Journal of Electrical and Computer Engineering

Clock/Frequency Generation Circuits and Systems

Guest Editors: Woogeun Rhee, Antonio Liscidini, Jae-Yoon Sim, Kenichi Okada, and Sudhakar Pamarti

Clock/Frequency Generation Circuits and Systems

Journal of Electrical and Computer Engineering

Clock/Frequency Generation Circuits and Systems

Guest Editors: Woogeun Rhee, Antonio Liscidini, Jae-Yoon Sim, Kenichi Okada, and Sudhakar Pamarti

Editorial Board

The editorial board of the journal is organized into sections that correspond to the subject areas covered by the journal.

Circuits and Systems

Communications

Signal Processing

Contents

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 941653, 2 pages
doi:10.1155/2012/941653

Editorial
Clock/Frequency Generation Circuits and Systems

Woogeun Rhee,[1] Antonio Liscidini,[2] Jae-Yoon Sim,[3] Kenichi Okada,[4] and Sudhakar Pamarti[5]

[1] *Institute of Microelectronics, Tsinghua University, Beijing 100084, China*
[2] *Department of Electronics, University of Pavia, 27100 Pavia, Italy*
[3] *Department of Electrical Engineering, Pohang University of Science and Technology, Kyungbuk 790-784, Republic of Korea*
[4] *Department of Physical Engineering, Tokyo Institute of Technology, Tokyo 152-8552, Japan*
[5] *Electrical Engineering Department, University of California, Los Angeles, CA 90095, USA*

Correspondence should be addressed to Woogeun Rhee, wrhee@tsinghua.edu.cn

Received 20 November 2011; Accepted 20 November 2011

Clock generation and frequency synthesis systems have played a critical role in modern communication system design. As data rate increases, low-noise and low-power clock/frequency generation is getting more important than ever for high-performance digital communication systems. Timing uncertainty directly affects the overall performance of microprocessors and I/O links. The quality of wireless transceiver systems is often determined by the phase noise performance of frequency synthesizers. Active researches in the areas of clocking, data synchronization, and frequency generation have been performed for broad applications. This special issue focuses on the new and existing clock/frequency generation circuits and systems for wireline and wireless system IC designs with a special emphasis given to advanced technical results.

This special issue consists of nine papers. Four papers focus on wireline applications: a paper on characterizing the jitter tracking performance of the receiver for high-speed source synchronous links, a paper on generating fault-tolerant clock for VLSI circuits, a paper on a wide lock range clock-and-data recovery (CDR) circuit, and a paper on a delay-locked-loop- (DLL-) based clock generation for USB 2.0 applications. Three papers focus on wireless applications: a paper on open-loop phase modulation techniques for wide-bandwidth transmitter design, a paper on high-speed, high-resolution autocalibration techniques for fractional-N phase-locked loops (PLLs), and a paper on a wideband LC-based voltage-controlled oscillator (VCO) design for reconfigurable radio transceivers. Other two subsequent papers present nontraditional ways of generating clock frequency: the one with an all-digital flying-adder-based method and the other with a semidigital PLL-based method.

One of the major factors limiting the maximum achievable I/O data rates occurs from the degradation of system timing margins by clock jitter. The paper by A. Ragab et al. presents a comparative analysis of several jitter tracking architectures employed in source synchronous high-speed links. Tracking the transmit clock jitter in the presence of clock skew and channel loss enables high data rate operation in these links. The paper by G. Fuchs and A. Steininger describes a generation scheme of fault-tolerant clock based on a tick synchronization algorithm. An ASIC chip, implemented in a 0.18-μm CMOS, demonstrates feasibility of this scheme creating a globally synchronized clock, which is tolerant to a configurable number of arbitrary faults.

For high-speed chip-to-chip communication, serial link protocol has been widely adopted in various computer-to-peripheral interfaces. The use of serial links for multi-purpose, however, still presents some challenges which must be overcome by circuit design. The paper by S.-K. Lee et al. presents a 650-Mb/s to 8-Gb/s referenceless CDR with automatic frequency acquisition. By utilizing a novel DLL-based frequency acquisition, the dual-loop CDR shows outstanding performance in terms of lock range, power consumption, and area. The USB 2.0 is one of the most popular wireline standards, but requiring a very small area and low power consumption for portable applications. For low-cost low-power clock generation, a clock generator based on an edge-combiner DLL is presented in the paper by T. Kawamoto et al. The clock generator generates 480-MHz

10-tap output signals from a 12-MHz reference signal and consists of three DLLs to shrink the design area so that it is smaller than the PLL-based one.

The rapid growth of new communication standards like LTE and WiMAX has led to high data rate, wide signal bandwidth and high peak-to-average power ratio. A wide-bandwidth phase modulator is one of the key building blocks for the design of next generation transceivers. The paper by N. Nidhi et al. presents open-loop phase modulation techniques for upcoming 60 GHz wireless communication and software-defined radio. The open-loop modulation technique achieves maximum flexibility with wide-bandwidth operation. In modern wireless transceiver systems, the $\Delta\Sigma$ fractional-N frequency synthesizer is an essential building block. Especially for a wide tuning-range fractional-N frequency synthesizer, high-speed and high-precision automatic calibration is important for shortening the lock time and improving the phase noise. In the paper by J. Shin and H. Shin, an automatic calibration technique for the VCO frequency and the loop gain in the fractional-N PLL design is presented. A simple and efficient autocalibration method based on a high-speed frequency-to-digital converter significantly reduces the calibration time.

Recently, dozens of wireless communication standards have been used for small mobile terminals, for example, GSM, UMTS, LTE, WiMAX, WLAN, Bluetooth, UWB, GPS, DTV, and RFID, and the standards use several frequency bands spreading in a quite wide range such as 800 MHz to 6 GHz. All of these RF front-ends require a wideband-tunable VCO, which is an indispensable component for the multiband radio in common. The paper by Y. Ito et al. presents a wide-band frequency synthesizer with a frequency range from 1 GHz to 6.6 GHz by employing a wide-tuning LC VCO and a tuning-range extension circuit. Such tuning allows to satisfy the requirements of all cellular and WiFi standards, becoming a very interesting solution for multistandard transceiver where the presence of several local oscillator would results in a too expensive design.

The use of advanced CMOS technologies makes the traditional phase-locked loop (PLL) design challenging as on-chip variability and modeling inaccuracy become severe in deep submicron CMOS. Large loop parameter variation makes it difficult to find the optimum bandwidth for phase noise, spur, and settling time. In addition, analog passive devices become a bottleneck for scalability and integrating the loop filter (LPF) has been a challenging task in the conventional PLL design. Therefore, clock generation based on digital or semidigital method has received great attention recently. The paper by P.-L. Chen and C.-C. Tsai describes a flying-adder-based digital frequency synthesizer with an interpolated multiplexer to improve cycle-to-cycle and *rms* jitter performance. This synthesizer, fabricated in a 0.18-μm CMOS, shows a frequency lock range of from 33 MHz to 286 MHz. The paper by N. Xu et al. introduces recent semidigital PLL architectures which relax technology dependency and provide low-cost low-power clock generation. With the absence of the time-to-digital converter (TDC), the semidigital PLL enables low-power linear phase detection and does not necessarily require advanced CMOS technology while maintaining a technology scalability feature.

Lastly, the Guest Editors of this special issue would like to thank all the reviewers for their dedicated efforts in ensuring a high standard for the selected papers. We hope that readers will find this issue interesting.

Woogeun Rhee
Antonio Liscidini
Jae-Yoon Sim
Kenichi Okada
Sudhakar Pamarti

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 267247, 17 pages
doi:10.1155/2012/267247

Research Article

480 MHz 10-tap Clock Generator Using Edge-Combiner DLL for USB 2.0 Applications

Takashi Kawamoto,[1] Kazuhiro Ueda,[2] and Takayuki Noto[3]

[1] *Hitachi Central Research Laboratory, 1-280, Higashi-Koigakubo Kokubunji-shi, Tokyo 185-8601, Japan*
[2] *Technology Development Unit, Mixed Signal Core Development Division, Converter Development Department, Renesas Electronics Corporation, 4-1 Mizuhara, Itani-shi, Hyogo 664-0005, Tokyo, Japan*
[3] *SoC Business Unit, 1st Industry & Network Business Division, Optical Disc Solutions Department, Renesas Electronics Corporation, 5-20-1 Josuihon-cho, Kodaira-shi, Tokyo 187-8588, Japan*

Correspondence should be addressed to Takashi Kawamoto, takashi.kawamoto.hv@hitachi.com

Received 28 June 2011; Accepted 8 September 2011

Academic Editor: Woogeun Rhee

A clock generator with an edge-combiner DLL (ECDLL) has been developed for USB 2.0 applications. The clock generator generates 480 MHz 10-tap output signals from a 12 MHz reference signal and consists of three DLLs to shrink the design area so that it is smaller than a conventional one based on a PLL. Each DLL is applied to our proposed shot pulse reset technique to prevent from a harmonic lock and is applied to a voltage-controlled delay line (VCDL) with a trimming function to operate against any process voltage temperature (PVT) variations. A 90 nm CMOS process was used to fabricate our proposed clock generator. The 480 MHz 10-tap output signals satisfy the USB 2.0 specifications. A power consumption is less than 1.3 mW and a locking time is less than 3.5 μs, which are far less than a conventional one, 10.0 μs. The design area is 200 $\times$ 225 μm, which is half that of the conventional one.

1. Introduction

The clock generator may be one of the largest blocks of the physical layer (PHY) in wireline communications, because it usually consists of a phase-locked loop (PLL) that has large capacitors for use as a lowpass filter. There are many reports proposing the shrinkage of the design area of PLLs. A capacitance-multiplication technique was reported to shrink the capacitance of the loop filter [1, 2]. However, the shrink ratio of a capacitor may be less than five when taking the leakage current of the capacitor and PVT variation into consideration. Thus, the total design area cannot be drastically reduced. An all digital PLL (ADPLL) technique has also been reported [3]. However, an issue with the accuracy operation against the PVT variation still remains. Therefore, a new approach is desirable for essentially reducing the design area.

The DLL has several advantages over the PLL. First, it can be designed to be smaller than the PLL. While the PLL is a higher-order system, the DLL is a first-order system and is always stable. Thus, the DLL needs small capacitors to keep the DLL loop stable while the PLL needs large capacitors to design a stable lowpass filter. Second, the DLL can achieve a shorter locking time than the PLL. Third, the DLL consumes less power than the PLL. The PLL has the VCO and a divider that consumes a large amount of power in order to reduce the jitter. However, the DLL has several disadvantages over the PLL when used as the clock generator. First, the DLL cannot generate faster clock signals than the PLL. Second, the DLL has a locking range limitation while the PLL does not. This means that the DLL cannot achieve a fractional multiplication ratio while the PLL can achieve a fractional-N PLL.

An edge-combiner DLL (ECDLL) has been reported as an alternate high-speed clock generator because the ECDLL is based on the DLL and can multiply the reference frequency [4–8]. The ECDLL has a potential for use as the clock generator although it has barely been used in this capacity because the ECDLL has several challenges that need to be

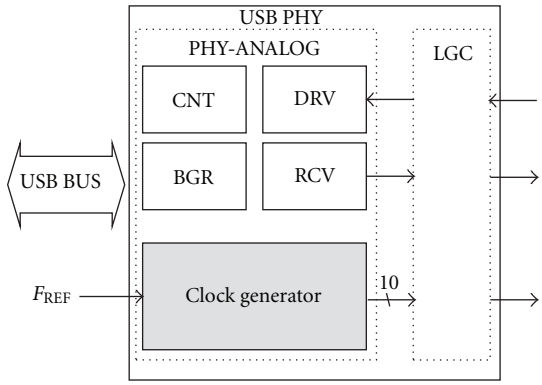


Figure 1: Block diagram of USB 2.0 PHY.

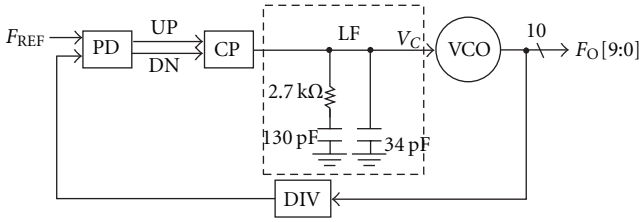


Figure 2: Conventional clock generator based on PLL.

overcome. The first is an operation against PVT variation. The second is the output signal frequency limitation.

From the viewpoint of the frequency limitation, the operation frequency of the DLL has been increasing in recent CMOS process. The DLL can operate at less than 1 GHz in a submicron CMOS process. Thus, the DLL might be able to be use as the clock generator for wireline communications whose operation frequency is less than about 1 GHz. USB 2.0 is the most popular wireline communication in the world and operates at 480 MHz. USB 2.0 PHY needs a small design area and a low power consumption level for use in portable devices. Therefore, USB 2.0 may be one of the most suitable applications for the clock generator with the ECDLL.

In order to apply the ECDLL for USB 2.0, we propose techniques hat overcome the above-mentioned challenges. The first is a shot pulse generator to prevent from a harmonic lock. The third is the VCDL trimming function to operate against PVT variation.

In this paper, we propose a clock generator applied to an ECDLL for USB 2.0 PHY to shrink the design area [1]. The organization of this paper is as follows. Section 2 describes the overall structure of the proposed clock generator architecture. Section 3 describes the ECDLL and DLL in detail. Section 4 presents the evaluation results of our measurements, and Section 5 concludes with a short summary of the key points.

2. Overall Clock Generator Architecture

Figure 1 shows a block diagram of a USB 2.0 PHY. The PHY consists of a clock generator, a band-gap reference (BGR), a controller (CNT), a driver (DRV), a receiver (RCV), and

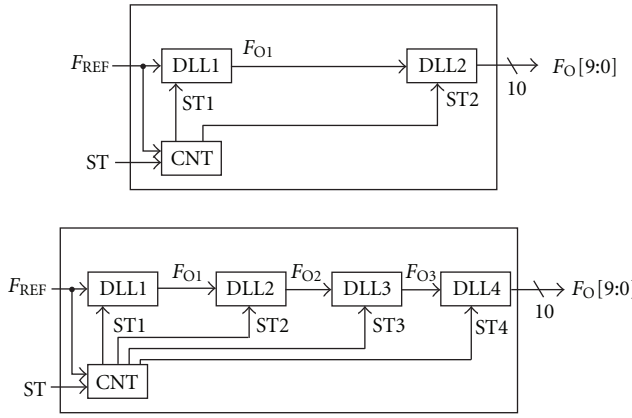


Figure 3: Block diagrams of candidate clock generator based on DLLs. Upper block diagram is the structure that consists of one ECDLL and one DLL. The ECDLL has a multiplication ratio of 40. Bottom block diagram is the structure that consists of three ECDLL and one DLL. The DLL1, DLL2, and DLL3 have multiplication ration of two, four, and five, respectively.

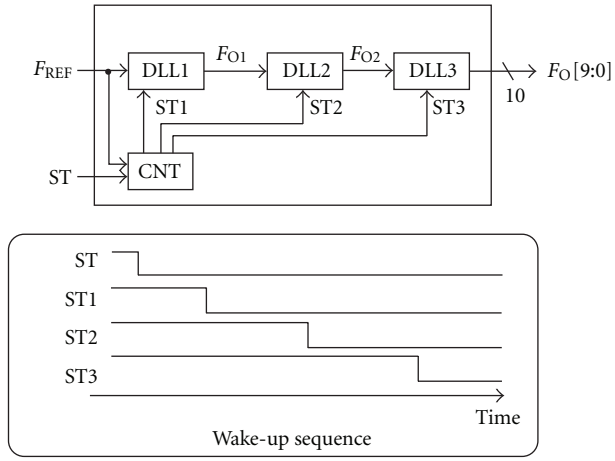


Figure 4: Proposed clock generator consisting of three DLLs and wake-up sequence. The DLL1 and DLL2 have multiplication ration of two and eight, respectively.

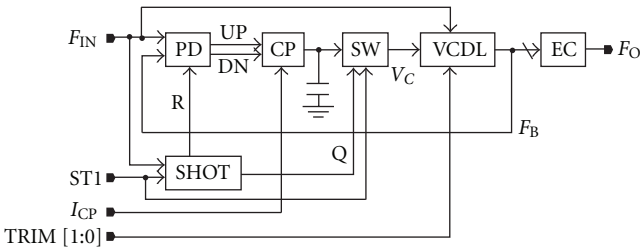


Figure 5: Block diagram of DLL1 and DLL2. In DLL1, F_{IN} and F_O are 12 MHz and 60 MHz, respectively. The VCDL generates 10-tap 12 MHz output signals. The capacitor is 10 pF. In DLL2, F_{IN} and F_O are 60 MHz and 480 MHz, respectively. The VCDL generates 16-tap 60 MHz output signals. The capacitor is 0.5 pF.

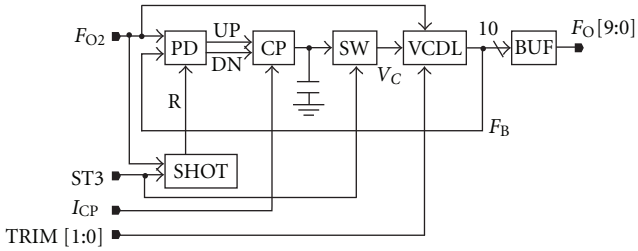


Figure 6: Block diagram of DLL. F_{O2} and F_O[9:0] are 480 MHz and 10-tap 480 MHz, respectively. The capacitor is 1 pF.

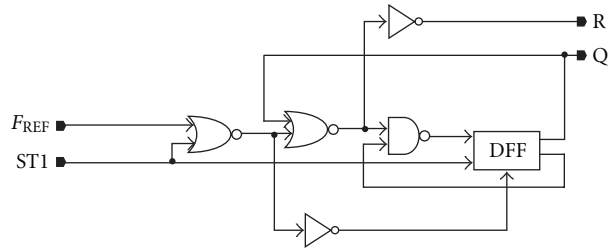


Figure 8: Block diagram of shot pulse generator.

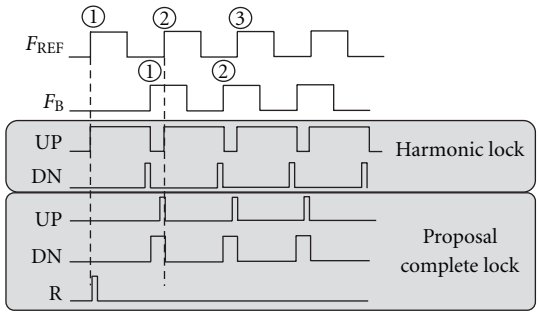


Figure 7: Explanation of harmonic lock and proposed shot pulse reset operation.

a logic block (LGC). Figure 2 shows a block diagram of a conventional clock generator based on the PLL with a ring oscillator. The reference signal (F_{REF}) is 12 MHz. The output signals (F_O[9:0]) are 10-tap 480 MHz. The design area is 200 $\times$ 550 μm^2. The loop bandwidth is designed at 1.6 MHz using a second-order lowpass filter that consists of 130 pF and 34 pF capacitances and a 2.7 kΩ resistance. Thus, the lowpass filter occupies a large portion of the design area. Therefore, we proposed the ECDLL as a clock generator to shrink the design area.

In this clock generator, there are three candidates, which are one ECDLL and one DLL, two ECDLL and one DLL, and three ECDLL and one DLL, as shown in Figures 3 and 4. The one ECDLL and one DLL structure consists of the DLL1 of the ECDLL that has the multiplication ratio of 40 and the DLL2 of the DLL that generates the 10-tap 480 MHz signals, as shown in Figure 3. The two ECDLL and one DLL structure consists of the DLL1 of the ECDLL that has the multiplication ratio of five, the DLL2 of the ECDLL that has the multiplication ration of eight, and the DLL3 of the DLL that has the same manner of the above DLL, as shown in Figure 4. The three ECDLL and one DLL structure consists of the DLL1 of the ECDLL that has the multiplication ratio of two, the DLL2 of the ECDLL that has the multiplication ratio of four, the DLL3 of the ECDLL that has the multiplication ratio of five, and the DLL4 of the DLL that has the same manner of the above DLL, as shown in Figure 3.

First, it is reasonable that the ECDLL has the multiplication ratio of less than 10, according to the design area and operation against from PVT variation. As the multiplication ratio is larger, the number of the VCDL stage is larger. It causes large design area. The one ECDLL and one DLL structure becomes two times as large as the two ECDLL and one DLL structure because the ECDLL of the multiplication ratio of 40 is large. The two ECDLL and one DLL structure is almost same design area as the three ECDLL and one DLL structure. And then, the previous DLL operates more slowly than the latter one. Thus, the delay cell size in the previous DLL may be smaller than that in the latter one. Therefore, to shrink the design area, the previous DLL might have smaller multiplication ratio than the latter one.

Second, the number of the cascade DLL block should be as low as possible because the operation of the whole clock generator could be stable and the settling period could be short. In our proposed clock generator, the standby sequence is necessary because the DLL may fall into the unlock state if the DLL starts to operate before the previous DLL completes the lock, as shown in Figure 4.

Finally, the two ECDLL and one DLL structure is proposed, considering above concern. The DLL1 and DLL2 have a multiplication ratio of five and eight, respectively.

The counter (CNT) generates the standby signals of each block (ST1, ST2, and ST3) using the standby signal of the clock generator (ST) to create a standby sequence for each DLL, as shown in Figure 4. If DLL2 starts the lock operation before DLL1 completes the lock, DLL2 might fall into the unlock state. Thus, the CNT controls the sequential wake-up operation by generating ST1, ST2, and ST3, as shown in Figure 4.

3. ECDLL and DLL in Detail

3.1. Shot Pulse Reset Technique. Figures 5 and 6 show a block diagram of our proposed ECDLL applied to DLL1 and DLL2 and DLL applied to DLL3. They consist of a phase detector (PD), a charge-pump (CP), a switch (SW), a capacitor, a voltage-controlled delay line (VCDL), and shot pulse generator (SHOT). And then the ECDLL has an edge-combiner (EC) and the DLL has the output buffer (BUF). The PD makes a comparison between the phase of the F_{REF} and a phase of the feedback clock (F_B) and generates result signals (UP/DN). The CP charges and discharges the capacitor. The VCDL generates the output signals from the F_{REF}. The delay time from F_{REF} to F_B is controlled by a controlled voltage (V_C). In Figure 6, the EC generates the output signal (F_{O1}) from VCDL output signals.

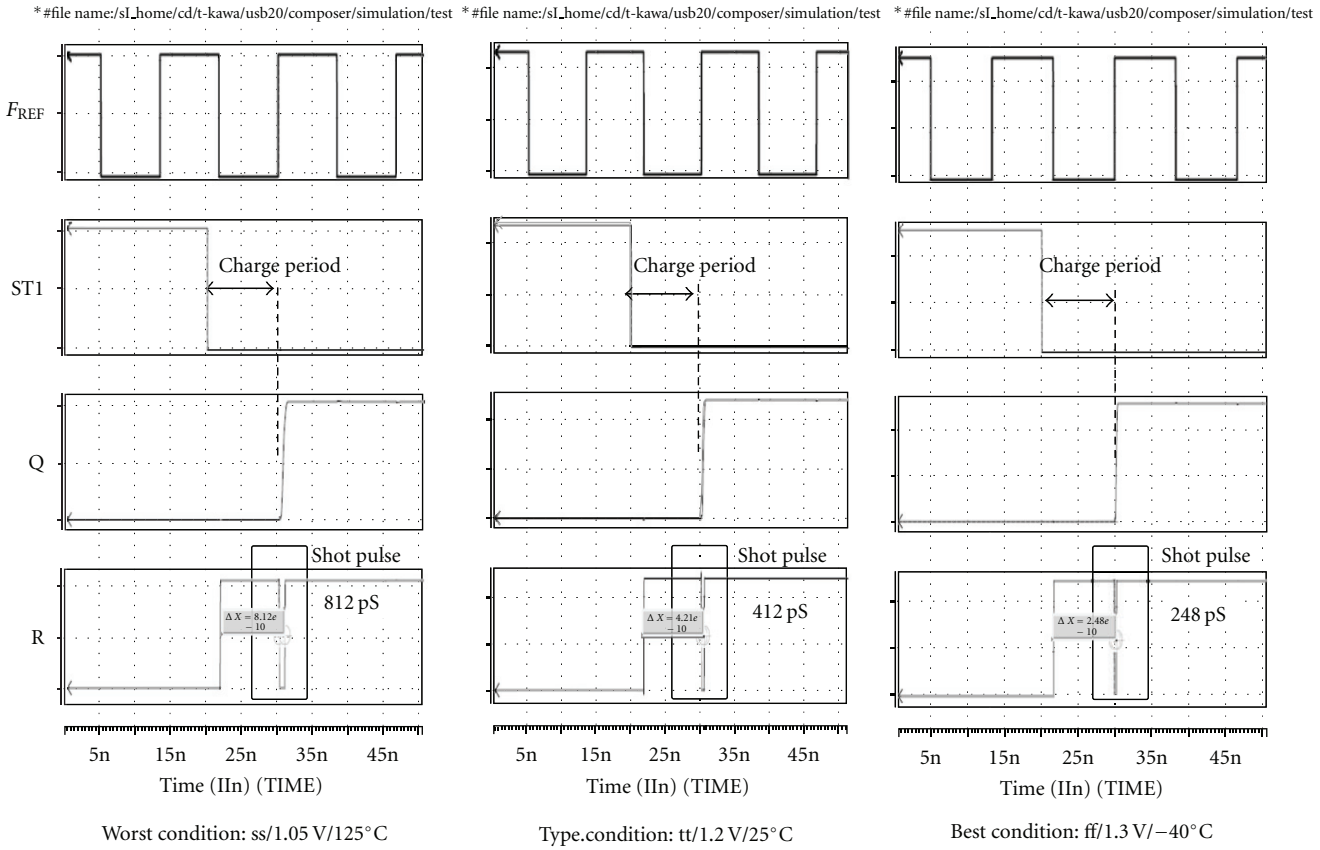


Figure 9: Simulation results from shot pulse generator.

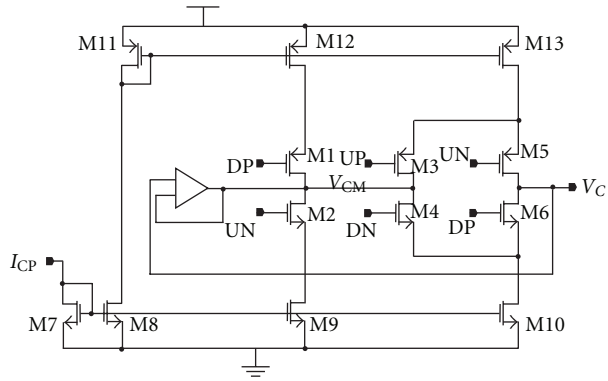


Figure 10: Circuit diagram of the CP.

Our ECDLL and DLL have an issue of a harmonic lock. We propose shot pulse reset technique by using the shot pulse generator to resolve this issue.

Figure 7 shows an explanation of the proposed shot pulse reset technique. The PD compares the rise edge of the F_{REF} and the rise edge of the F_B and generates the UP and DN signals. When the second rise edge of the F_{REF} comes after the first rise edge of the F_B, a harmonic lock occurs that the PD compares between first rise edge of the F_{REF} and the first rise edge of the F_B. It is an actual operation that the PD compares between the second rise edge of the F_{REF} and the first rise edge of the F_B. A shot pulse reset shown in Figure 7 is proposed to prevent this malfunction from happening. The operation of the PD is reset by R when the PD compares the first rise edge of the F_{REF} with the first rise edge of the F_B. Thus, the PD can compare the second rise edge of the F_{REF} with the first rise edge of the F_B. Even though the PD is reset by R, the harmonic lock occurred and the PD compared the third rise edge of the F_{REF} with the first rise edge of the F_B, when the first rise edge of the F_B comes after the third rise edge of the F_{REF}. To prevent this malfunction from happening, the capacitor is charged by the SW and the voltage level of the

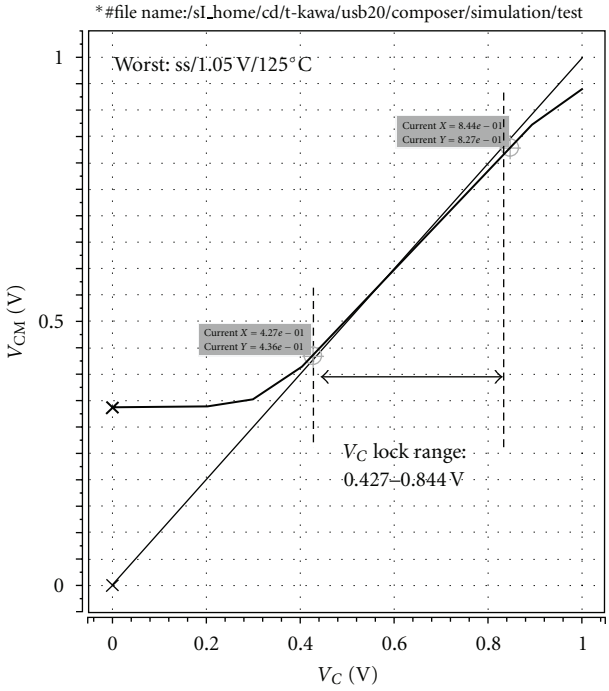


Figure 11: Simulation results from CP.

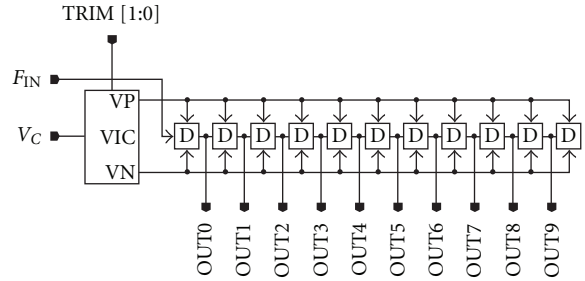


Figure 12: Block diagram of VCDL in DLL1 and DLL3.

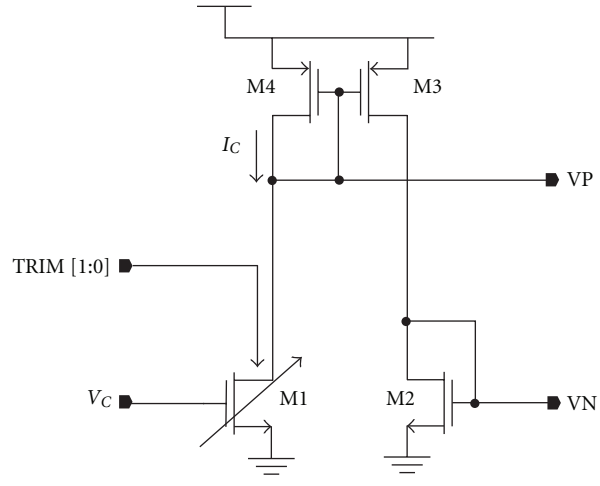


Figure 13: Circuit diagram of VIC.

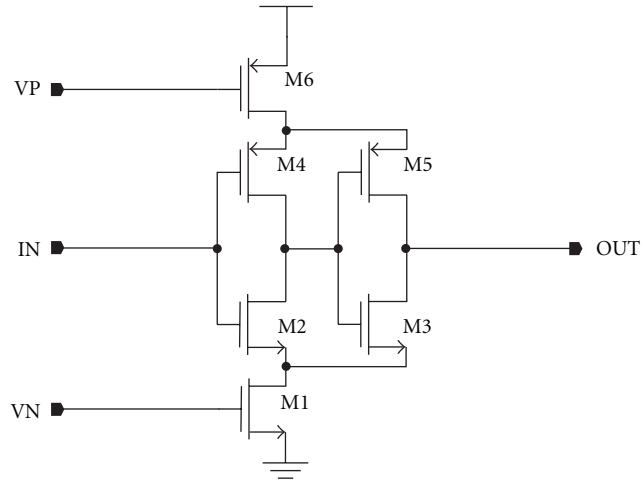


Figure 14: Circuit diagram of delay cell.

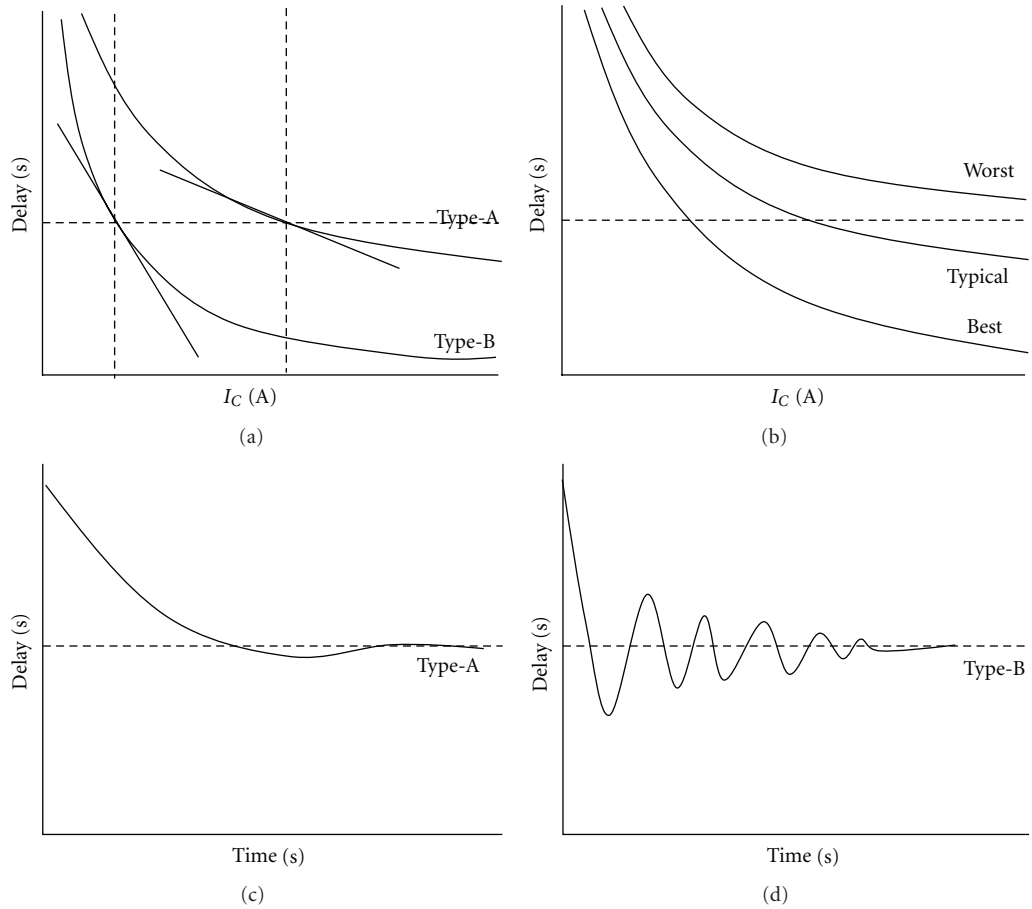


Figure 15: Explanation of the VCDL and DLL operation. (a) The explanation of the VCDL delay-current characteristics. The type-A VCDL has the smooth sensitivity and the type-B has the steep sensitivity. (b) The explanation of the influence of the VCDL delay-current characteristics due to the PVT variation. (c) The explanation of the DLL settling operation by using type-A VCDL. The settling operation is smooth because the sensitivity of the VCDL is not steep. (d) The explanation of the DLL settling operation by using type-B VCDL. The settling operation is not smooth because the sensitivity of the VCDL is steep.

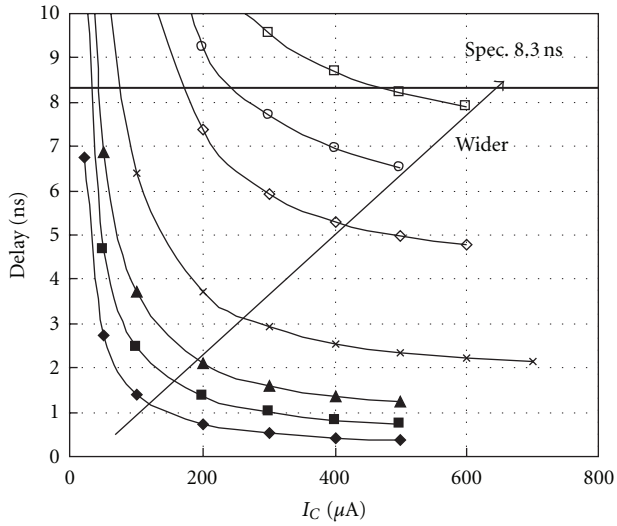


Figure 16: Postlayout simulation results of VCDL delay-current characteristics by using variable delay cell.

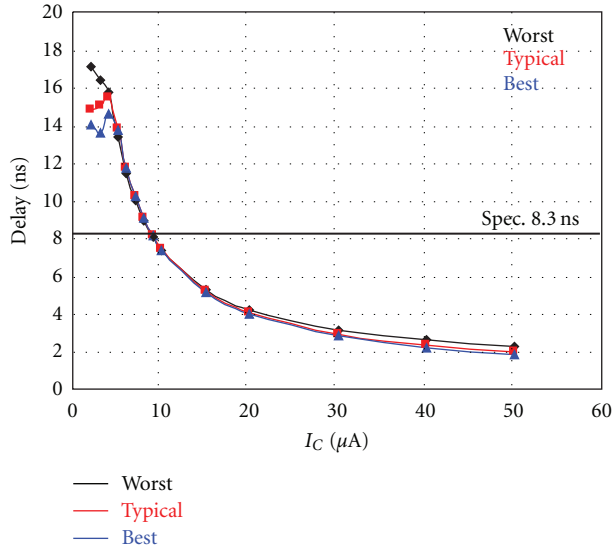


Figure 17: Postlayout simulation results of VCDL delay-current characteristics for DLL1.

V_C is likely to be V_{DD}. This allows the first rise edge of the F_B to come before the second rise edge of the F_{REF} when the PD starts operation.

Figures 8 and 9 show a circuit diagram of the shot pulse generator (SHOT) and the simulation results from the SHOT. After the ST1 is set to low, the R operates the pulse reset and the Q is set to high at the rise edge of the F_{REF}. The pulse of the R resets the PD operation and the SW charges the capacitor during the period between the fall edge of the ST1 and the rise edge of the Q. After charging the capacitor, the V_C is almost the V_{DD}. Our ECDLL and DLL can be operated accurately because of the SHOT.

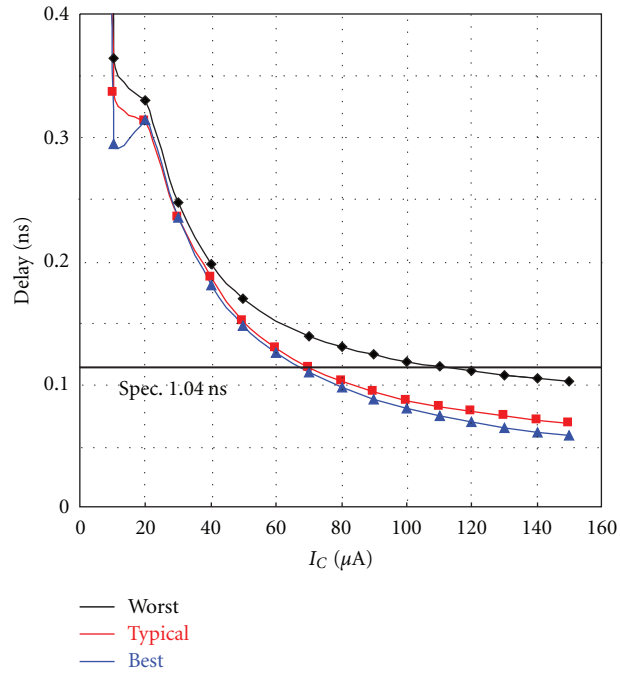


Figure 18: Postlayout simulation results of VCDL delay-current characteristics for DLL2.

3.2. Charge Pump. Figure 10 shows a circuit diagram of the CP. M5 and M6 are the switches that charge and discharge the capacitor. When the CP charges the current, the UP, UN, DP, and UN are high, low, low, and high, respectively. The charge current, which is the M13 drain current, passes through the switch M5 and charges the capacitor that is connected at the V_C, and the M12 drain current passes through the switches (the M1 and M4) and flows to M10. When the CP discharges the current, the UP, UN, DP, and UN are low, high, high, and low, respectively. The discharge current, which is the M10 drain current, passes through the switch at M6 and discharges the capacitor that is connected at the V_C, and the M13 drain current passes through the switches (the M3 and M2) and flows to M9. The Op-Amp is designed in the CP to structure the common-mode feedback. When the V_C is not equal to the voltage of the V_{CM}, the difference between the charge current and the discharge current is larger. This causes a constant phase error. Figure 11 shows the simulation results from the CP. The lock range, which is the range in which the V_C is equal to the voltage of the V_{CM}, is 0.427–0.884 V under the worst conditions (ss/1.05 V/125°C).

3.3. VCDL. Figure 12 shows a block diagram of the voltage-controlled delay line (VCDL) for DLL1 and DLL3. It consists of a voltage-current converter (VIC) and the delay chain consisting of eleven delay cells. The VCDL for DLL2 has the seventeen delay cells. Figures 13 and 14 show the circuit diagrams of the VIC and the delay cell. The VIC converts the control voltage (V_C) to a control current (I_C). The VCDL delays the F_{IN}, which is the amount of the delay controlled by the I_C. The trimming signal (TRIM[1:0]) controls the sensitivity of the current-voltage characteristics of the M1 by

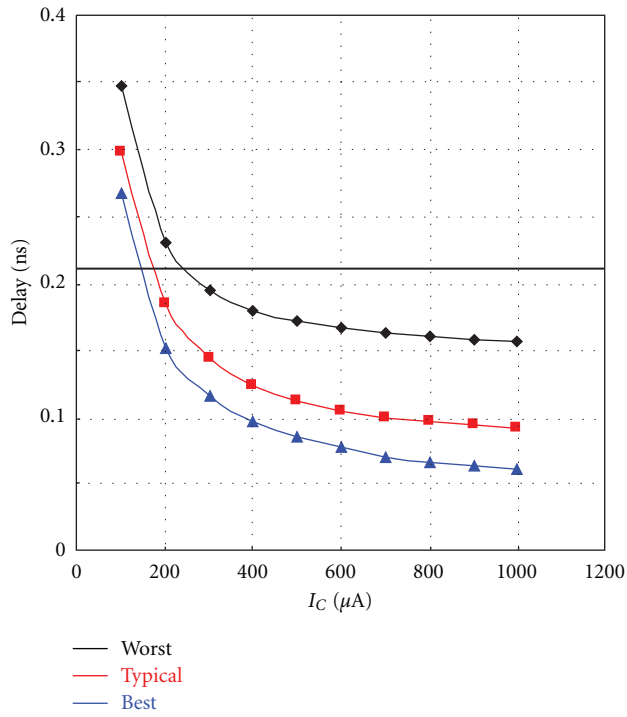


Figure 19: Postlayout simulation results of VCDL delay-current characteristics for DLL2.

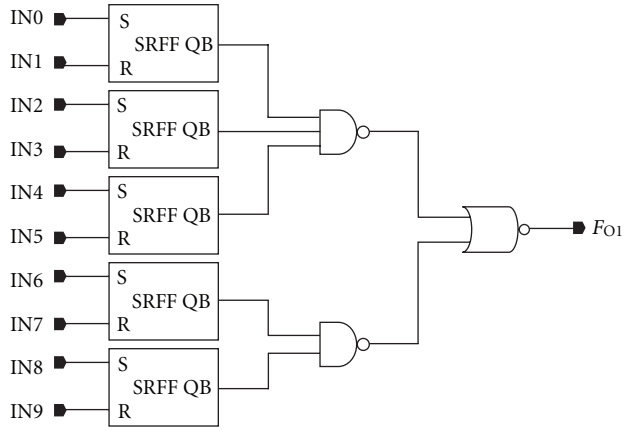


Figure 20: Block diagram of EC for DLL1.

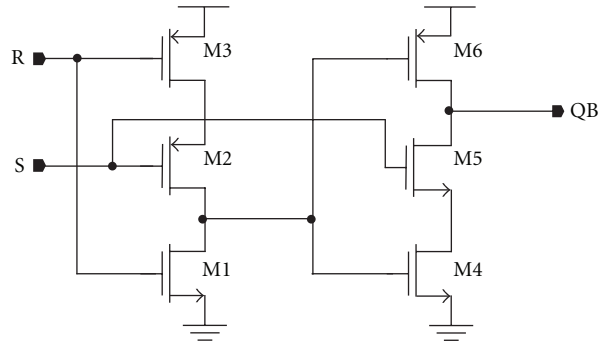


Figure 21: Circuit diagram of SRFF.

