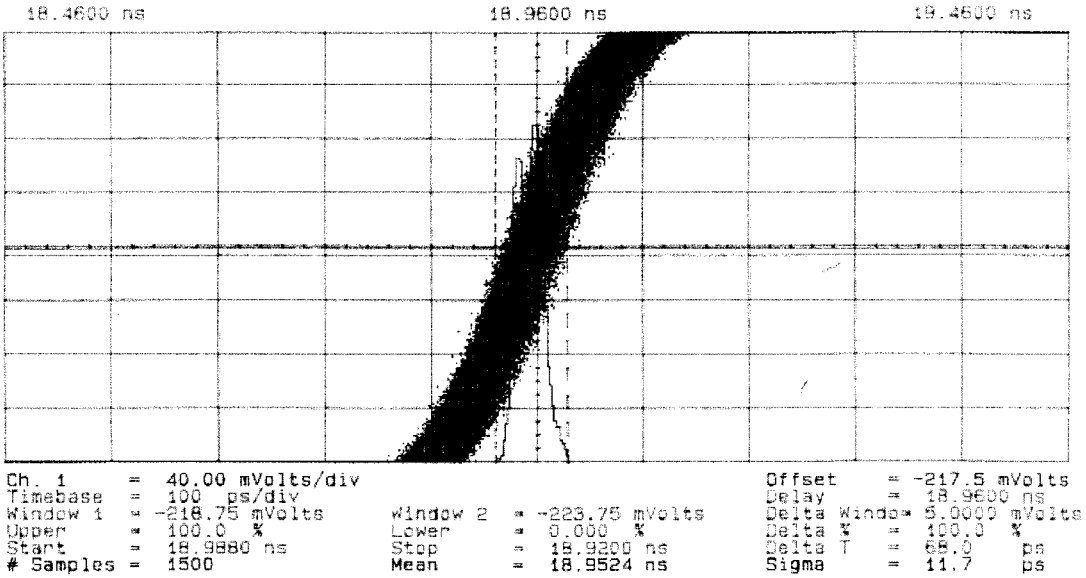


Fig. 15. Jitter histogram with quiet supply.

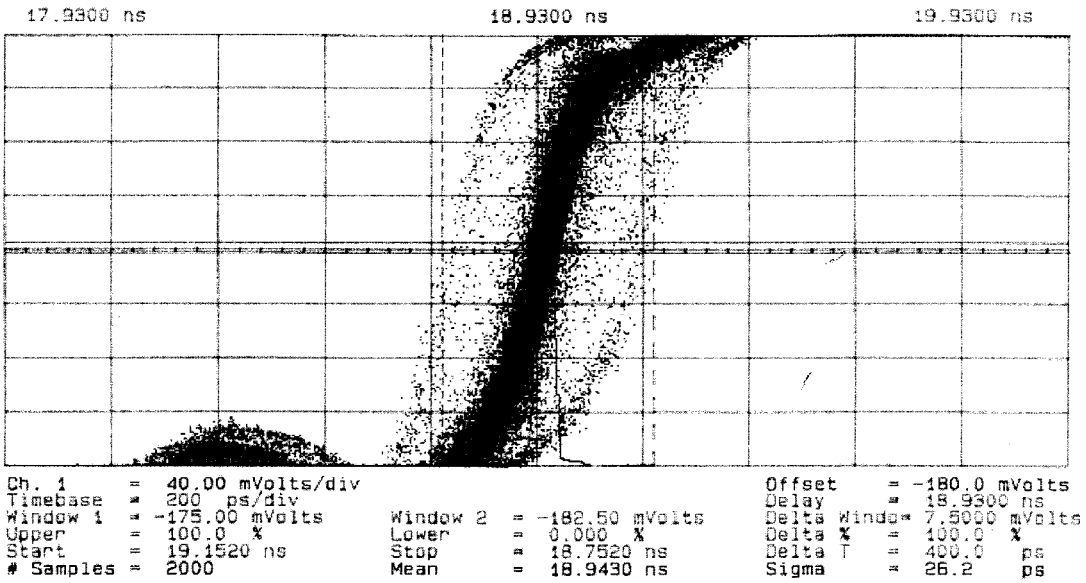


Fig. 16. Jitter histogram with 1-MHz 750-mV square wave supply noise.

case jitter (68 ps) with quiescent supply. The jitter histogram consists of the superposition of two Gaussian distributions resulting from the switching of the peripheral loop between two adjacent interpolation intervals. The distance between the peaks of the two superimposed distributions is about 40 ps, which is in fair agreement with the simulation results. With the noise generation circuits injecting a 750-mV 1-MHz square wave on the chip supply, the peak-to-peak jitter increases to 400 ps (Fig. 16). It should be noted that simulation results indicate that approximately 50% of this jitter is not inherent to the loop, but is due to the supply sensitivity of the succeeding static CMOS clock buffer and off-chip driver.

Fig. 17 illustrates the linearity of the interpolation process in the peripheral loop. The figure shows the histogram of the output clock with the peripheral loop FSM continuously rotating that clock. The histogram was generated by keeping the reference clock to a constant voltage while the input clock ran at its nominal frequency of 250 MHz. The histogram valleys correspond to the interpolation interval boundaries. The spacing of the valleys is within 10% of their nominal 333-ps distance, indicating good matching of the delays of the core loop buffers. The absence of one valley at the 180° interpolation boundary indicates a slight offset in the core loop. The fact that the magnitude of the highest peak of the histogram is smaller than the magnitude of the deepest valley indicates that the interpolator achieves the 4-b target linearity (the 4-b linearity of the interpolator was also confirmed by a similar histogram of a single interpolation interval). Thus the overall linearity of the DLL is limited by the steps at the interpolation interval boundaries.