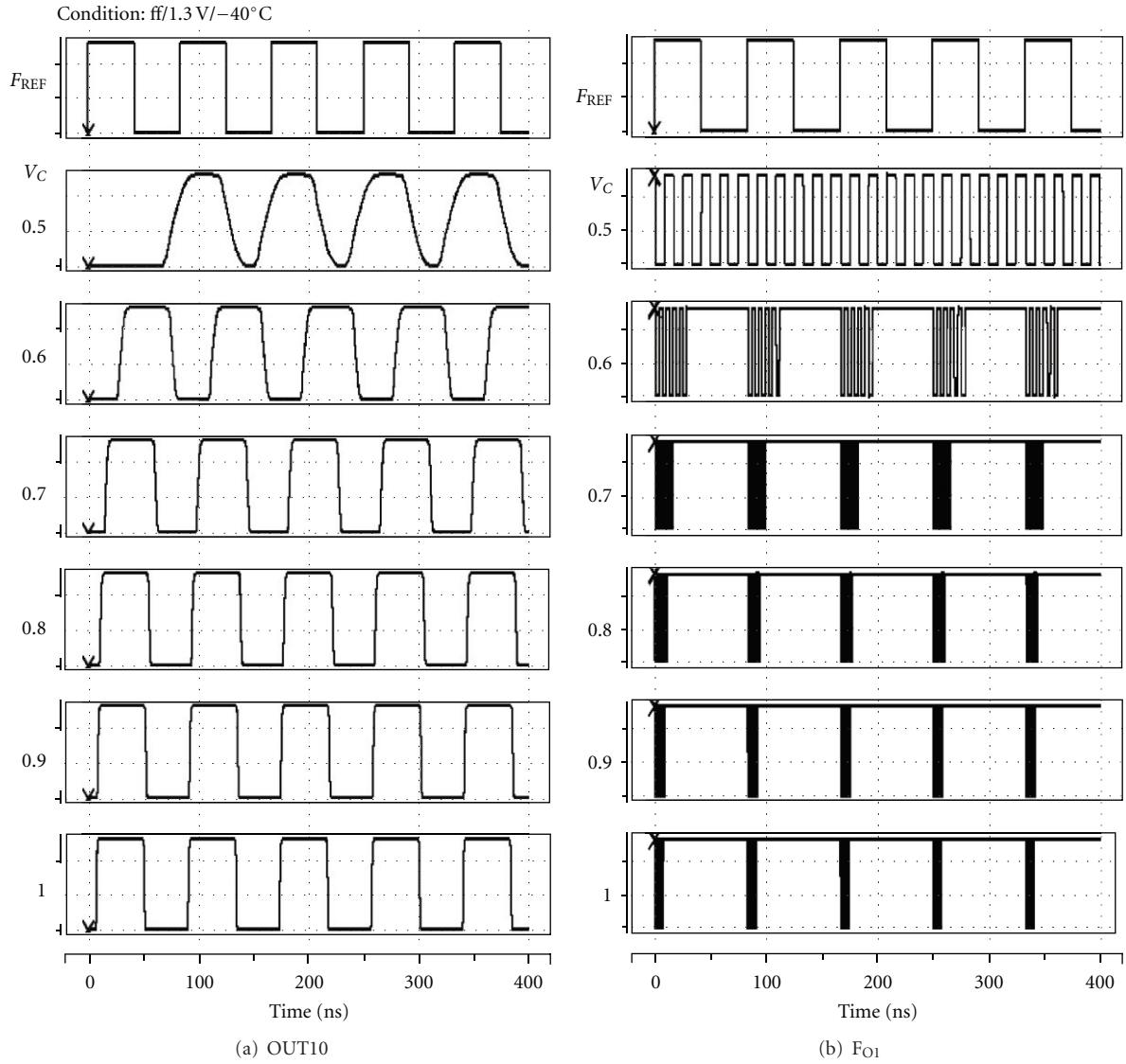


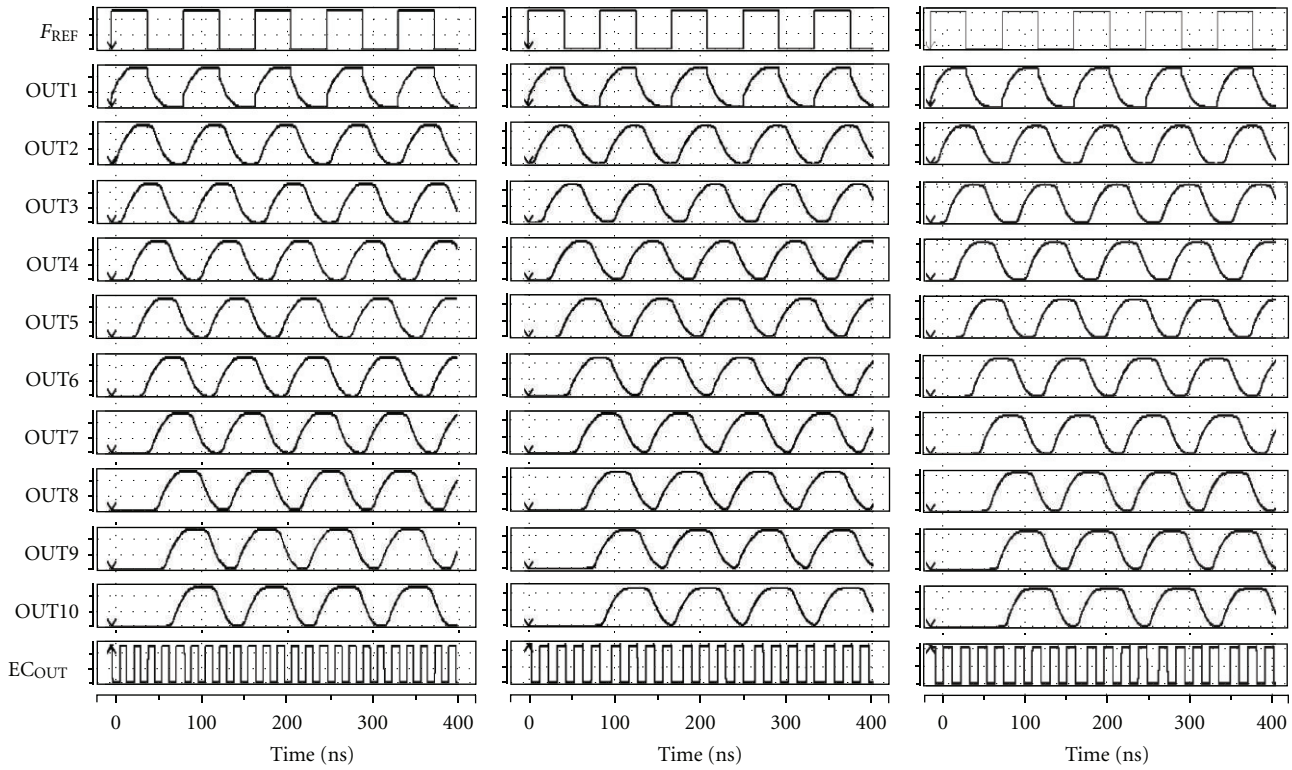
Figure 22: Postlayout simulation results from VCDL and EC applied to DLL1 for variable V_C. The simulation condition is ff/1.30 V/−40°C.

changing the gm of M1. The larger the TRIM[1:0] is, the larger the I_C is. The delay cell consists of the two inverter buffers (M4-M2 and M5-M6) and current sources (M6 and M1).

In the VCDL, the sensitivity of the VCDL is important for DLL operation. Figure 15 shows the explanation of the sensitivity of the VCDL and the DLL settling operation. If the sensitivity of the VCDL delay-current characteristics is larger at the lock point, the DLL settling operation may not be stable as shown in Figure 15(d). It is the reason that the overshoot is large because the magnitude of the delay change per one clock cycle is large. To prevent from this unstable state, the VCDL sensitivity is designed small by using large delay cell for VCDL, as shown in Figure 15(a). However, this design causes large power consumption and the malfunction may be caused in the worst condition if the sensitivity is designed too small, as shown in Figure 15(b).

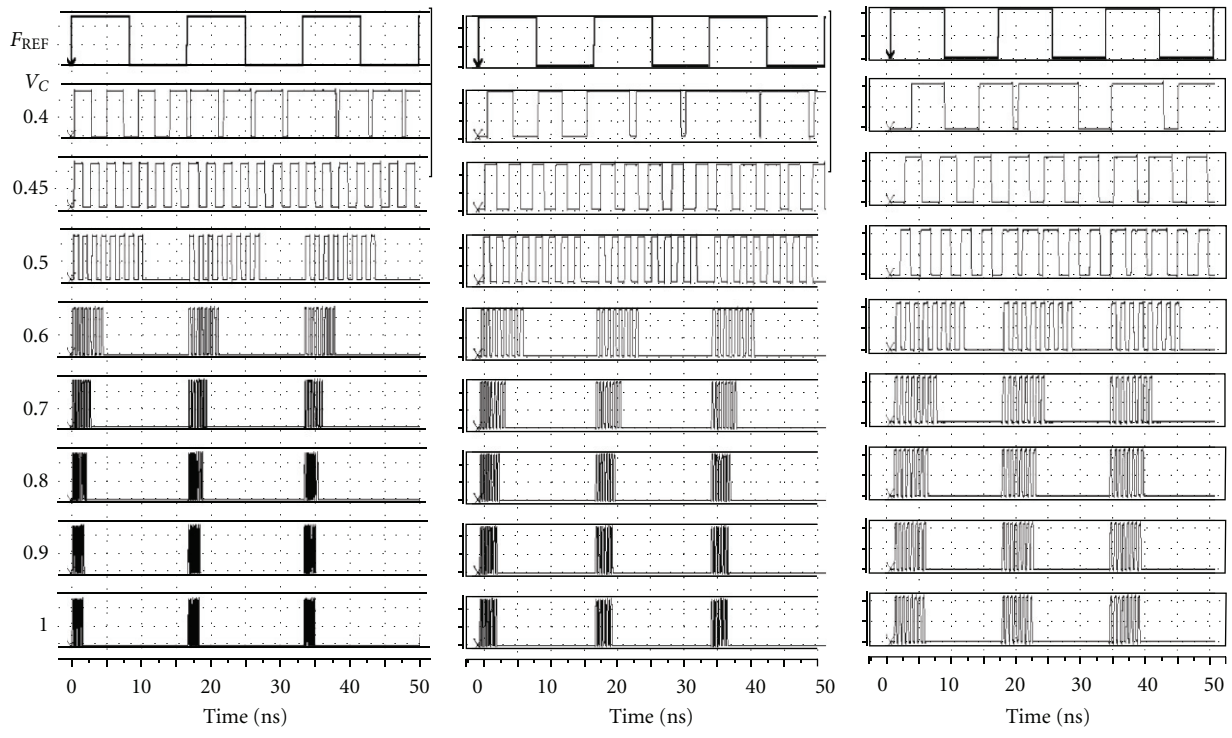
The delay is mainly generated as the control current and input capacitor, which is gate capacitor of M4 and M2, and a parasitic capacitor between delay cells. If the buffer MOSs (M4-M2 and M5-M3) are designed small, the necessary delay is obtained by small current. However, this causes large sensitivity. Thus, the buffer MOSs are not designed small. Figure 16 shows the VCDL delay-current characteristics by using variable delay cell. As the size of the delay cell is larger, the sensitivity at the necessary delay point is smoother.

Figures 17, 18, and 19 show the postlayout simulation results of the VCDL delay-current characteristics for DLL1, DLL2, and DLL3, respectively. The VCDL for DLL1 can achieve a target delay of 8.3 ns at about 9 μA under variable conditions. The VCDL for DLL2 can achieve a target delay of 1.04 ns at 80 μA under typical and the best conditions, which are tt/1.20 V/25°C and ff/1.35 V/−40°C, respectively. However, under the worst condition, which is ss/1.05 V/125°C,



(a) ff/1.30 V/−40°C (b) tt/1.20 V/25°C (c) ss/1.05 V/125°C

Figure 23: Postlayout simulation results from VCDL and EC applied to DLL1 for variable condition at $V_C = 0.5$ V.



(a) ff/1.30 V/−40°C (b) tt/1.20 V/25°C (c) ss/1.05 V/125°C

Figure 24: Postlayout simulation results from VCDL and EC applied to DLL2 for variable V_C.

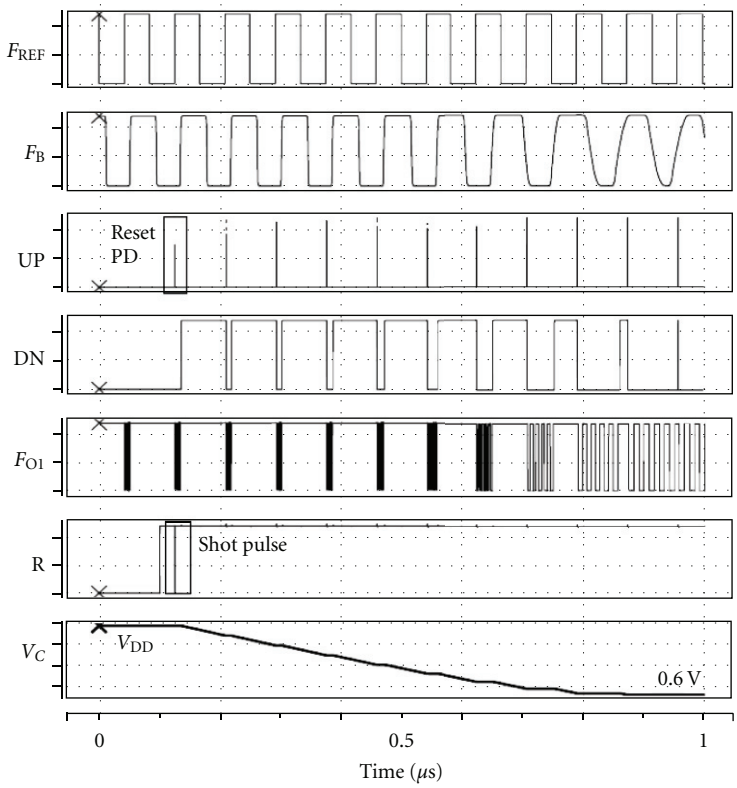


Figure 25: Postlayout simulation results from DLL1 locking operation. The simulation condition is tt/1.2 V/25°C.

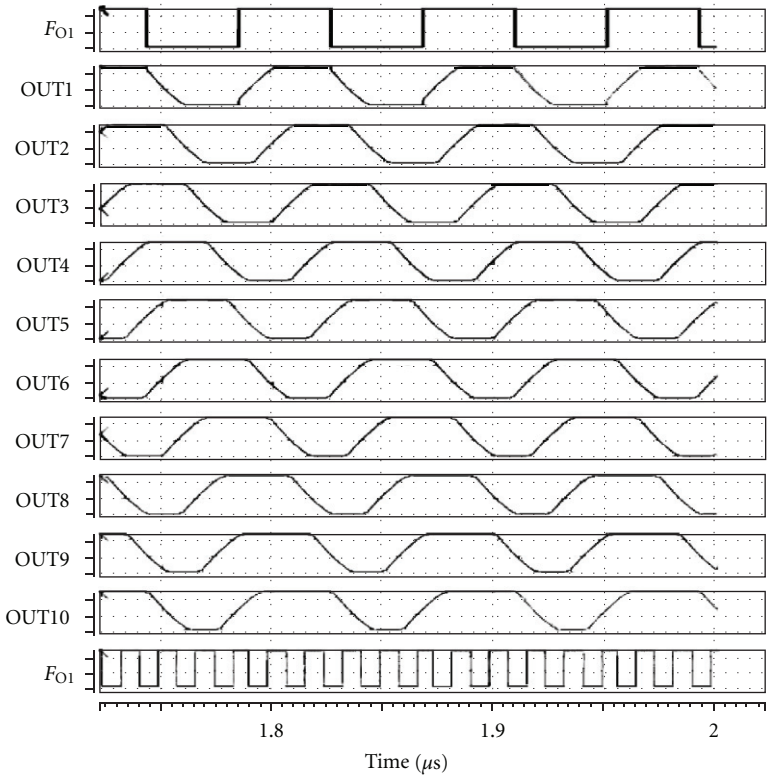


Figure 26: Postlayout simulation result of VCDL output signals from DLL1 settling operation. The simulation condition is tt/1.2 V/25°C.

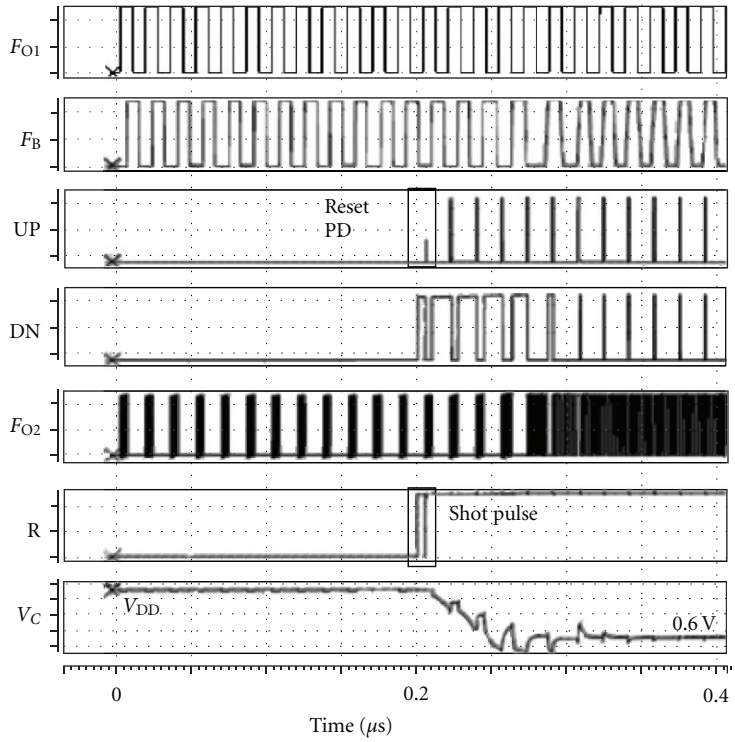


Figure 27: Postlayout simulation results from DLL2 locking operation. The simulation condition is tt/1.2 V/25°C.

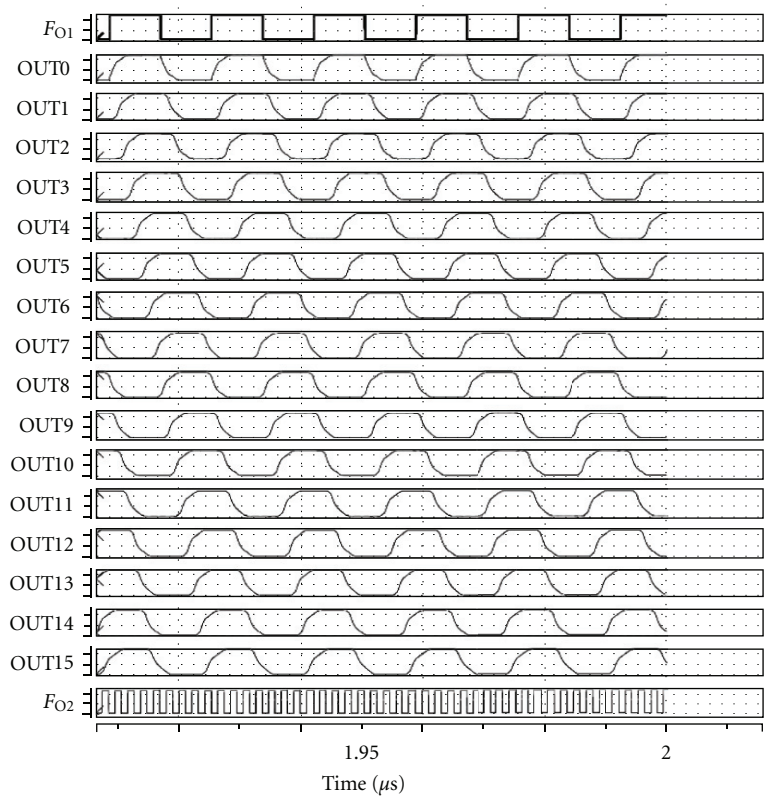


Figure 28: Postlayout simulation result of VCDL output signals from DLL2 settling operation. The simulation condition is tt/1.2 V/25°C.

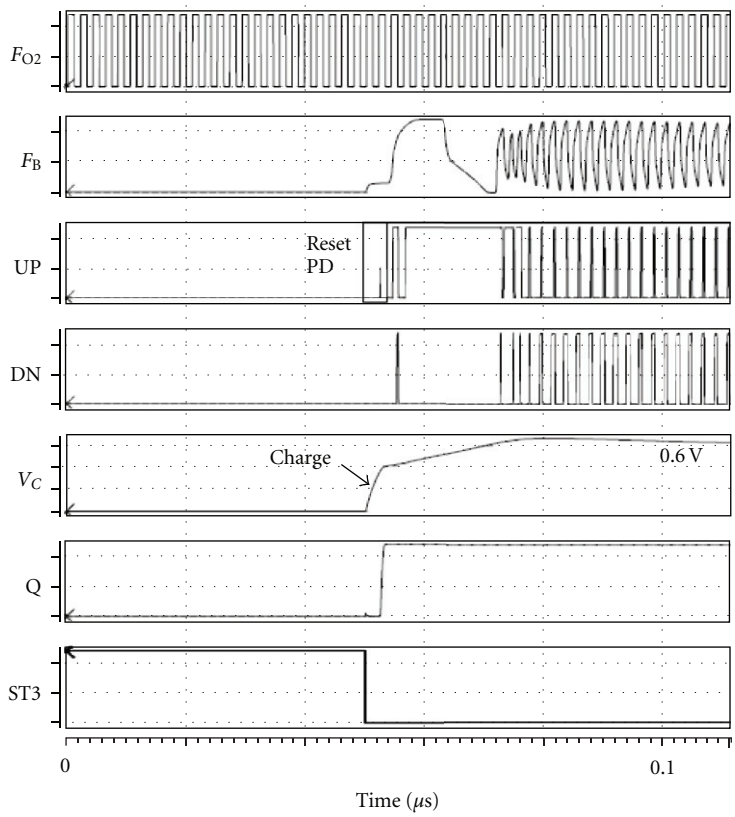


Figure 29: Postlayout simulation results from DLL3 locking operation. The simulation condition is tt/1.2 V/25°C.

the target delay can be achieved at an I_C of about 120 μA. This is adjusted by the trimming bits (TRIM[1:0]), as shown in Figure 14. The VCDL for DLL3 can achieve a target delay of 0.208 ns at a V_C of about 200 μA. However, the VCDL is trimmed by the TRIM[1:0].

3.4. Edge-Combiner. Figure 20 shows a block diagram of the EC for DLL1. It consists of SR flip-flops (SRFFs) and NANDs and a NOR. Figure 21 shows a circuit diagram of the SRFF. It consists of six MOSs. There are two floating nodes (M3 drain node and M5 source node). Figures 22 and 23 show the postlayout simulation results from the VCDL and EC for DLL1. The one cycle delay is obtained at about $V_C = 0.5$ V in ff/1.30 V/−40°C as shown in Figure 22(a). The EC can operate variable input signal as shown in Figure 22(b). If the SRFF cannot operate accurately by the leakage current, the output signal of the EC slips the clock in part. However, the EC can get all clock edges of the each signal at variable V_C as shown in Figure 22(b) and it can operate at variable conditions as shown in Figure 23.

Figure 24 shows the postlayout simulation results from the VCDL and EC for DLL2. The one cycle delay is obtained at between 0.45 V and 0.50 V in ff/1.30 V/−40°C and tt/1.20 V/25°C as shown in Figures 23(a) and 23(b), and at between 0.50 V and 0.60 V in ss/1.05 V/125°C. The EC can get all clock edges of the each signal at variable V_C and it can operate at variable conditions as shown in Figure 24.

3.5. Lock Operation. Figure 25 shows the postlayout simulation results of the DLL1 locking operation. The simulation condition is tt/1.2 V/25°C. The DLL1 has a capacitor of 10 pF. After ST1 is set to low at about 100 ns, the R is set to high, and then a shot pulse occurs. The PD operation is reset by the shot pulse, as shown by the UP and DN signals in Figure 25. After that, the PD generates a wide DN pulse and then the V_C decreases. Finally, DLL1 completes the lock at about 1 μs. When DLL1 completes the lock, the V_C is about 0.6 V. Figure 26 shows the VCDL output signals after the DLL1 completes the lock. The EC can generate the output signal (F_{O1}) of 60 MHz.

Figure 27 shows the postlayout simulation result of the DLL2 locking operation. The simulation condition is tt/1.2 V/25°C. The DLL2 has a capacitor of 0.5 pF. After ST2 is set to low at about 200 ns, the R is set to high and then a shot pulse occurs. The PD operation is reset by the shot pulse, as shown by the UP and DN signals in Figure 27. After that, the PD generates a wide DN pulse and then the V_C decreases. Finally, DLL2 completes the lock at about 400 ns. When DLL2 completes the lock, the V_C is about 0.6 V. Figure 28 shows the VCDL output signals after the DLL2 completes the lock. The EC can generate the output signal (F_{O2}) of 480 MHz.

Figure 29 shows the postlayout simulation results of the DLL3 locking operation. The simulation condition is tt/1.2 V/25°C. The DLL2 has a capacitor of 1 pF. After ST3 is set to low at about 50 ns, the PD operation is reset by

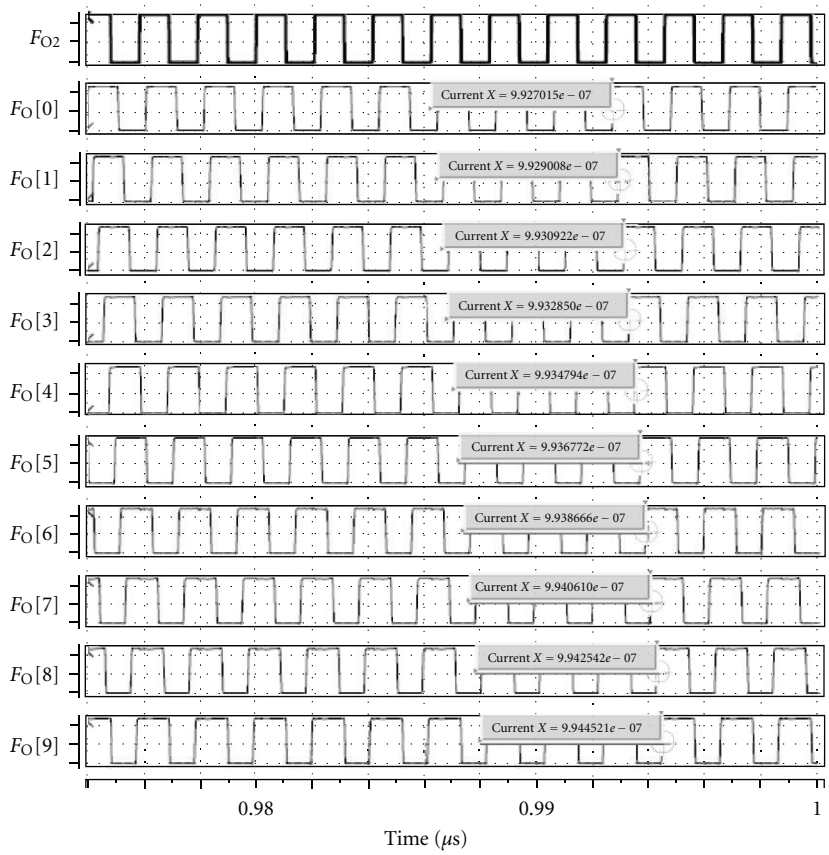


Figure 30: Postlayout simulation result of VCDL output signals from DLL3 settling operation. The simulation condition is tt/1.2 V/25°C.

the R shot pulse, as shown by the UP and DN signals in Figure 28. After that, the PD generates a wide UP pulse and then the V_C increases. Finally, DLL3 completes the lock at about 100 ns. When DLL3 completes the lock, the V_C is about 0.6 V. Figure 30 shows the VCDL output signals after the DLL3 completes the lock. The DLL3 can generate the output signals of 480 MHz.

Figure 31 shows the postlayout simulation results from a clock generator that consists of DLL1, DLL2, and DLL3. After ST is set to low, ST1 is set to low first. At this time, ST2 and ST3 remain high. F_{REF} inserts DLL1 and the V_C is nearly V_{DD} due to a precharge. The PD generates a wide DN pulse at first because of the precharge. The V_C decreases due to the wide DN pulse. At about 2 μs, DLL1 completes the lock and generates F_{O1}, which is the 60 MHz clock signal. ST2 is set to low at about 1 μs. It is essentially set to low after DLL1 completes the lock. However, in this simulation, it is set to low before the DLL lock time. The V_C in DLL2 is almost V_{DD} due to the precharge. After ST2 is set to low, the PD in DLL2 generates a wide DN pulse and then the V_C is soon almost 0.5 V. At about 2 μs, DLL2 completes the lock. ST3 is set to low at about 2 μs. It is essentially set to low after DLL2 completes the lock, but in this simulation, it is set to low before the DLL lock time, too. After ST3 is set to low, the V_C at first remains almost V_{DD}. The V_C decreases at about 2.6 μs and finally is about 0.5 V. DLL3 completes the lock and generates the 10-tap 480 MHz clock signals. The total lock time of the clock generator is about 3.0 μs. In general, the locking time of the DLL is defined by the capacitance and CP current. When the CP current is large for the capacitance, the locking time is short, but the locking operation is barely stable. DLL1 is designed to be stable because the phase error of DLL1 directly influences the other DLLs. DLL2 and DLL3 are designed to achieve a fast locking time because the clock generator can achieve it. Then, DLL2 and DLL3 start to operate before the forward DLL completes the fast locking time. When DLL2 starts to operate, F_{O1} is almost 60 MHz. Thus, DLL2 can accurately operate.

4. Measurement Results

A 90 nm CMOS process was used to fabricate our proposed clock generator for use as a USB 2.0 PHY. Figure 32 shows the measurement results of the output signal $F_O[9]$. The measurement signal is $F_O[9]$ divided by eight. The clock generator output signal frequency is 480 MHz. The jitter is less than 0.8 psrms. Figure 33 shows the measurement results of the EYE pattern for the USB 2.0 specifications. The USB 2.0 PHY with our proposed clock generator can pass these specifications. Figure 34 shows the measurement results of the random data pattern in the USB 2.0 specifications. The USB 2.0 PHY with our proposed clock generator can operate random data that meets the USB 2.0 specifications. Figure 35 shows the layout of the chip. Our proposed clock

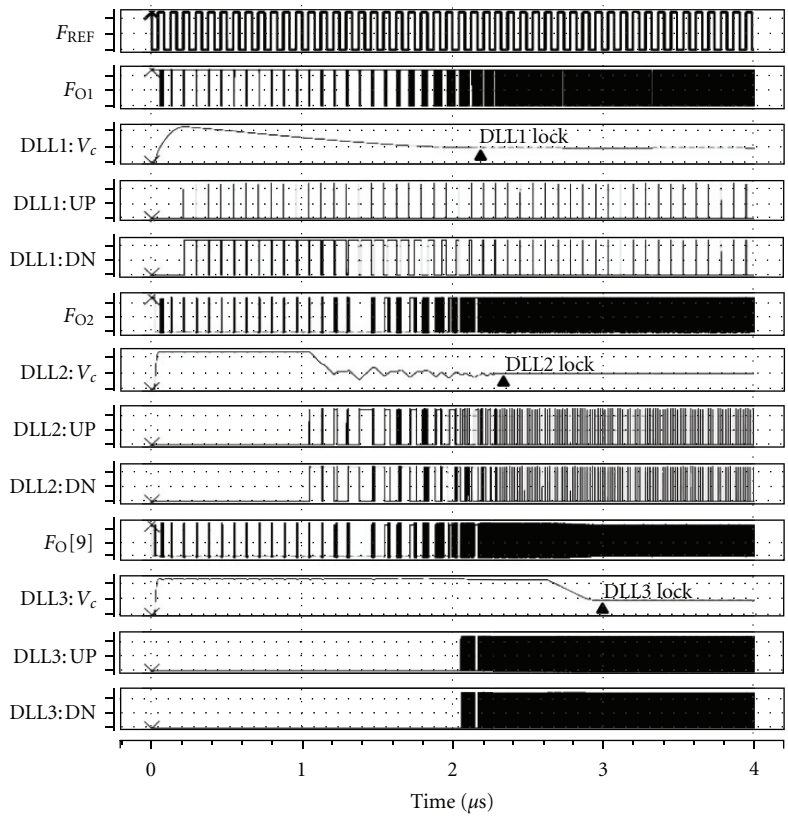


Figure 31: Postlayout simulation results from clock generator wake-up operation. The simulation condition is tt/1.2 V/25°C.

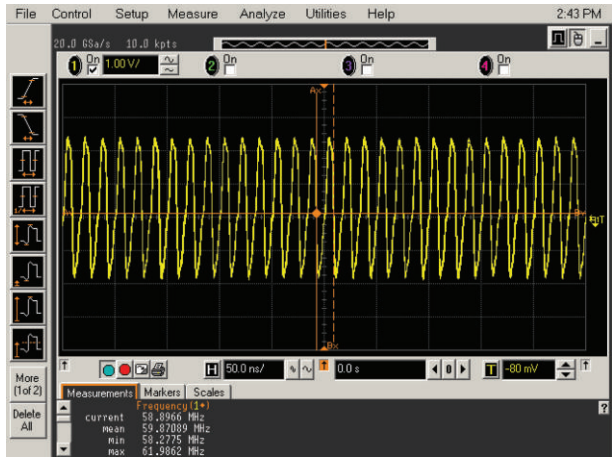


Figure 32: Measurement results from output signal F_O[9]. The measured signal is F_O[9]/8.

Table 1: Comparison table.

Item	Unit	Proposed clock generator	Conventional PLL in Figure 2
Structure	—	2 ECDLL + 1 DLL	1 PLL
Frequency	MHz	480	480
Locking-time	μs	3.5	10.0
Output jitter	ps	0.8	2.0
Technology	nm	90	90
Power	mW	1.3	3.8
Area	μm^2	200×225	200×500

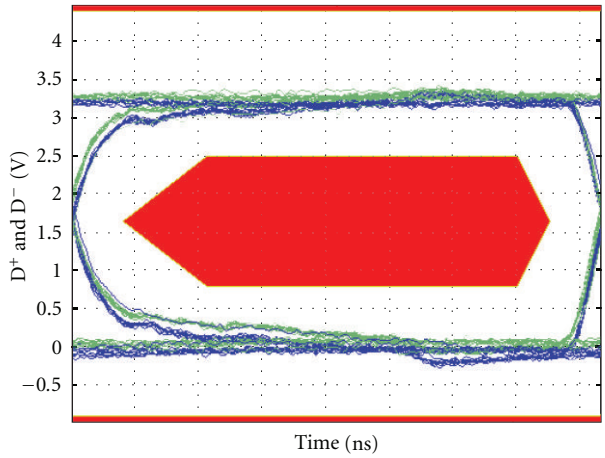


Figure 33: Measurement results from EYE pattern of fabricated PHY. The results meet the USB 2.0 specifications.

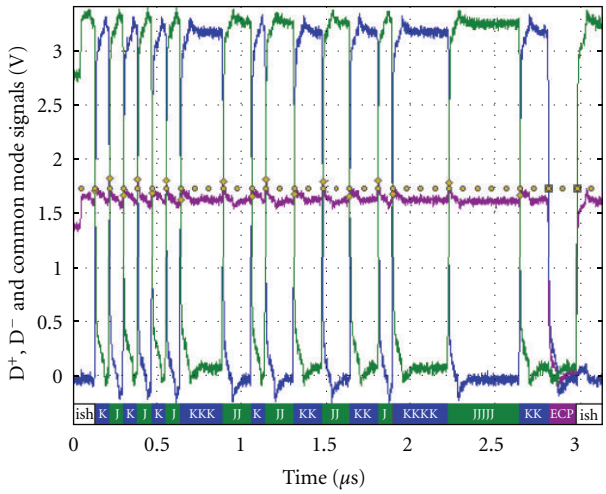


Figure 34: Measurement results from random data pattern of USB 2.0 specifications.

generator consists of three DLLs that is half the design area as that of the conventional one that consists of the PLL. Our clock generator consists of three DLLs. However, each DLL has a small capacitor to maintain the loop stability. Thus, our clock generator is smaller than the conventional one that has a large capacitor in the loop filter. Table 1 is a comparison table. The proposed clock generator has a power consumption of 1.3 mW, which is less than that of the conventional one, which is based on the PLL as shown in Figure 2. The ECDLL operates at the necessary reference signal frequency in the DLL loop that includes the VCDL. Thus, the power consumption is less than that of a PLL that has a VCO and a divider. A locking time of less than 3.5 μs can also be achieved.

5. Conclusion

We proposed novel clock generator architecture to shrink the design area. The proposed clock generator consists of two edge-combiner DLLs and a DLL. A shot pulse generator is used in the DLLs to prevent from harmonic lock and a CP with common-mode feedback is used in the DLLs to reduce the pattern jitter due to a constant phase error. A controller is used to control the wake-up sequence to prevent malfunctions. Our proposed clock generator is fabricated using a 90 nm CMOS process. It can achieve 10-tap 480 MHz clock signals that meet the USB 2.0 specifications. A power consumption of less than 1.3 mA was also achieved. Our USB 2.0 PHY with this clock generator also meets the USB 2.0 specifications. Our proposed clock generator needs only half the design area of the conventional one, which is based on the PLL.

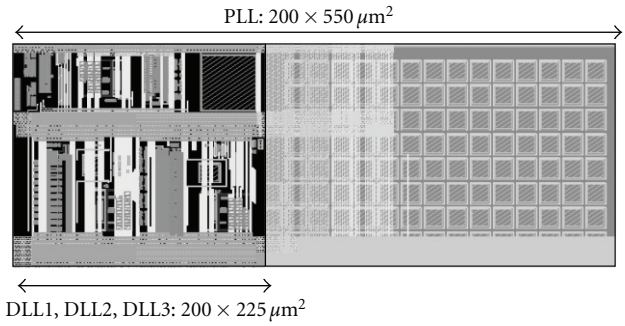


Figure 35: Chip layout. Our fabricated DLL occupies 200 × 225 μm.

References

[1] Y. Moon, G. Ahn, H. Choi, N. Kim, and D. Shim, "A quad 6 Gb/s multi-rate CMOS transceiver with TX rise/fall-time control," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '06)*, pp. 79–84, February 2006.

[2] T. Kawamoto, T. Takahashi, S. Suzuki, T. Noto, and K. Asahina, "Low-jitter fractional spread-spectrum clock generator using fast-settling dual charge-pump technique for serial-ATA application," in *Proceedings of the 35th European Solid-State Circuits Conference (ESSCIRC '09)*, pp. 380–383, September 2009.

[3] W. Grollitsch, R. Nonis, and N. Da Dalt, "A 1.4psrms-period-jitter TDC-less fractional-N digital PLL with digitally controlled ring oscillator in 65 nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '10)*, pp. 478–479, February 2010.

[4] "USB 2.0 transceiver macrocell interface (UTMI) specification version 1.05 3/29/2001," 2001, http://developer.intel.com/technology/usb/download/2_0_Xcvr_Macrocell_1_05.pdf .

[5] G. Chien and P. R. Gray, "A 900 MHz local oscillator using a DLL-based frequency multiplier technique for PCS applications," in *Proceedings of the IEEE International Solid-State Circuits Conference 47th Annual ISSCC*, pp. 202–203, February 2000.

[6] K. H. Cheng, S. M. Chang, Y. L. Lo, and S. Y. Jiang, "A 2.2 GHz programmable DLL-based frequency multiplier for SOC applications," in *Proceedings of the IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, pp. 72–75, August 2004.

[7] H. Y. Huang and J. H. Shen, "A DLL-based programmable clock generator using threshold-trigger delay element and circular edge combiner," in *Proceedings of the IEEE Asia-Pacific*

Conference on Advanced System Integrated Circuits, pp. 76–79, August 2004.

[8] K. H. Cheng, S. M. Chang, S. Y. Jiang, and W. B. Yang, "A 2 GHz fully differential DLL-based frequency multiplier for high speed serial link circuit," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 2, pp. 1174–1177, May 2005.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 235843, 9 pages
doi:10.1155/2011/235843

Research Article

Semidigital PLL Design for Low-Cost Low-Power Clock Generation

Ni Xu, Woogeun Rhee, and Zhihua Wang

Institute of Microelectronics, Tsinghua University, Beijing 100084, China

Correspondence should be addressed to Woogeun Rhee, wrhee@tsinghua.edu.cn

Received 15 May 2011; Revised 5 September 2011; Accepted 9 September 2011

Academic Editor: Sudhakar Pamarti

This paper describes recent semidigital architectures of the phase-locked loop (PLL) systems for low-cost low-power clock generation. With the absence of the time-to-digital converter (TDC), the semi-digital PLL (SDPLL) enables low-power linear phase detection and does not necessarily require advanced CMOS technology while maintaining a technology scalability feature. Two design examples in 0.18 μm CMOS and 65 nm CMOS are presented with hardware and simulation results, respectively.

1. Introduction

As the system integration complexity increases, robust low-cost frequency generation is highly demanded. Especially, the use of advanced CMOS technologies makes the traditional phase-locked loop (PLL) design challenging as on-chip variability and modeling inaccuracy become severe in deep submicron CMOS. Large loop parameter variation makes it difficult to find the optimum bandwidth for phase noise, spur, and settling time. In addition, analog passive devices become a bottleneck for scalability and integrating the loop filter (LPF) has been a challenging task in the conventional PLL design. Figure 1 depicts an example showing large area contribution of the on-chip loop filter to the PLL. Since the capacitor takes a significant portion of the whole LPF area, the gate leakage current by the on-chip MOS capacitor becomes substantial enough to affect the PLL performance, degrading the static phase error or reference spur performance. As a result, thick-oxide MOSFETs or metal-to-metal capacitors are used for the PLL loop filters at the cost of using an extramask.

2. Design Issues in All-Digital PLL

While integrating a loop filter has been a challenging task in the conventional PLL design, removing the analog loop filter is considered an alternative solution in the recent PLL works [1–13]. However, the all-digital PLL (ADPLL) requires a high-resolution complex time-to-digital converter (TDC) which requires advanced CMOS technology. Use of the bang-bang phase detector (BBPD) relaxes the TDC requirement but suffers from a nonlinear PLL bandwidth control [2]. In this paper, we present recent architectures of hybrid PLL systems which reduce technology dependency.

In the ADPLL design, high resolution of the TDC as shown in Figure 2 is important not only to enhance linearity but also to reduce in-band phase noise of the ADPLL. For the given reference clock frequency F_{REF} and the VCO frequency F_{VCO}, the in-band phase noise L of the ADPLL due to the TDC time resolution Δt_{res} is given by [1]

$$L = \frac{(2\pi)^2}{12}\left(\frac{\Delta t_{\mathrm{res}}}{T_{\mathrm{VCO}}}\right)^2 \cdot \frac{1}{F_{\mathrm{REF}}}. \tag{1}$$

The equation implies that finer TDC resolution is required for higher VCO output frequency. In fact, this is analogous to the fact that noise contribution of the phase detector (PD) increases with high division ratio N by the factor of $20 \log N$ in the conventional analog PLL design. Therefore, the ADPLL design also has difficulty in achieving low in-band phase noise performance with high division ratio. Besides, the ADPLL requires advanced CMOS technology for low in-band noise performance based on the above equation, which is different from the analog PLL. In addition to the advanced technology requirement, the TDC is sensitive

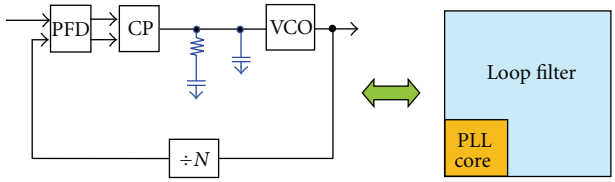


Figure 1: Loop filter area contribution to PLL in advanced CMOS.

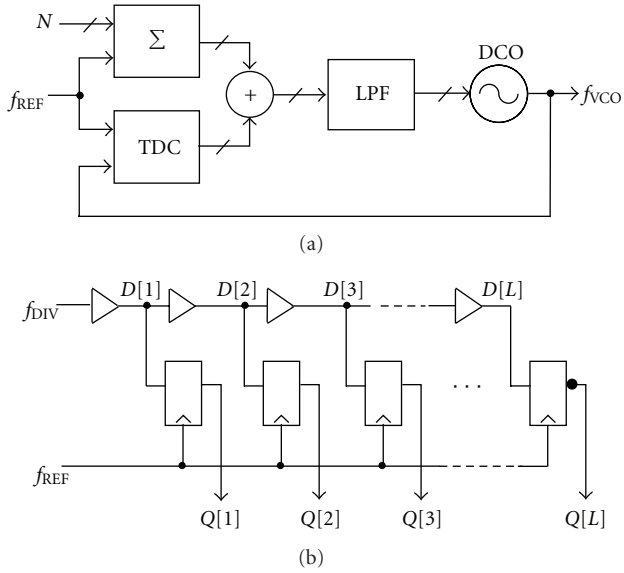


Figure 2: ADPLL with linear TDC [1].

to PVT variation. Typical delay time variation of a single inverter exhibits nearly 50% variation over process and temperature. Such a high sensitivity can cause poor linearity and nonuniform phase detector gain, resulting in widespread spur generation.

Table 1 shows architecture comparison between the ADPLL and the conventional analog PLL which typically consists of the phase-frequency detector (PFD) and the charge pump (CP). The conventional analog PLL suffers from poor scalability and leakage current sensitivity mainly due to the analog loop filter and does not offer good control of loop parameters compared to the ADPLL. On the other hand, the ADPLL features high scalability and reconfigurability with digital implementation but suffers from design complexity and nonlinear loop dynamics. Since the digitally controlled oscillator (DCO) has many switches with parasitic capacitance and the TDC requires fine-timing resolution using an advanced CMOS technology is highly demanded for the high performance ADPLL design.

3. Technology Scalable Semidigital PLL

In this paper, we consider a low-cost TDC-less semidigital PLL architectures [14–17] which do not require a large integration capacitor in the LPF, achieving technology scalability and leakage current immunity like the ADPLL.

Table 1: ADPLL versus conventional PLL.

	ADPLL	Conventional PLL
Power	Fair (depends on tech)	Good
Reconfigurability	Good	Poor
Scalability	Good	Poor
I_{Leak} Immunity	Good	Poor
Linear BW control	Fair	Good
Design complexity	High	Fair
Tech. dependency	High	Fair

3.1. Basic Concept. The type II PLL inherently provides an integral path which tracks frequency offset independently so that, in theory, the static phase error can be zero even with the frequency offset. Figure 3 shows how the type II PLL obtains frequency acquisition without generating a static phase error. As far as phase tracking is concerned, the integral path is a large-signal path while the proportional-gain path is a small-signal path. When the large-signal path slowly tunes the VCO to the desired frequency, the small-signal path does not have to provide additional DC information for frequency acquisition. Therefore, different implementation for each path is possible in the type II PLL design, namely, the integral path in digital and the proportional-gain path in analog.

Figure 4 shows the basic concept of the semi-digital loop control [14], and a linear model is shown in Figure 4(b). Since the control path of the type II PLL can be decomposed into a proportional-gain path and an integration path as discussed, independent implementation is considered for each path. For the proportional-gain path, the conventional analog control is used except the absence of the integrating capacitor. Since the capacitance values for high-order poles are not high, either the MOS capacitor with negligible leakage current or the MIM capacitor can be used. As for the integral path, digital implementation is done with the BBPD and the FSM to compensate for the limited frequency tracking capability of the proportional-gain path. The $\Delta\Sigma$ modulator is used to provide fine frequency resolution as done in the ADPLL. The main purpose of the digital integration path is to provide frequency tracking rather than phase tracking. Accordingly, the time constant of the digital integration path can be much longer than the analog small-signal path, resulting in overdamped loop dynamics. With the overdamped loop dynamics, the PLL bandwidth is linearly controlled by the charge pump current whose value can be digitally programmable in the design. In summary, the proposed hybrid loop control with the analog proportional path and the digital integration path provides linear phase tracking, leakage-insensitive loop filtering, technology scalability, and uniform PD gain capability.

3.2. Architecture Comparison with Other PLLs. For the SDPLL design with an LC VCO, three different topologies can be considered for the analog proportional-gain path as shown in Figure 5. Figure 5(a) is based on the CP PLL topology as already presented previously [14]. Other way

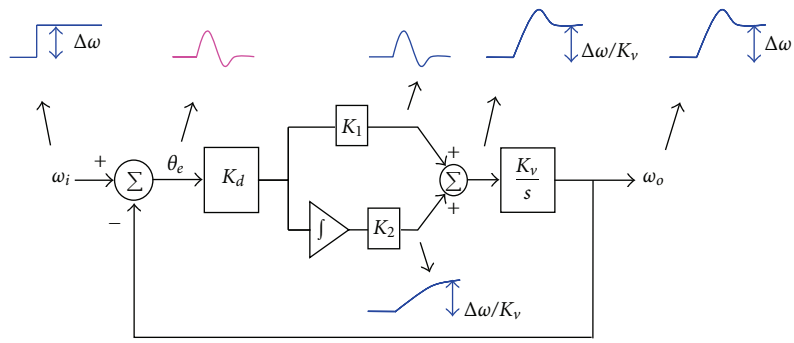
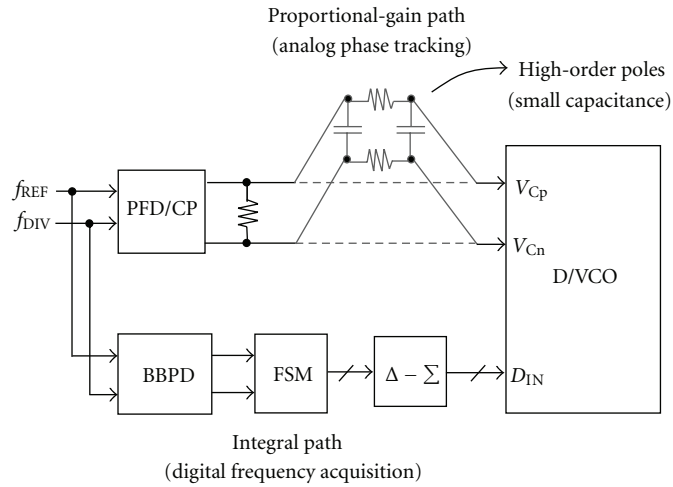
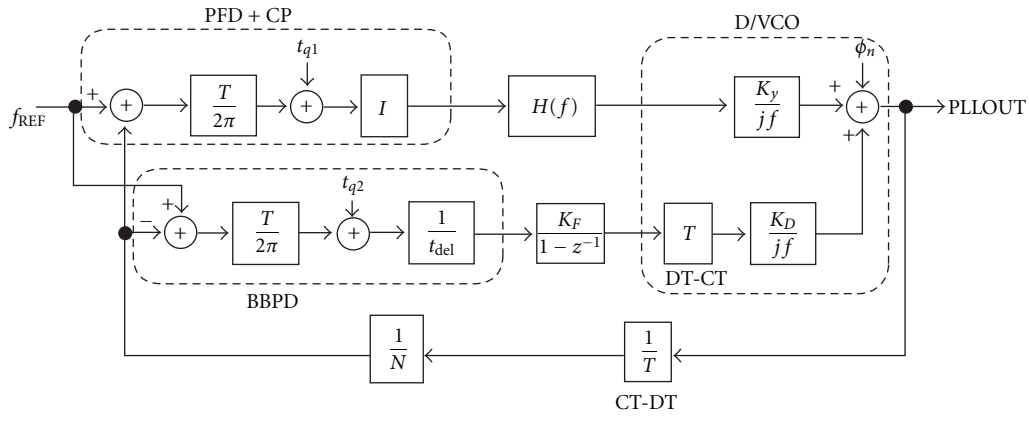


Figure 3: State-variable model of type II PLL.



(a)



(b)

Figure 4: (a) Basic concept and (b) linear model.

is to have the PFD output directly connected to the VCO input [15]. In this case, a linear amplifier (LA) is needed to set the optimum common mode voltage for maximum varactor tuning range. The additional LA degrades VCO noise performance. However, the LA noise contribution can be suppressed by the PLL bandwidth since the LA is placed after the LPF and gets high-pass noise transfer function by the PLL. Without the CP, Figure 5(b) can achieve better in-band phase noise performance. When the PLL bandwidth is narrow and requires good phase noise performance of the VCO, the topology from Figure 5(a) should be chosen. In addition, the phase detector gain can be well regulated over PVT variations if the bias current of the charge pump is generated by an on-chip resistor and a bandgap reference

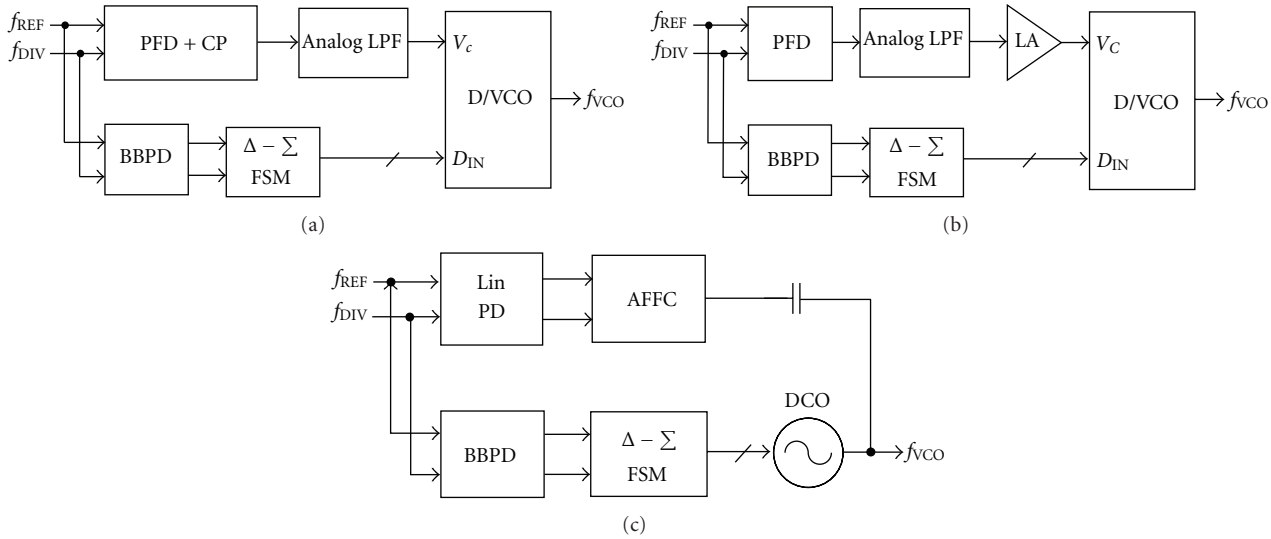


Figure 5: (a) PFD/CP based, (b) PFD/LA based, and (c) AAFC based.

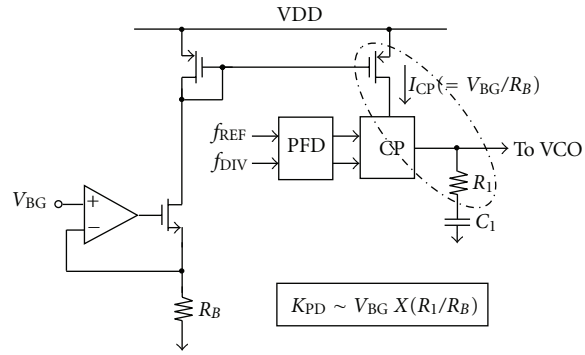


Figure 6: CP biasing for uniform PD gain.

voltage [16], which is illustrated in Figure 6. The last one shown in Figure 5(c) uses analog feed-forward circuits (AFFC) to provide a linear phase modulation path [17]. However, having the AC-coupling path at the VCO output requires more complicated design efforts than employing the dual-control path at the VCO input since dealing with high frequency signals is more difficult. Moreover, the analog RC filter in the AFFC connected to both supply and ground can cause a coupling path to the supply noise.

As depicted in Figure 7, the SDPLL offers moderate performance between the conventional analog PLL and the ADPLL. A mixed-mode loop control with an analog proportional path and a digital integration path offers a leakage-insensitive and technology scalable architecture comparable to the digital PLL, while maintaining low-cost linear phase detection like the analog PLL. The PFD/CP-based proportional-gain path provides linear loop dynamics in which tracking bandwidth is simply set by the PD gain, the passive LPF transfer function, and the analog VCO gain. In addition, with the absence of the linear TDC, power consumption can be reduced and using advanced

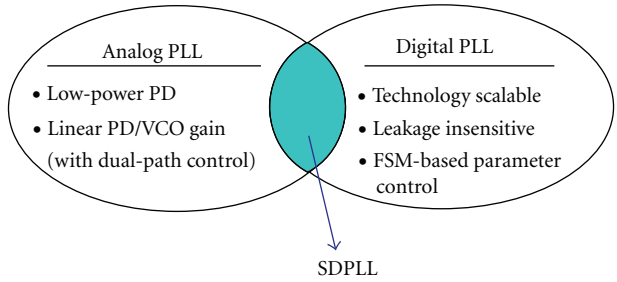


Figure 7: Architecture comparison.

CMOS technology is not a must for achieving good noise performance.

4. Design Examples

In this paper, two SDPLL design examples are presented; one designed in 0.18 μm CMOS for digital clock generation and

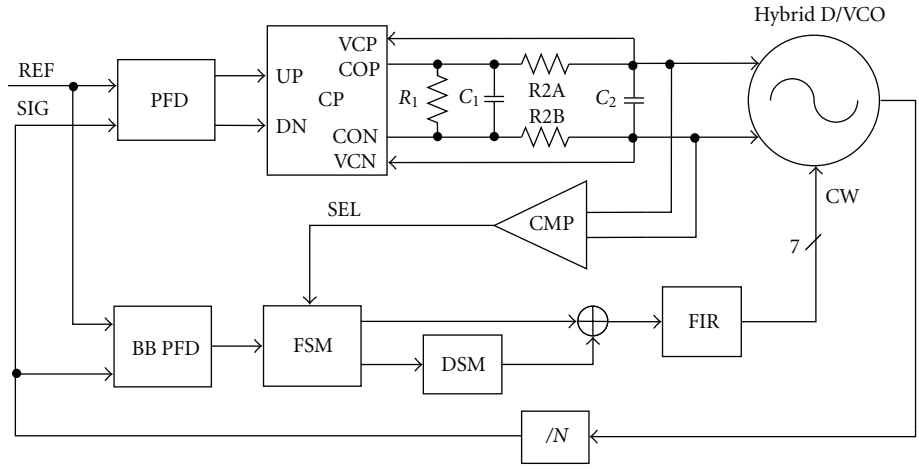


Figure 8: SDPLL block diagram.

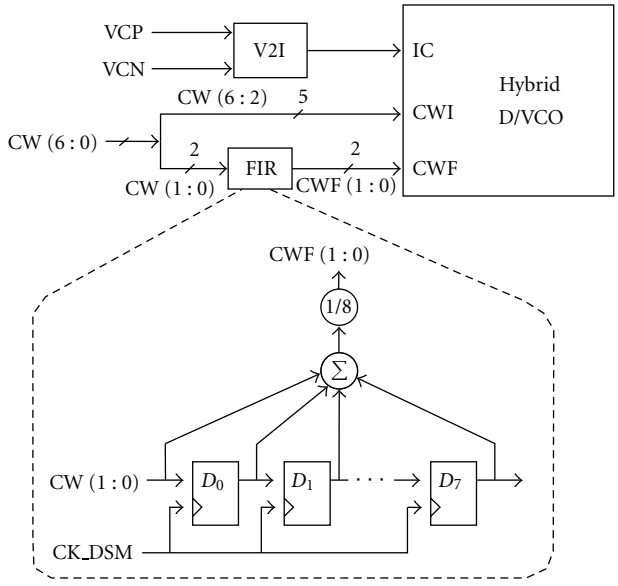


Figure 9: Hybrid D/VCO with embedded FIR filtering.

the other in 65 nm CMOS for wireless applications (Table 2). The former shows that the hybrid loop control is successfully verified in hardware, and the latter shows promising low-power feature of the SDPLL for two-point modulation with a small area comparable to the ADPLL-based modulation.

4.1. 0.18 μm CMOS SDPLL for Digital Clock Generation. Figure 8 shows a block diagram of the SDPLL [14]. To minimize noise coupling, a differential charge pump followed by a differential loop filter is designed in the analog control path. For the 3rd- and 4th-order poles, the MIM capacitor is used to have good isolation from the substrate noise coupling. In the digital integration path, the BBPD is used to provide bi-level information to the 18-bit FSM, where the 7-bit output from the MSB is the integral part and the following 8-bit output is the fractional part. The remaining 3-bit output is used for averaging function. The frequency resolution set by the digital tuning loop is about 2.1 MHz per LSB. Since the digital integration path has slow frequency acquisition, an adaptive bandwidth scheme is designed. For that purpose, a voltage comparator is added to provide transition information to the FSM.

Figure 9 shows the hybrid DCO block diagram with the FIR-based Δ-Σ control, where a 7-bit control input is used. A 2nd-order MASH modulator is used for its simple structure. The 2nd-order MASH modulator has 2-bit output CW (1 : 0), so only the last two bits from the LSB are controlled by the modulator. The 5-bit static input, CW (6 : 2) directly controls the digital input of the hybrid DCO. Since the hybrid DCO has a 5-bit static input and a 2-bit dynamic modulated by the 10-bit modulator, the total frequency resolution of 15 bit is obtained in the digital control path.

Figure 10 shows the behavioral simulation results of the PLL settling behavior, (a) without digital path and (b) with

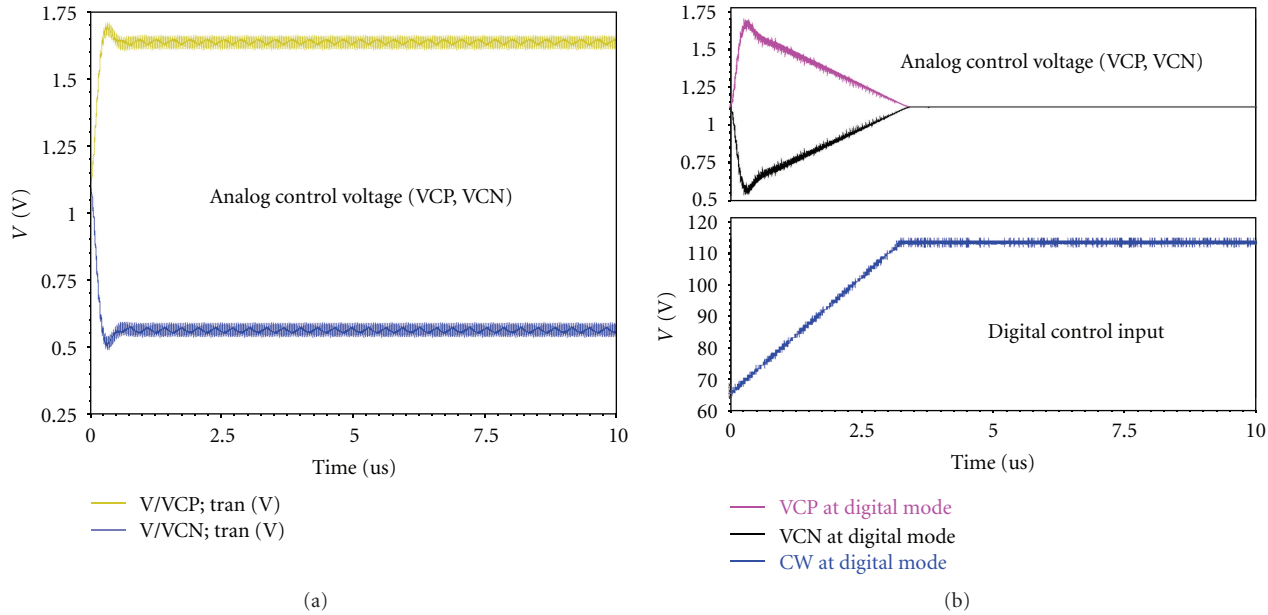


Figure 10: Simulated transient settling voltage: (a) with digital path disabled, and (b) with digital path enabled.

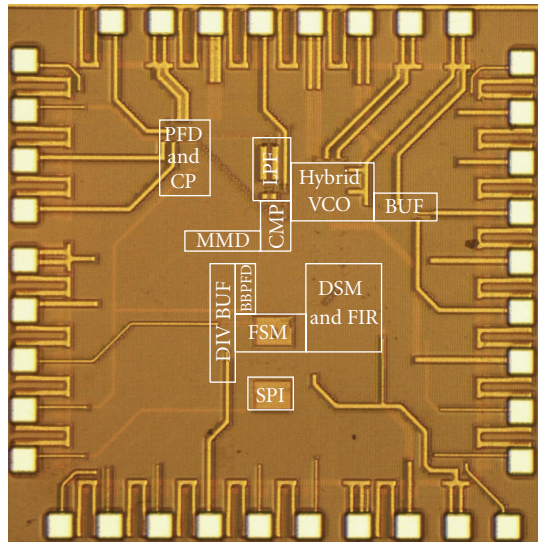


Figure 11: Chip micrograph [14].

digital path. When the digital path is not enabled, frequency acquisition is done only by the proportional-gain path, resulting in an analog type I PLL. As a result, large static phase error is observed when the PLL needs to track the frequency offset. With the digital path enabled, the frequency is tracked by the digital integration path. Consequently, the control voltage is settled within a very small range even with the frequency offset as shown in Figure 10(b), showing that the type II PLL is realized.

Figure 11 shows the micrograph of the test chip fabricated in 0.18 μm CMOS. The active core area is 0.6 mm^2 where only 0.01 mm^2 is occupied by the analog loop filter. Figure 12 shows the measured output spectra at 870 MHz frequency with the reference clock frequency of 30 MHz. The upper plot shows the output spectrum with the digital path disabled. Since the PLL becomes a type I PLL without the digital integration path, a large static phase offset is generated, resulting in the spur level as high as –30 dBc. When the digital integration path is enabled, the reference spur is reduced by more than 20 dB. Also, the DC control voltage range is settled within ±0.02 V when the digital path is enabled. The experimental results prove that the type-II PLL is realized with the hybrid loop control.

Since the phase noise contribution from the analog control path is worse than expected, it is difficult to see the effect of the hybrid FIR filter. Figure 13 shows the phase

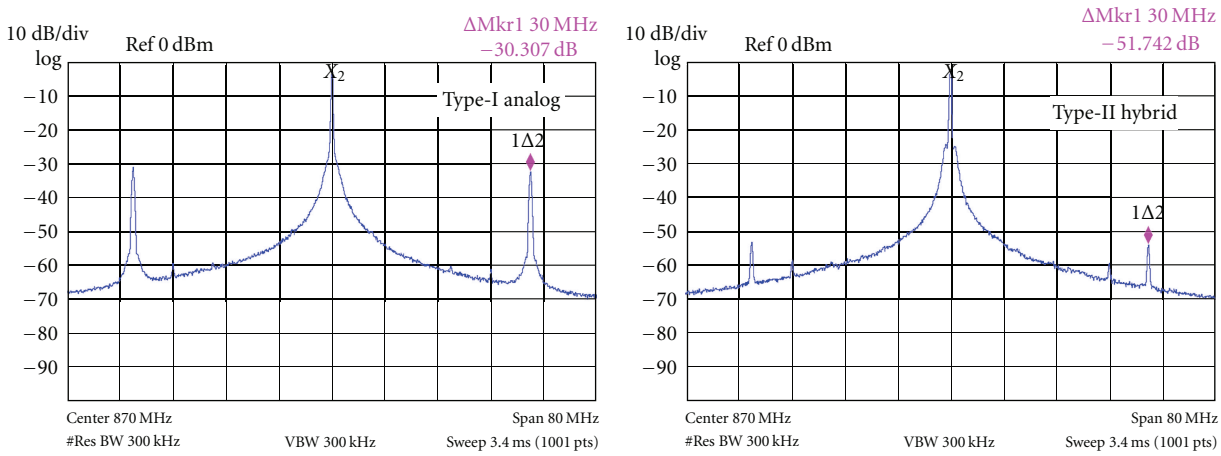


Figure 12: Measured output spectra: (a) with digital path disabled and (b) with digital path enabled.

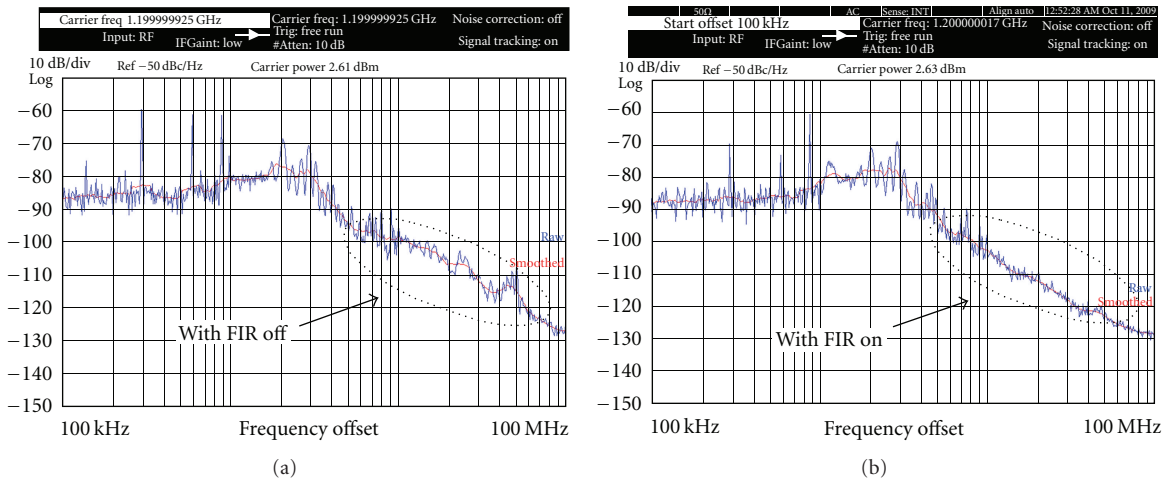


Figure 13: Measured output spectra: (a) with FIR disabled and (b) with FIR enabled.

Table 2: Measured performance summary [14].

Process	0.18 μm CMOS
Supply Voltage	1.8 V for analog, 1.5 V for digital
Power consumption	Total: 16.8 mW (Analog: 11.9 mW, Digital: 4.9 mW)
Occupied area	Active area: ~0.6 mm^2 (LPF < 0.04 mm^2)
VCO tuning range	790–925 MHz
Reference clock	30 MHz
Reference spur	< −52 dBc
Phase noise	< −81 dBc/Hz
Integrated RMS noise	100 kHz~100 MHz: 12.6$^\circ_{\text{rms}}$ 10 MHz~100 MHz: 1.1$^\circ_{\text{rms}}$

noise performance of the SDPLL output from the other test site, in which the hybrid FIR filter embedded in the D/VCO clearly reduces the high-frequency noise caused by the Δ-Σ modulation.

4.2. 65 nm CMOS SDPLL for Two-Point Modulation. The SDPLL architecture can be further extended to accommodate two-point phase modulation for RF transmitter systems, which has been well demonstrated by the ADPLL in the literature [1]. Figure 14 shows a simplified block diagram of the fractional-N SDPLL having the two-point modulation feature for GSM/GPRS applications. Similar to the ADPLL, the DCO gain can be calibrated easily by measuring the frequency step for the 1-LSB change since phase modulation is done in the digital domain with the DCO input control. Also, the group delay mismatch can be controlled to a certain degree by embedding the high-frequency DFFs in the FSM. Since the noise transfer functions of the DCO and the fractional-N divider are still controlled by the PLL loop dynamics, the use of the PFD and the CP offers linear control. Without using the TDC, overall power consumption

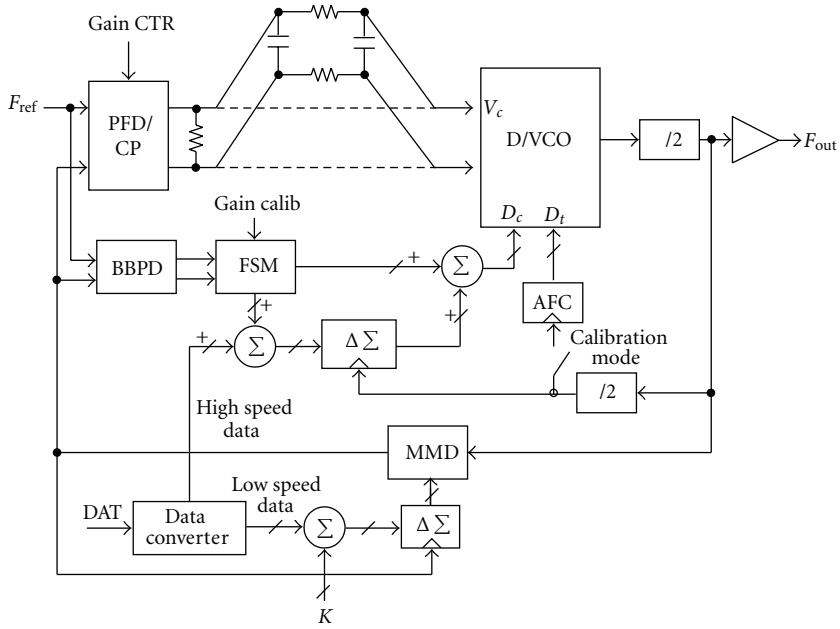


Figure 14: SDPLL with two-point modulation for RF transmitter systems.

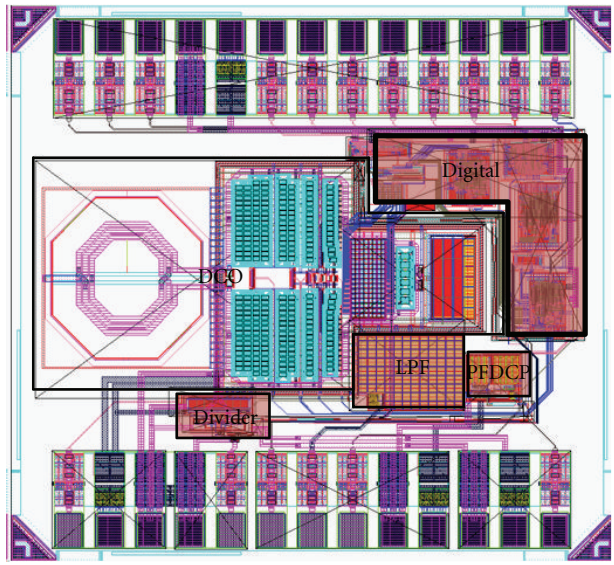


Figure 15: SDPLL layout in 65 nm CMOS (active area: 0.52 mm^2).

of the SDPLL can be further reduced while the total area is comparable to the ADPLL.

Figure 15 shows a layout of the SDPLL in 65 nm CMOS, occupying the active area of only 0.52 mm^2. The active area is much smaller than that of the conventional analog PLL and comparable to that of the ADPLL. Except the ΔΣ modulator and high-frequency control blocks, most digital operations for the two-point modulation are realized by external control with FPGA. Figure 16 shows power consumption analysis of the SDPLL based on postlayout simulation results. The D/VCO consumes nearly 80% of the total power. The power contribution of the PFD and the CP is less than 5% of the total power, showing that the SDPLL can save more power than the ADPLL by not employing the power-hungry TDC.

5. Conclusion

In this paper, some design issues in recent clock generation systems are reviewed and the SDPLL is presented as a low-power technology-friendly architecture. We also show that

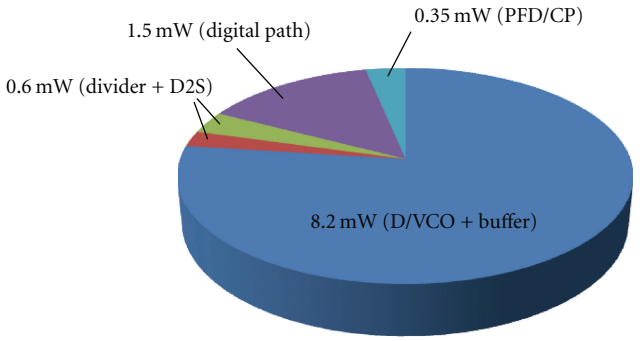


Figure 16: Power consumption analysis.

the SDPLL architecture can be further tailored for various applications without necessarily requiring advanced CMOS technology.

References

[1] R. B. Staszewski, J. L. Wallberg, S. Rezeq et al., "All-digital PLL and transmitter for mobile phones," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2469–2480, 2005.

[2] J. A. Tierno, A. V. Rylyakov, G. J. English, D. Friedman, and M. Meghelli, "A wide power supply range, wide tuning range, all static CMOS all digital PLL in 65 nm SOI," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 42–51, 2008.

[3] C. M. Hsu, M. Z. Straayer, and M. H. Perrott, "A low-noise wide-BW 3.6-GHz digital ΔΣ fractional-N frequency synthesizer with a noise-shaping time-to-digital converter and quantization noise cancellation," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, Article ID 4684627, pp. 2776–2786, 2008.

[4] M. Lee, M. E. Heidari, and A. A. Abidi, "A low-noise wideband digital phase-locked loop based on a coarse-fine time-to-digital converter with subpicosecond resolution," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 10, article 23, pp. 2808–2816, 2009.

[5] M. Zanuso, S. Levantino, C. Samori, and A. Lacaita, "A 3 MHz-BW 3.6 GHz digital fractional-N PLL with sub-gate-delay TDC, phase-interpolation divider, and digital mismatch cancellation," pp. 476–477.

[6] E. Temporiti, C. Weltin-Wu, D. Baldi, M. Cusmai, and F. Svelto, "A 3.5 GHz wideband ADPLL with fractional spur suppression through TDC dithering and feedforward compensation," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 12, Article ID 5604330, pp. 2723–2736, 2010.

[7] S.-K. Lee, Y.-H. Seo, H.-J. Park, and J.-Y. Sim, "A 1 GHz ADPLL with a 1.25 ps minimum-resolution sub-exponent TDC in 0.18 μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 12, Article ID 5609226, pp. 2874–2881, 2010.

[8] H.-H. Chang, P.-Y. Wang, J. H. C. Zhan, and B. Y. Hsieh, "A fractional spur-free ADPLL with loop-gain calibration and phase-noise cancellation for GSM/GPRS/EDGE," in *Proceedings of the IEEE International Solid State Circuits Conference, (ISSCC, '08)*, pp. 200–201, February 2008.

[9] W. Grollitsch, R. Nonis, and N. Da Dalt, "A 1.4 psrms-period-jitter TDC-less fractional-N digital PLL with digitally controlled ring oscillator in 65nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '10)*, pp. 478–479, February 2010.

[10] P.-H. Hsieh, J. Maxey, and C. K. K. Yang, "A phase-selecting digital phase-locked loop with bandwidth tracking in 65-nm CMOS technology," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, Article ID 5437483, pp. 781–792, 2010.

[11] D.-S. Kim, H. Song, T. Kim, S. Kim, and D. K. Jeong, "A 0.3–1.4 GHz all-digital fractional-N PLL with adaptive loop gain controller," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 11, Article ID 5607237, pp. 2300–2311, 2010.

[12] T. Tokairin, M. Okada, M. Kitsunezuka, T. Maeda, and M. Fukaishi, "A 2.1-to-2.8-GHz low-phase-noise all-digital frequency synthesizer with a time-windowed time-to-digital converter," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 12, Article ID 5604672, pp. 2582–2590, 2010.

[13] M. Chen, D. Su, and S. Mehta, "A calibration-free 800 MHz fractional-N digital PLL with embedded TDC," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 12, Article ID 5610982, pp. 2819–2827, 2010.

[14] R. He, C. Liu, X. Yu et al., "A low-cost, leakage-insensitive semi-digital PLL with linear phase detection and FIR-embedded digital frequency acquisition," in *Proceedings of the IEEE Asian Solid-State Circuits Conference, (A-SSCC '10)*, pp. 197–200, 2010.

[15] W. Yin, R. Inti, and P. K. Hanumolu, "A 1.6 mW 1.6ps-rms-Jitter 2.5 GHz digital PLL with 0.7-to-3.5 GHz frequency range in 90 nm CMOS," in *Proceedings of the 32nd Annual Custom Integrated Circuits Conference, (CICC '10)*, September 2010.

[16] W. Rhee, H. Ainspan, D. J. Friedman, T. Rasmus, S. Garvin, and C. Cranford, "A uniform bandwidth PLL using a continuously tunable single-input dual-path LC VCO for 5 Gb/s PCI express Gen2 application," in *Proceedings of the IEEE Asian Solid-State Circuits Conference, (A-SSCC '07)*, pp. 63–66, November 2007.

[17] P.-Y. Wang, J. H. C. Zhan, H. H. Chang, and H. M. S. Chang, "A digital intensive fractional-N PLL and all-digital self-calibration schemes," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 8, Article ID 5173739, pp. 2182–2192, 2009.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 871385, 11 pages
doi:10.1155/2011/871385

Research Article

An Interpolated Flying-Adder-Based Frequency Synthesizer

Pao-Lung Chen and Chun-Chien Tsai

Department of Computer and Communication Engineering, National Kaoshiung First University of Science and Technology, No. 2, Jhuoyue Road, Nanzih District, Kaohsiung City 811, Taiwan

Correspondence should be addressed to Pao-Lung Chen, plchen@nkfust.edu.tw

Received 8 June 2011; Revised 25 August 2011; Accepted 25 August 2011

Academic Editor: Jae-Yoon Sim

This work presents an interpolated flying-adder- (FA-) based frequency synthesizer. The architecture of an interpolated FA, which uses an interpolated multiplexer (MUX) to replace the multiplexer in conventional flying adder, improves the cycle-to-cycle jitter and root-mean-square (RMS) jitter performance. A multiphase all-digital phase-locked loop (ADPLL) provides steady reference signals for the interpolated flying adder. This paper reveals implementation skills of a multiphase ADPLL, as well as an interpolated flying adder. In addition, analytical details of the jitter performance are derived. A test chip for the proposed interpolated FA-based frequency synthesizer was fabricated in a standard 0.18 μm CMOS technology, and the core area was 0.143 mm^2. The output frequency had a range of 33 MHz ~ 286 MHz at 1.8 V with peak-to-peak (P_k-P_k) jitter 215.2 ps at 286 MHz/1.8 V.

1. Introduction

The frequency synthesizer is a key component for numerous systems to generate a desired frequency for frequency conversion in digital communication or in system on chip (SoC) for signal synchronization [1]. Conventional approaches in [2–4] have utilized phase-locked loop (PLL) to generate different high-frequency outputs with a low-frequency crystal clock by setting the frequency control word that is widely used in the industry. While PLL provides flexible frequency synthesis, the loop parameters, such as damping factor and loop bandwidth, must be adjusted to minimize jitter and to ensure that each output frequency and frequency control word is stable. The loop bandwidth should be approximately 1/20 of the reference frequency. To reduce cost and enhance loop stability, an all-digital PLL (ADPLL) in [5] utilized a digital loop filter with a seven-cycle lock time. However, the output frequency range is less than 100 MHz. With a fractional-N synthesis technique [6], finer frequency control can be achieved. But the system typically has a narrow bandwidth that has tight control on loop parameters.

Direct digital synthesis (DDS) in [7] uses memory and logic to generate the desired output frequency. It normally consists of a phase accumulator, an ROM lookup table and a linear digital-to-analog converter (DAC). The performance of a DAC is the bottleneck of this technique. In recent years, flying-adder frequency synthesis has provided a new way of generating frequency on chip [8–16]. The flying-adder-based frequency synthesizer solves problem that cannot be dealt easily with conventional PLL-based or DDS-based frequency synthesizer. The flying-adder frequency synthesizer is also called a direct digital period synthesizer (DDPS) in [17]. The flying-adder-based frequency synthesizer has been applied in many commercial chips such as the triple DAC graphics digitizer, digital speaker, LCD monitors, HDTV chips, and NTSC video decoders [11–13].

The basic structure of a flying adder is shown in Figure 1. The structure consists of an N equally spaced phases, a multiplexer (MUX), a D flip-flop (DFF), and an n-bit accumulator. The n-bit accumulator is composed of an n-bit adder and an n-bit register. The n-bit register is divided into the integer (Int) and fractional (Fract) parts. The frequency control word (FREQ) is inputted to the flying adder. The Fract is used for accumulating the fractional part. The multiplexer selects one of the input signals from the N equally spaced phases according to Int value. The output signal from the multiplexer triggers a D flip-flop. The function of D flip-flop is similar to a frequency divider

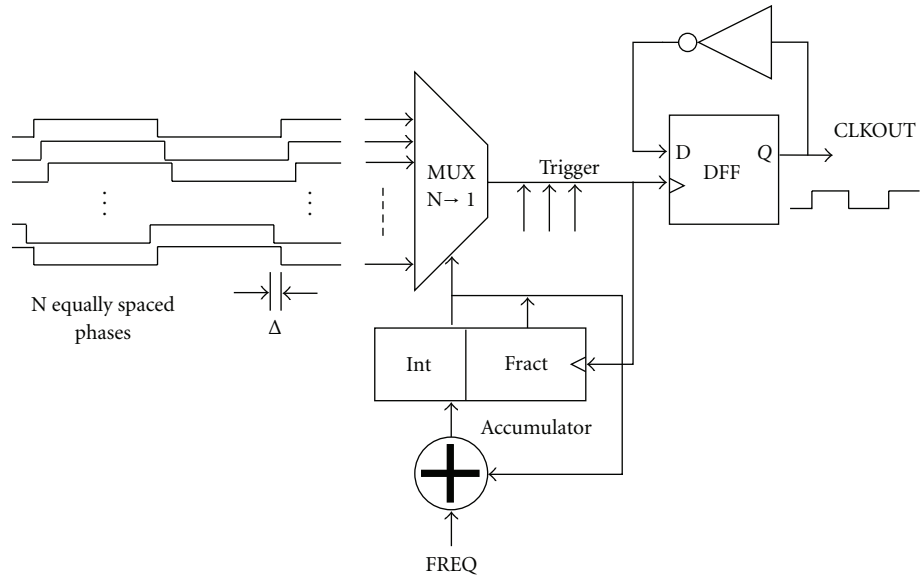


Figure 1: Block diagram of conventional flying adder [8].

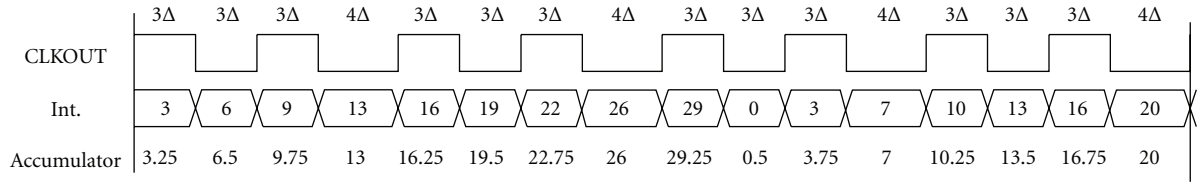


Figure 2: Numerical operation of [8]'s flying adder.

by-2 circuit. Meanwhile, the n-bit register for the MUX address is updated. Let us assume that FREQ is a 10-bit frequency control word with 5 bits for Int and 5 bits for Fract, respectively. The timing resolution of a 32 equally spaced phases with "Δ" is 0.2 ns. If a 769 MHz (1.3 ns) signal is the desired output, then the FREQ[9 : 0] can be calculated as follows:

$$\text{FREQ}[9:0] \cdot 2 \cdot 0.2\,\text{ns} = 1.3\,\text{ns}. \quad (1)$$

The FREQ[9 : 0] is equal to $3.25 = 00011.01000_{\text{b}}$. As illustrated in this numerical example, the clock cycle at that particular time is 0.2 ns (i.e., Δ) longer than a normal cycle whenever the value of fractional part is propagated to the integer part. This is called the cycle prolong in a flying-adder frequency synthesizer.

To explain clearly the cycle prolong of the flying-adder-based frequency synthesizer of [8], we assume that the FREQ = Int + Fract = 3.25 as in (1). The Int part is equal to 3, and the fractional part is 0.25.

The numerical operation of Figure 1's flying adder is illustrated in Figure 2. Let us assume that the phase resolution of N equally spaced signals is Δ. Whenever there is a rising edge from the MUX output, the value in the accumulator is accumulated with FREQ. The values of the accumulator are 3.25, 6.5, 9.75, and 13. The values of Fract part are 0.25, 0.5, 0.75, and 0. The generated trigger intervals are 3Δ, 3Δ, 3Δ, and 4Δ. The trigger interval between 3Δ and 4Δ is 1Δ because the accumulation of Fract part is propagated into the Int part. Therefore, the cycle-to-cycle jitter is 1Δ when the cycle is prolonged, as indicated in [8, 11, 14, 15]. The poor cycle-to-cycle jitter degrades the output clock's performance. Reference [11] solved the prolonged cycle using a post divider. However, the generated frequency is divided down. We propose an interpolated flying-adder structure to reduce jitter which is caused by the cycle prolong.

The rest of this paper is arranged as follows: Section 2 describes the proposed interpolated flying adder. The implementation of a frequency synthesizer using an interpolated flying adder is described in Section 3. In Section 4, the jitter analysis of proposed interpolated flying adder is provided. The experimental result is in Section 5. Finally, Section 6 presents a summary and conclusions.

2. Proposed Interpolated Flying Adder

We propose an interpolated flying-adder structure, which can reduce the cycle-to-cycle jitter to one-half, to solve the cycle prolong. The interpolated flying-adder structure is an improvement of [9] two path's flying-adder structure. The two-path flying adder is interlocked through two AND gates and XOR/XNOR gates between path A and path B as shown in Figure 3.

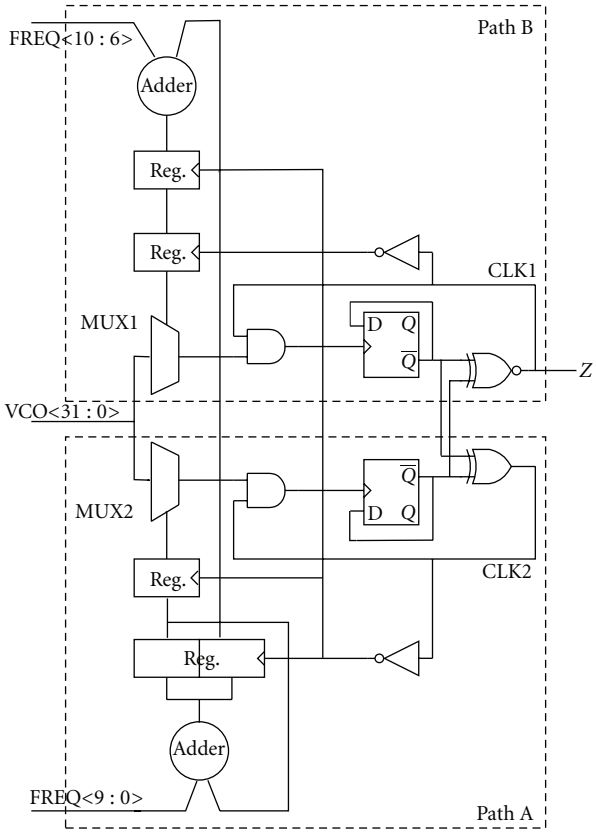


Figure 3: Two-path flying-adder frequency synthesizer [9].