Table I summarizes the loop performance characteristics. With a 3.3-V supply, the loop operates from 80 kHz to

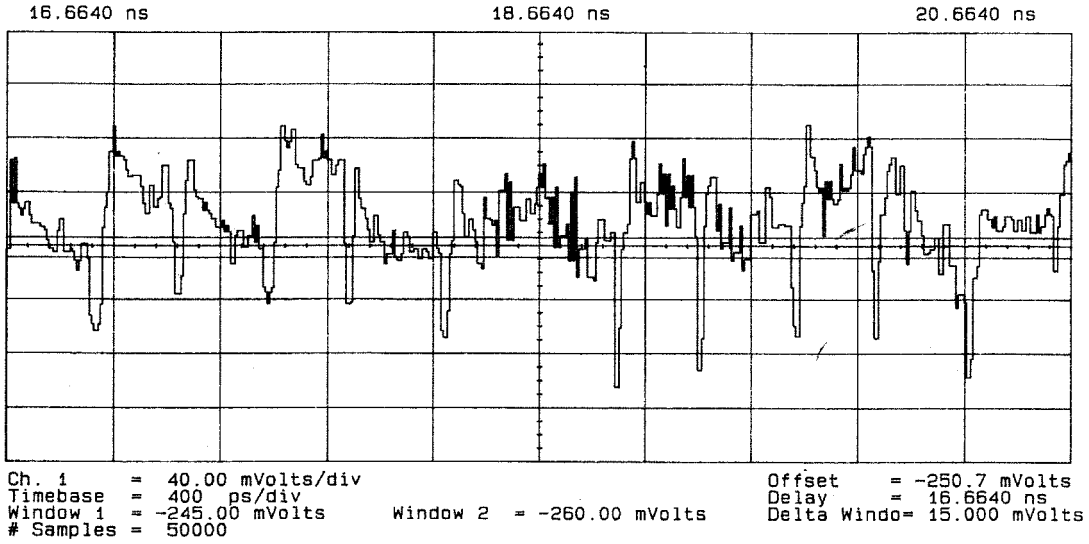


Fig. 17. Interpolation process linearity.

TABLE I
PROTOTYPE PERFORMANCE SUMMARY

Process	**1 μm (drawn), 0.8 μm (effective) CMOS nwell**
Active Area	**0.8 mm^2**
Supply Voltage	**3.3 V**
Power Dissipation	**102 mW (@ 250 MHz)**
Operating Range	**0.08-400 MHz**
Phase Offset	**<40 ps**
Jitter	**68 ps pk-pk/11 ps RMS (@250 MHz)**
Supply sensitivity	**0.4 ps/mV (@250 MHz-closed loop)**

400 MHz. The phase offset between the reference clock and the output clock of the loop is less than 40 ps. Operating at 250 MHz, the dual DLL draws 31 mA dc from a 3.3-V power supply.

V. SUMMARY

Although DLL's are easier to design than PLL's and offer better jitter performance, their main disadvantage is their limited phase capture range. This disadvantage limits their application to completely synchronous environments and complicates start-up circuitry. This paper presented a dual DLL architecture which removes this limitation by using a core DLL to generate coarsely spaced clocks which are then used by a peripheral DLL to generate the output clock by using phase interpolation. This architecture has unlimited (modulo 2π) phase shift capability, therefore removing boundary conditions and phase relationship constraints between the system clocks. The only requirement is that the DLL input and reference clocks are plesiochronous, making the dual DLL suitable for clock recovery applications. In addition, the digital nature of the peripheral loop control enables implementation of complicated phase alignment algorithms in a straightforward manner.

A prototype using a linear self-biased core loop has been implemented in a 0.8-μm technology. The prototype achieves 68-ps peak-to-peak jitter, 0.4-ps/mV supply sensitivity, and 0.08–400 MHz operating range.

ACKNOWLEDGMENT

The authors are grateful to M. Johnson, T. Lee, J. Maneatis, and K. Yang for helpful discussions.

REFERENCES

[1] M. Johnson and E. Hudson, "A variable delay line PLL for CPU-coprocessor synchronization," *IEEE J. Solid-State Circuits*, vol. 23, Oct. 1988.
[2] T. Lee *et al.*, "A 2.5 V CMOS delay-locked loop for an 18 Mbit, 500 MB/s DRAM," *IEEE J. Solid-State Circuits*, vol. 29, pp. 1491–1496, Dec. 1994.
[3] M. Izzard *et al.*, "Analog versus digital control of a clock synchronizer for a 3 Gb/s data with 3.0 V differential ECL," in *Dig. Tech. Papers 1994 Symp. VLSI Circuits*, June 1994, pp. 39–40.
[4] M. Horowitz *et al.*, "PLL design for a 500 MB/s interface," in *Dig. Tech. Papers Int. Solid State Circuits Conf.*, Feb. 1993, pp. 160–161.
[5] S. Sidiropoulos and M. Horowitz, "A semi-digital delay locked loop with unlimited phase shift capability and 0.08–400 MHz operating range," in *Dig. Tech. Papers Int. Solid State Circuits Conf.*, Feb. 1997, pp. 332–333.
[6] J. Maneatis and M. Horowitz, "Precise delay generation using coupled oscillators," *IEEE J. Solid-State Circuits*, vol. 28, pp. 1273–1282, Dec. 1993.
[7] J. Maneatis, "Low-jitter process-independent DLL and PLL based on self-biased techniques," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1723–1732, Nov. 1996.

Stefanos Sidiropoulos (S'93), for a photograph and biography, see p. 690 of the May 1997 issue of this JOURNAL.

Mark A. Horowitz (S'77–M'78–SM'95), for a photograph and biography, see p. 690 of the May 1997 issue of this JOURNAL.