The two AND gates and the feedback self-clocking of the registers ensure that at any given time, there is only one path switched on and one path switched off. The two-path flying-adder makes the 32 VCO ticks look like 64 ticks [19]. utilized an all-digital PLL with delay chain for generation of the 32 VCO ticks for two-path flying-adder frequency synthesizer. A cell-based implementation of two-path flying adder with dual resolution is indicated in [20]. However, the problem of cycle prolong is still unsolved in both one-path and two-path flying adders. Figure 4 is the block diagram of proposed interpolated flying adder. We modify the original two-path flying-adder structure. A digital interpolator is applied to interpolate the output from path A and path B. Both paths are simultaneously switched on in interpolated flying adder which is different from the two-path operation.

Path A is similar to path B. However, the difference is the bit width of accumulator. The input signal of Reg. A is the integer (Int) part of accumulator. The value of Reg. B is accumulated with the value in Reg. A and input phase control. The phase control is used to control the phase relationship between path A and path B. The output signals of path A and path B are interpolated by an interpolator. The interpolator receives two signals, path A_out and path B_out, to produce third output signal, CLKOUT. The rising edge of the out signal is in the midpoint of path A_out and path B_out as illustrated in Figure 5.

When FREQ = 3.25 is the same as in Figure 2 for the flying adder of [8], the PHASE = 3 is equal to the integer part of FREQ. Figure 6 shows the numerical operation of the proposed interpolated flying adder. The Int part value of the accumulator is stored in Reg. A. The Reg. A's value will be accumulated with phase into Reg. B triggered by the output MUX B. The output of path A and path B will be interpolated at the output of interpolator. Then the interpolated intervals are 3Δ, 3Δ, 3.5Δ, and 3.5Δ, as calculated in Figure 6. The trigger interval between 3Δ and 3.5Δ is 0.5Δ. Therefore, the cycle-to-cycle jitter of the proposed interpolated flying adder is improved 50% as compared with the flying adder in Figure 2.

3. Implementation of Interpolated Flying-Adder-Based Frequency Synthesizer

The interpolated flying-adder-based frequency synthesizer consists of two subsystems: one is the multiphase all-digital phase lock loop (ADPLL) to generate 32 equally spaced phases; the second subsystem is the interpolated flying-adder as described in Figure 4.

The operation of the system is based on the FREQ < 28 : 0 > which is similar to [8]. The FREQ < 28 : 0 > is the input frequency control word. The 5 MSB bits are used directly for selecting the MUX 32 inputs. The 24 LSB bits are the fractional part. Let us assume that the output frequency of multiphase ADPLL is running at 156.25 MHz (6.4 ns).

The time delay between the two adjacent outputs is 6.4 ns/32 = 0.2 ns. If a 161.29 MHz (6.2 ns) is the desired output, then the FREQ < 28 : 0 > can be calculated as follows: FREQ < 28 : 0 > ·2 · 0.2 ns = 6.2 ns $\Longrightarrow$ FREQ < 28 : 0 > = 15.5 = 01111.1000 0000 0000 0000 0000 0000_2.

3.1. Multiphase All-Digital Phase Locked Loop. The function of the multiphase all-digital phase locked loop (ADPLL) is to generate 32 equally spaced phases for the proposed interpolated flying-adder-based frequency synthesizer. The block diagram of multiphase ADPLL is indicated in Figure 7.

The multiphase ADPLL consists of two major function units. The first function unit is the timing control unit and SAR, and the second part is the digital-to-voltage converter (DVC) and multiphase voltage-controlled oscillator (VCO). The timing control unit performs timing selection of reference frequency as well as output-generated frequency. Successive approximation register (SAR) control is the key unit for frequency searching. The major advantages of the SAR unit are a compact structure as well as a regular layout [21].

The digital-to-voltage converter (DVC) generates the Vtune signal to control VCO, and detailed DVC can be found in [22]. The VCO has 32 NAND gates that are illustrated in Figure 8. The VCO has 16 stages, and each stage has two NAND gates. The output of each NAND gate is connected to the next stage, and the feedback is connected to the input of the same stage's NAND gate [11]. The structure of the VCO is highly regular. The output frequency of VCO ranges

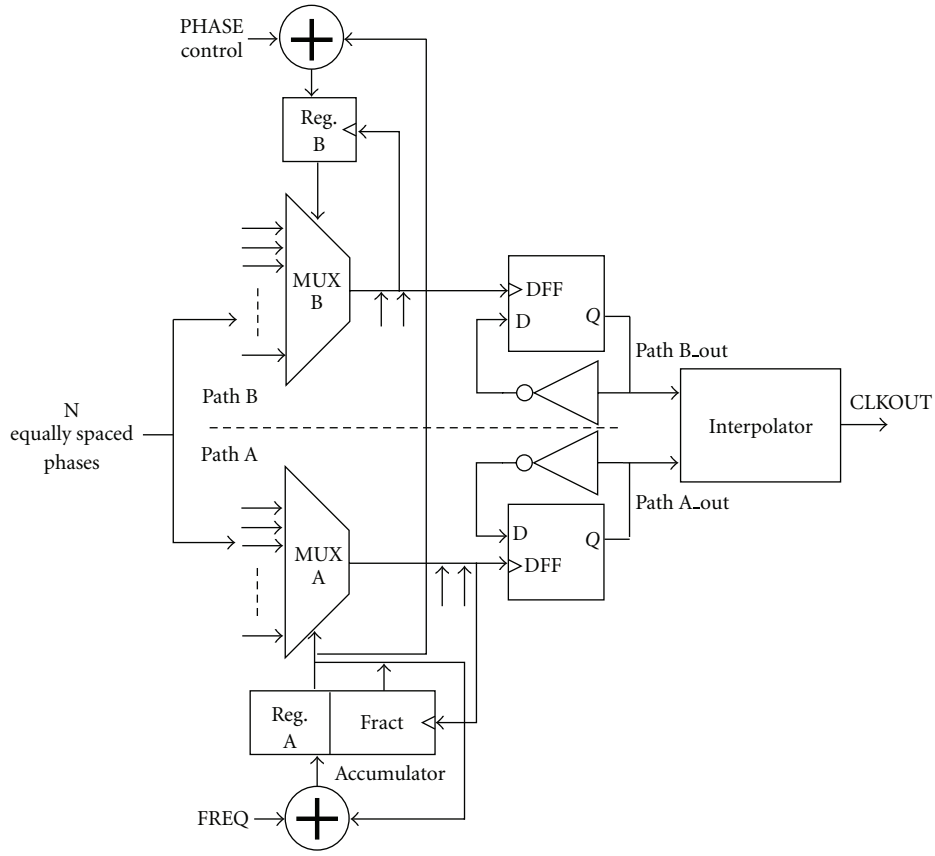


Figure 4: Block diagram of proposed interpolated flying adder.

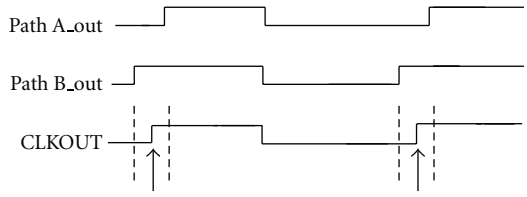


Figure 5: Function of the interpolator.

from 35 MHz to 235 MHz as shown in Figure 9. The output frequency of the multiphase ADPLL can be represented as

$$f_{\text{ADPLL}} = f_{\text{Ref}} \cdot \frac{N}{2}, \tag{2}$$

where f_{ADPLL} is the output frequency and f_{Ref} is the input reference frequency. The N is controlled by $m_0 \sim m_5$. To generate 50% duty cycle of the output frequency, the VCO output frequency passes a divide-by-2 circuit. Therefore, the output frequency of a multiphase ADPLL is $N/2$ times of the reference frequency.

3.2. Interpolated Flying Adder. The subsystem of an interpolated flying adder consists of path A, path B, and an interpolator as shown in Figure 4. Each path has one 32-to-1 MUX, accumulator, flip-flop, and an inverter. The bit width of accumulator in the bottom is 29 bits, and the bit width of upper accumulator is 5 bits. The core element of an accumulator is the adder design. For simplicity of the adder design and taking advantage of signal synchronization, we adopted a modified carry-select adder with linear increasing the length of each subadder. The first 2 bits are a simple ripple adder, and followed by a carry-save adder with the length of 2 bits, 3 bits, 4 bits, 5 bits, 6 bits, and 7 bits. Therefore, the total length of the adder is 29 bits as shown in Figure 10.

The MUX design is a key issue and the implementation of an interpolator. The 32-to-1 MUX can be constructed by the tree structure of 2-to-1 MUX [24]. However, this tree structure creates delay mismatch as indicated in Figure 11. T_S and T_D denote the delays for control inputs (CS0 ~ CS4) to the MUX and for the data, respectively. The data have different delay phases to the output depending on their control inputs. If the control input only CS0 changes, the delay is $T_S + 4T_D$. In contrast, the delay is T_S when only CS4 changes. The delay mismatch creates deterministic jitter.

To simplify the MUX design, we use a parallel transmission gate to reduce the switching delay as well as signal delay. To reduce the loading of the parallel-connected transmission, we combine 16 parallel transmission gates in one group. Then, we used a 2-to-1 MUX to select the signal. Figure 12 shows the implemented structure of a 32-to-1 MUX. Furthermore, we simplified the encoder circuit design, as shown in Figure 13. We divide the encoder circuit into 16 groups. Each group has one P-MOS load and four NMOSs.

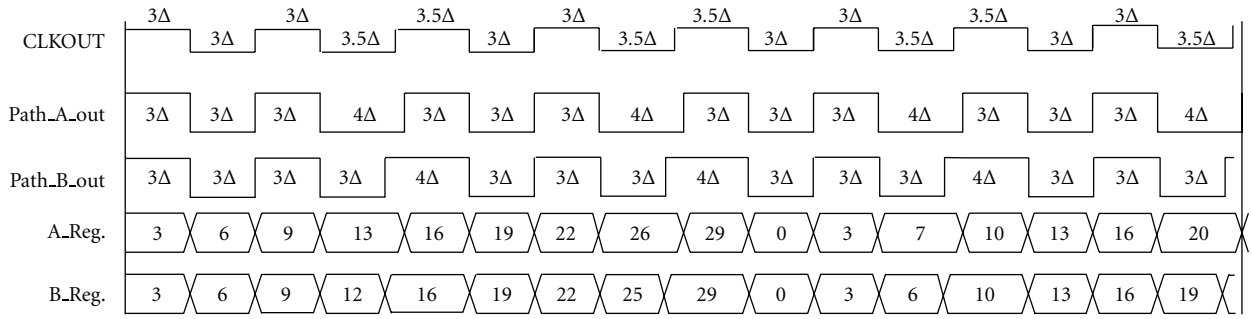


Figure 6: Numerical operation of proposed interpolated flying adder.

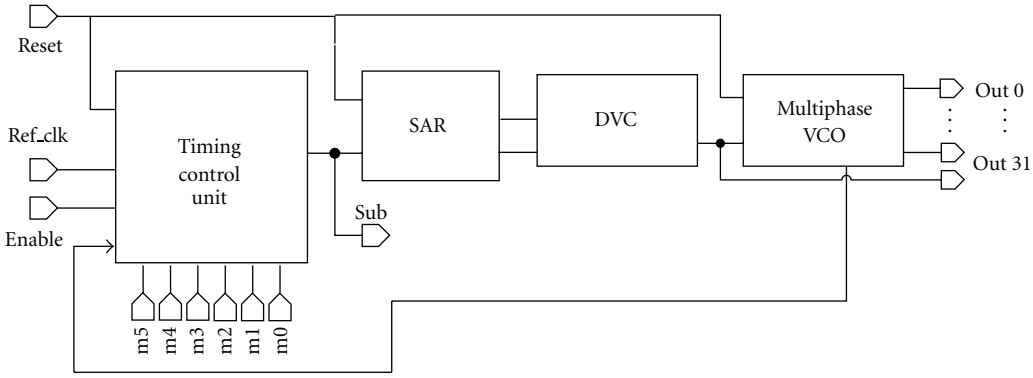


Figure 7: Block diagram of multiphase ADPLL.

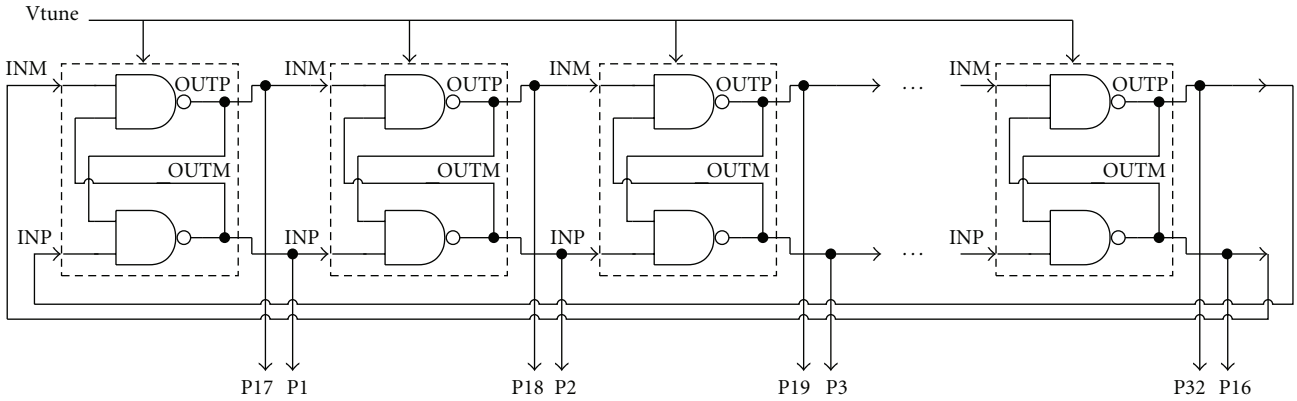


Figure 8: Block diagram of multiphase VCO.

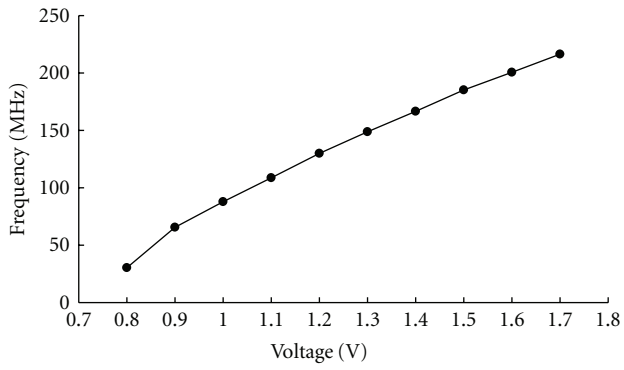


Figure 9: VCO output frequency.

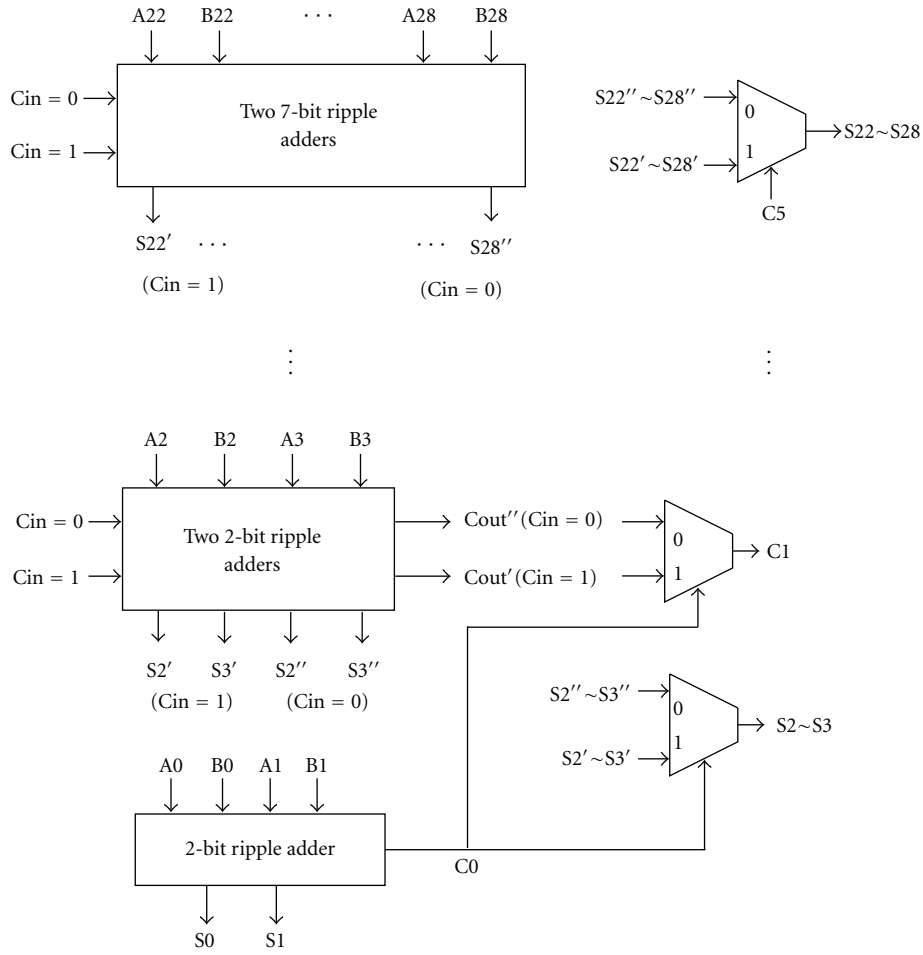


Figure 10: A modified 29-bit carry-select adder.

A digital interpolator in [18] has been used to adjust the delay that is smaller than the propagation delay of inverter in the delay-locked loop (DLL) as indicated in Figure 14(a). The digital interpolator can provide the function of interpolating two input rising edges as shown in Figure 14(b). The two input signals path A_out and path B_out are interpolated by the interpolator to achieve the out signal. Therefore, the interpolator performs the function in Figure 5.

4. Jitter Analysis of Interpolated Flying Adder

The basic architecture of interpolated flying adder is discussed in Section 2. A simple numerical example involving FREQ = 3.25 is illustrated in Figure 6. The interpolated flying adder can reduce the cycle-to-cycle jitter from 1Δ to Δ/2 as explained in Figures 2 and 6. However, more detailed analysis should be provided to compare the difference between the flying adder and interpolated flying adder. For the flying adder, jitter performance analysis has been intensively investigated in [8, 25, 26].

If a frequency $f = 1/T$ must be generated by the interpolated flying-adder-based frequency synthesizer and the multiphase signals are equally spaced with Δ resolution, then the frequency control word FREQ can be calculated as

$$T = \text{FREQ} \cdot \Delta, \tag{3}$$

$$\text{FREQ} = \frac{T}{\Delta} = I + r, \tag{4}$$

where the I is the integer part and r is the fractional part. Two types of cycles, short cycle T_S, and long cycle T_L are in the synthesized signal. The short cycle T_S and long cycle T_L are represented as

$$\begin{aligned} T_S &= I \cdot \Delta, \\ T_L &= \left(I + \frac{1}{2}\right) \cdot \Delta. \end{aligned} \tag{5}$$

The peak-to-peak jitter can be shown as

$$J_{P_k\text{-}P_k} = T_L - T_S = \frac{\Delta}{2}. \tag{6}$$

The possibility of a long cycle and a short cycle can be divided into two cases: $0 < r < 0.5$ and $0.5 \leqq r < 1$.

Table 1: Jitter performance comparison.

	Interpolated flying adder	Reference [8]'s flying adder
J_{mean}	0	0
J_{rms} (maximum)	$\Delta/4$	$\Delta/2$
$J_{P_k\text{-}P_k}$ (maximum)	$\Delta/2$	Δ

Table 2: Measured results versus jitter analysis.

Items	Measured results		
Synthesized output frequency	33 MHz	120 MHz	286 MHz
Power consumption (includes 28 I/O pads)	16.2 mW	21.3 mW	28.2 mW
Frequency, cycle time (Δ), and peak-to-peak jitter of multiphase ADPLL	42.7 MHz, Δ = 732 ps, 213 ps	171 MHz, Δ = 183 ps, 150.3 ps	171 MHz Δ = 183 ps, 150.3 ps
Peak-to-peak jitter	435.6 ps	248.7 ps	215.2 ps
Peak-to-peak jitter, jitter of multiphase ADPLL	222.6 ps < $\Delta/2$	98.4 ps ~ $\Delta/2$	64.9 ps < $\Delta/2$
RMS jitter	62.6 ps < $\Delta/4$	41.5 ps < $\Delta/4$	40.2 ps < $\Delta/4$

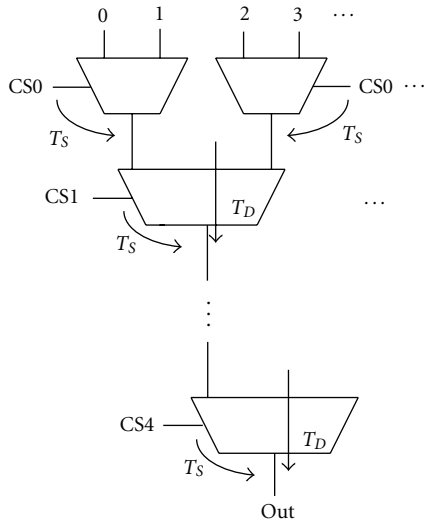


Figure 11: Delay mismatch in 32-to-1 MUX based on tree structure.

Case 1 ($0 < r < 0.5$). The possibility of a long cycle and a short cycle is

$$\begin{aligned} P_L &= 2r, \\ P_S &= 1 - P_L = 1 - 2r. \end{aligned} \tag{7}$$

Then, the synthesized average cycle time is calculated as

$$\begin{aligned} T_{\text{avg}} &= (P_L \cdot T_L) + (P_S \cdot T_S) \\ &= (2r) \cdot \left(I + \frac{1}{2}\right)\Delta + (1 - 2r) \cdot (I)\Delta \\ &= (I + r) \cdot \Delta \end{aligned} \tag{8}$$

and is the same as (4). The mean jitter can be derived as

$$\begin{aligned} J_{\text{mean}} &= P_L(T_L - T) + P_S(T_S - T) \\ &= 2r \cdot \left[\left(I + \frac{1}{2}\right) \cdot \Delta - (I + r) \cdot \Delta\right] \\ &\quad + (1 - 2r) \cdot [I \cdot \Delta - (I + r) \cdot \Delta] \\ &= 0. \end{aligned} \tag{9}$$

Finally, the root-mean-square (RMS) jitter can be shown as

$$\begin{aligned} J_{\text{rms}} &= \sqrt{P_L(T_L - T)^2 + P_S(T_S - T)^2} \\ &= \sqrt{2r \cdot \mathcal{A} + (1 - 2r) \cdot [I \cdot \Delta - (I + r) \cdot \Delta]^2} \\ &= \Delta\sqrt{\frac{r}{2} - r^2}. \end{aligned} \tag{10}$$

We donate that $[(I + (1/2)) \cdot \Delta - (I + r) \cdot \Delta]^2 = \mathcal{A}$.

When $r = 1/4$, the J_{rms} has maximum jitter value $\Delta/4$ and is one-half of the conventional flying adder of [8].

Case 2 ($0.5 \leqq r < 1$). The possibility of a long cycle and a short cycle is

$$\begin{aligned} P_L &= 2r - 1, \\ P_S &= 1 - P_L = 2 - 2r \end{aligned} \tag{11}$$

and is different from $0 < r < 0.5$. However, the $T_{\text{avg}} = (I + r)\,\Delta$ and $J_{\text{mean}} = 0$ are the same as in Case 1.

The rms jitter can also be derived as

$$\begin{aligned} &J_{\text{rms}} \\ &= \sqrt{P_L(T_L - T)^2 + P_S(T_S - T)^2} \\ &= \sqrt{(2r - 1) \cdot [(I + 1) \cdot \Delta - (I + r) \cdot \Delta]^2 + (2 - 2r) \cdot \mathcal{A}} \\ &= \Delta\sqrt{-r^2 + \frac{3}{2}r - \frac{1}{2}}. \end{aligned} \tag{12}$$

When $r = 3/4$, the J_{rms} has maximum jitter value $\Delta/4$ and is also one-half of the flying adder of [8].

Table 1 is the summary of the jitter performance comparison between the proposed interpolated flying adder and conventional flying adder [8]. Both of the proposed interpolated flying adder and the flying adder of [8] have

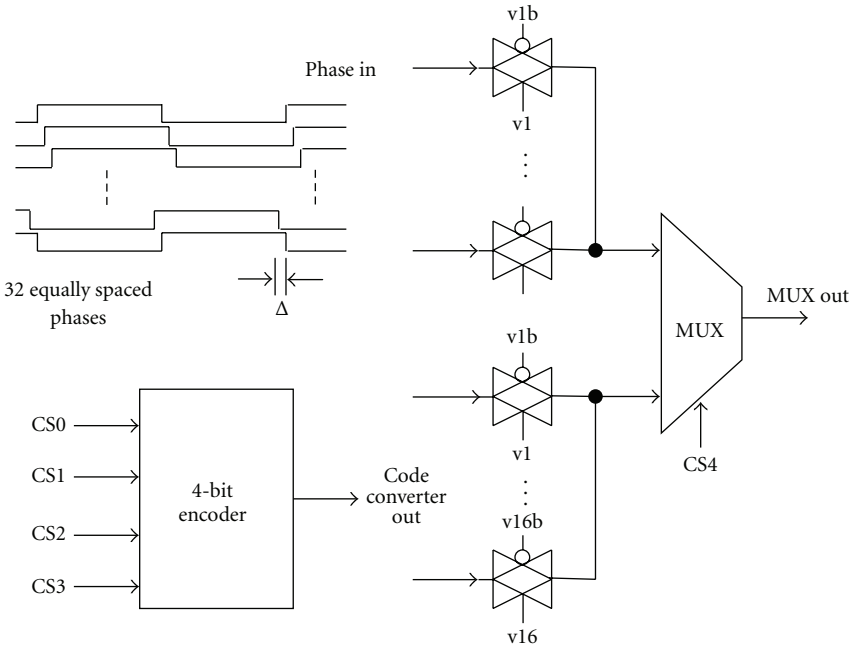


Figure 12: Transmission gate-based 32-to-1 MUX.

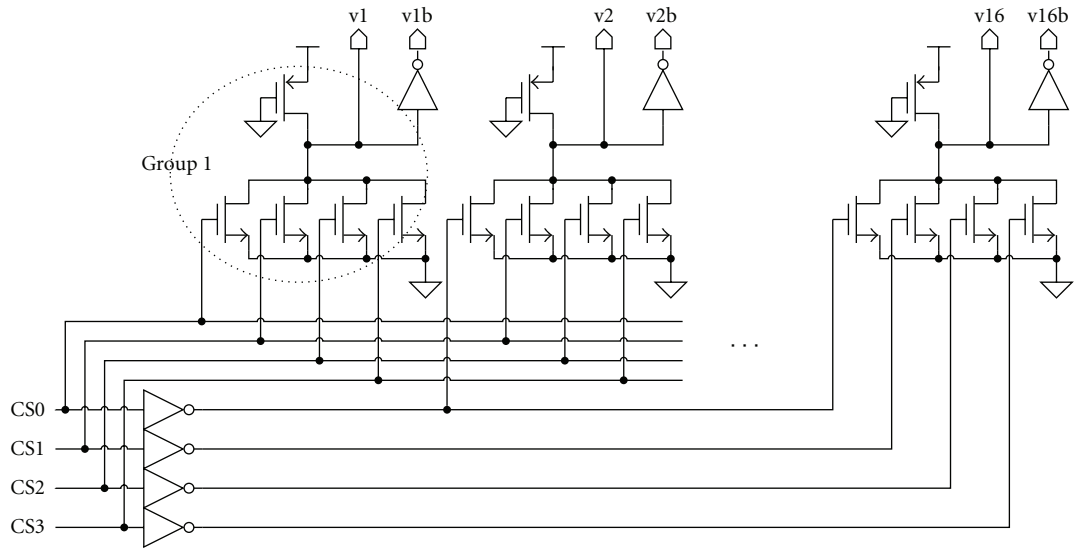


Figure 13: Simplified 4-bit encoder circuit.

the same average cycle time (T_{avg}) and mean jitter (J_{mean}). However, the RMS jitter and peak-to-peak jitter of the interpolated flying adder were improved to one-half of [8]'s flying adder.

5. Experimental Results

The test chip was fabricated in a standard 0.18 μm CMOS process. Figure 15 displays the microphotograph of the proposed interpolated flying adder. It includes two subsystems. The first subsystem is the multiphase ADPLL. The area of multiphase ADPLL is 283 μm $\times$ 132 μm. The reference clock of multiphase ADPLL input is 10 MHz. The output frequency of multiphase ADPLL is 171 MHz when the multiplication factor is 34 $(m_5, m_4, \ldots, m_0)$. Figure 16 shows the measured frequency of multiphase ADPLL at 171 MHz with 150.3 ps peak-to-peak jitter.

The other subsystem is the interpolated flying adder whose area is 295 μm $\times$ 353 μm as shown in Figure 15. When the frequency control word FREQ $< 28 : 0 >$ is 22.75 and the frequency of multiphase ADPLL is 171 MHz, the measured interpolated flying-adder output frequency is 120 MHz as indicated in Figure 17(a). The power consumption is 21.3 mW including 28 I/O pads. The peak-to-peak jitter is

Table 3: Performance comparison.

	Reference [8]'s FA frequency synthesizer	Reference [19]'s two-path FA synthesizer	Reference [20]'s two-path FA synthesizer	Reference [23]'s two-path ADPLL	This work
Process	0.6 μm, 3.3 V	0.18 μm, 1.8 V	0.18 μm, 3.3 V	0.13 μm, 1.3 V	0.18 μm, 1.8 V
Area (flying-adder)	1350 μm $\times$ 1260 μm	400 μm $\times$ 400 μm	690 μm $\times$ 630 μm	300 μm $\times$ 300 μm	295 μm $\times$ 353 μm
Output frequency range	57.27 MHz ~ 130 MHz	39.38 MHz ~ 226 MHz	62 KHz ~ 62.5 MHz	10 MHz ~ 500 MHz	33 MHz ~ 286 MHz
Accumulator width	32 bits	11 bits	11 bits		29 bits
Power consumption (multiphase PLL)	~40 mW	3.6 mW	32.4 μW		28.2 mW (I/O pads: ~14 mW, core:14.2 mW)
Power consumption (flying adder)	~150 mW				
Peak-to-peak jitter	1132 ps at 120.05 MHz	130 ps at 187.5 MHz	410 ps	288 ps at 191.4 MHz	215.2 ps at 286 MHz
RMS jitter	165 ps at 120.05 MHz	50 ps at 187.5 MHz		39 ps at 191.4 MHz	40.2 ps at 286 MHz
FoM(GHz/W)	0.63	17.8			18.2

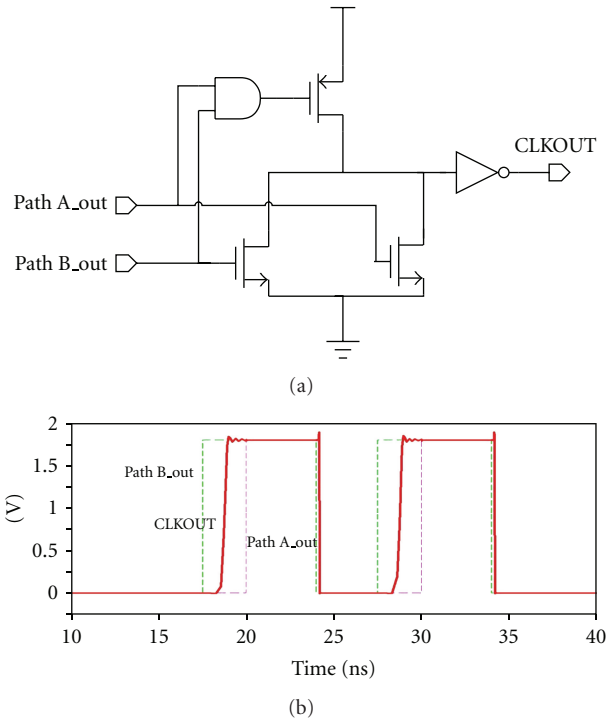


Figure 14: (a) Digital interpolator [18], (b) Hspice simulation.

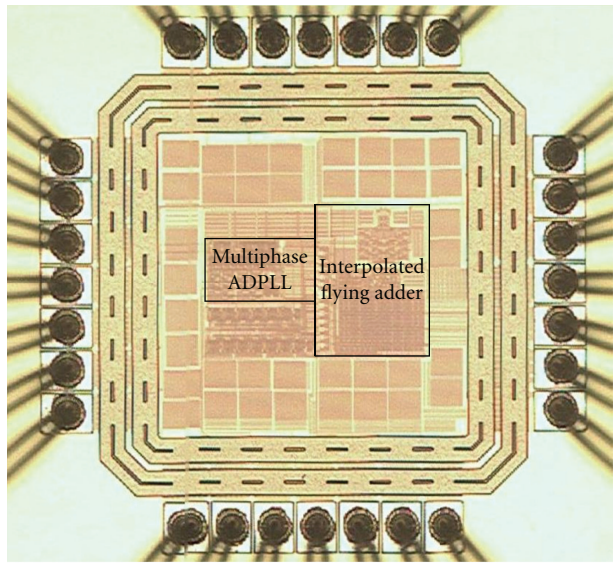


Figure 15: Microphotograph of the proposed all-digital frequency synthesizer using an interpolated flying adder.

249 ps, and the RMS jitter is 41.4 ps. Figure 17(b) is the measured result of interpolated flying adder at 286 MHz with 215.2 ps peak-to-peak jitter and 40.2 ps RMS jitter when the frequency control word FREQ < 28 : 0 > is 9.57. Only when the frequency of multiphase ADPLL is 42.7 MHz with 213 ps peak-to-peak jitter, the measured result of minimum output frequency is 33 MHz. Figure 18 shows that the output peak-to-peak jitter is 435 ps, and the RMS jitter is 60.5 ps.

The peak-to-peak jitter 150.3 ps of the multiphase signals will directly contribute jitter to output frequency as indicated in [17]. The output jitter caused by the interpolated flying adder is 98.4 ps and 64.9 ps at 120 MHz and 286 MHz, respectively. The RMS jitter is roughly equal to $\Delta/4 = 41$ ps in (12). When the output frequency is 33 MHz with multiphase ADPLL's cycle time (Δ) 732 ps, the jitter caused by the interpolated flying adder is 222.6 ps. Table 2 is the jitter measurements as compared with jitter analysis. Therefore, the measurement of peak-to-peak jitter and RMS jitter verifies the jitter analysis in Section 4.

To evaluate the performance of flying-adder-based frequency synthesizer, an easy measured and calculated figure

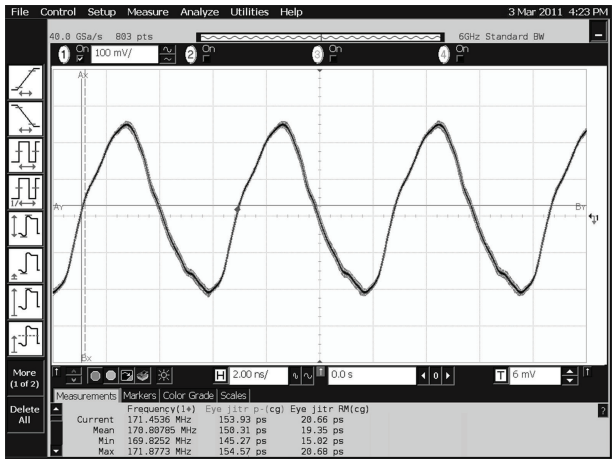
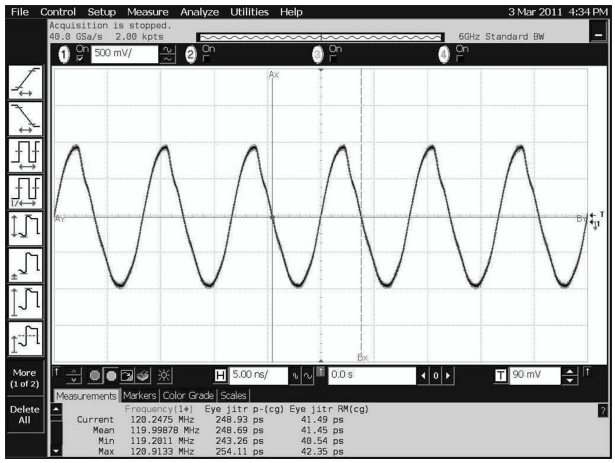
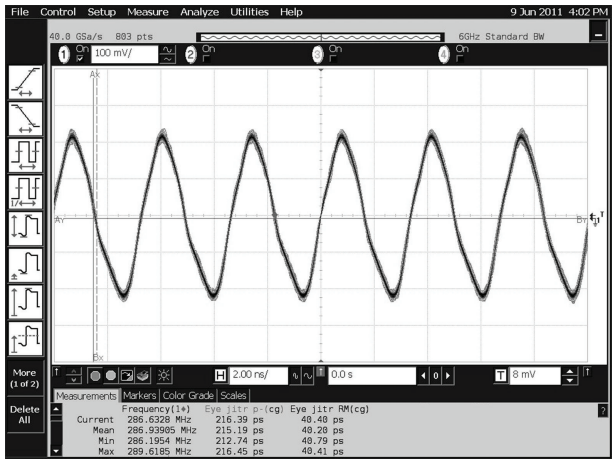


Figure 16: Measured results of multiphase ADPLL's output at 171 MHz with P_k-P_k = 150.3 ps.



(a)



(b)

Figure 17: Measured results of interpolated flying adder (multiphase ADPLL at 171 MHz), (a) Output at 120 MHz with $P_k - P_k$ = 248.7 ps, (b) Output at 286 MHz with P_k-P_k = 215.2 ps.

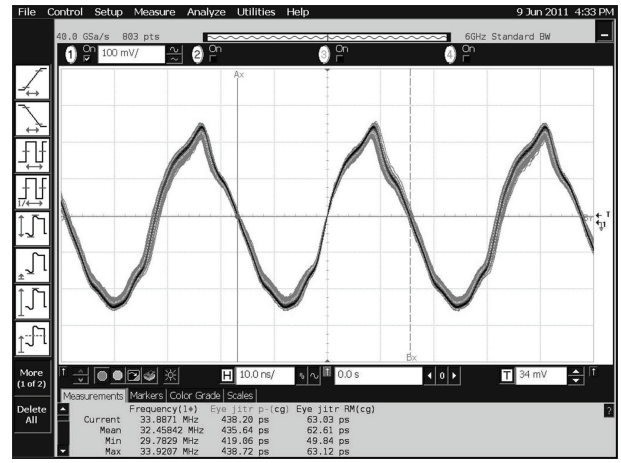


Figure 18: Measured results of interpolated flying-adder output at 33 MHz with P_k-P_k = 435.6 ps (multiphase ADPLL at 42.7 MHz).

of merit (FoM) must be defined from a combination of performance parameters. The power efficiency FoM including the effective number of bit (ENOB) is defined in [7]. The bit width of accumulator is important to frequency tuning resolution as indicated in [8]. Similarly, we define the power efficiency FoM related with normalized bit width of accumulator as follows:

$$\text{FoM} = \frac{[\text{Output frequency (GHz)} \cdot \text{ normalized bit width}]}{\text{power(W)}}. \tag{13}$$

The normalized bit width is the bit width of accumulator divided by bit width of [8]. According to Table 2's measurement results, the power consumption of the 28 I/O pads is approximately 14 mW, and the core power is 14.2 mW at 286 MHz. The FoM of proposed frequency synthesizer is 18.2 (GHz/W).

Table 3 is the performance comparison among flying-adder-based frequency synthesizers. The proposed frequency synthesizer achieved the smallest area and wide output frequency range as compared with [19, 20] in 0.18 μm process technology. By setting the worst case in the fractional part of frequency control word, we accomplish in peak-to-peak jitter and RMS jitter performance improvement as compared with [8]'s flying-adder structure. This works also has better jitter performance than [23]'s two-path ADPLL in 0.13 μm process.

6. Conclusion

This paper presents an all-digital frequency synthesizer using an interpolated flying adder with improved jitter performance. The 32 equally spaced phases are generated by a multiphase ADPLL which consists of a 32-phase NAND gate-based VCO, a DVC circuit, and an SAR controller. The interpolated flying adder is implemented with two low-cost pseudo-N MUXs, a modified 29-bit carry-select adder as well as a digital interpolator. The jitter performance

of the proposed interpolated flying-adder-based frequency synthesizer is also obtained. The output frequency of the prototype chips has the range of 33 MHz ~ 286 MHz at 1.8 V. Moreover, the peak-to-peak (P_k-P_k) jitter of the output clock at 286 MHz is 215.2 ps, and the RMS jitter is 40.2 ps.

Acknowledgments

The authors would like to thank L. Xiu, NovaTek, for useful discussions. They also would like to thank the Chip Implementation Center (CIC) of the National Science Council in Taiwan for chip fabrication. This work was supported by National Science Council of Taiwan, Taiwan, under Grant no. Nsc99-2221-E-327-047.

References

[1] W. F. Egan, *Frequency Synthesis by Phase Lock*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1999.

[2] P. Park, D. Park, and S. Cho, "A low-noise and low-power frequency synthesizer using offset phase-locked loop in 0.13 μm CMOS," *IEEE Microwave and Wireless Components Letters*, vol. 20, no. 1, Article ID 5339111, pp. 52–54, 2010.

[3] A. Yamagishi, M. Ugajin, and T. Tsukahara, "A 2.4-GHz PLL synthesizer for a 1-V Bluetooth RF transceiver," *IEICE Transactions on Electronics*, vol. E87-C, no. 6, pp. 895–900, 2004.

[4] K.-Y. Lee, H. Ku, and Y. B. Kim, "A fast switching low phase noise CMOS frequency synthesizer with a new coarse tuning method for PHS applications," *IEICE Transactions on Electronics*, vol. E89-C, no. 3, pp. 420–428, 2006.

[5] T. Watanabe and S. Yamauchi, "An all-digital PLL for frequency multiplication by 4 to 1022 with seven-cycle lock time," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 2, pp. 198–204, 2003.

[6] S. Dosho, T. Morie, K. Okamoto, Y. Yamada, and K. Sogawa, "A -90 dBc@10 kHz phase noise fractional-N frequency synthesizer with accurate loop bandwidth control circuit," *IEICE Transactions on Electronics*, vol. 89, no. 6, pp. 739–744, 2006.

[7] X. Geng, F. F. Dai, J. D. Irwin, and R. C. Jaeger, "An 11-bit 8.6 GHz direct digital synthesizer MMIC with 10-bit segmented sine-weighted DAC," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, Article ID 5405153, pp. 300–313, 2010.

[8] H. Mair and L. Xiu, "Architecture of high-performance frequency and phase synthesis," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 6, pp. 835–846, 2000.

[9] L. Xiu and Z. You, "A flying-adder architecture of frequency and phase synthesis with scalability," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 5, pp. 637–649, 2002.

[10] L. Xiu and Z. You, "A new frequency synthesis method based on flying-adder architecture," *IEEE Transactions on Circuits and Systems II*, vol. 50, no. 3, pp. 130–134, 2003.

[11] L. Xiu, "A Flying-Adder On-chip frequency generator for complex SoC environment," *IEEE Transactions on Circuits and Systems II*, vol. 54, no. 12, pp. 1067–1071, 2007.

[12] L. Xiu, "A novel DCXO module for clock synchronization in MPEG2 transport system," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 8, pp. 2226–2237, 2008.

[13] L. Xiu, "A Flying-Adder PLL technique enabling novel approaches for video/graphic applications," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 591–599, 2008.

[14] L. Xiu, "The concept of time-average-frequency and mathematical analysis of flying-adder frequency synthesis architecture," *IEEE Circuits and Systems Magazine*, vol. 8, no. 3, Article ID 4609962, pp. 27–51, 2008.

[15] L. Xiu, "Some open issues associated with the new type of component: digital-to-frequency converter," *IEEE Circuits and Systems Magazine*, vol. 8, no. 3, Article ID 4609966, pp. 90–94, 2008.

[16] L. Xiu, "A fast and power-area-efficient accumulator for flying-adder frequency synthesizer," *IEEE Transactions on Circuits and Systems I*, vol. 56, no. 11, Article ID 4785490, pp. 2439–2448, 2009.

[17] D. E. Calbaza and Y. Savaria, "A direct digital period synthesis circuit," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 8, pp. 1039–1045, 2002.

[18] T. Saeki, M. Mitsuishi, H. Iwaki, and M. Tagishi, "1.3-cycle lock time, non-PLL/DLL clock multiplier based on direct clock cycle interpolation for 'clock on demand'," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1581–1590, 2000.

[19] G. N. Sung, S. C. Liao, J. M. Huang, Y. C. Lu, and C. C. Wang, "All-digital frequency synthesizer using a flying adder," *IEEE Transactions on Circuits and Systems II*, vol. 57, no. 8, Article ID 5545381, pp. 597–601, 2010.

[20] Y. A. Chau, Y.-Y. Yang, and J.-F. Chen, "All-Digital frequency synthesizer with dual resolution," in *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems*, pp. 630–633, Tottori, Japan, 2006.

[21] A. Rossi and G. Fucili, "Nonredundant successive approximation register for A/D converters," *Electronics Letters*, vol. 32, no. 12, pp. 1055–1057, 1996.

[22] P.-L. Chen, C.-F. Liu, and T.-H. Lin, "Amultiphase digital controlled oscillator with DVC techniqu," in *Proceedings of the 15th Workshop on Synthesis and System Integration of Mixed Information Technologies*, pp. 473–476, Okinawa, Japan, March 2009.

[23] W. Liu, W. Li, P. Ren, C. Lin, S. Zhang, and Y. Wang, "A PVT tolerant 10 to 500 MHz all-digital phase-locked loop with coupled TDC and DCO," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, Article ID 5405141, pp. 314–321, 2010.

[24] H. Lu, C. Su, and C. N. J. Liu, "A tree-topology multiplexer for multiphase clock system," *IEEE Transactions on Circuits and Systems I*, vol. 56, no. 1, pp. 124–131, 2009.

[25] P. P. Sotiriadis, "Theory of flying-adder frequency synthesizers—Part I: modeling, signals' periods and output average frequency," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 8, Article ID 5424107, pp. 1935–1948, 2010.

[26] P. P. Sotiriadis, " Theory of flying-adder frequency synthesizers—Part II: time- and frequency-domain properties of the output signal," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 8, pp. 1949–1963, 2010.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 507381, 12 pages
doi:10.1155/2011/507381

Review Article

Open-Loop Wide-Bandwidth Phase Modulation Techniques

Nitin Nidhi,[1] Pin-En Su,[2] and Sudhakar Pamarti[1]

[1] *Electrical Engineering Department, University of California, Los Angeles, CA 90095-1594, USA*
[2] *Broadcom Corporation, Irvine, CA 92617-3038, USA*

Correspondence should be addressed to Nitin Nidhi, nitin@ee.ucla.edu

Received 31 May 2011; Accepted 16 August 2011

Academic Editor: Kenichi Okada

The ever-increasing growth in the bandwidth of wireless communication channels requires the transmitter to be wide-bandwidth and power-efficient. Polar and outphasing transmitter topologies are two promising candidates for such applications, in future. Both these architectures require a wide-bandwidth phase modulator. Open-loop phase modulation presents a viable solution for achieving wide-bandwidth operation. An overview of prior art and recent approaches for phase modulation is presented in this paper. Phase quantization noise cancellation was recently introduced to lower the out-of-band noise in a digital phase modulator. A detailed analysis on the impact of timing and quantization of the cancellation signal is presented. Noise generated by the transmitter in the receive band frequency poses another challenge for wide-bandwidth transmitter design. Addition of a noise transfer function notch, in a digital phase modulator, to reduce the noise in the receive band during phase modulation is described in this paper.

1. Introduction

The rapid growth of new communication standards like LTE and WiMAX has led to high data rate, wide signal bandwidth, and high peak-to-average-power ratio. Additionally, transmission of 1-2 GHz bandwidth signals in the unlicensed frequency band at 60 GHz is also gaining momentum. Since total power consumption is largely determined by the efficiency of the power amplifier used, high-efficiency architectures, like polar [1–5] and out-phasing [6, 7], are preferable for future designs. Both of these architectures require a phase modulator as one of their key building blocks. The bandwidth of these modulators increases with the bandwidth of the transmitted signal. The concept of a software-defined radio, which can support multiple programmable carrier frequencies and provides maximum flexibility in data rates, is being looked at as a desirable and viable enhancement for future radios. Such a transmitter may also require a wide-bandwidth phase modulator.

Besides being wideband, a digital implementation of the phase modulator should be favored as it comes with several advantages—it (1) enables implementation of calibration algorithms to correct for transmitter nonlinearity, PVT variations, and, in a polar architecture, AM/PM path delay mismatch, (2) allows shaping of quantization noise transfer function to reduce in-band noise, (3) allows dynamic element matching to make linearity insensitive to component mismatches, (4) permits reconfigurability to meet the requirements for more than one communication standard, and (5) eases porting of design from one process to the next and hence readily benefits from technology scaling. On the other hand, the digital implementation poses new challenges in terms of quantization noise and spectral images. Furthermore, PVT variation, nonlinearity and power of the front-end digital-to-phase block, although minimized by digital techniques, still require optimization.

Traditionally, phase modulators have been implemented using a phase-locked loop (PLL) for narrow-band modulation [2–5, 8–11]. A second input port, inside a PLL, was employed to enable wideband modulation capability to such modulators [7, 12–17]. But these techniques have not been successful for applications beyond GSM/EDGE [2–5, 13, 17] and WCDMA [7, 12]. LTE and WiMAX both require wider-bandwidth modulators. Recently, open-loop phase switching technique, that dynamically selects a signal from a bank of signals at the carrier frequency but with different phase offsets, was proposed for digital wide-bandwidth phase modulation. Quadrature signals at the output of a frequency

divider [18], output signals of a ring oscillator [19], and phase interpolation [20] were used to form the bank of reference signals. An overview of both open-loop and closed-loop techniques for phase modulation is presented in this paper, contrasting their performance limitations. Finite resolution in the digital phase modulator results in phase quantization noise (PQN). A PQN cancellation technique was proposed in [20] to reduce the out-of-band emission from the phase modulator. The signal processing details of this technique along with methods to further improve its effectiveness are described in this paper.

The quantization noise added by the modulator must not violate the transmit spectrum mask and the receive band noise requirement of a frequency division duplexing (FDD) system. FDD is commonly employed in cellular systems like GSM/EDGE, WCDMA, and LTE. Recent advances towards a software-defined radio (SDR) promotes coexistence of multiple radios on an integrated circuit. This trend will further increase the requirements on emission in the receive band frequencies. The noise component due to PLL phase noise can be reduced to meet the receive band noise requirements for WCDMA without requiring additional filtering. However, a high Q band pass filter, like a SAW filter, will be required to filter out the quantization noise component. Such a filter is costly and must be avoided to achieve a fully integrated transmitter. A more elegant solution will be to reduce the noise at the receive band frequency by design. This will not only save the cost of an additional costly board component but also provide the flexibility in changing the position and order of the notch. A detailed description of this approach is presented in this paper.

The paper is organized as follows. Section 2 presents an overview of prior art on phase modulation techniques and describes the concept of digital phase modulation. Section 3 describes the concept and analysis of a phase quantization noise cancellation technique, and Section 4 describes a solution to meet the receive band noise specification. A conclusion is given in Section 5.

2. Wide-Bandwidth Digital Phase Modulation

In one of the simplest implementations of phase or frequency modulation, the modulation data is applied to the control voltage of a voltage-controlled oscillator (VCO). Although, this technique is open-loop and wideband, it suffers from frequency drift, VCO transfer function nonlinearity and variations over PVT, high close-in phase noise, and loss of transmission during periods of frequency lock. In order to find a solution for these problems, broadly two categories of phase modulators have emerged—closed-loop and open-loop phase modulation.

2.1. Closed-Loop Phase Modulation. If continuous feedback control is applied to a VCO, to arrive at a conventional PLL, close-in phase noise is improved and carrier frequency is tightly controlled. In this case, modulation data has been successfully applied using a multimodulus feedback divider (MMD) in a fractional-N PLL [8, 9]. In this technique,

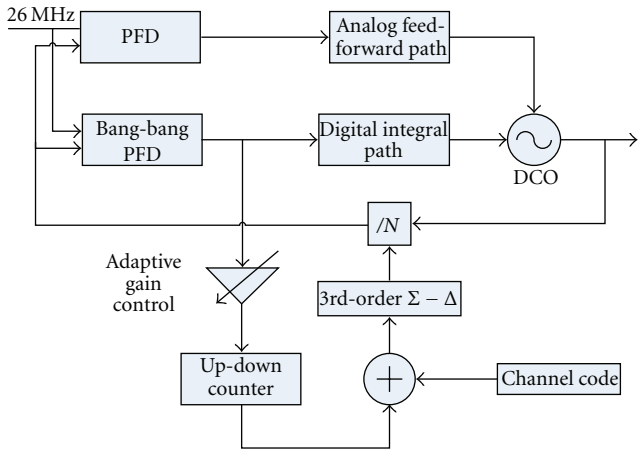


Figure 1: Block diagram of DCO gain calibration as used in [21].

however, the high-frequency content of the modulation data is filtered out by the loop filter of the PLL making it unsuitable for wide-bandwidth phase modulation. Bandwidth extension methods like phase noise cancellation [10, 22–26], multiphase fractional-N PLL [27], type I fractional-N PLL with sharp loop filter [28], and digital pre-emphasis [11] have been applied, but even the best in state-of-the-art designs have not exceeded 3 MHz. Additionally, this technique can create low-frequency fractional spurs at PLL output.

In order to obtain further increase in bandwidth, the so-called two-point modulation has been often used [12–17]. Since the injection of modulation data at any node of the PLL loop is either high-pass filtered or low-pass filtered, it is injected at two nodes simultaneously such that the sum of the two transfer functions becomes wideband. The most commonly used injection nodes are the MMD and the VCO control voltage to achieve wide-bandwidth FM. A common problem encountered in this approach is the loss in SNR due to gain and phase mismatch between the two paths. Since K_{VCO} must be known for gain matching, an on-chip K_{VCO} estimation method becomes important. Nonetheless, this technique has been successfully used to generate transmit signals meeting GSM/EDGE and WCDMA requirements.

In [29], gain matching was obtained by applying a square wave input to the two modulation input nodes. The calibration loop then minimizes the error voltage developed across the loop filter zero setting resistor. The approach taken in [21] to measure digitally controlled oscillator (DCO) gain and PLL loop gain was to measure the phase error in response to a DCO control change (Figure 1). The PLL loop, reduced to type I operation by holding the digital integral path, compensates for the frequency error by adjusting the phase error at phase detector input. The calibration loop then senses the phase error using a bang-bang PFD and modulates the fractional part of the feedback divider until the phase error is reduced to a very small value. Using the new frequency divider value so obtained, the DCO gain was calculated.

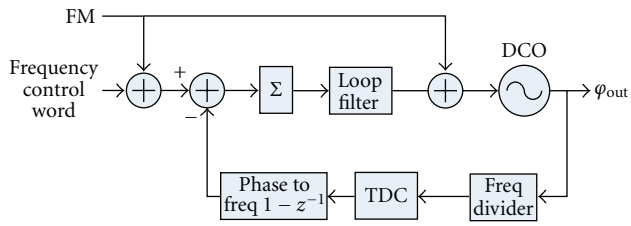


Figure 2: Two-point FM in an all-digital PLL [13].

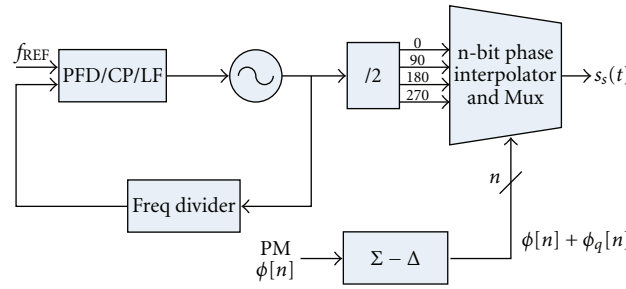


Figure 3: Open-loop phase modulator [18].

The development of an all-digital PLL (ADPLL) [13, 30–33] has opened up new possibilities for accomplishing phase modulation, as the signals within the PLL loop have become more predictable. Besides this, injection of digital data can be readily achieved in the digital domain, at most of the internal nodes of the PLL. The design presented in [13] took advantage of this feature of a digital PLL and injected the frequency modulation data to the control word of DCO and the carrier frequency control word of the PLL (Figure 2).

The design presented in [12] introduced VCO transfer function linearization technique for two-point modulation for WCDMA. It utilized a local negative feedback loop around the VCO to obtain a fairly constant K_{VCO}. This local loop employed an analog technique to measure the VCO frequency, which forms the feedback signal. Although gain and phase calibration techniques have helped to increase the robustness and bandwidth of phase modulators, to the best of the authors' knowledge, their application have not been demonstrated on even wider modulation standards such as WLAN, WiMAX, and LTE.

2.2. Open-Loop Phase Modulation. The bandwidth limitation and gain and phase mismatch issue associated with two-point modulation techniques can be avoided by using open-loop modulation techniques, where modulation is performed outside the frequency synthesizer loop. Essentially, it isolates the carrier frequency generation block from the data modulation block, yielding a modulator which does not involve a low-pass filter (the loop filter) in its path. Hence, these modulators can achieve very wide bandwidth.

In a typical open-loop phase modulator, a phase generator block produces multiple phases at the carrier frequency. It is followed by a phase multiplexer whose output is controlled by the phase modulation data. For a given sequence of desired digital phase values $\varphi[n]$, a $\Sigma-\Delta$ modulator quantizes each phase sample $\varphi[n]$ to one of the M available phases, $2\pi \times k/M$, $k = 0, 1, \ldots, M-1$. The output of the $\Sigma - \Delta$ modulator controls a digital-to-phase converter, whose inputs are M phase signals, $s_k = \cos(\omega_c + 2\pi \times k/M)$. Note that ω_c is the carrier frequency in rad/s. The digital-to-phase converter can be a phase multiplexer with M phase inputs or a digital phase interpolator. The resultant synthesized signal can be written as

$$s_s(t) = \cos\left(\omega_c t + \sum_{n=1}^{\infty}\left(\phi[n] + \phi_q[n]\right) \cdot p(t - nT_s)\right), \quad (1)$$

where pulse-shaping function, $p(t)$, is nominally a rectangular pulse of unit amplitude with duration T_s and $\phi_q[n]$ is the error in quantizing $\phi[n]$. The synthesized signal $s_s(t)$ well approximates the desired phase modulated carrier signal for large M and a high switching frequency, $F_s = 1/T_s$. The ability to dynamically switch between multiple phases can be easily extended to synthesize a frequency at a small offset from the VCO frequency by applying a ramp signal as input to the $\Sigma - \Delta$ modulator, with the appropriate slope.

This approach for phase modulation suffers from three critical issues—phase quantization noise (PQN), nonlinearity of digital-to-phase converter, and spectral images. Phase quantization noise arises due to the quantization process involved in the generation of output phases. Simple truncation results in white quantization noise and a flat power spectral density (PSD) stretching over a band $[-F_s/2, F_s/2]$ around the carrier frequency, resulting in poor in-band signal-to-noise ratio (SNR) and error vector magnitude (EVM). Using a $\Sigma - \Delta$ modulator to quantize $\phi[n]$ imparts a high-pass shape [32] to $\phi_q[n]$, thereby suppressing the close-in PQN and improving EVM. However, the PQN at offsets of $F_s/2$ from the carrier frequency are amplified resulting in elevated out-of-band noise that can violate the spectral mask requirements for the transmitter. The PQN can be reduced by increasing the phase switching speed and/or by increasing the resolution of the digital phase interpolator. The discrete-time nature of the phase modulator causes spectral images to appear at integer multiples of F_s. The zeroth-order hold operation results in only modest filtering ($\sin c^2(\cdot)$) of these images, and hence a high oversampling factor or interpolation is required to reduce them. At most, 5–10 dB of further suppression results from the LC tanks typically employed in a tuned power amplifier.

Due to a digital implementation of the modulator, this technique easily lends itself for quantization noise shaping, digital predistortion to compensate for errors due to PVT variation, dynamic element matching, and other similar digital techniques.

The design presented in [18] used four quadrature phases at the output of a frequency divider and digitally switched between them to realize phase modulation (Figure 3). Since only four-level quantization was used, the noise floor was quite high. With $\Sigma - \Delta$ quantization noise shaping, the in-band noise was lowered at the expense of large out-of-band noise (-25 dBr observed from the measured spectrum). While transmitting at 403.2 MHz, the measured FSK errors

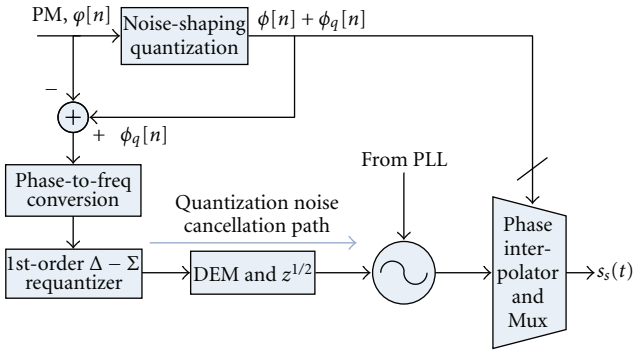


Figure 4: Block diagram of phase quantization noise cancellation technique [20].

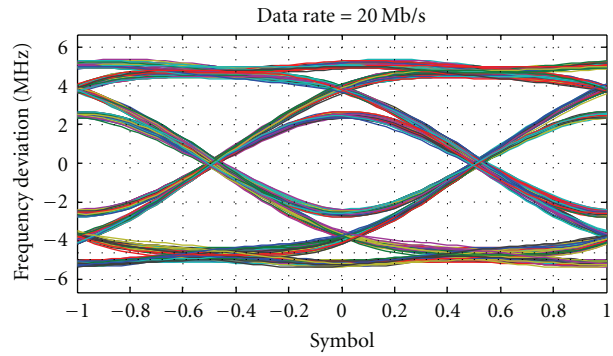


Figure 5: Eye diagram of the transmitted signal.

at 6 Mb/s data rate was 4.1% and 11.6% for 2-FSK and GFSK modulations, respectively.

In [7], a hybrid of many of the techniques mentioned earlier was designed for GSM and WCDMA. The outphasing angle was generated using an 8-bit phase interpolator, while phase modulation was generated by two-point modulation. However, it also used phase-to-digital converters in a negative feedback loop to correct for the nonlinearity of phase interpolators, thereby reducing the available bandwidth. Phase modulation techniques at 60 GHz are also being researched. Reference [34] implemented a novel method to obtain phase modulation at 60 GHz by digitally controlling the effective dielectric constant of a differential transmission line. This was achieved by digitally switching in and out a 4-bit bank of floating M6 and M7 strips placed underneath the transmission line, which leads to a digitally variable phase of S21. However, its dynamic performance under data transmission was not presented.

3. Quantization Noise Cancellation

PQN can be contrasted with the quantization noise added by baseband DACs in an I–Q architecture. In the latter, out-of-band quantization noise and spectral images of baseband DACs can be removed by baseband low-pass filters, although the total power consumption also gets increased. Furthermore, the noise generated by the low-pass filters adds on top of the contributions from the mixer and amplifiers in the transmit chain. Such a filter in a digital phase modulator will have to be RF band pass, requiring high Q passive components or a SAW filter. A PQN cancellation technique can, therefore, significantly improve the performance of the phase modulator.

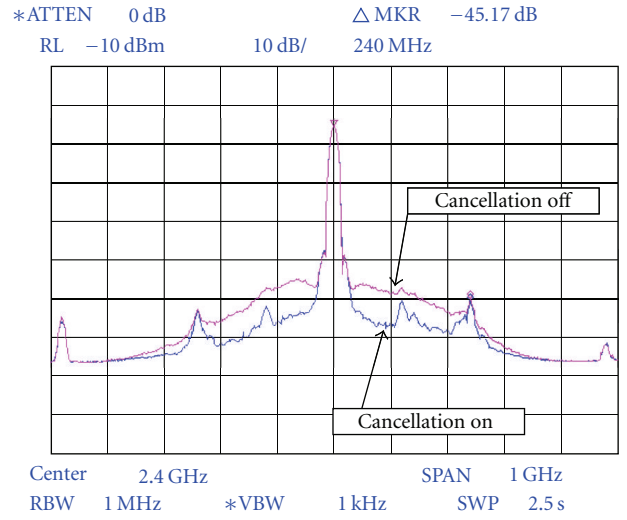
Through a second VCO control port, a quantization noise cancellation signal was added to the modulator output in [20], as shown in Figure 4. The required cancellation signal is obtained by first subtracting the input signal, $\phi[n]$, from the quantized signal, $\phi[n] + \phi_q[n]$. Thus, the quantized phase data $\phi[n] + \phi_q[n]$ is applied through the digital-to-phase converter, while the PQN cancellation signal of $-\phi_q[n]$ is applied through a cancellation path through the VCO.

In the cancellation path, frequency of the VCO can be controlled in a straightforward manner through an analog control signal, or through a digital word in the case of a digitally controlled oscillator. Its phase, on the other hand, is the outcome of an integration of the resultant frequency. Since the cancellation signal is a phase quantity, while the VCO input port controls its frequency, the required cancellation phase is differentiated, $(1 - z^{-1})$ to obtain an equivalent VCO frequency deviation. It must also be attenuated by VCO control voltage-to-frequency gain, K_{VCO}.

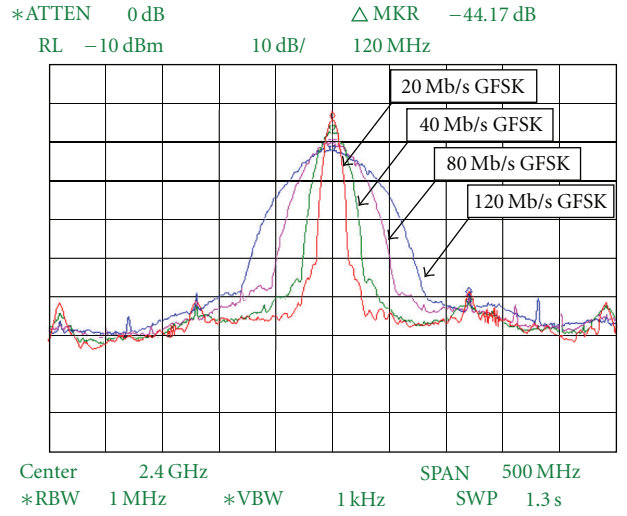
In [20], a state-of-the-art implementation of PQN cancellation was presented on a 2.4 GHz wide-bandwidth open-loop GFSK transmitter IC. The phase cancellation path was implemented by adding a second VCO port to control a 4-bit capacitor bank, with DEM logic incorporated in the selection process. Figure 5 shows the GFSK transmitted eye diagram at 20 Mb/s GFSK modulation. Figure 6(a) shows 9 dB improvement in the measured output spectrum, after enabling the phase noise cancellation technique. The corresponding GFSK transmitted modulation rms error is 3.2% rms at 20 Mb/s. The wide-bandwidth capability of the transmitter is demonstrated in Figure 6(b), which shows overlaid spectrum of transmitter output under 20, 40, 80, and 120 Mb/s rates.

It should be noted that the normal functioning of the PLL is not impacted by the cancellation path. The cancellation signal at the VCO input port goes through the same transfer function as seen by the VCO phase noise, resulting in its high-pass filtering. The crucial difference being that, as opposed to VCO phase noise, the high-frequency content of the cancellation signal is a desired signal which lowers the quantization noise added from phase interpolator input. It may be perceived that since the low-frequency content of the cancellation signal is filtered out, it will degrade the efficacy of the technique. But, since quantization noise is already low at these frequencies, this results in negligible loss in effectiveness.

One of the subtle features of the cancellation path is the implication of discrete-time differentiation applied to obtain the required VCO frequency deviation. The integration of frequency inherent in a VCO is, on the other hand, continuous. Consider the case of rectangular pulse shape for the input phase and cancellation signal, where both of them are applied at a frequency, $F_s = 1/T_s$. Since the cancellation signal is updated every T_s time period, while VCO frequency is continuously being integrated, perfect cancellation is



(a)



(b)

Figure 6: Transmitter output spectrum (a) with noise cancellation ON and OFF; (b) under high speed data transmission [20].

obtained only at the end of each time period, which is not very effective. Figure 8(b) depicts the timing diagram of phase quantization noise, the applied cancellation signal, and the uncancelled PQN. In [20], the cancellation signal was advanced by $T_s/2$ to improve the effectiveness of noise cancellation. In the following subsections, results from detailed system simulation are used to illustrate this technique and bring up the associated trade-off, along with mathematical expressions for residual quantization noise. It will also be shown how the effectiveness of cancellation changes with a reduction in integration time and after quantization of the cancellation signal. Since the techniques are applicable to all forms of PQN shaping algorithm and for any resolution in the phase data path, a 2nd-order $\Sigma - \Delta$ noise shaping with a 5-bit quantizer in the forward path is used to illustrate these

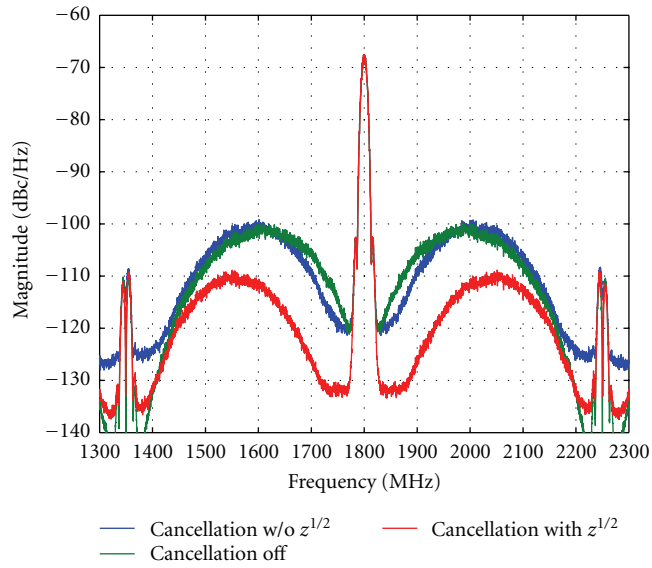


Figure 7: The simulated PSD of a GFSK-modulated signal showing PQN cancellation obtained with and without $T_s/2$ advancement.

techniques. For all system simulations, input data is 20 Mb/s GFSK modulated.

3.1. Advancement of Cancellation Signal. The control voltage-to-frequency response of VCO to a cancellation signal was found to be low pass with a very high cutoff frequency ($>F_s$). Hence, the cancellation frequency can be modeled as a rectangular pulse shape, and the cancellation phase becomes its integrated form. The transfer function of the uncancelled PQN can be expressed mathematically as

$$S_{\phi_n,\text{un-cancel}} = \text{NTF}_1(f) \times P_1(f) \times K_1(f), \tag{2}$$

where $\text{NTF}_1(f)$ is the NTF of $\Sigma - \Delta$ modulator quantizing $\phi[n]$, $P_1(f)$ models the pulse-shaping function of uncancelled $\phi_q[n]$, and $K_1(f)$ models the effect of PQN cancellation. When PQN cancellation is off, $P_1(f)$ is a sinc function, $\sin(\pi f Ts)/(\pi f Ts)$, representing the zeroth-order hold, and $K_1(f)$ is 1. However, when a cancellation signal is applied through the VCO control port, two changes take place: (1) pulse shape of uncancelled quantization noise becomes saw-tooth and (2) magnitude of quantization noise becomes a first-order difference, $(1 - z^{-1})$ of the initial quantization noise. These changes can be easily observed in the timing plot of Figure 8(b). As a result, $P_1(f)$, for a saw-tooth pulse shape, attains a DC value of -6 dB and $K_1(f)$ becomes $4 \times \sin^2(\pi f Ts)$. Saw-tooth pulse shape for PQN and the additional first-order noise shaping results in lower quantization noise at low frequencies, but higher noise at high frequencies (Figure 7). Hence noise cancellation is limited to frequencies below 40 MHz offset; while it increased by 2 dB at higher offset frequencies due to the additional first-order noise shaping.

In order to improve the PQN cancellation mechanism, the cancellation signal can be advanced by $T_s/2$. As a result, the residual quantization noise rises to only half of the value

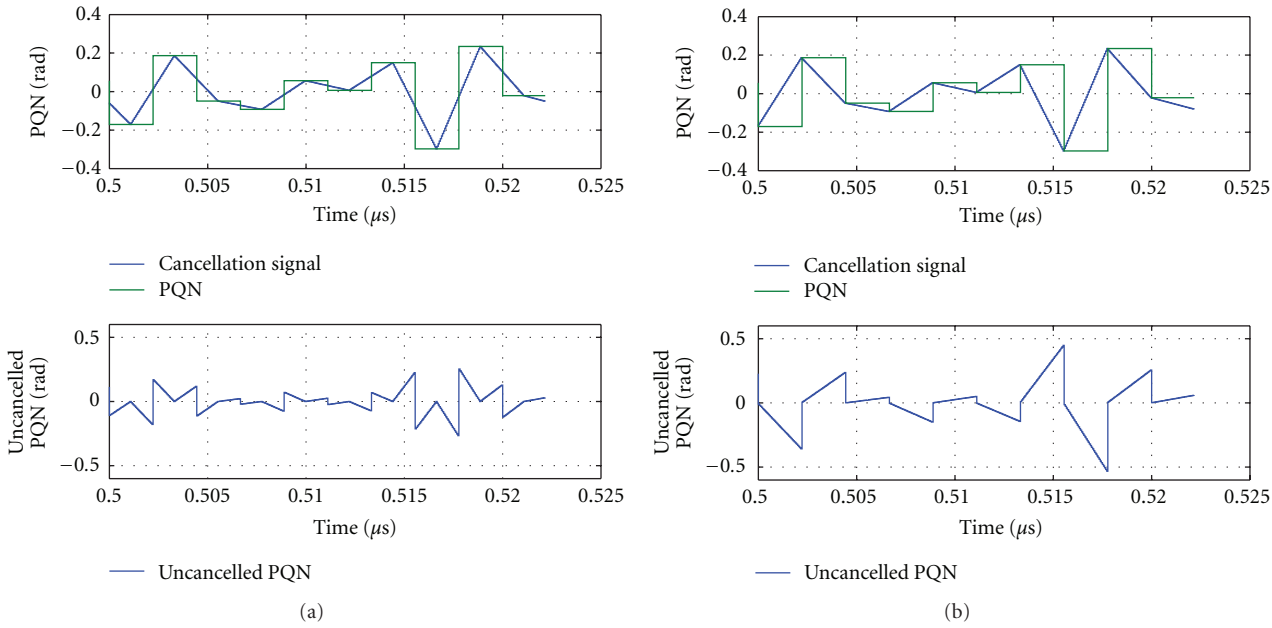


FIGURE 8: Timing diagram for PQN cancellation technique for (a) $T_s/2$ advancement in cancellation signal and (b) no advancement in cancellation signal.

attained in the earlier case and afterwards its sign gets flipped (Figure 8(a)). Hence, $P_1(f)$ has a zero at DC and $K_1(f)$ is reduced by 6 dB. The combined effect of these two changes is a reduction in peak quantization noise by 10 dB, along with a maximum improvement of 17 dB, at a lower frequency (Figure 7). $P_1(f)$ and the expression of uncancelled PQN, $S_{\phi_n,\text{un-cancel}}(f)$, for the three cases are plotted in the appendix. It should be noted that half-period advancement results in the highest achievable noise cancellation, compared to other values for signal advancement.

3.2. Reduction in Integration Time. Further improvement in PQN cancellation can be obtained by using a return-to-zero (RZ) DAC or an equivalent DCO to control the deviations in VCO frequency. In this case, integration of frequency input to the VCO is performed for a shorter duration of time, and hence perfect phase cancellation is obtained for a longer duration as opposed to one time instant. The required frequency deviation must also be increased in proportion to the reduction in integration time, such that the phase accumulated in one time period equals the required cancellation phase value. Figures 9(a) and 9(b) depict the time waveforms for the cases when integration time is reduced to $T_s/2$ and $T_s/4$, respectively.

The resultant improvement in PQN cancellation is 6 dB for an integration time of $T_s/2$ as shown in Figure 10. If the integration time is reduced further while simultaneously increasing the cancellation frequency signal, output PQN reduces by 6 dB for each octave reduction in integration time. In the limit of an impulse in frequency cancellation signal, perfect cancellation is obtained at all times. Since shorter integration time requires a larger frequency deviation from the VCO, there exists a trade-off between noise cancellation and VCO frequency deviation. The minimum integration time and the resultant noise cancellation are limited by the maximum frequency deviation that can be linearly obtained from the VCO. For instance, for a maximum achievable frequency deviation of 100 MHz and a 4-bit phase interpolator, the maximum cancellation phase required is 0.4 radians, and hence the minimum integration time allowed becomes 625 ps.

3.3. Combination of Cancellation Signal Advancement and Reduction in Integration Time. The improvement in noise cancellation can be further increased by combining both signal advancement and reduction in integration time. When integration time is reduced to $T_s/2$, the optimal signal advancement changes from $T_s/2$ to $T_s/4$ so that the peak magnitude of PQN splits equally in its positive and negative cycle.

In general, the optimal clock advancement reduces by a factor of 2 when the integration time is reduced by a factor of 2. The PSD of the output signal when $T_s/4$ clock advancement and $T_s/2$ integration time are used is plotted in Figure 12. For comparison, PSD plots for zero clock advance and T_s integration time and $T_s/2$ clock advance and $T_s/2$ integration time are also plotted. The peak PQN has reduced by an additional amount of 10 dB due to simultaneous application of the two techniques.

3.4. Optimized Modulator for PQN Shaping. A second-order $\Sigma - \Delta$ modulator results in excessive quantization noise at a frequency offset of $F_s/2$. This increase can lead to spectral mask violation. By including a pair of complex poles in the NTF [35], the high-frequency noise can be reduced at the expense of higher noise at low frequencies. The improvement in noise reduction after the inclusion of a pair of complex

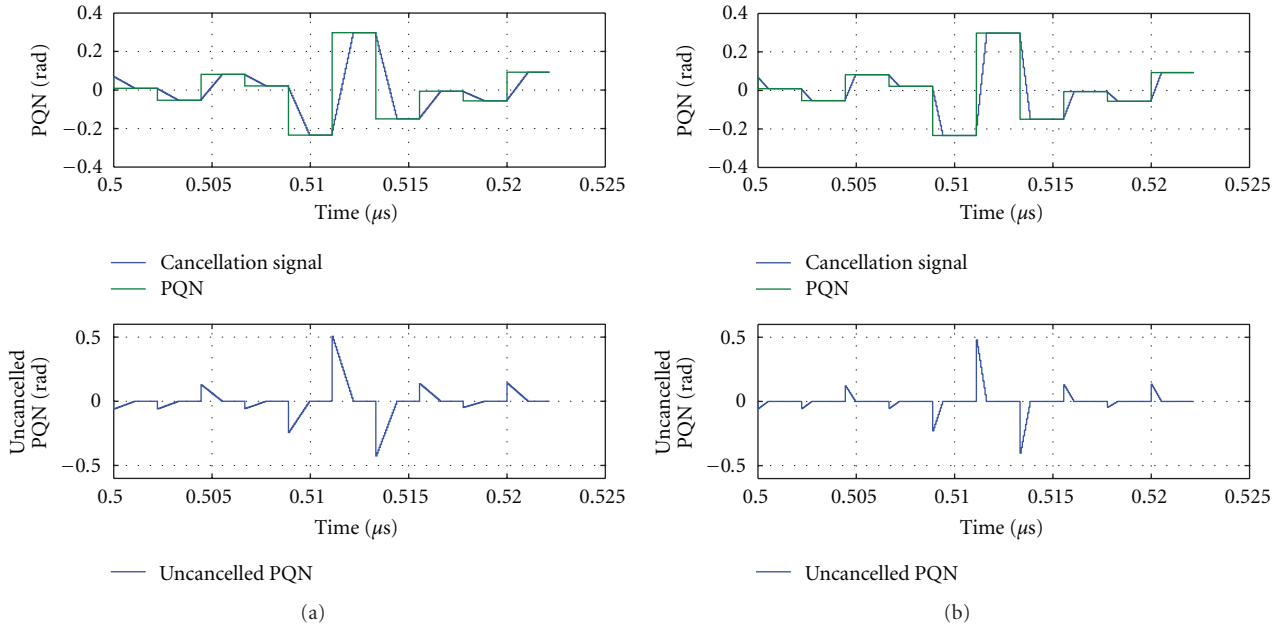


Figure 9: Timing diagram for PQN cancellation technique for (a) integration time of $T_s/2$ and (b) integration time of $T_s/4$, within each time period.

poles in the NTF is depicted in Figure 11(c). In addition, the low frequency noise degradation observed in 2nd-order $\Sigma - \Delta$ modulator due to the advancement of cancellation signal has also improved resulting in lower noise for the optimized NTF.

3.5. Quantization of Cancellation Signal. Since, in a digital implementation of the cancellation path, the frequency data fed to the second VCO port must go through a quantization process, its impact also requires a careful attention. If the frequency data is uniformly quantized within the input dynamic range of VCO control port, its quantization noise has a constant PSD between $-F_s/2$ and $+F_s/2$. Within the loop bandwidth of PLL, this noise will be tracked by the negative feedback loop and its impact will be nullified. Outside the loop bandwidth, VCO integrates this noise due to frequency to phase conversion, and hence an amplified version will appear at its output. Mathematically, the PSD of quantization noise at VCO output can be expressed as

$$S_{\varphi_{n,\text{freq}}}(f) = \frac{1}{4}\frac{\Delta^2}{12F_s}\ \text{sin}\ c^2\left(\frac{\pi f}{F_s}\right)\frac{K_{\text{VCO}}^2}{f^2}\text{NTF}_2(f), \quad (3)$$

where Δ is quantization step size in VCO control voltage, F_s is sampling frequency, f is frequency offset from carrier, K_{VCO} is VCO gain, and $\text{NTF}_2(f)$ is noise transfer function. For an example case with K_{VCO} of 100 MHz/V, F_s of 450 MHz, and Δ of 41 mV (5-bit quantization), the calculated output noise is plotted in Figure 11(a) for uniform quantization and 1st-order and 2nd-order $\Sigma - \Delta$ noise shaping. If the integration time is reduced to $T_s/2$, the expression for quantization noise changes to (Figure 11(b))

$$S_{\varphi_{n,\text{freq}}}(f) = \frac{1}{16}\frac{\Delta^2}{12F_s}\ \text{sin}\ c^2\left(\frac{\pi f}{2\times F_s}\right)\frac{K_{\text{VCO}}^2}{f^2}\text{NTF}_2(f). \quad (4)$$

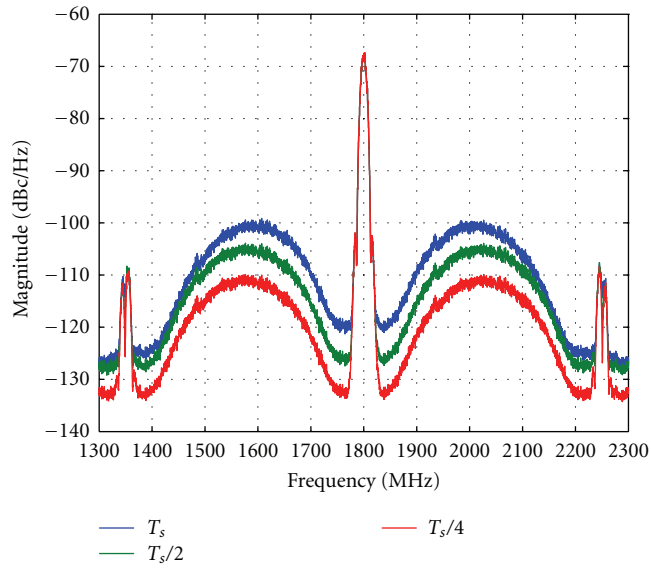


Figure 10: The simulated PSD of a GFSK-modulated signal showing PQN reduction with decrease in integration time.

Clearly, for both cases, the noise at VCO output will suffer from both EVM and ACPR degradation if uniform quantization is applied. In order to cancel the pole in the VCO transfer function, a first- or higher-order zero is required in the NTF of the cancellation signal quantizer. First-order noise shaping results in a flat noise PSD close to DC, while second-order noise shaping has a zero at DC in its noise transfer function. Hence, the noise shaping employed must be of at least second order.

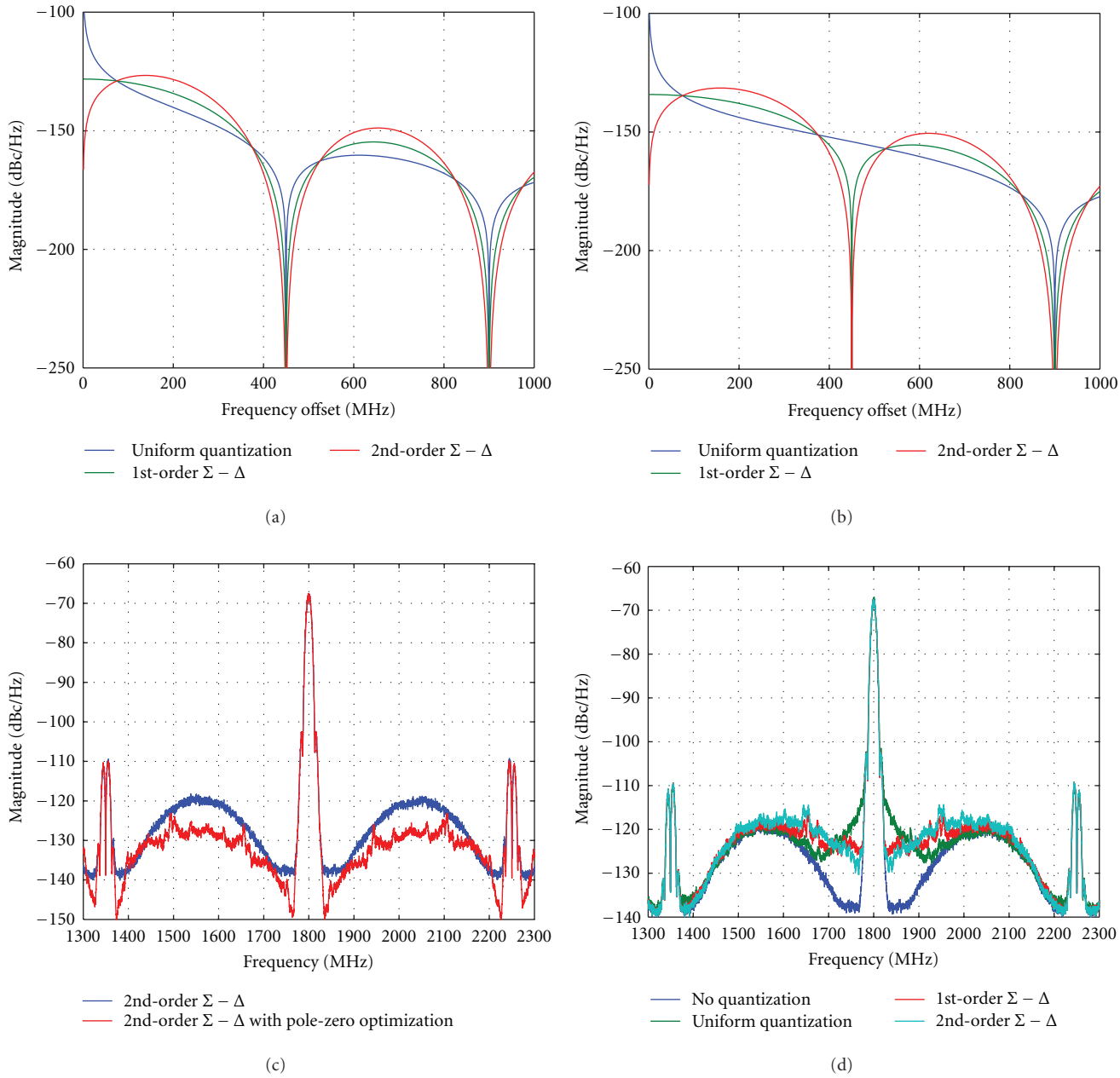


FIGURE 11: Plot of analytical expression for output quantization noise due to quantization in the cancellation path for (a) integration time of T_s and (b) integration time of $T_s/2$. The simulated PSD of a GFSK-modulated signal showing (c) PQN reduction with poles in the NTF of phase path and (d) impact of quantization in the cancellation path using three types of quantizers.

The PSD of a GFSK-modulated signal with quantized cancellation signal is shown in Figure 11(d). Clearly, the residual quantization noise matches the behavior expected for uniform, 1st-order and 2nd-order $\Sigma - \Delta$ noise shaping in the cancellation path, from the preceding analysis.

The overall improvement obtained from the cancellation technique can now be compared with the case when it is off. From Figure 11(d), it can be observed that the peak quantization noise is −128 dBc/Hz, which is 27 dB lower than the modulator with cancellation off (Figure 7).

4. Receive Band Noise

In a Frequency division duplexing (FDD) system, both transmitter and receiver are operational simultaneously. For instance, in the LTE-FDD frequency planning Tx-Rx separation varies from 30 MHz in band XII (700 MHz) to 400 MHz in band X (Tx in 1710–1770 MHz and Rx in 2110–2170 MHz). Due to a finite duplexer Tx to Rx isolation, the transmitter noise in the receive frequency band leaks into the receiver, which can desensitize the receiver (Figure 13). If the transmitter noise in the receive band is −160 dBc/Hz and

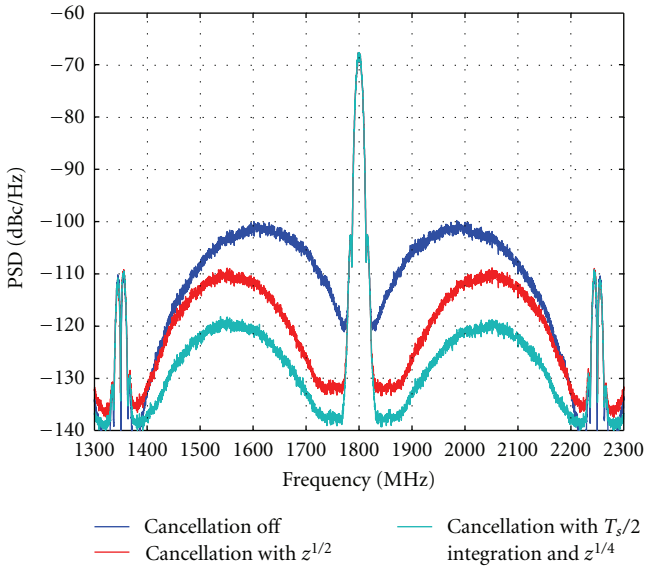


Figure 12: The simulated PSD of a GFSK-modulated signal showing PQN reduction with optimized cancellation signal advancement and integration time.

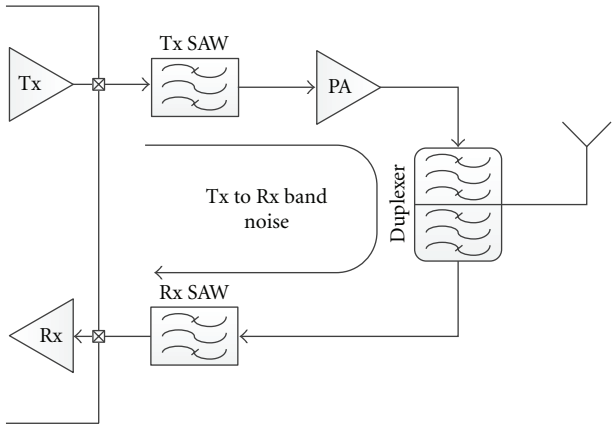


Figure 13: Diagram showing Tx signal leakage in a conventional transceiver.

the duplexer Tx-to-Rx isolation in the receive band is 47 dB, then noise power at LNA input is given by

$$N_{\text{Rx}} = -160\,\text{dBc}/Hz - 47\,\text{dB} + (24 + 1.5 + 1.0)\,\text{dBm}, \quad (5)$$

assuming 24 dBm power at the antenna, 1.5 dB of duplexer Tx insertion loss, and 1.0 dB of antenna switch insertion loss. As a result of this additional noise at LNA input, the receiver noise figure can degrade by 0.5 dB if it was 3 dB in the beginning (without including switch and duplexer loss). In the phase path of a polar transmitter, this noise is composed of PLL phase noise and quantization noise added by the digital phase modulator. In practice, the phase noise of the PLL can be reduced to meet the requirement in the receive band. Consequently, the quantization noise of the modulator becomes the dominant source of noise, requiring additional filtering by off-chip components.

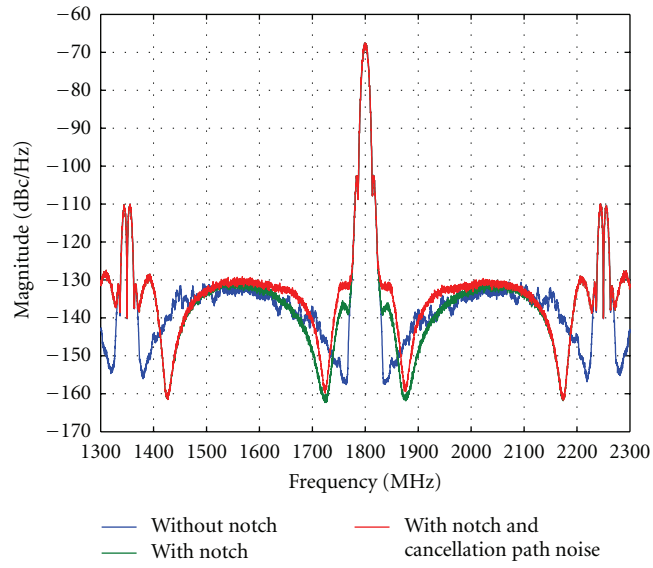


Figure 14: The simulated PSD of a GFSK-modulated signal before and after the inclusion of a notch at 80 MHz offset. Degradation by 2 dB is observed due to quantization noise in the cancellation path.

However, the additional components can be avoided by positioning a quantization noise transfer function (NTF) notch at the receive band frequency. The quantizer of the phase modulator can be modified to include a zero in the quantization noise transfer function. Due to high-pass shaping of quantization noise, the 30–70 MHz Rx band offset poses less design challenge. Improvement in noise at the receive band due to NTF notch at 80 MHz is shown in Figure 14. The resolution of digital phase modulator was increased from 5 bits to 6 in this simulation, to lower its noise contribution. The modulator also employs PQN cancellation technique with $T_s/4$ advancement in the cancellation path for an integration time of $T_s/2$. For comparison, the transfer function without the notch, but including the same PQN cancellation technique, is also shown. An improvement of 12 dB is obtained due to the notch in the transfer function. When the cancellation path is quantized to 5 bits, the noise performance degrads by 2 dB due to the additional noise of the cancellation path. The ACPR performance of the modulator is better than 62 dB in the out-of-band region which meets the requirement of both LTE and WiMAX. Although the ACPR specification can be met with a lower resolution, at least 4 bits are generally required to keep the frequency deviation in the cancellation path within a reasonable range.

5. Conclusion

Transmitters for upcoming wireless standards, 60 GHz band, and software-defined radio require a digital wide-bandwidth phase modulator for a reduction in power consumption and for achieving maximum flexibility in transmission. Several circuit and system techniques for designing such a modulator were reviewed in this paper. Signal processing details of

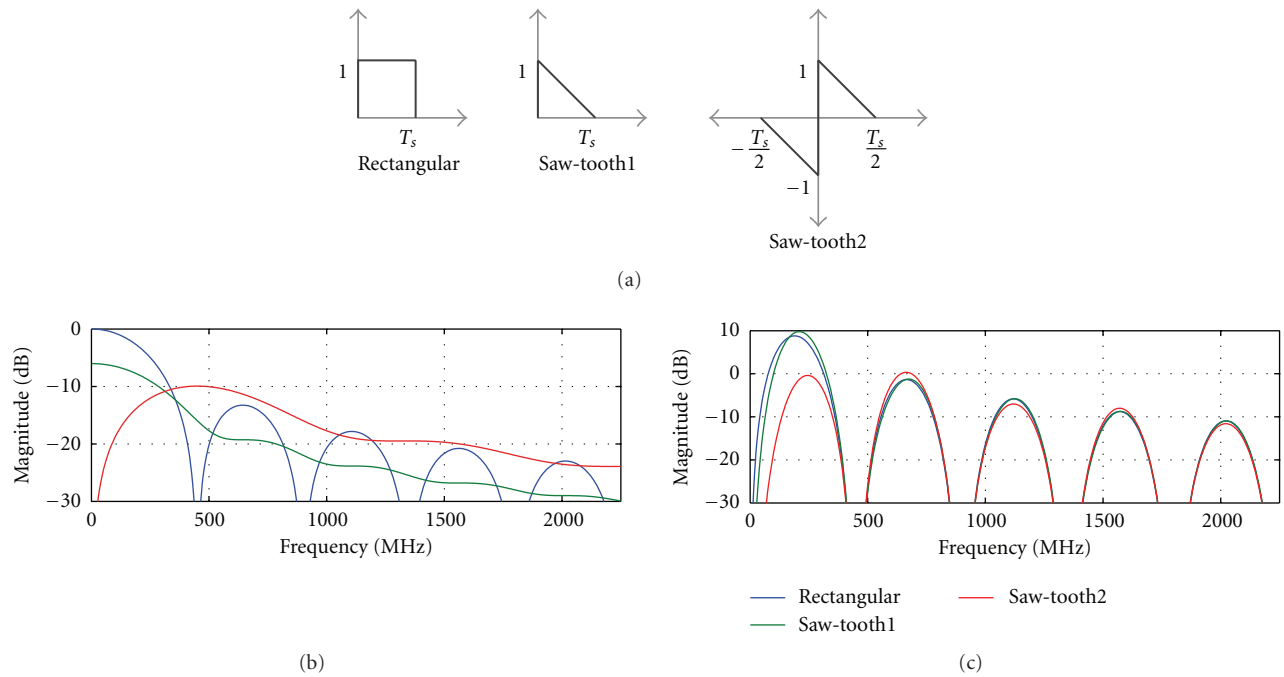


FIGURE 15: (a) Pulse shapes for rectangular, saw-tooth1, and saw-tooth2 functions. (b) Pulse-shaping function $P_1(f)$ for rectangular, saw-tooth, and modified saw-tooth functions. (c) Calculated transfer function of uncancelled PQN for the three cases ($T_s = 1/450$ MHz).

phase quantization noise cancellation were presented with emphasis on advancement of cancellation signal, reduction in integration time, and impact of quantization in the cancellation signal. In the final system, the residual PQN for a 2nd-order $\Sigma - \Delta$ quantized digital phase modulator was lower by 27 dB. Inclusion of a noise transfer function notch was presented to meet the specification of receive band noise in a digital phase modulator.

Appendix

Pulse-shaping function for a rectangular pulse, saw-tooth pulse (saw-tooth1), and modified saw-tooth pulse is obtained after advancing cancellation signal by $T_s/2$ (saw-tooth2):

$$P_1(f)_{\text{rect}} = \sin c^2(\pi f T_s),$$

$$P_1(f)_{\text{saw-tooth1}} = \frac{1}{4\pi^2 f^2 T_s^2}\left((1-\sin c(\pi f T_s))^2 + 4 \times \sin c(\pi f T_s)\ \sin^2\left(\frac{\pi f T_s}{2}\right)\right),$$

$$P_1(f)_{\text{saw-tooth2}} = \frac{4}{4\pi^2 f^2 T_s^2}(1-\sin c(\pi f T_s))^2. \tag{A.1}$$

For a 2nd-order $\Sigma - \Delta$ modulator quantizing $\phi[n]$, $\text{NTF}_1(f)$ is given by

$$\text{NTF}_1(f) = 16 \times \sin^4(\pi f T_s). \tag{A.2}$$

$K_1(f)$, which models the effect of noise cancellation, for the three cases can be written as

$$K_1(f)_{\text{rect}} = 1,$$
$$K_1(f)_{\text{saw-tooth1}} = 4 \times \sin^2(\pi f T_s),$$
$$K_1(f)_{\text{saw-tooth2}} = \frac{4 \times \sin^2(\pi f T_s)}{4}. \tag{A.3}$$

Figure 15(b) depicts the pulse-shaping functions $P_1(f)_{\text{rect}}$, $P_1(f)_{\text{saw-tooth1}}$, and $P_1(f)_{\text{saw-tooth2}}$. Figure 15(c) depicts the calculated transfer function of uncancelled PQN $P_1(f) \times \text{NTF}_1(f) \times K_1(f)$ for the three cases.

Acknowledgment

This research was funded in parts by National Science Foundation awards ECCS-0824279 and ECCS-0955330.

References

[1] V. Petrovic and W. Gosling, "Polar-loop transmitter," *Electronics Letters*, vol. 15, no. 10, pp. 286–288, 1979.

[2] M. R. Elliott, T. Montalvo, B. P. Jeffries et al., "A polar modulator transmitter for GSM/EDGE," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 12, pp. 2190–2199, 2004.

[3] T. Sowlati, D. Rozenblit, R. Pullela et al., "Quad-band GSM/GPRS/EDGE polar loop transmitter," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 12, pp. 2179–2189, 2004.

[4] A. W. Hietala, "A quad-band 8PSK/GMSK polar transceiver," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 5, pp. 1133–1141, 2006.

[5] Y. Akamine, S. Tanaka, M. Kawabe et al., "A polar loop transmitter with digital interface including a loop-bandwidth calibration system," in *Proceedings of the IEEE International Solid-State Circuits Conference: Digest of Technical Papers (ISSCC '07)*, pp. 348–349, San Francisco, Calif, USA, February 2007.

[6] D. C. Cox, "Linear amplification with nonlinear components," *IEEE Transactions on Communications*, vol. 22, no. 12, pp. 1942–1945, 1974.

[7] M. E. Heidari, M. Lee, and A. A. Abidi, "All-digital outphasing modulator for a software-defined transmitter," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, Article ID 4804977, pp. 1260–1271, 2009.

[8] T. A. D. Riley and M. A. Copeland, "A simplified continuous phase modulator technique," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 41, no. 5, pp. 321–328, 1994.

[9] N. M. Filiol, T. A. D. Riley, C. Plett, and M. A. Copeland, "An agile ISM band frequency synthesizer with built-in GMSK data modulation," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 7, pp. 998–1008, 1998.

[10] S. Pamarti, L. Jansson, and I. Galton, "A wideband 2.4-GHz delta-sigma fractional-N PLL with 1-Mb/s in-loop modulation," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, pp. 49–62, 2004.

[11] M. H. Perrott, T. L. Tewksbury III, and C. G. Sodini, "A 27-mW CMOS fractional-N synthesizer using digital compensation for 2.5-Mb/s GFSK modulation," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 12, pp. 2048–2059, 1997.

[12] M. Youssef, A. Zolfaghari, H. Darabi, and A. Abidi, "A low-power wideband polar transmitter for 3G applications," in *Proceedings of the IEEE International Solid-State Circuits Conference: Digest of Technical Papers (ISSCC '11)*, pp. 378–379, San Francisco, Calif, USA, February 2011.

[13] R. B. Staszewski, J. L. Wallberg, S. Rezeq et al., "All-digital PLL and transmitter for mobile phones," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2469–2482, 2005.

[14] S. A. Yu and P. Kinget, "A 0.65-V 2.5-GHz fractional-N synthesizer with two-point 2-Mb/s GFSK data modulation," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 9, Article ID 5226689, pp. 2411–2425, 2009.

[15] W. W. Si, D. Weber, S. Abdollahi-Alibeik et al., "A single-chip CMOS bluetooth v2.1 radio SoC," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, Article ID 4684630, pp. 2896–2904, 2008.

[16] K. C. Peng, C. H. Huang, C. J. Li, and T. S. Horng, "High-performance frequency-hopping transmitters using two-point delta-sigma modulation," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 11, pp. 2529–2535, 2004.

[17] S. Lee, J. Lee, H. Park, K. Y. Lee, and S. Nam, "Self-calibrated two-point delta-sigma modulation technique for RF transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 7, Article ID 5481987, pp. 1748–1757, 2010.

[18] Y. H. Liu and T. H. Lin, "A wideband PLL-based G/FSK transmitter in 0.18 μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 9, Article ID 5226692, pp. 2452–2462, 2009.

[19] H. Mair and L. Xiu, "Architecture of high-performance frequency and phase synthesis," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 6, pp. 835–846, 2000.

[20] P. E. Su and S. Pamarti, "A 2.4 GHz wideband open-loop GFSK transmitter with phase quantization noise cancellation," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 3, pp. 615–626, 2011.

[21] P. Y. Wang, J. H. C. Zhan, H. H. Chang, and H. M. S. Chang, "A digital intensive fractional-N PLL and all-digital self-calibration schemes," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 8, Article ID 5173739, pp. 2182–2192, 2009.

[22] K. J. Wang, A. Swaminathan, and I. Galton, "Spurious tone suppression techniques applied to a wide-bandwidth 2.4 GHz fractional-N PLL," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, Article ID 4684634, pp. 2787–2797, 2008.

[23] E. Temporiti, G. Albasini, I. Bietti, and R. Castello, "A 700-kHz bandwidth $\Sigma\Delta$ fractional synthesizer with spurs compensation and linearization techniques for WCDMA applications," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1446–1454, 2004.

[24] M. Gupta and B. S. Song, "A 1.8-GHz spur-cancelled fractional-N frequency synthesizer with LMS-based DAC gain calibration," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 12, pp. 2842–2851, 2006.

[25] A. Swaminathan, K. J. Wang, and I. Galton, "A wide-bandwidth 2.4 GHz ISM-band fractional-N PLL with adaptive phase-noise cancellation," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2639–2650, 2007.

[26] S. E. Meninger and M. H. Perrott, "A 1-MHZ bandwidth 3.6-GHz 0.18-μm CMOS fractional-N synthesizer utilizing a hybrid PFD/DAC structure for reduced broadband phase noise," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 4, pp. 966–980, 2006.

[27] P. E. Su and S. Pamarti, "A 2-MHz bandwidth Δ—Σ fractional-N synthesizer based on a fractional frequency divider with digital spur suppression," in *Proceedings of the IEEE Radio Frequency Integrated Circuits Symposium (RFIC '10)*, pp. 413–416, Anaheim, Calif, USA, May 2010.

[28] H. Hedayati, W. Khalil, and B. Bakkaloglu, "A 1 MHz bandwidth, 6 GHz 0.18 μm CMOS type-I $\delta\sigma$ fractional-N synthesizer for WiMAX applications," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 12, Article ID 5342360, pp. 3244–3252, 2009.

[29] M. Nilsson, S. Mattisson, N. Klemmer et al., "A 9-band WCDMA/EDGE transceiver supporting HSPA evolution," in *Proceedings of the IEEE International Solid-State Circuits Conference: Digest of Technical Papers (ISSCC '11)*, pp. 366–367, San Francisco, Calif, USA, February 2011.

[30] C. M. Hsu, M. Z. Straayer, and M. H. Perrott, "A low-noise wide-BW 3.6-GHz digital $\Delta\Sigma$ fractional-N frequency synthesizer with a noise-shaping time-to-digital converter and quantization noise cancellation," *IEEE Journal of Solid State Circuits*, vol. 43, no. 12, Article ID 4684627, pp. 2776–2786, 2008.

[31] M. Lee, M. E. Heidari, and A. A. Abidi, "A low-noise wideband digital phase-locked loop based on a coarse-fine time-to-digital converter with subpicosecond resolution," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 10, pp. 2808–2816, 2009.

[32] M. Zanuso, S. Levantino, C. Samori, and A. Lacaita, "A 3MHz-BW 3.6 GHz digital fractional-N PLL with sub-gate-delay TDC, phase-interpolation divider, and digital mismatch cancellation," in *Proceedings of the IEEE International Solid-State Circuits Conference: Digest of Technical Papers (ISSCC '10)*, pp. 476–477, San Francisco, Calif, USA, February 2010.

[33] R. B. Staszewski, K. Muhammad, D. Leipold et al., "All-digital TX frequency synthesizer and discrete-time receiver for Bluetooth radio in 130-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 12, pp. 2278–2291, 2004.

[34] T. LaRocca, J. Liu, F. Wang, and F. Chang, "Embedded DiCAD linear phase shifter for 57–65 GHz reconfigurable direct frequency modulation in 90nm CMOS," in *Proceedings of the IEEE Radio Frequency Integrated Circuits Symposium (RFIC '09)*, pp. 219–222, Boston, Mass, USA, June 2009.

[35] *Delta-SigmaToolBox*,http://www.mathworks.com/matlab-cen-tral/fileexchange/19.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 982314, 15 pages
doi:10.1155/2011/982314

Research Article

Receiver Jitter Tracking Characteristics in High-Speed Source Synchronous Links

Ahmed Ragab,[1] Yang Liu,[1,2] Kangmin Hu,[2,3] Patrick Chiang,[3] and Samuel Palermo[1]

[1] *Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843, USA*
[2] *Broadcom Corporation, Analog and Mixed-Signal Group, Irvine, CA 92618, USA*
[3] *School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA*

Correspondence should be addressed to Samuel Palermo, spalermo@ece.tamu.edu

Received 14 June 2011; Accepted 22 August 2011

Academic Editor: Sudhakar Pamarti

High-speed links which employ source synchronous clocking architectures have the ability to track correlated jitter between clock and data channels up to high frequencies. However, system timing margins are degraded by channel skew between clock and data signals and high-frequency loss. This paper describes how these key channel effects impact the jitter performance and influence the clocking architecture of high-speed source synchronous links. Tradeoffs in complexity and jitter tracking performance of common per-channel de-skew circuits are discussed, along with how band-pass filtering can be leveraged to provide additional jitter filtering at the receiver. Jitter tolerance analysis for a 10 Gb/s system shows that a near all-pass delay-locked loop (DLL) and phase-interpolator- (PI-) based de-skew performs best under low skew conditions, while, at high skew, architectures which leverage band-pass clock filtering or a phase-locked loop (PLL) for increased jitter filtering are more suitable. De-skew based on injection-locked oscillators (ILOs) offer a reduced complexity design and competitive jitter tolerance over a wide skew range.

1. Introduction

Interface architectures which allow for high data rates at improved power efficiency levels are required to satisfy the growing I/O bandwidth in power-constrained environments ranging from data centers [1] to mobile systems [2]. Links which leverage source synchronous clocking, such as high-bandwidth multichannel parallel connections from processor to processor or memory chips (Figure 1) [3, 4], have potential to achieve these objectives due to their wide bandwidth jitter tracking and reduced clock circuitry complexity relative to embedded clock systems [5].

One of the major factors limiting the maximum achievable I/O data rates occurs from the degradation of system timing margins by clock jitter. In a multichannel source synchronous system, jitter can be decomposed into sources which are correlated or common among the clock and data links, such as phase-locked loop (PLL) and supply-noise jitter, and uncorrelated sources, such as driver random and intersymbol-interference-(ISI-) induced jitter. A key advantage of source synchronous systems is that correlated jitter can be tracked over an extremely wide bandwidth, as the clock which synchronizes the transfer of data onto the channels is also forwarded to the receiver to perform the data sampling operation. Thus, only uncorrelated jitter and jitter at frequencies beyond this high tracking bandwidth degrade the data capture process.

Delay variations between the clock and data signal paths, which occur due to circuit board trace mismatches, forwarded clock buffer/regeneration delays, and multichannel clock distribution, have a major impact on link performance. While matched delay elements in the data path have been implemented to mitigate this clock/data skew in lower-speed memory interfaces [6], implementing this at very high speed conflicts with the paramount objective of improving I/O power efficiency. Ultimately, clock-to-data skew can approach the ns-range and places a limit on the maximum jitter frequency that the receiver should track for optimal timing margins. Jitter amplification of the forwarded clock over the low-pass channel is another important effect which influences the link architecture.

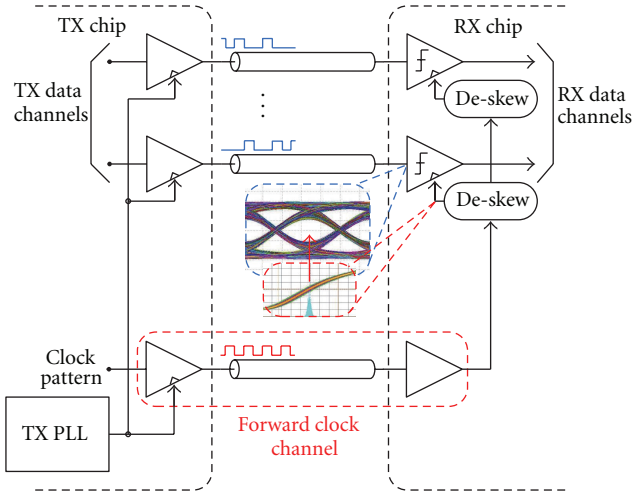


Figure 1: Source synchronous link.

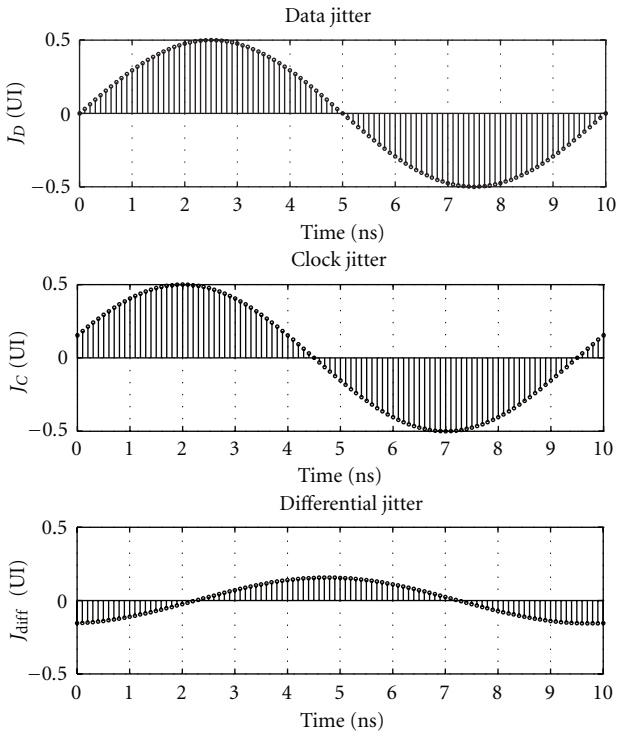


Figure 2: 100 MHz differential jitter in a 10 Gb/s system with 5 UI (500 ps) clock-to-data skew.

This paper presents an analysis of key channel effects and how different receiver clock de-skew structures impact the jitter performance of high-speed source synchronous links. Section 2 gives an overview of source synchronous link architectures and explains the effects of clock and data skew and channel loss on system jitter. The operation and jitter tracking properties of different receiver de-skew circuits are discussed in Section 3. Section 4 details how applying band-pass filtering to the forwarded clock impacts clock jitter performance. A comparison of clock de-skew architectures' jitter tracking performance for different channel conditions is made in Section 5. Finally, Section 6 concludes the paper.

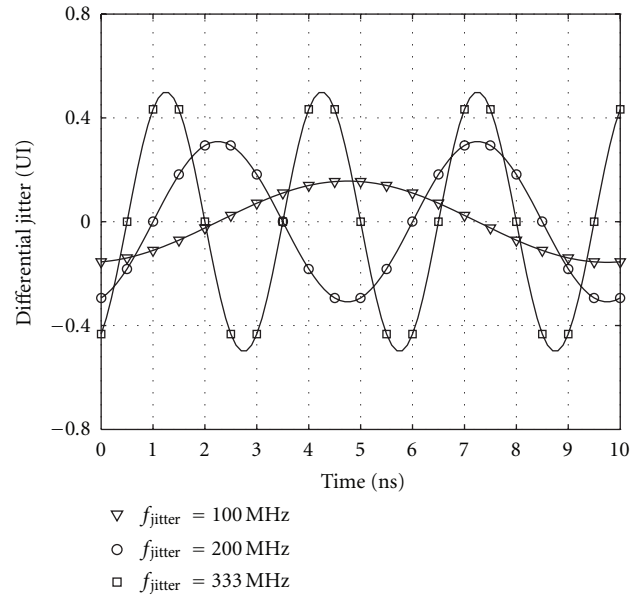


Figure 3: Differential jitter at 100 MHz, 200 MHz, and 333 MHz in a 10 Gb/s system with 5 UI (500 ps) clock-to-data skew.

2. Source Synchronous Link Jitter Properties

2.1. Source Synchronous Architecture. Source synchronous link architectures (Figure 1) employ an extra channel to transmit the clock signal from transmitter to receiver chip for data sampling. In order to maximize the jitter correlation between clock and data paths, a replica transmitter with the same power-supply jitter sensitivity as the data transmitter drives a clock pattern onto the clock channel. Since the low-pass channel will attenuate the clock signal, a receiver clock amplifier is often used to compensate the channel filtering and drive the clock signal over the on-chip distribution network. Clock de-skew circuits adjust the sampling clock phase at each receiver channel independently to maximize link timing margins. In sampled data receivers, the de-skew circuits introduce a 0.5 UI spacing to align the forwarded-clock signal near the center of the data pulses. While for integrating receiver frontends, the forwarded clock signal is aligned near the ends of the data pulses [7].

2.2. Clock Skew Impact on Jitter. In source synchronous systems, emphasis is placed on matching the data and clock path circuit design in order to ensure similar supply noise sensitivity and maximize jitter correlation. As the clock which synchronizes the data transfer onto the channels is also forwarded to the receiver to perform the data sampling operation, this ideally results in zero differential jitter during sampling for correlated jitter with frequency content up to the clock de-skew circuits' jitter tracking bandwidth (JTB).

However, delay mismatch in the correlated clock and data jitter due to skew degrades system timing margins.

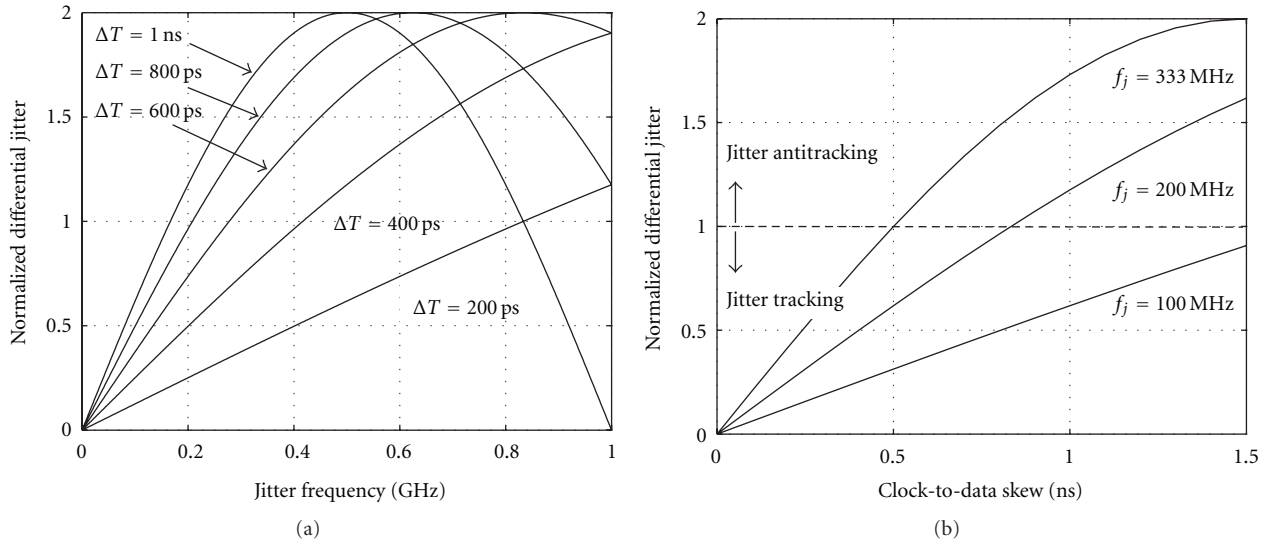


Figure 4: (a) Normalized differential jitter versus frequency with skew ranging from 2 UI (200 ps) to 10 UI (1 ns). (b) Normalized differential jitter versus skew for jitter frequencies from 100 MHz to 333 MHz. Note: the normalized differential jitter is symmetric for negative skew values.

This is important because it is difficult to match the trace lengths of the clock and all data signals in practical systems, resulting in different signal propagation delays. Moreover, circuit mismatches in the clock and data paths introduce additional skew.

Consider a data jitter sequence given by

$$J_D = J_P \sin\left(2\pi f_j t\right), \tag{1}$$

where J_P is the jitter amplitude and f_j is the jitter frequency. Neglecting jitter filtering from receiver de-skew circuitry and channel jitter amplification effects, clock jitter at the sampler is expressed as

$$J_C = J_P \sin\left(2\pi f_j t + mUI\right), \tag{2}$$

where UI is the bit period and m is the skew in bit periods. Differential jitter at the receiver samplers is given by

$$J_{\text{diff}} = J_D - J_C. \tag{3}$$

If no skew exists between the clock and data signals, $J_D = J_C$ and $J_{\text{diff}} = 0$, which implies that the system provides ideal data and clock jitter tracking. However, skew-induced phase shift between the jitter terms results in nonzero differential jitter, as shown in the example of Figure 2 for a 10 Gb/s system with 5 UI (500 ps) skew and a 100 MHz correlated jitter component. Here a peak differential jitter of approximately 0.16 UI results from 0.5 UI jitter amplitude. Figure 3 shows that as jitter frequency increases from 100 to 333 MHz, resulting in a larger phase shift for the 500 ps skew value, differential jitter also increases. This increased differential jitter will ultimately degrade system bit-error rate.

Figures 4(a) and 4(b) illustrate the relationship between skew, jitter frequency, and normalized differential jitter, J_{NOR}, using the frequency domain transformation of a system with a skew of ΔT:

$$|J_{\text{NOR}}(\omega)| = \left|1 - e^{-j\omega\Delta T}\right|. \tag{4}$$

Note that this expression neglects jitter filtering from receiver de-skew circuitry, which will be introduced later in Section 5. Figure 4(a) shows that as jitter frequency increases from low to moderate frequencies, a larger phase shift develops and differential jitter increases. A steeper increase in differential jitter is observed at lower frequencies as the skew is increased. While there may be a small set of dominant jitter frequencies in a multichannel system, the performance impact will differ due to variations in the per-channel clock-to-data skew. Figure 4(b) shows that, for a given jitter frequency, the normalized differential jitter increases with clock-to-data skew. The minimum frequency for which the differential jitter amplitude is equal to the input jitter is inversely proportional to the clock-to-data skew:

$$|J_{\text{NOR}}| = 1 \longrightarrow f_j = \frac{1}{(6\Delta T)}. \tag{5}$$

This is also shown in Figure 3, where for a skew of 500 ps, the 333 MHz differential jitter amplitude is equal to the 0.5 UI input jitter amplitude. The differential jitter is amplified for moderate jitter frequencies above this value.

The differential jitter is periodic with a frequency of $1/\Delta T$, peaking at a value of twice the input jitter amplitude at a frequency of $1/(2\Delta T)$ when the jitter is 180° out of phase and reducing for higher frequencies. However, there is generally little correlation between jitter components at these higher frequencies. This increased differential jitter with frequency implies that an all-pass jitter tracking response is not optimal if clock skew exists in the system and that implementing receiver de-skew circuits with jitter filtering could provide performance benefits.

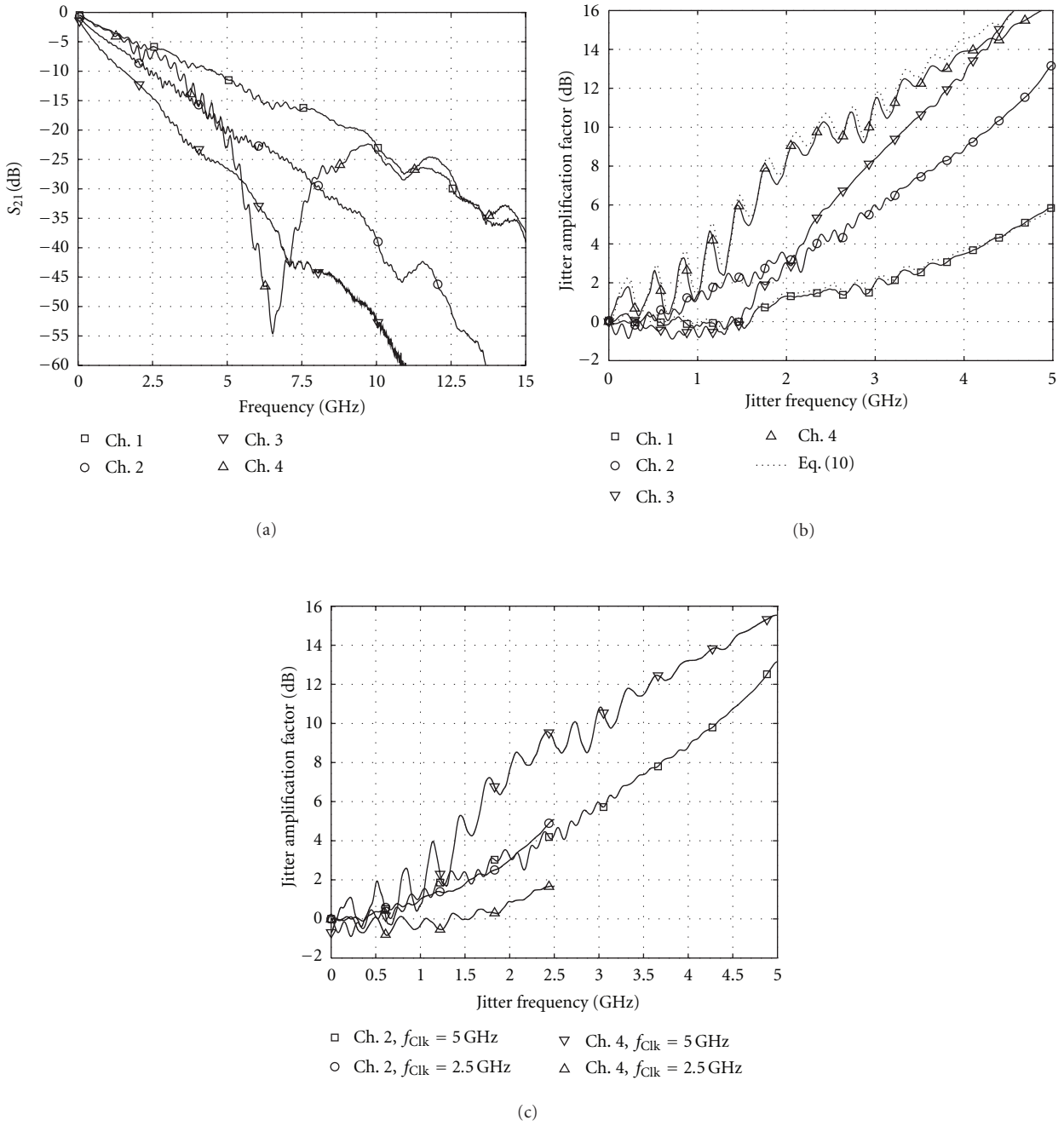


Figure 5: Channel jitter amplification. (a) Frequency response of 4 backplane channels. (b) Channel jitter transfer characteristics for a 5 GHz clock signal. (c) Jitter transfer characteristics for different clock rates for channels 2 and 4 extracted from the channel JIR.

2.3. Channel Clock Jitter Amplification. High-frequency channel loss (Figure 5) also impacts the jitter performance of source synchronous links, as the low-pass channel response causes input jitter to be amplified in a high-pass manner [8, 9]. In order to investigate channel clock jitter amplification, consider the following time domain expression for a clock signal at frequency f_c that contains a sinusoidal jitter component with amplitude J_p and frequency f_j:

$$c(t) = A\cos\left(2\pi f_c t + J_p \sin 2\pi f_j t\right), \tag{6}$$

which, for small J_p values, can be expressed in the frequency domain by

$$S_c(f) = A\left[\frac{1}{2}\delta(f - f_c) - \frac{J_p}{4}[\delta(f - f_L) - \delta(f - f_H)]\right], \tag{7}$$

where $f_L = f_c - f_j$ and $f_H = f_c + f_j$ [8]. The main clock component and the jitter sidebands experience different

channel scaling factors, resulting in the following received signal

$$S_r(f) = \alpha_c A\left[\frac{1}{2}\delta(f-f_c) - \frac{J_P}{4\alpha_c}[\alpha_L\delta(f-f_L) - \alpha_H\delta(f-f_H)]\right], \tag{8}$$

where α_c, α_L and α_H are the channel response at f_c, f_L, and f_H, respectively. Transforming this filtered signal back to the time domain results in

$$r(t) = \alpha_c A\cos\left(2\pi f_c t + J_{pr}\sin 2\pi f_j t\right). \tag{9}$$

Thus, the ratio of received jitter to transmitted jitter can be approximated as

$$\frac{J_{pr}}{J_p} \approx \frac{\alpha_L + \alpha_H}{2\alpha_c}, \tag{10}$$

which implies the potential for received jitter amplification for typical low-pass channels.

To validate this for the 4 backplane channels of Figure 5(a), the jitter impulse response [5, 10–12] is generated by extracting the channel output jitter pattern from a 5 GHz clock input with a 1 ps impulse applied to a rising edge. The channels' jitter transfer functions are obtained by performing a DFT on this output jitter pattern and are shown in Figure 5(b) plotted up to the clock frequency to capture duty cycle distortion (DCD) jitter. As predicted by (10), the jitter amplification factor is highest in channels with the most severe frequency-dependent loss. This further motivates the use of receiver de-skew circuits that provide jitter filtering which, in addition to filtering uncorrelated high-frequency jitter, also mitigate the impact of this clock jitter amplification.

System designers will often choose to forward a lower frequency clock in an attempt to reduce jitter amplification. However, the clock jitter amplification is not so much determined by the absolute loss at the clock frequency, but rather the slope of the loss near the clock frequency. Note that while the 5 GHz loss is similar in channels 2 and 4, the jitter amplification factor is much higher in channel 4 with the 7 GHz resonant null versus the smooth loss channel 2. As shown in Figure 5(c), forwarding a 2.5 GHz clock over channel 4 provides much less jitter amplification due to the relatively shallow loss slope near 2.5 GHz versus the steep loss slope at 5 GHz due to the resonant null. However, for channel 2 which has a relatively uniform loss slope, the jitter amplification is similar for both 2.5 GHz and 5 GHz forwarded clocks. Thus, channel loss characteristics should be carefully considered in the decision to send a lower clock frequency, as this does not always mean a reduction in jitter amplification.

3. Jitter Tracking in Different Receiver Architectures

As alluded to in the previous section, the impact of channel skew and loss on system jitter performance has an influence on the desired receiver jitter filtering properties. This section discusses the operation and jitter filtering or tracking properties of common receiver de-skew circuits.

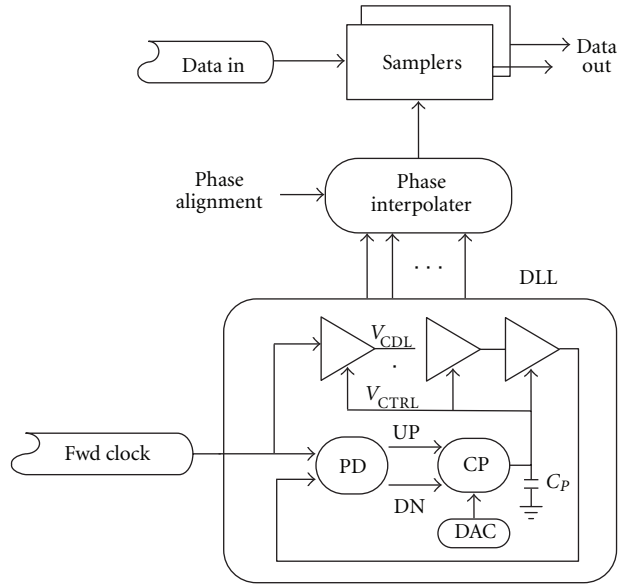


Figure 6: Source synchronous receiver with DLL-PI de-skew.

3.1. DLL-Phase Interpolator De-Skew. Figure 6 shows a receiver de-skew architecture which utilizes a delay-locked loop (DLL) followed by a phase interpolator (PI). The DLL feedback system generates uniformly spaced clock phases by passing the input clock through a multicell delay line set to typically be one or one-half the input clock cycle. High-resolution mixing is then performed by the PI with a pair of these coarselyspaced clock phases in order to generate the optimal sampling clock position.

As the clock passes directly through the DLL delay line and is simply phase shifted by the PI, ideally this DLL-PI de-skew system displays an all-pass jitter transfer function. However, the delay induced by the delay line in the DLL feedback system introduces frequency peaking.

In order to investigate this peaking behavior, consider the DLL z-domain model shown in Figure 7 [13]. The DLL jitter transfer is

$$\frac{\varphi_{\text{out}}}{\varphi_{\text{in}}} = \frac{(1 + K_{\text{CP}}K_{\text{DL}})z - 1}{z - (1 - K_{\text{CP}}K_{\text{DL}})}, \tag{11}$$

where

$$K_{\text{CP}} = \frac{I_{\text{CP}}T_S}{2\pi C_P}, \tag{12}$$

I_{CP} is the charge pump current, C_P is the loop filter capacitor, T_S is the sampling period, and K_{DL} is the delay line gain.

The frequency peaking observed in the 5 GHz DLL jitter transfer function of Figure 8 results in undesired amplification of high-frequency input jitter and degradation of system timing margins. Introducing an additional high-frequency pole, ω_p, within the DLL can reduce this high-frequency jitter amplification. A common example of this additional pole is powering the delay line with a linear regulator which introduces extra filtering in the loop [14].

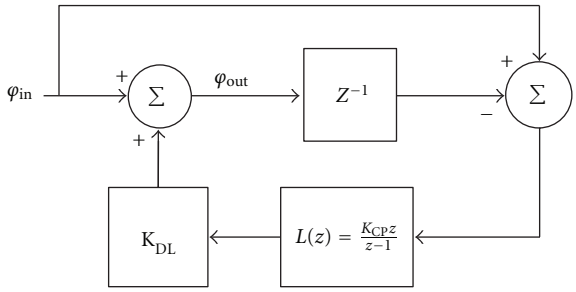


Figure 7: DLL z-domain model.

With this additional pole, the overall DLL loop filter response is modified to

$$L(z) = \frac{K_{\mathrm{CP}} z^2}{z-1}\left(\frac{1-e^{-\omega_p T_s}}{z-e^{-\omega_p T_s}}\right), \tag{13}$$

and the DLL jitter transfer becomes

$$\frac{\varphi_{\mathrm{out}}}{\varphi_{\mathrm{in}}} = \frac{(z-1)(z-e^{-\omega_p T_s}) + K_{\mathrm{CP}}K_{\mathrm{DL}}(1-e^{-\omega_p T_s})z^2}{(z-1)(z-e^{-\omega_p T_s}) + K_{\mathrm{CP}}K_{\mathrm{DL}}(1-e^{-\omega_p T_s})z}. \tag{14}$$

As observed in Figure 8, introducing an additional 250 MHz pole reduces the high-frequency jitter amplification. In order to compensate for the residual frequency peaking, it is possible to cascade an injection-locked oscillator after the DLL [13] or leverage band-pass filtering of the clock signal [15] prior to the DLL for additional filtering.

The power supply noise performance of the de-skew circuitry is also an important design consideration in these multichannel source synchronous links, as switching noise from multiple transmitters, receivers, and core logic can couple into the receiver clock de-skew circuitry. As a DLL exhibits a high-pass response to noise coupled into the power supply of the delay line [16], the DLL high-pass bandwidth, which is set by the pole in (11), should be increased to minimize the impact of power supply noise. However, a tradeoff exists in the DLL-PI architecture between power supply noise filtering and peaking in the jitter transfer function, as increasing the DLL pole frequency results in increased peaking. Thus, the DLL pole frequency location should be set to balance these two system design considerations.

3.2. PLL-Phase Interpolator De-Skew. Figure 9 shows a receiver de-skew architecture which utilizes a phase-locked loop (PLL) followed by a PI. Similar to the DLL-PI de-skew, the PLL generates uniformly spaced clock phases with a voltage-controlled oscillator (VCO) which is phase-locked to the input clock. In addition, the PLL can provide frequency multiplication of a lower-frequency forwarded clock.

The overall jitter transfer function is the PLL phase transfer function, as ideally the PI only bypasses the PLL output signal jitter to the sampling clock. Utilizing a common series

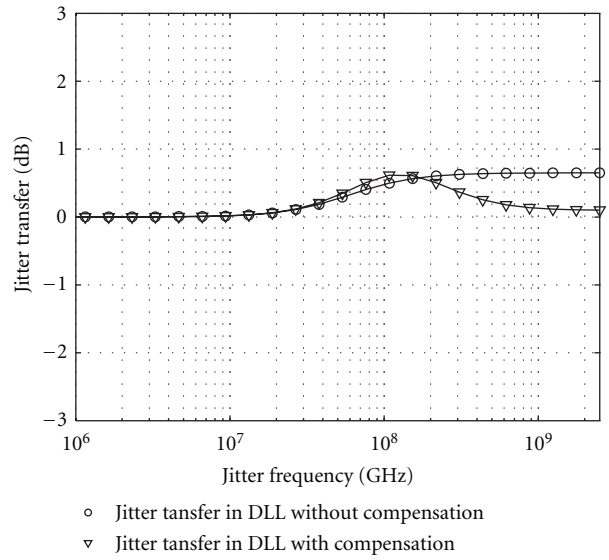


Figure 8: DLL jitter transfer with 5 GHz reference clock.

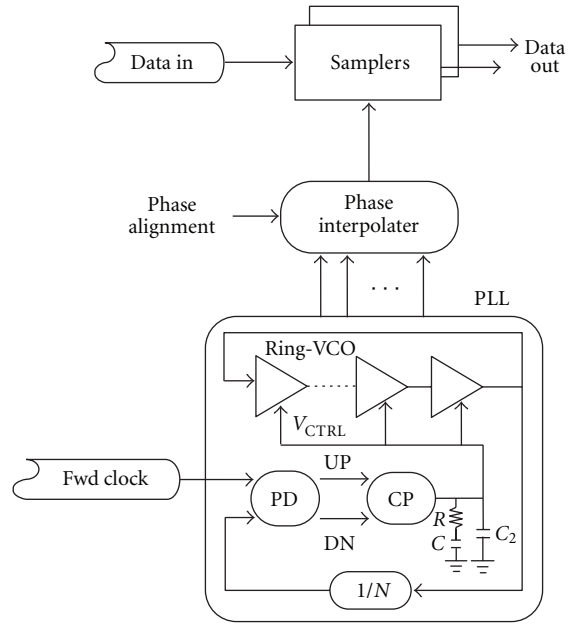


Figure 9: Source synchronous receiver with PLL-PI de-skew.

R-C filter and neglecting any secondary parallel filtering cap, C_2, the PLL jitter transfer function is

$$H_{\mathrm{PLL}}(s) = \frac{(I_{\mathrm{CP}}K_{\mathrm{VCO}}/2\pi C)(RCs+1)}{s^2 + (I_{\mathrm{CP}}/2\pi)(K_{\mathrm{VCO}}/N)Rs + (I_P/2\pi C)(K_{\mathrm{VCO}}/N)}, \tag{15}$$

where I_{CP} is the charge pump current and K_{VCO} is the VCO gain. This expression can be rewritten as

$$H(s) = N\frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{16}$$

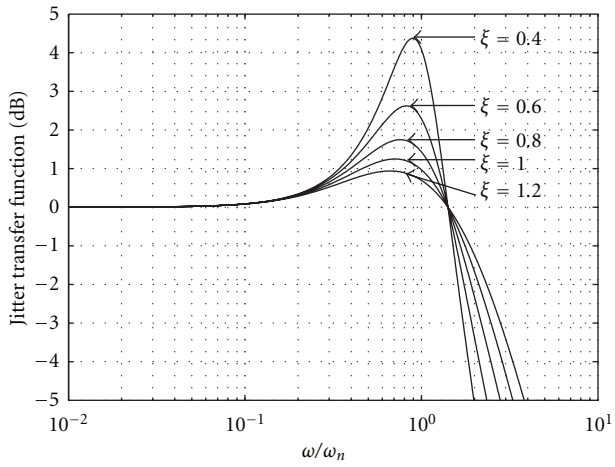


Figure 10: Second-order PLL jitter transfer characteristic for different damping factors.

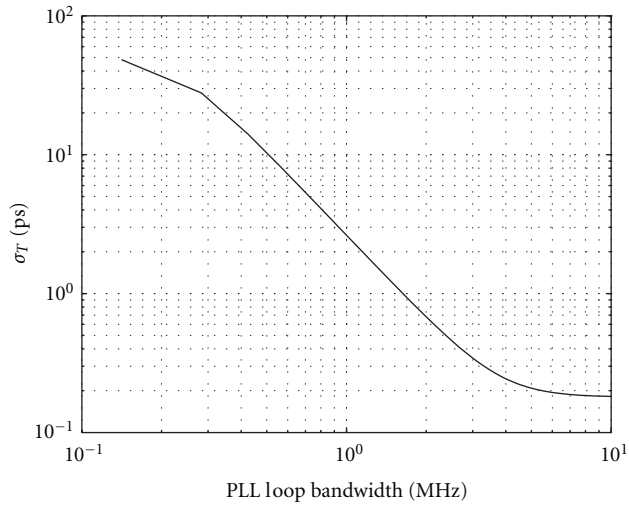


Figure 11: VCO random jitter versus PLL loop bandwidth.

where

$$\omega_n = \sqrt{\frac{I_{CP}}{2\pi C}\frac{K_{VCO}}{N}}, \qquad \zeta = \frac{R}{2}\sqrt{\frac{I_{CP}C}{2\pi}\frac{K_{VCO}}{N}}. \tag{17}$$

Setting the damping factor, ζ, too low in the second-order PLL jitter transfer function will result in peaking that amplifies jitter on the forwarded clock. As shown in Figure 10, this peaking exceeds 1 dB for damping factors less than 1.2. While this peaking can be reduced by increasing the damping factor further, there is the potential for instability and additional frequency peaking if the damping factor is increased excessively due to the secondary pole introduced by the extra filter capacitor. A PLL damping factor of $\zeta = 1.2$ is assumed for the remainder of this paper.

If increased jitter filtering is desired due to channel skew or loss effects, PLL bandwidth can be lowered by reducing charge pump current or increasing filter capacitance. However, excessive reduction in loop bandwidth increases both PLL settling time, which is a problem for low-power systems that require fast wakeup from power-down modes [17], and VCO accumulated jitter, which will degrade timing margins:

At the PLL output, VCO phase noise exhibits a high-pass transfer function.

$$H_{VCO}(s) = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \tag{18}$$

The accumulated VCO jitter is a random jitter (RJ) component that has to be considered in the link timing budget. In the time domain, VCO random jitter will accumulate up to a time inversely proportional to the PLL bandwidth. The variance of the VCO random jitter is calculated from the VCO phase noise profile and the PLL transfer function by [18]

$$\sigma_T^2 = \frac{4}{\omega_o^2}\int_0^\infty S_\varphi(f)\,df, \tag{19}$$

where

$$S_\varphi(f) = |\mathrm{JTF}_{VCO}(s)|^2 S_{VCO}. \tag{20}$$

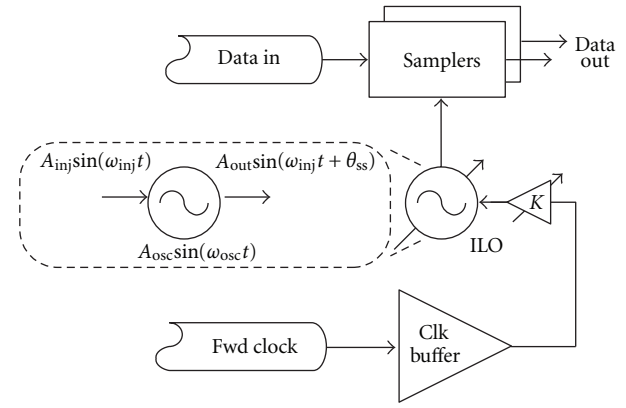


Figure 12: Source synchronous receiver with ILO de-skew.

Figure 11 plots calculated σ_T values for the VCO phase noise profile of Figure 14 and verifies that VCO accumulated jitter reduces as PLL loop bandwidth is increased. Thus, in setting PLL loop bandwidth, system designers must balance the tradeoff between filtering input forwarded clock jitter and VCO accumulated jitter.

PLLs are also susceptible to power supply noise, especially the noise coupled through the VCO supply. As a PLL exhibits a band-pass response to noise coupled into the VCO power supply [18], the PLL bandwidth should be reduced to minimize the impact of power supply noise. However, this will come at the expense of reducing the jitter tracking bandwidth and also VCO random jitter accumulation.

3.3. ILO De-Skew. Relative to DLL or PLL-PI architectures, a simpler approach involves utilizing an injection-locked oscillator (ILO) to obtain the required per-channel de-skew, as shown in Figure 12. Under injection lock, the oscillator runs at the same frequency of the injected clock signal, but with an output phase shift of θ_{ss} that is a function of the relative injection clock signal strength, $K =$

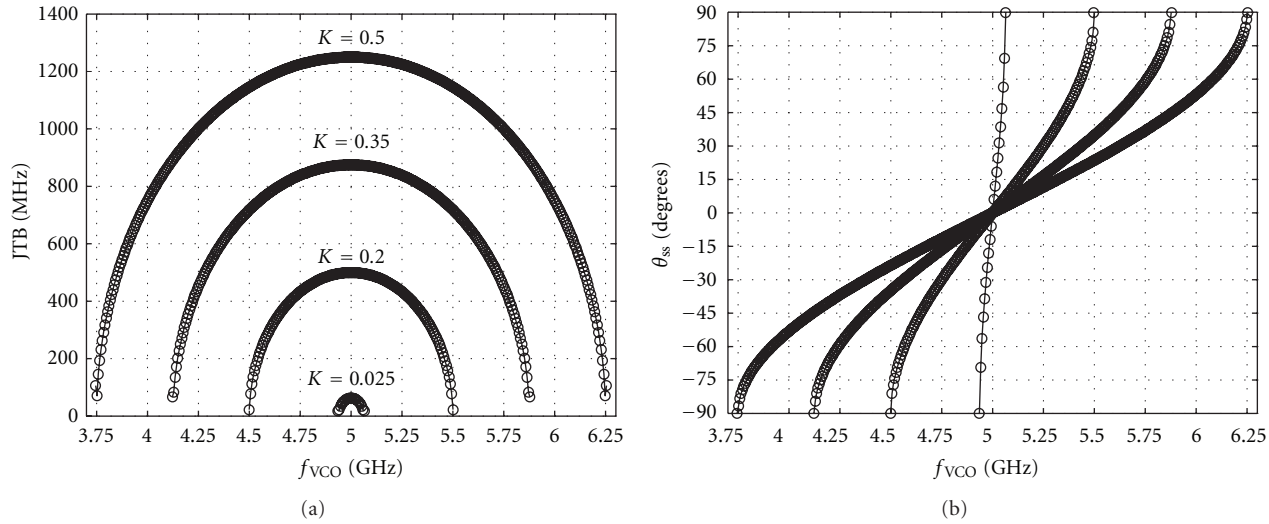


Figure 13: ILO characteristics as free-running frequency is varied relative to the 5 GHz injection signal. (a) Jitter tracking bandwidth. (b) Output phase shift.

$|A_{inj}|/|A_{osc}|$, and the difference between the oscillator's free-running frequency, ω_{osc}, and the injected clock frequency, ω_{inj}. Derived in [19, 20], the output phase shift can be expressed as

$$\Delta\omega = \omega_{osc} - \omega_{inj} \approx \frac{K}{A}\sin\theta_{ss}, \tag{21}$$

where A varies according to the oscillator topology,

LC oscillator:

$$A = \frac{2Q}{\omega_{osc}}, \tag{22}$$

ring oscillator:

$$A \approx \frac{n}{2\omega_{osc}}\sin\left(\frac{2\pi}{n}\right), \tag{23}$$

and Q is the LC oscillator tank quality factor and n is the number of delay stages in the ring oscillator. Theoretically, ILOs are only capable of achieving a phase de-skew range of $\pm 90°$, which is the minimum required phase shift for two clock phases in a half-rate receiver architecture. However, the injection locking is weak at this extreme phase shift. In order to make the system more robust and provide additional phase shift, additional weighted phase inversions of the injected signal can be employed [21].

An ILO provides first-order low-pass jitter filtering on the incoming clock signal

$$H_{ILO}(f) = \frac{1}{1 + jf/f_p}, \tag{24}$$

where f_p is the ILO jitter tracking bandwidth. Tuning the output phase shift by adjusting the oscillator free-running frequency and injection strength will also impact the ILO jitter tracking bandwidth:

$$\omega_p = \sqrt{\frac{K^2}{A^2} - \Delta\omega^2} = \frac{K}{A}\cos\theta_{ss}. \tag{25}$$

Assuming a 4-stage ring oscillator with a 5 GHz free-running frequency and a 6.5 MHz minimum frequency step, the jitter tracking bandwidth and phase shift are plotted in Figure 13 for various injection strengths. A maximum jitter tracking bandwidth is obtained with zero phase shift, with a bandwidth degradation of less than 10% for de-skew settings within $\pm 36°$. However, the bandwidth falls off sharply as de-skew settings approach the $\pm 90°$ theoretical maximum phase shift.

ILO jitter tracking implies that the output phase noise can actually be superior to the inherent oscillator phase noise, provided that the injection signal has lower phase noise, as is often the case where an LC-PLL is used at the transmitter chip to generate the forwarded clock and ring oscillators at the receiver serve as per-channel ILOs. If S_{inj} is the injected clock phase noise and S_{osc} is the de-skew oscillator phase noise, then the output phase noise, S_{out}, at a given frequency, ω, can be expressed in terms of frequency offset $\Delta\omega$ or de-skewed output phase shift θ_{ss} [21]:

$$\begin{aligned} S_{out} &= |\mathrm{JTF}_{inj}|^2 S_{inj} + |\mathrm{JTF}_{osc}|^2 S_{osc}, \\ S_{out} &= \frac{\left(K^2/A^2 - \Delta\omega^2\right)S_{inj} + \omega^2 S_{osc}}{\left(K^2/A^2 - \Delta\omega^2\right) + \omega^2}, \\ S_{out} &= \frac{((K/A)\cos\theta_{ss})^2 S_{inj} + \omega^2 S_{osc}}{((K/A)\cos\theta_{ss})^2 + \omega^2}. \end{aligned} \tag{26}$$

Using this expression, output phase noise is plotted for several de-skew settings in Figure 14. As the free-running frequency offset is tuned higher to generate a larger phase shift, the output phase noise deviates from the injection phase noise and begins to track the free-running oscillator phase noise at lower frequencies.

ILO accumulated jitter, obtained by integrating the ILO output phase noise of Figure 14 with (19), is shown as the de-skew phase is varied in Figure 15. Higher output jitter

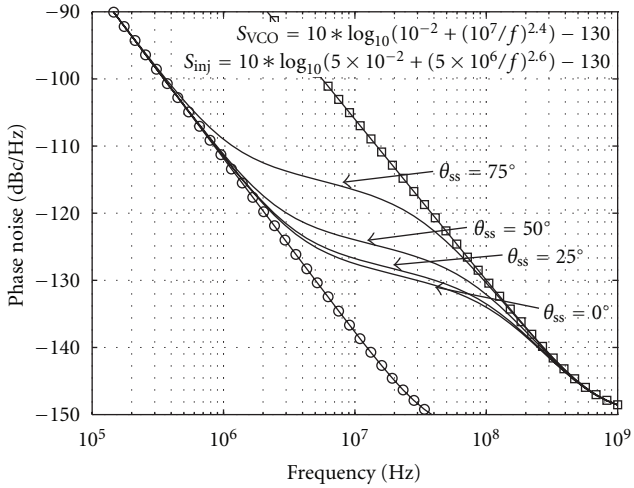


Figure 14: Phase noise transfer for ILO at different θ_{ss} settings.

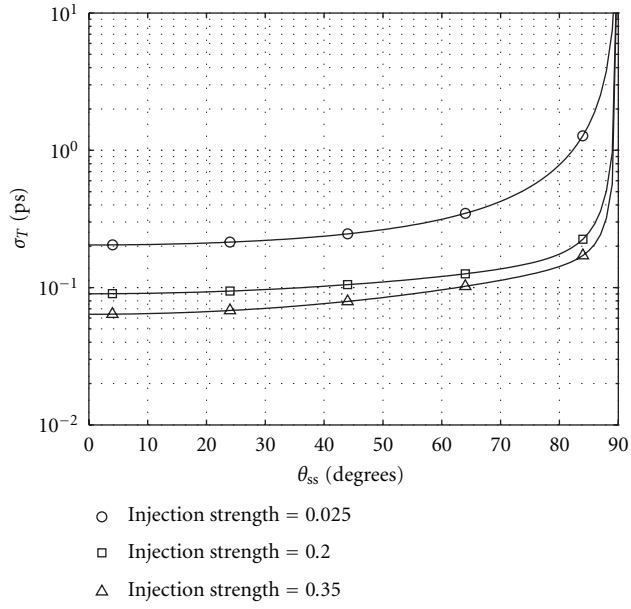


Figure 15: Variation of ILO random jitter with θ_{ss}.

is observed as the de-skew phase is increased due to more of the free-running oscillator phase noise spectrum being integrated.

An increase in injection strength allows for a higher jitter tracking bandwidth and allows the output phase noise spectrum to track the injection phase noise over a larger frequency range, resulting in a reduced amount of accumulated jitter for a given de-skew setting. Note that near the edge of the de-skew range the accumulated jitter rises sharply, which would dramatically degrade receiver timing margins. This motivates the use of quarter-rate receiver architectures [22] which only require phase de-skew of four ILO clock phases over a range of ±45° to cover the necessary ±0.5 UI tuning. If the phase de-skew is limited to a maximum ±45° de-skew range, the ILO jitter tracking bandwidth is not degraded significantly and the oscillator accumulated jitter is significantly reduced.

ILOs are also sensitive to any noise coupled from their power supply, with the severity dependent on the specific oscillator topology [23]. In setting ILO design parameters such as LC oscillator Q and ring oscillator n, designers should balance these parameters' impact on supply noise sensitivity and jitter tracking bandwidth.

4. Band-Pass Filtering for Forwarded Clock Links

The use of band-pass filters can also provide jitter filtering, as an alternative to the de-skew circuits in the previous section. In a forwarded clock system, band-pass filtering can be leveraged to provide jitter filtering in a DLL de-skew system or decouple the dependency of jitter filtering with VCO jitter accumulation present in a PLL or ILO system. Band-pass filtering has been implemented in the receiver input clock amplifier by replacing the common differential resistive load with an LC tank designed to center the filter at the forwarded clock frequency [15]. Inductive termination has also been used to resonate at the clock frequency with the capacitance of a multichannel distribution network [24], resulting in a band-pass response that both provides jitter filtering and reduced clock distribution power by increasing the effective distribution impedance.

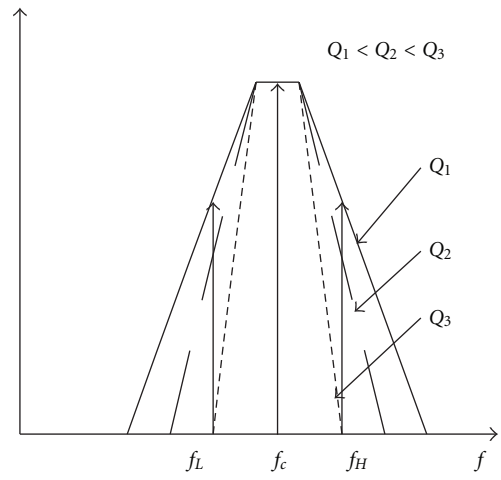


Figure 16: Band-pass filtering of sinusoidal jitter.

In order to investigate the jitter filtering offered by band-pass filters, we consider the expression stated earlier in (10). For a band-pass filter properly centered at the input clock frequency (Figure 16), $\alpha_c > (\alpha_L \approx \alpha_H)$. Thus, $J_{pr} < J_p$ and the jitter of the transmitted clock has been reduced by band-pass filtering.

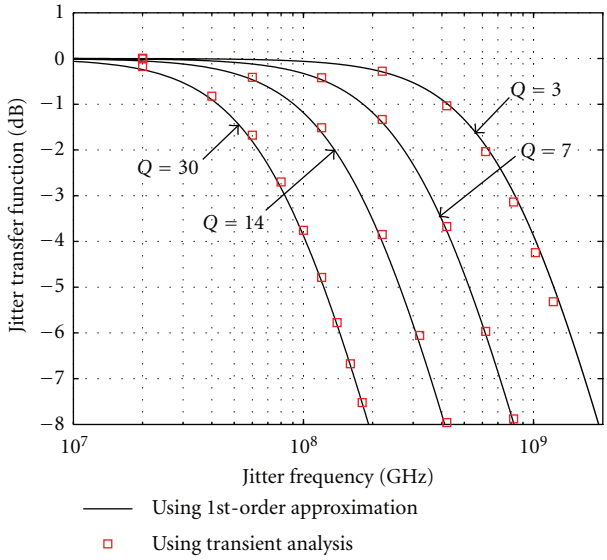


Figure 17: Jitter transfer function of a 5 GHz band-pass system.

For up to moderate jitter frequency offsets, the band-pass function can be approximated as a low-pass function with respect to frequencies offset from f_c:

$$|H(f_c - f)| \approx |H(f_c + f)| \approx \frac{\left|H\left(f = f_c\right)\right|}{\left|1 + jf/f_p\right|}, \quad (27)$$

where

$$f_p = \frac{f_{\mathrm{BW}}}{2} = \frac{f_c}{2Q}, \quad (28)$$

and f_{BW} is the bandwidth of a band-pass filter with quality factor, Q. The band-pass filter's fitter transfer function is approximated by

$$\mathrm{JTF}_{\mathrm{BP}}(j2\pi f) = \frac{|H(f_c - f)| + |H(f_c + f)|}{2|H(f_c)|} = \left|\frac{1}{1 + jf/f_p}\right|. \quad (29)$$

Figure 17 shows that jitter filtering increases with Q value. At a 5 GHz center frequency, a Q of 3 yields a jitter tracking bandwidth near 800 MHz. This Q value can be realized with a passive-inductor-based band-pass filter [15]. Large-signal simulations with an active-inductor band-pass filter [25] show that a Q of 30 is possible, which would yield a jitter tracking bandwidth near 80 MHz. Band-pass filters which allow for Q tuning [25] provide the potential for an adjustable jitter tracking bandwidth that can be set independent of the de-skew position and also avoid the jitter accumulation present in a PLL or ILO system.

5. Comparison of Source Synchronous Clocking Architectures

The previous sections discussed the jitter transfer characteristics of the channel and of different receiver blocks that a forwarded clock could encounter. This section examines how jitter tracking bandwidth impacts system differential jitter and compares the jitter tolerance performance of different source synchronous clock architectures for various channel skew conditions.

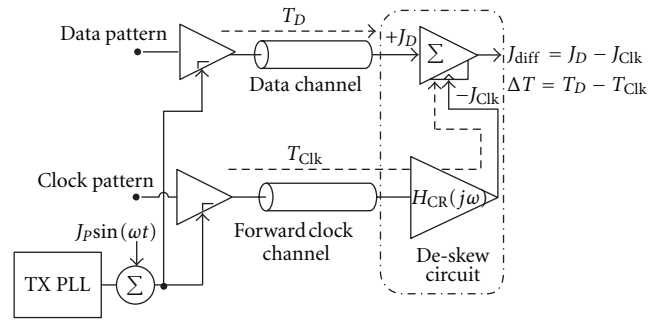


Figure 18: Source synchronous link frequency domain model.

To understand how jitter tracking bandwidth impacts receiver performance, the system model of Figure 18 is used. Including the receiver circuitry jitter transfer function $H_{\mathrm{CR}}(f)$, the differential jitter seen at the sampler is

$$J_{\mathrm{diff}} = J_p \left| 1 - e^{-j\omega * \Delta T} |H_{\mathrm{CR}}(f)| \right|. \quad (30)$$

In the case of DLL-PI de-skew, if the peaking due to the delay line is neglected, the jitter transfer function can be approximated as all-pass, that is, $H_{\mathrm{CR}}(f) = 1$, and would not alter the differential jitter function. While, for systems which use PLL-PI or ILO de-skew or include a band-pass filter in the clock path, the forwarded clock jitter is attenuated by a low-pass function. For these systems, a first-order low-pass function serves as a good approximation for the jitter transfer function.

$$H_{\mathrm{CR}}(f) = \frac{1}{1 + jf/f_p}. \quad (31)$$

In order to illustrate how varying the jitter tracking bandwidth affects differential jitter, consider the results for jitter at a common power-supply resonance frequency of 200 MHz [26], shown in Figure 19. Low skew values result in small relative phase shifts for the jitter on the data and clock signals, allowing for minimal filtering of the 200 MHz jitter on the receiver clock and an optimum jitter tracking bandwidth near or above 1 GHz. Phase shift between this correlated jitter increases with skew, resulting in a pronounced optimal jitter tracking bandwidth for skew values above 500 ps, which is as low as 60 MHz for a skew of 1 ns. If the jitter tracking bandwidth is increased beyond this optimal point, the differential jitter increases and can even be amplified as the correlated clock and data jitter combine out of phase. This implies that, along with general system constraints (i.e., power/area consumption and wake-up time), clock-to-data skew should be considered in selecting the receiver jitter tracking bandwidth.

Optimal jitter tracking bandwidth depends on the location of the dominant jitter frequency terms, as shown by

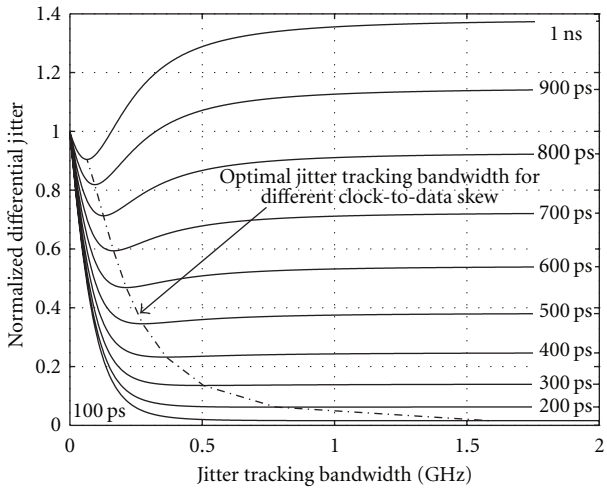


Figure 19: Normalized differential jitter for jitter frequency at 200 MHz versus jitter tracking bandwidth of receiver for skew of 100 ps to 1 ns.

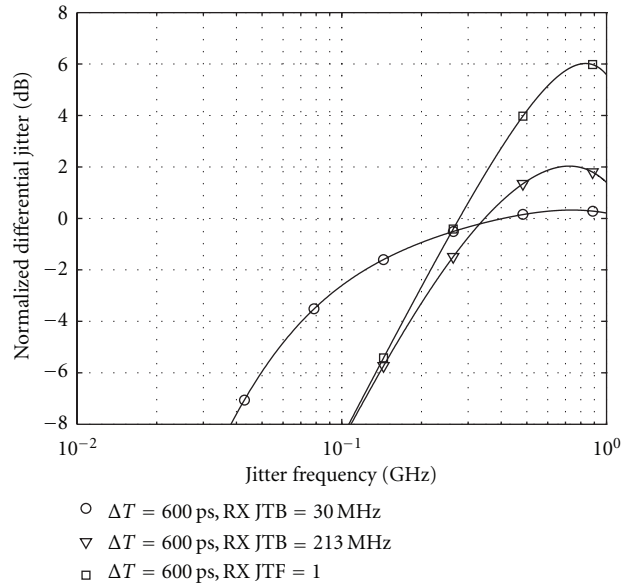


Figure 20: Normalized differential jitter resulting with a skew of 6 UI (600 ps) and different receiver jitter tracking bandwidths.

plotting the normalized differential jitter over a wide frequency range for a skew of 600 ps in Figure 20. As predicted by Figure 19, the 213 MHz optimal jitter tracking bandwidth displays the lowest normalized differential jitter for a 200 MHz jitter frequency term. If jitter is dominant at higher frequencies, the system would benefit from the use of a lower jitter tracking bandwidth to filter the jitter terms combining out of phase. However, this lower jitter tracking bandwidth will result in higher differential jitter at lower frequencies, as the 100 MHz performance is significantly worse with a 30 MHz jitter tracking bandwidth. Note that the absence of jitter filtering, or a jitter transfer function of unity, results in significantly worse differential jitter at higher frequencies, peaking at 6 dB at a frequency of $1/(2\Delta T)$ where the correlated jitter terms combine with a 180° phase shift.

A key receiver performance metric involves quantifying the maximum amount of sinusoidal jitter the receiver can tolerate for a given bit error rate (BER) specification, known as jitter tolerance [27]. For a sampled data receiver with an ideal 0.5 UI timing margin, the maximum tolerable phase error or differential jitter is

$$\varphi_{\text{diff}} = \varphi_D - \varphi_C \leq 0.5\,\text{UI}. \tag{32}$$

Considering Figure 18 source synchronous model results in a maximum tolerable sinusoidal jitter amplitude of [26]

$$J_{\text{Pmax}} = J_{\text{TOL}} = \frac{0.5\,\text{UI}}{\left|1 - e^{-j\omega * \Delta T}\right| H_{\text{CR}}(f)\,|\,|}, \tag{33}$$

which will vary based on the amount of clock-to-data skew and jitter transfer function of the receiver clocking circuitry. In addition to these effects, any VCO accumulated jitter will subtract from the system timing margin:

$$\varphi_{\text{diff}} \leq (0.5\,\text{UI} - Q_{\text{BER}} \cdot \sigma_T), \tag{34}$$

where

$$Q_{\text{BER}} = \sqrt{2}erfc\left(1 - \frac{\text{BER}}{\rho_T}\right), \tag{35}$$

and ρ_T is the transition density, assumed 0.5 for random data signals [28]. Thus, the jitter tolerance expression is modified for the ILO and PLL-PI de-skew architectures to include the oscillator accumulated jitter, which, as discussed in Section 3, is a function of the jitter tracking bandwidth:

$$J_{\text{TOL}} = \frac{0.5\,\text{UI} - Q_{\text{BER}} \cdot \sigma_T}{\left|1 - e^{-j\omega * \Delta T}\right| H_{\text{CR}}(f)\,|\,|}. \tag{36}$$

In order to compare the jitter tolerance performance of the key source synchronous clock architectures at different skew conditions, a 10 Gb/s half-rate architecture with a 5 GHz forwarded clock is modeled with the following assumptions for the different receiver clock circuits. The first-order model of Figure 7 is used for the DLL-PI case, while a variable bandwidth PLL with a 5 GHz reference clock ($N = 1$) and a maximum jitter tracking bandwidth of 150 MHz is assumed for the PLL-PI de-skew. For the ILO modeling, a four-stage ring oscillator is considered with injection strength assumed to be variable from an extremely high value of $K = 0.5$ [26] to a minimum value of 0.025 to allow for a de-skew resolution near 36 phase settings within 1 UI [22]. This yields an ILO jitter tracking bandwidth which ranges from 54 MHz to 1.25 GHz for the 5 GHz forwarded clock frequency. For systems which leverage a BPF, the filter Q is variable from 3 to 30, resulting in a potential jitter tracking bandwidth from 833 MHz to 83 MHz.

5.1. Zero Clock-to-Data Skew (0 UI). While ensuring identical delays on the clock and data paths poses a major challenge, the zero clock-to-data skew is first considered in

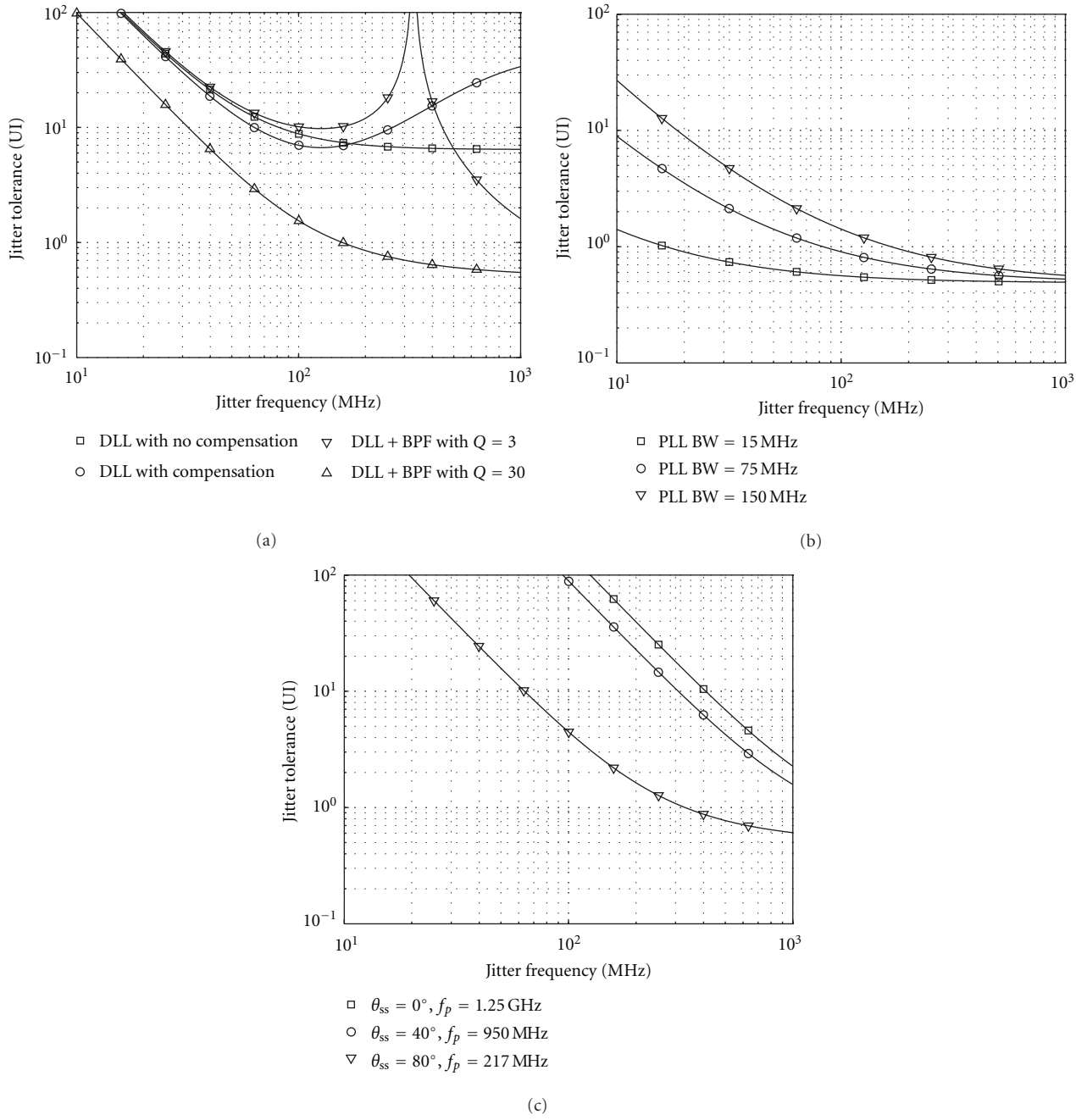


Figure 21: Jitter tolerance assuming no clock-to-data skew for (a) DLL both compensated and uncompensated along with BPF of varying Q, (b) PLL for loop bandwidth of 15, 75, and 150 MHz, (c) ILO with $K = 0.5$ at varying de-skew settings.

order to differentiate the receiver structures' performance for systems which approach this ideal case.

An ideal DLL with a unity jitter transfer function over all frequencies would hypothetically provide infinite jitter tolerance over all frequencies for this zero-skew case. While peaking in real DLLs degrades this ideal performance, the system still tolerates multiple UIs of jitter at high-frequency, as shown in Figure 21(a). Compensating the DLL with an additional pole can further improve the jitter tolerance at high frequencies. Leveraging band-pass filtering provides the potential for jitter filtering with DLL-PI de-skew. In the ideal zero skew case, a low-Q band-pass filter provides improved DLL compensation in the 100–400 MHz range at the cost of increased amounts of in-phase correlated jitter being filtered at higher frequencies. If the BPF Q is increased to a high value, the jitter tolerance degrades over all frequencies due to an increased amount of this correlated jitter filtering.

The limited PLL bandwidth causes the PLL-PI de-skew to have less jitter tolerance relative to the DLL-PI architecture, as Figure 21(b) shows the jitter tolerance falling below 1 UI

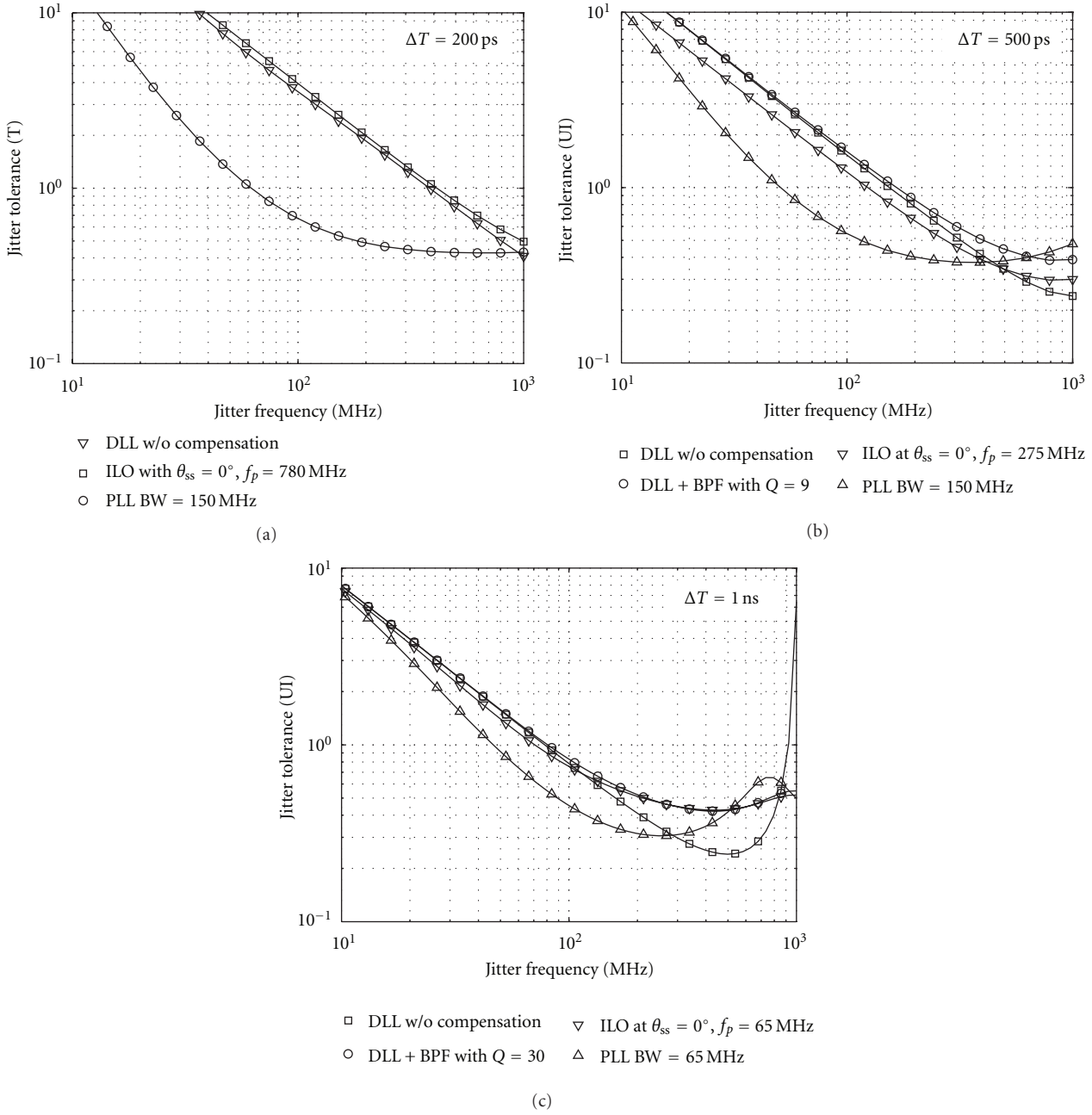


Figure 22: Jitter tolerance for varying clock-to-data skew (a) at 200 ps (b) 500 ps (c) 1 ns. Performance shown for DLL without any compensation techniques, ILO with K varying from 0.312 to 0.026, and PLL with BW varied from a maximum of 150 MHz to 65 MHz.

near 200 MHz for the maximum 150 MHz PLL bandwidth. Performance degrades further as PLL bandwidth is reduced due to increased filtering of in-phase correlated jitter and also additional VCO jitter accumulation subtracting from the overall timing margin.

Figure 21(c) shows a similar trend for the ILO architecture, with the maximum 1.25 GHz tracking bandwidth at a 0° de-skew setting achieving more than 10 UI jitter tolerance at 200 MHz. Changing the ILO free-running frequency to obtain an output phase shift results in a lower jitter tracking bandwidth and increased oscillator jitter accumulation. Note that, for moderate output phase shifts, the overall high tracking bandwidth and lack of jitter peaking still allows the ILO to outperform the other de-skew architectures up to near 500 MHz. However, if an extreme phase shift is required, the dramatically reduced jitter tracking bandwidth and oscillator jitter accumulation causes the ILO system to have worse jitter tolerance than the DLL-PI architecture for frequencies greater than 50 MHz.

5.2. Low Clock-to-Data Skew (2 UI). Introducing a low skew value of 200 ps degrades the jitter tolerance performance

for all the presented de-skew architectures, as shown in Figure 22(a). DLL-PI without band-pass filtering and ILO de-skew display similar performance and can tolerate 2 UI of jitter near 200 MHz. Here, the ILO bandwidth is reduced to 700 MHz to compensate for the 200 ps skew. The PLL-PI de-skew, with maximum bandwidth setting of 150 MHz, displays the lowest jitter tolerance at this low skew value due to excessive filtering of in-phase correlated jitter.

5.3. Mid Clock-to-Data Skew (5 UI). An increased amount of jitter filtering benefits the jitter tolerance performance for systems with a moderate skew value of 500 ps. A DLL-PI-based de-skew system that includes a BPF with a Q of 9, resulting in an overall jitter tracking bandwidth of 275 MHz, provides the best performance at 200 MHz with jitter tolerance close to 0.9 UI. The performance is similar for the DLL-PI system without band-pass filtering at low frequencies, but begins to diverge above 100 MHz due to inadequate filtering of out-of-phase correlated jitter. ILO de-skew jitter tolerance is slightly degraded relative to the DLL system due to increased oscillator accumulated jitter associated with the reduced jitter tracking bandwidth. While the 150 MHz PLL-PI system still performs the worst at low and moderate frequencies, it does offer superior jitter tolerance relative to the stand-alone DLL and ILO systems for jitter frequencies above 400 MHz due to increased filtering of this out-of-phase correlated jitter.

5.4. High Clock-to-Data Skew (10 UI). As skew is increased to 1 ns, the PLL-PI-based de-skew with a 65 MHz bandwidth provides more comparable performance to the other de-skew architectures. While peaking in the PLL transfer function degrades the jitter tolerance at the lower frequencies, the PLL-PI system achieves 0.3 UI jitter tolerance at 200 MHz and superior performance relative to the stand-alone DLL for jitter frequencies above 300 MHz. At 200 MHz, the ILO de-skew with 65 MHz bandwidth and the BPF-DLL-PI with 83 MHz bandwidth achieve the best jitter tolerance of 0.5 UI. Here an active-inductor-based band-pass filter is assumed to achieve the Q of 30 required for the 83 MHz bandwidth.

6. Conclusion

This work presented an analysis of key channel effects that impact the jitter performance of high-speed source synchronous links. Skew between the clock and data signals degrades source synchronous system timing margins, motivating the use of receiver clock circuits that provide filtering of high-frequency jitter components that would otherwise combine out-of-phase and increase differential jitter. High-frequency channel loss characteristics influence system forwarded clock frequency choice, as the differences in the loss slope impacts the amount of high-pass jitter amplification.

Also discussed was the tradeoffs in complexity and jitter tracking properties of common receiver de-skew circuits, along with how band-pass filtering can be leveraged to provide additional jitter filtering. Jitter tolerance modeling indicates that an all-pass DLL-PI or high jitter tracking bandwidth ILO structure performs best in low skew systems. For systems with large amounts of skew, PLL-PI de-skew becomes competitive with low-bandwidth ILOs and DLL-PI systems which leverage additional clock band-pass filtering. Overall, ILO-based de-skew holds the potential for high jitter tolerance over wide skew ranges at a low complexity level relative to other receiver clock topologies.

Acknowledgments

The authors would like to thank Yohan Frans, Brian Leibowitz, Jihong Ren, Sam Chang, and Masum Hossein of Rambus and Younghoon Song of Texas A&M University for advice and comments on this work. This work was supported by SRC Grant 1836.060.

References

[1] J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, no. 3, Article ID 034008, 2008.

[2] G. Delagi, "Harnessing technology to advance the next-generation mobile user-experience," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '10). Digest of Technical Papers*, pp. 18–24, February 2010.

[3] E. Prete, D. Scheideler, and A. Sanders, "A 100mW 9.6Gb/s transceiver in 90nm CMOS for next generation memory interfaces," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '06). Digest of Technical Papers*, pp. 253–262, San Francisco, Calif, USA, February 2006.

[4] S. Sawant, U. Desai, G. Shamanna et al., "A 32nm Westmere-EX Xeon® enterprise processor," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '10). Digest of Technical Papers*, pp. 74–75, February 2010.

[5] B. Casper and F. O'Mahony, "Clocking analysis, implementation and measurement techniques for high-speed data links—a tutorial," *IEEE Transactions on Circuits and Systems I*, vol. 56, no. 1, pp. 17–39, 2009.

[6] J. B. Lee, K. H. Kim, C. Yoo et al., "Digitally-controlled DLL and I/O circuits for 500Mb/s/pin ×16 DDR SDRAM," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '01). Digest of Technical Papers*, pp. 68–69, February 2001.

[7] J. Zerbe, B. Daly, L. Luo et al., "A 5 Gb/s link with matched source synchronous and common-mode clocking techniques," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 974–985, 2011.

[8] G. Balamurugan and N. Shanbhag, "Modeling and mitigation of jitter in high-speed source-synchronous inter-chip communication systems," in *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*, vol. 2, no. 2, pp. 1681–1687, November 2003.

[9] S. Chaudhuri, W. Anderson, J. Bryan, J. McCall, and S. Dabral, "Jitter amplification characterization of passive clock channels at 6.4 and 9.6 Gb/s," in *Proceedings of the 14th Topical Meeting on Electrical Performance of Electronic Packaging and Systems (EPEP '06)*, pp. 21–24, October 2006.

[10] J. Ren, D. Oh, and S. Chang, "High-speed I/O jitter modeling methodologies," in *Proceedings of the 19th IEEE Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS '10)*, pp. 25–27, October 2010.

[11] C. Madden, S. Chang, D. Oh, and C. Yuan, "Jitter amplification considerations for PCB clock channel design," in *Proceedings of the 16th IEEE Topical Meeting on Electrical Performance of Electronic Packaging (EPEP '07)*, pp. 135–138, October 2007.

[12] W. Beyene, "Modeling and analysis techniques of jitter enhancement across high-speed interconnect systems," in *Proceedings of the 16th IEEE Topical Meeting on Electrical Performance of Electronic Packaging (EPEP '07)*, pp. 29–32, October 2007.

[13] M. J. E. Lee, W. J. Dally, T. Greer et al., "Jitter transfer characteristics of delay-locked loops—theories and design techniques," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 4, pp. 614–621, 2003.

[14] G. Y. Wei, J. Kim, D. Liu, S. Sidiropoulos, and M. A. Horowitz, "Variable-frequency parallel I/O interface with adaptive power-supply regulation," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1600–1610, 2000.

[15] T. M. Hollis and D. J. Comer, "Bandpass filtering of high-speed forwarded clocks," *Analog Integrated Circuits and Signal Processing*, vol. 54, no. 3, pp. 171–186, 2008.

[16] Y. C. Bae and G. Y. Wei, "A mixed PLL/DLL architecture for low jitter clock generation," in *Proceedings of the IEEE International Symposium on Cirquits and Systems (ISCAS '04)*, pp. V-788–V-791, May 2004.

[17] F. O'Mahony, G. Balamurugan, J. E. Jaussi et al., "The future of electrical I/O for microprocessors," in *Proceedings of the IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT '09)*, pp. 31–34, April 2009.

[18] M. Mansuri and C. K. K. Yang, "Jitter optimization based on phase-locked loop design parameters," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1375–1382, 2002.

[19] Adler, "A study of locking phenomena in oscillators," *Proceedings of the IEEE*, vol. 61, no. 10, pp. 1380–1385, 1973.

[20] L. J. Paciorek, "Injection locking of oscillators," *Proceedings of the IEEE*, vol. 53, no. 11, pp. 1723–1728, 1965.

[21] M. Hossain and A. C. Carusone, "CMOS oscillators for clock distribution and injection-locked deskew," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 8, Article ID 5173768, pp. 2138–2153, 2009.

[22] K. Hu, T. Jiang, J. Wang, F. O'Mahony, and P. Y. Chiang, "A 0.6 mW/Gb/s, 6.47.2 Gb/s serial link receiver using local injection-locked ring oscillators in 90 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, Article ID 5437482, pp. 899–908, 2010.

[23] V. Kratyuk, I. Vytyaz, U. K. Moon, and K. Mayaram, "Analysis of supply and ground noise sensitivity in ring and LC oscillators," in *Proceedings of the International Symposium on Circuits and Systems (ISCAS '05)*, pp. 5986–5989, May 2005.

[24] J. Poulton, R. Palmer, A. M. Fuller et al., "A 14-mW 6.25-Gb/s transceiver in 90-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2745–2757, 2007.

[25] H. Xiao and R. Schaumann, "A 5.4-GHz high-Q tunable active-inductor bandpass filter in standard digital CMOS technology," *Analog Integrated Circuits and Signal Processing*, vol. 51, no. 1, pp. 1–9, 2007.

[26] M. Hossain and A. C. Carusone, "A 6.8mW 7.4Gb/s clock-forwarded receiver with up to 300MHz jitter tracking in 65nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 6, pp. 1336–1348, 2011.

[27] B. Razavi, *Design of Integrated Circuits for Optical Communications*, McGraw-Hill, Boston, Mass, USA, 2002.

[28] S. Hall and H. Heck, *Advanced Signal Integrity for High-Speed Digital Designs*, John Wiley & Sons, Hoboken, NJ, USA, 2009.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 936712, 23 pages
doi:10.1155/2011/936712

Research Article

VLSI Implementation of a Distributed Algorithm for Fault-Tolerant Clock Generation

Gottfried Fuchs and Andreas Steininger

Embedded Computing Systems Group (E182/2), Technische Universität Wien, Treitlstraße 3, 1040 Vienna, Austria

Correspondence should be addressed to Andreas Steininger, steininger@ecs.tuwien.ac.at

Received 15 June 2011; Accepted 12 August 2011

Academic Editor: Jae-Yoon Sim

We present a novel approach for the on-chip generation of a fault-tolerant clock. Our method is based on the hardware implementation of a tick synchronization algorithm from the distributed systems community. We discuss the selection of an appropriate algorithm, present the refinement steps necessary to facilitate its efficient mapping to hardware, and elaborate on the key challenges we had to overcome in our actual ASIC implementation. Our measurement results confirm that the approach is indeed capable of creating a globally synchronized clock in a distributed fashion that is tolerant to a (configurable) number of arbitrary faults. This property facilitates eliminating the clock as a single point of failure. Our solution is based on purely asynchronous design, obviating the need for crystal oscillators. It is capable of adapting to parameter variations as well as changes in temperature and power supply–properties that are considered highly desirable for future technology nodes.

1. Introduction

Throughout the last decades progress in VLSI technology has constantly fueled an incredible advancement in complexity, speed, functionality, and power efficiency of digital circuits [1]. This trend has always created new opportunities, but at the same time has been accompanied by various challenges for the design of these circuits [2]. Contemporary chip design seems to be dominated by the following issues.

(i) *Fault Tolerance*. It is commonly agreed that technology nodes smaller than 65 nm tend to become increasingly vulnerable to single-event upsets, due to their small critical charges and the low voltage swing [3–5]. As a consequence the need for fault tolerance emerges, even for non-safety-critical applications.

(ii) *Power Efficiency*. With a growing number of transistors per unit area, the power density is increasing, even in spite of technological progress. This leads to problems with power distribution and with heat dissipation.

(iii) *Variation Tolerance*. The fabrication tolerances of new technology nodes lead to uncertainties in the timing behavior, power consumption, and so forth, where traditional corner-case design is too pessimistic [6]. Therefore design techniques are sought that are capable of sustaining reliable operation even under these variations.

In the light of these substantial challenges even one of the foundations of digital design is being questioned, namely, the globally synchronous paradigm. While the abstraction of the chip being a perfect isochronous region facilitates an efficient design, retaining a reasonable synchrony all over a large and complex chip with a sub-nanosecond precision has become extremely cumbersome. To minimize the skew within the clock network, not only sophisticated geometries are applied, but in addition large numbers of clock buffers and deskewing units have to be placed at carefully chosen positions [7–9]. As a result an appreciable share of the power budget goes into the clock distribution network [10, 11]. This stands in contrast with the above stated requirement for power efficiency. Recently, parallelism is being introduced to increase processing power while maintaining clock speed. This trend is accompanied by new communication schemes, so called networks on chip. As a recent example, the Godson-3B [12, 13] comprises two distinct groups of four tightly coupled cores per group.

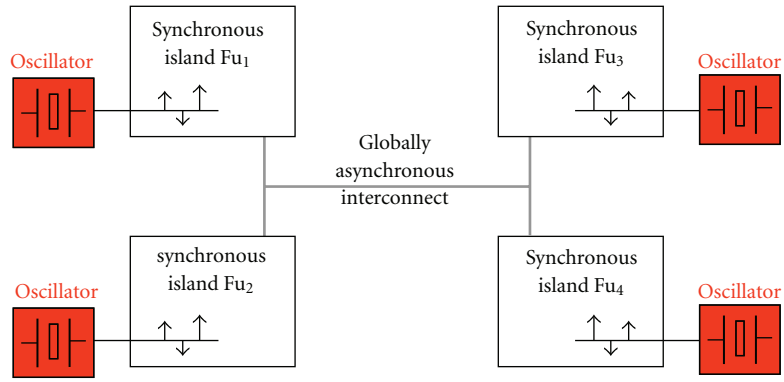


Figure 1: Globally asynchronous locally synchronous (GALS) architecture.

Item 3 on the above list, namely the increasing fabrication tolerances, unfortunately forms another obstacle for maintaining an isochronous region all over the chip: The synchronous design paradigm rests upon the intimate knowledge of the circuit timing that becomes blurred for newer technologies.

And finally the globally synchronous clocking approach proves to be problematic with respect to fault tolerance as well. Although synchrony is an important foundation for many fault-tolerance schemes (such as TMR or duplication and comparison), the single, central clock source forms a single point of failure even in such a replicated architecture. This issue has long been neglected, as the clock network is considered robust due to its strong drivers and its relatively high capacitance. Recently, however, concerns have been raised about the vulnerability of the clock nets, and specifically clock repeaters, as well [14].

In addition to all these challenges, however, newer technologies also introduce new possibilities as well. The architectures found in systems on chip have very much in common with traditional distributed architectures. In the latter a globally synchronous clock source has hardly ever been employed—their components are rather loosely coupled, employing several local clock sources that are then synchronized on a higher level of abstraction by distributed algorithms, if desired. In this paper we will review options for generating a fault-tolerant clocking scheme that is feasible for modern technologies and architectures, and we will present a novel scheme for fault-tolerant clock generation that is based on a distributed algorithm and thus exploits the typical architecture found in systems on chip (SoC). In addition to its superior fault tolerance the proposed scheme is extremely robust against process variations.

2. Related Work

As motivated above traditional, globally synchronous design may not be able to meet all upcoming challenges to future computer architectures. Subsequently, several promising alternatives will be surveyed that have been proposed in the literature.

2.1. Globally Asynchronous, Locally Synchronous. The Globally Asynchronous Locally Synchronous (GALS) approach [15] is based on the generic architecture depicted in Figure 1. Small (local) synchronous islands implement functions (subtasks) of the whole system. Each local island's function is executed using the traditional synchronous design style, whereas global interaction follows an asynchronous communication style. Each island is provided with its own oscillator as clock source for the locally synchronous computations. Compared to the high effort for global clocking of a purely synchronous system, in local synchronous islands skew optimization of the clock signal is much easier to attain. Although GALS simplifies the clock distribution to some extent, some other issues are still left. The need for a dedicated oscillator for each synchronous island adds additional components to the system, which clearly decreases reliability. The often used quartz oscillators are sensitive to, for example, vibration, temperature, shock, and so forth, while on-chip RC oscillators are known for their strong dependence on operating conditions like temperature and supply voltage, which leads to frequency changes in the range of 10 to 30%.

Beyond that, if compared to a synchronous system, the GALS concept has two major fundamental disadvantages. Firstly, a GALS design does not implicitly provide the convenient systemwide notion of time which most hardware designers are used to and design tools are made for. All clock sources are free running—the local clocks may drift arbitrarily apart from each other. Communication leaving a local island's clock domain introduces the need for synchronization. The fact that the interface between globally asynchronous communication and locally synchronous data processing has to incorporate some sort of synchronizer circuits poses the second, probably the most severe, disadvantage of GALS. Unfortunately, synchronizing clock domains with arbitrary, possibly changing, relation to each other, cannot be solved in a safe way. Metastability issues might even upset the synchronizer circuits [16] and can only be made more unlikely by adding further synchronizer stages. Taking parameter variations and clock jitter into

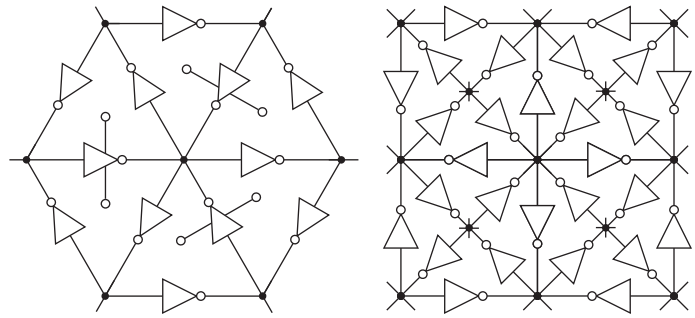


Figure 2: Interconnected ring oscillator architectures.

account synchronizers have to be designed very conservatively, thus introducing significant performance penalties into the asynchronous/synchronous interfaces.

Recent GALS implementations incorporate stoppable (plausible) and/or stretchable clocks [17, 18] to reduce performance loss at the clock domain interfaces. This, however, comes at the price of a reduction of clock accurracy and stability.

2.2. Interconnected Rings and Oscillators. This concept proposed by Maza and Aranda in [19, 20] presents an alternative approach for generating and distributing GHz clocks. The design relies on the self-oscillation property when interconnecting an odd number of inverters in a ring topology (shown in Figure 2) and achieves high clock frequencies due to its simplicity. Inverter and buffer placement of the proposed architecture determines wiring costs (in terms of wire length), speed, and skew of the generated clocks. The design especially fits as on-chip clocking scheme for the previously introduced GALS systems. It can be seen as a refinement of the GALS RC-oscillator clocking. Due to the fact that all inverters of the clock generation scheme are interconnected directly (locally) or indirectly (globally, through some additional inverter stages) with each other, the local islands of a GALS system cannot arbitrarily desynchronize (at least in the fault-free case). This property severely eases synchronization within the GALS design since the synchronizers can take advantage of the fact that the local clocks are not entirely unrelated.

2.3. Distributed Clock Generator. The scheme of a distributed clock generator (DCG) introduced by Fairbanks and Moore [21, 22] represents a special form of asynchronous FIFO implementation for the purpose of on-chip generation and distribution of a synchronized clock. Similarly to the approach by Maza and Aranda, interconnected clock generation hardware is distributed in a grid all over the chip, but the locally generated clocks are generated at approximately the same instant having only small skew. Every DCG instance is interconnected with its four neighbors, and half of the DCG units are initialized with a clock token. Due to the asynchronous FIFO implementation of each DCG the so-called Charlie effect [22] ensures that clock tokens are passed over to neighboring nodes in a synchronous way, generating a chip wide synchronized clock signal (the Charlie effect describes the force that slows down a subsequent token within a FIFO if it is closing in on a previous one).

2.4. Purely Asynchronous Design. Asynchronous design styles [23] are considered a viable alternative for synchronous design in the future, specifically for application fields like low power [24] or high performance [25, 26]. With asynchronous design the burden of clock distribution can be entirely eliminated and the clock tree be substituted by far less timing critical local handshake signals. Parameter variations are much less problematic in the context of, for example, quasi delay-insensitive circuits [27] since, due to the indication principle, only performance but not the correct function is influenced by variations. Furthermore, the inherent robustness of asynchronous design styles allows to address the issue of increased failure rates in future VLSI technology [28, 29] to some extent. On the downside, the variety of existing asynchronous design styles and delay models distracts not only designers who are not expert in the field, but also EDA companies whose design and verification tools are crucial enablers for a general acceptance of the asynchronous design paradigm. Nonnegligible area overhead, higher design complexity, and the intricate circuit testing issues add to these problems. Even though a good robustness is inherent to asynchronous designs, the "wait for all" paradigm implied by the indication principle prevents established system-level fault-tolerance techniques, like triple modular redundancy (TMR), from being directly applied to asynchronous systems [30].

2.5. Discussion. In the presented approaches the incorporation of fault tolerance as well as the robustness required for coping with unexpected faults as well as parameter variations is mostly lacking. GALS in general has issues with interfacing multiple uncorrelated clock domains, and its lack of a global time severely complicates the design process (which is also the case for purely asynchronous approaches). The interconnected rings and oscillators as well as the distributed clock generator approach are not able to cope with failures. To be able to tolerate arbitrary failures in a clock synchronization process, theory shows that almost

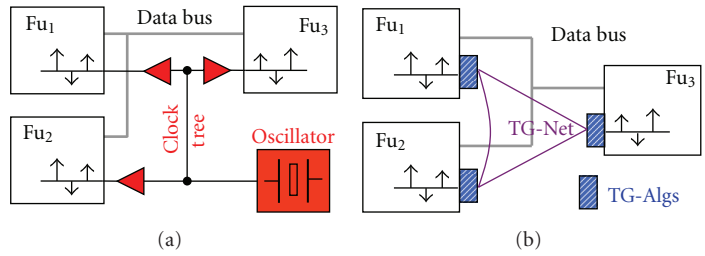


Figure 3: Replacing synchronous clocking by fault-tolerant distributed tick generation.

fully connected networks are needed [31] which is clearly not fulfilled by those two approaches. Therefore, a transient fault might lead to major clock deviation, overclocking phenomena or could even stop the whole clock generation process.

The work described in this paper focuses on the development of a robust clocking scheme for future dependable systems. Especially in safety- and mission-critical environments like in the automotive and aerospace domain robustness against arbitrary faults is of utmost importance. Similar to GALS, our approach provides strong local synchrony. In contrast to GALS, however, a fault-tolerant time base is maintained on the global level as well (albeit with slightly relaxed synchrony assumptions).

3. The DARTS Concept

Figure 3 illustrates the key principle of our approach: we replace the central clock source (crystal oscillator) by a set of tick generation units. Each of these units implements an instance of the same distributed algorithm in hardware (therefore we call them *TG-Algs* further on). This algorithm is based on communication between the individual TG-Algs, through which the TG-Algs mutually stimulate each other, thus creating an oscillation that can be viewed as a globally synchronized clock. Through the choice of an appropriate algorithm and by means of a careful implementation, the TG-Algs' local perceptions of this global clock remain within a bounded precision, even if the communication network, called *TG-Net* in Figure 3, introduces considerable delays and skew. Each of the TG-Algs is attached to one or more functional units of the SoC for which it provides a local clock that is in synchrony with all other local clocks generated by the other TG-Algs for their respective functional units (Fu_i). Based on these local clocks the functional units can internally be operated according to the traditional synchronous design paradigm, which is desirable and unproblematic as long as their local extent is limited.

From a high-level perspective distributed algorithms suitable for tolerating multiple Byzantine faults can be incorporated to get a robust clocking scheme. If we use such an algorithm for our TG-Algs, our clock generation will stay operational even if some of the TG-Algs and/or links of the TG-Net should fail arbitrarily. Clearly, the functional units connected to the failed TG-Algs will no more be supplied with a proper clock, but this can be compensated by their appropriate replication along with connection of the replica to different TG-Algs. Note that this solves a notorious problem in classical fault-tolerant VLSI systems: fault tolerance is considerably easier to implement under the assumption of global synchrony, while the establishment of global synchrony by means of a central clock introduces a single point of failure. The key advantage of our approach is to provide a globally synchronized clock that is at the same time fault tolerant. On this foundation it is straightforward to build a fault-tolerant architecture on the level of functional modules. Furthermore, the global synchrony allows metastability-free communication between the functional units without the need for synchronizers [32]. Another advantage of our approach is its insensitivity to delays in the communication links as well as to the TG-Alg's propagation delays; as we will show later, even considerable tolerances, drift and jitter, can be accommodated. This relieves the designer from using strong drivers, which in turn increases power efficiency.

We have designed, formally proven, simulated, implemented, and evaluated the proposed concept in the course of the research project *Distributed Algorithms for Robust Tick-Synchronization (DARTS)*. While an in-depth analysis of the formal aspects of the DARTS approach can be found in [33], this paper will be more concerned with the implementation-related issues of DARTS. In particular, throughout the remainder of this section we will investigate the foundations for our concept, namely, the selection of a suitable algorithm and the constraints that have to be considered when implementing that abstract algorithm as VLSI chip design.

3.1. Finding a Suitable Distributed Algorithm. Distributed computing research provides the required algorithms for fault-tolerant generation of synchronized clock ticks. The class of distributed algorithms considered in the DARTS approach is based on message passing, with a set of particular properties to meet the requirements for tick generation. In short these characteristics of tick-generation algorithms are as follows.

(i) The algorithm consists of a set of rules which are evaluated whenever a message arrives at a node. These rules conditionally update the respective node's local memory and trigger the transmission of messages to other nodes.

```
1: variables
2:    k: integer :=0
3: end variables
4: if C^{k-1}(t) = kP then //ready to start C^k
5:    → broadcast(TICK(k))
6: end if
7: if accepted the message(TICK(k)) then //according to a selection/voting function
8:    → C^k(t) := kP + α
9: end if
```

Algorithm 1: Nonauthenticated algorithm for clock synchronization at node p [36].

(ii) To implement a tick generation approach, the class of distributed algorithms is restricted to those that send only messages containing ascending natural numbers, that is, it is demanded that every node p sends messages $\langle 0\rangle, \langle 1\rangle, \langle 2\rangle, \ldots$ in the given order during its executions. When mapping tick generation to hardware the natural numbers of messages TICK(k) mod 2 can be seen as discrete *up* and *down* transitions of a hardware clock.

(iii) To achieve synchronization among all nonfaulty nodes of a distributed system, a tick generation algorithm has to solve the synchronization problem following Lamport's definition [34, 35] if synchronization precision π and accuracy shall hold.

(iv) Furthermore, the algorithm is called fault-tolerant if it maintains the conditions described above even in the presence of faults.

An algorithm presented by Srikanth and Toueg in [36] fulfils all these criteria. It comprises two parts. According to Algorithm 1, the so-called nonauthenticated clock synchronization part, node p broadcasts message TICK(k) as soon as its clock counter $C^{k-1}(t)$ reaches a threshold value indicating that the next tick has to be issued. The internal clock counter is directly driven by a local oscillator. The round-based threshold value is given by kP, where P denotes the predefined resynchronization interval.

When reaching the threshold count node p is ready to change from round $k-1$ to round k. However, in order to keep the system synchronized, this transition must be coordinated with the other nodes. This is accomplished by resynchronizing the clock counter $C^k(t)$ at the instant an "accepted TICK(k)" message is received from another node (actually, $C^k(t)$ is adjusted to $kP+\alpha$, where α denotes a constant ensuring that the clock always steps forward in time, see Figure 4).

The function responsible for generating these "accept TICK(k)" messages forms the second part of the algorithm and is shown in Algorithm 2. It comprises three parallel rules for message processing. In response to the reception of particular TICK(k) messages from at least $f+1$ distinct nodes, either via *init* or *echo* messages, each node relays an *echo* TICK(k) message to all other nodes (Relay rules). The actual generation of an "acceptance event" for advancing the clock, however, requires the reception of at least $2f+1$ distinct

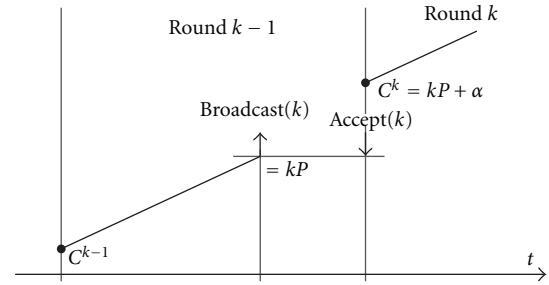


Figure 4: Nonauthenticated broadcast execution at node p.

echo TICK(k) messages (Accept Rule). It has been shown by Srikanth and Toueg that in a system of $n \geq 3f+1$ nodes Algorithm 1 in cooperation with the consistent broadcast primitive of Algorithm 2 solves the clock synchronization problem, even in the presence of up to f Byzantine faulty nodes if the conditions hold that

(i) the local clocks' maximum drift rate is known and bounded by ρ,

(ii) message end-to-end delays are within a certain known bound of $[d, d+\varepsilon]$,

(iii) two specific timing assumptions are ensured by properly chosen values for P and α.

Let us reconsider the initial motivation for taking a closer look at tick generation algorithms, namely, to get synchronized clocks without the need for local clock sources. The nonauthenticated algorithm for clock synchronization, presented above in Algorithm 1, still requires a local clock source at each node to supply the local clock counter. Fortunately, some modifications of Algorithm 1 yield a solution which no longer requires a local counter and also removes the distinction of *init* and *echo* events, which largely eases message handling. This algorithm is shown below (Algorithm 3).

It is derived from the algorithm originally proposed by Widder and Schmid [37], however, with the following important change: in order to facilitate an implementation in hardware (see Section 3.2) it no longer relies on infinite TICK(k) numbers. Under the additional constraint that every TICK(k) message is only sent [**once**] this simplification can be accomplished without spoiling the analysis of [37].

```
1: variables
2:    k: integer :=0
3: end variables
4: for each correct process do
5:    if received(init,TICK(k)) from at least f + 1 distinct nodes then //Init Relay Rule
6:       → send(echo,TICK(k)) to all
7:    end if
8:    if received(echo,TICK(k)) from at least f + 1 distinct nodes then //Echo Relay
9:       → send(echo,TICK(k)) to all
10:   end if
11:  if received(echo,TICK(k)) from at least 2f + 1 distinct nodes then //Accept Rule
12:     → accept(TICK(k))
13:  end if
14: end for
```

Algorithm 2: Acceptance function selecting valid clock ticks [36].

```
1: variables
2:    k: integer :=0
3: end variables
4: initially send TICK(0) to all [once]
5: if received TICK(ℓ) from at least f + 1 rem. processes with ℓ ≥ k then //Relay Rule
6:    send TICK(k),..., TICK(ℓ) to all [once]
      K := ℓ
7: end if
8: if received TICK(k) from at least 2f + 1 remote processes then //Increment Rule
9:    send TICK(k + 1) to all [once]
      K := k + 1
10: end if
```

Algorithm 3: Byzantine-tolerant tick generation suitable for bounded tick numbers.

With this algorithm the previously used assumption on message delays $[d, d+\varepsilon]$ can be weakened to the one that for any two messages in transit m_1, m_2 it has to hold that

$$\frac{\delta(m_1)}{\delta(m_2)} \leq \Theta \tag{1}$$

with $\delta(m_1)$ and $\delta(m_2)$ being the respective message delays of m_1 and m_2, and Θ being constant. The analyses in [37] show that despite the presented substantial simplifications, Algorithm 3 still solves the clock synchronization problem—in this particular case, this is maintaining precision π as well as accuracy even in the presence of Byzantine faults. Algorithm 3 processes like this: the "Relay Rule" of a correct node fires as soon as TICK(ℓ) messages from at least $f + 1$ distinct nodes have been received—given that f is the maximum number of faults the system is supposed to tolerate, this ensures that at least one of these TICK(ℓ) messages has been issued by a correct node. Notice that in the case of triggering the "Relay Rule" the node does not immediately set its local clock k to ℓ since this would lead to skipping some values of k if the respective node is lagging more than one tick behind. The strategy followed in Algorithm 3 explicitly ensures that all messages TICK(k),..., TICK(ℓ) are issued when catching up with faster nodes, resulting in a continuous progression of the clock without potentially troublesome leaping effects. Especially when recalling the targeted application of clocking synchronous circuits, skipped clock ticks might result in inconsistent state progression over different functional modules.

As a result we now have a distributed algorithm available that is able to generate an ascending sequence of clock ticks k in a fault-tolerant manner without relying on a local clock source.

3.2. Hardware Implementation Challenges. So far our approach has been to use a distributed tick generation algorithm for generating an ascending sequence of tick numbers, and the mapping to rising and falling edges in the hardware implementation simply implies a mod 2 operation. The above algorithm provides all required features; however, substantial implementation-related problems originate in the fact that this algorithm (like virtually all other distributed algorithms) has been designed on a very high level of abstraction, at best with a software implementation in mind.

In general, a tick generation algorithm operates on unbounded natural numbers, whereas a hardware clock signal simply toggles between the two logic values *high* and *low* (Figure 5). For our ultimate purpose of clocking our

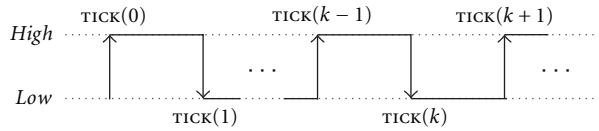


FIGURE 5: Hardware clock signal versus tick numbers.

functional units we do not need the history information contained in the individual tick numbers, and we cannot afford to convey it. With clock frequencies ranging into hundreds of MHz or even some GHz the value of k rapidly reaches gigantic dimensions, and in fact its unbounded nature prohibits any concrete implementation. Moreover, each value has to be repeatedly transferred at this high frequency as part of the tick generation process, which obviously causes an excessive data rate.

At the same time we cannot completely get rid of k, since the abstract operational principle of tick generation algorithms relies on counting up this integer tick number. In order to facilitate a hardware implementation, we have to change the algorithm such that it can accommodate bounded values for the TICK(k) numbers and the resulting wrap-around effects in their numerical representation, that is, after sending the largest value of k in the chosen integer representation, the smallest one, for example, TICK(*0*) follows. From the hardware point of view the bound on k should be as low as possible. In practice, however, this minimization of k is limited by the boundary conditions that

> Alg-R1: it must be ensured that no TICK(k) messages of different wrap-around phases can interfere with each other, and a bound on the maximum offset of any two clocks holds;
>
> Alg-R2: the two parallel rules (Increment- and Relay-Rule) executed on a node p never generate and sequentially transmit the same TICK(k) message.

To accommodate for bounded values of k, the algorithm presented by Widder and Schmid [37] was augmented by the requirement that every TICK(k) message is only sent [**once**] regardless of the fact that multiple rules might be eligible to generate this particular TICK(k) message. This change is already reflected in Algorithm 3.

Based on the refinements of our original algorithm presented in this chapter, and considering the boundary conditions we have identified, we can concentrate our efforts on finding a suitable mapping of algorithmic statements to hardware building blocks in the next chapter.

4. Hardware Implemented Fault-Tolerant Tick-Generation

4.1. Hardware-Related Requirements. Even with the modified tick synchronization algorithm by Widder and Schmid (Algorithm 3) there are still several difficulties when attempting to map the software-based (high-level) tick generation algorithm to the restrictions of hardware design. Most of them are not due to algorithm-related requirements, but rather originate from the need to find a fast and area-efficient projection of the algorithm to hardware, since these issues are by no means considered in the high-level description of the algorithm so far. In the following we will give a list of these challenges.

> HW-R1: *Tick Generation Network*. Integer TICK(k) messages have to be conveyed in a way to keep the clock network as simple as possible without compromising clock speed. Therefore, strategies having more than a single wire per clock signal are assumed too costly.
>
> HW-R2: *Tick Messages*: Simple TICK(k) messages have to be used to enable highest possible speed, while still operating on a single rail per clock signal—clock transitions (*up/down*) on a single signal rail, as depicted in Figure 5, seem to be the only viable implementation option.
>
> HW-R3: *Unique Sender Identification*: The algorithm is based on the assumption that the receiver of a TICK(k) message can uniquely identify the respective sender. In the light of HW-R2, appending a sender ID is not feasible, hence we need a point-to-point connection between any two nodes, that is, a fully meshed network. This in turn confirms the claim for a lightweight interconnection posed in HW-R1.
>
> HW-R4: *Asynchronous Design*: Since the TG-Algs' task is to generate a clock, they do not have a clock available for their own operation in the first place. (In principle, the provision of a local clock to each TG-Alg would be possible, but it would counteract the original intention of the approach, namely generating a clock, and it would suffer metastability problems at the clock domain boundaries that would inevitably emerge then.) As a consequence their implementation needs to follow an asynchronous design paradigm. From the available approaches the (quasi) delay insensitive one is most attractive, since it does not make assumptions on the individual path delays, thus increasing the desired robustness.
>
> HW-R5: *Atomicity of Actions*: For all distributed computing models that the authors are aware of, atomic computing steps at the level of a single node are assumed. However, this abstraction cannot be adopted when implementing an algorithm directly in (asynchronous) hardware, where computations are performed by numerous concurrently operating digital logic gates. The most challenging part in our case is given by the parallel processing of the two algorithmic rules ("Relay Rule" and "Increment Rule" of Algorithm 3) in conjunction with concurrently arriving TICK() messages. To handle requirement Alg-R1, explicit synchronization of local computations is needed.
>
> HW-R6: *Fast Operation*: The theoretical analysis of the algorithm [38] confirms the intuition that the

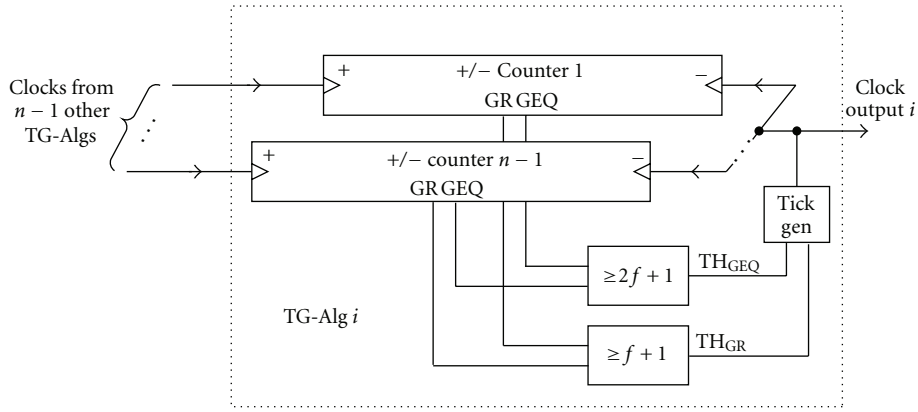


Figure 6: Tick generation architecture handling relative tick numbers.

oscillation frequency delivered by the DARTS scheme as well as the attainable precision is determined essentially by the round-trip times of the transitions, that is, the time from the generation of a particular transition until the generation of the next one as a result of its reception by all other nodes. Consequently, in order to obtain fast operation, the hardware implementation shall minimize the number of gate delays in these paths.

The most demanding design requirement is certainly HW-R4, the need for an entirely asynchronous hardware implementation: this restriction completely rules out the employment of the well-established synchronous design methods. Systematic asynchronous design styles (e.g., [39]) follow a handshake-based flow control to ensure that no old data interferes with new one—this provides *interlocking* between subsequent data waves. In this context it is important that a transition at a gate input causes an effect at the gate output, before the next input transition is allowed to occur. This so-called *indication principle* is crucial for handling concurrency (without having to introduce timing assumptions), and it can be maintained by handshaking, as long as the function of the gate is such that every input transition actually causes an output transition, that is, no input is being masked or disregarded.

However, in the context of fault-tolerant design a fundamental problem arises with this indication principle: if a module is waiting until *all* REQs have properly arrived before issuing an ACK and proceeding with its operation (as it is implied by the handshake procedure), this would allow a single faulty unit to inhibit any further processing. Hence, a strategy has to be followed where processing is halted only until an algorithm-dependent *threshold* of REQs has been reached. While such an approach now enables the incorporation of fault tolerance, it necessarily breaks the implementation's REQ/ACK feedback loops for the slowest paths, thus inhibiting their indication. Without additional measures or constraints such open loops tend to run out of sync, endangering the correct operation of the whole system. In order to obtain a fault-tolerant asynchronous design the traditional closed REQ/ACK control loops have to be augmented by explicit timing constraints, this way supporting an interlocking scheme for consecutive data waves for all, even the slowest paths. We will elaborate on these timing constraints later on. For the moment we will just summarize this insight in a further requirement.

HW-R7: *Timing Requirements*: In the absence of a global clock we are forced to apply the principles of asynchronous design, according to which all activities must be involved in a REQ/ACK handshake cycle. Due to requirement HW-R1 we do not want to extend this handshaking principle to the TG-Net. Furthermore, fault tolerance techniques like a threshold function essentially contradict the "wait-for all" paradigm implied by the handshaking. For these reasons handshaking shall not be employed in a completely consequent fashion, and appropriate timing conditions shall be elaborated to constrain the implementation in such a way that proper operation is still ensured where the handshake loops are broken.

4.2. Block Diagram of the Implementation. Figure 6 shows the basic architecture of a single TG-Alg's hardware design resulting from the above described specifications. The most notable peculiarity of this design is the dissemination strategy for TICK(k) messages. In accordance with HW-R2 no explicit tick numbers are transmitted over the TG-Net. Anonymous *up* and *down* signal transitions (zero-bit messages) are used instead of conveying integer values. From an abstract point of view this means that the sender just conveys "differential" information in the shape of a plain transition, while each receiver is equipped with a counter to integrate this information into the actual tick number k. In this way the message size can be reduced to the absolute minimum, while, however, every node now requires one separate counter per incoming link (i.e., $n-1$), further called "remote counter" (RC), in addition to the single one for maintaining its local count (LC).

A closer look at Algorithm 3 reveals that only a *relative* comparison between LC and RC is carried out, while their absolute value is not relevant: the relay rule checks whether the remote count l is greater than the local count k, while

the increment rule checks whether RC $\geq$ LC. Therefore, as shown in Figure 6, we can employ up/down counters, further referred to as +/− Counters to just maintain the *difference* between LC and RC rather than their absolute values. Note that this difference never becomes larger than the precision—this is guaranteed by the function of Algorithm 3. With this knowledge we can safely minimize the size of the difference counters without having to consider a potential wrap-around further on.

In accordance with Algorithm 3, in the next stage we join the results of the "greater" (GR) comparisons and those of the "greater or equal" (GEQ) comparisons from all inputs of a node and check whether the number of positive comparison results reaches the threshold of $f + 1$ implied by the relay rule or $2f + 1$ from the increment rule, respectively. This is done by the units called "threshold gates," in Figure 6 marked by their respective threshold values. The task of an m-of-n-threshold gate is simply to activate its output as soon as m or more of its n inputs are at logic HI.

At this point it is interesting to note that, although implied by Algorithm 3, we do not consider a self-reception in our hardware implementation, that is, a node only receives the tick messages sent by *all other* nodes, not the one produced by itself. The reason is that in practice the self-reception path is very likely to be much faster than all the other message delays. Now recall from Section 3.1 that the attainable precision largely depends on a constant Θ which is derived from the ratio of the fastest and slowest feedback delays within the tick generation scheme. Hence the imbalance caused by an extraordinarily fast self-reception path would unnecessarily increase Θ, thus degrading the attainable precision. As a consequence of omitting the self-reception path, the presented tick generation system has to comprise at least $3f + 2$ TG-Algs instead of the usually applied (lower bound of) at least $3f + 1$ nodes to attain the targeted degree of Byzantine fault tolerance.

The activation of a threshold gate's output corresponds to the firing of a rule in Algorithm 3, and as there are two concurrent rules in the algorithm, we have two threshold gates operating in parallel, with the appropriate thresholds of $f + 1$ and $2f + 1$. Finally, a *tick generation unit* ("Tick Gen" in Figure 6) takes care of properly combining these outputs into a tick that can be conveyed over the TG-Net. This is the place where the "once" statement from Algorithm 3 and requirement HW-R5 become important: care must be taken to issue only one tick per k value in spite of the concurrent operation of the two threshold gates.

The block diagram developed so far has been sufficiently accurate to allow for a thorough formal analysis of the TG-Alg design [33], yielding several implementation constraints. Nevertheless, from a hardware designer's point of view the abstraction of the TG-Alg design has to undergo further steps of detailing to enable a successful mapping to an ASIC manufacturing process. This will be the subject of the next section. (Our previous work [40] treated an FPGA prototype, whereas the main focus of this paper is on the presentation of the ASIC implementation and the evaluation of a DARTS cluster based on the manufactured ASICs.)

5. DARTS Implementation

In this section the TG-Alg's hardware implementation is presented in more detail. To this end Figure 7 presents a more accurate architecture of a single TG-Alg.

In the top part of the figure the +/− Counter module is shown, decomposed into a "Local Pipeline," a "Remote Pipeline," a "Diff Module," and a "Pipe Compare Signal Generator." Note that this counter module just corresponds to one remote input; overall $n - 1$ of these modules are required per node to handle all incoming links from the TG-Net. This fact is illustrated by the further counter modules shown in the bottom left part of the figure. The modules termed "Threshold Modules," on the bottom right in the figure, comprise the threshold gates and the "Tick Generation" unit.

To understand the proposed structure of the $\pm$ Counter recall from Section 4 that message transmission is differential, that is, we use *transitions* to convey the TICK(k) messages. As we do not know (and actually do not want to assume) any phase relationship between the local ticks and the remote ticks, TICK($\uparrow$) and TICK($\downarrow$) transitions may occur arbitrarily close to each other, hence introducing the potential for metastability. Instead of building a flip-flop-based counter, as one would usually do in synchronous logic, we decided to go for a consequent implementation in *transition logic*. In transition logic the expressiveness is limited to the causal ordering of events in a basically time-free system. However, to retain its delay insensitiveness the class of allowed circuit elements is fairly restricted. Permitted elementary units are for instance Muller C-Elements, inverters, XOR gates, and a few rather complicated and quite exotic building blocks like the toggle unit [41–43]. Even simple logic operations have to be treated in a different way in the scope of transition signaling. The behavior of a conventional OR gate that is generating an output as soon as the first (rising) input event has occurred would, for example, destroy the causality relation of the late input with the issued output transition. Its use is therefore not permitted in transition logic; the same is true for the AND gate.

Given this limited set of available functions, our approach to implement the +/− Counter is as follows: we provide a buffer for incoming transitions, both from the local and from the remote side. An implementation based on the well-known transition signaling elastic pipeline approach made famous by Sutherland [43] can be employed here. These are the modules termed "Remote Pipeline" and "Local Pipeline" in Figure 7. The "Diff Module" is connected to the outputs of both these pipelines and removes pairs of "matching" transitions (i.e., such with matching (virtual) k values) from both sides. In the static case one pipeline is always empty while the other one contains the difference of ticks. The nonempty pipeline is the one that has experienced a higher number of ticks (indicating the sign of the difference), while the number of pipeline entries equals the absolute value of the difference. As a result, the two elastic pipelines together with the Diff Module form the desired +/− Counter for the differential ticks. Note that, while due to requirement HW-R1 no acknowledge is given

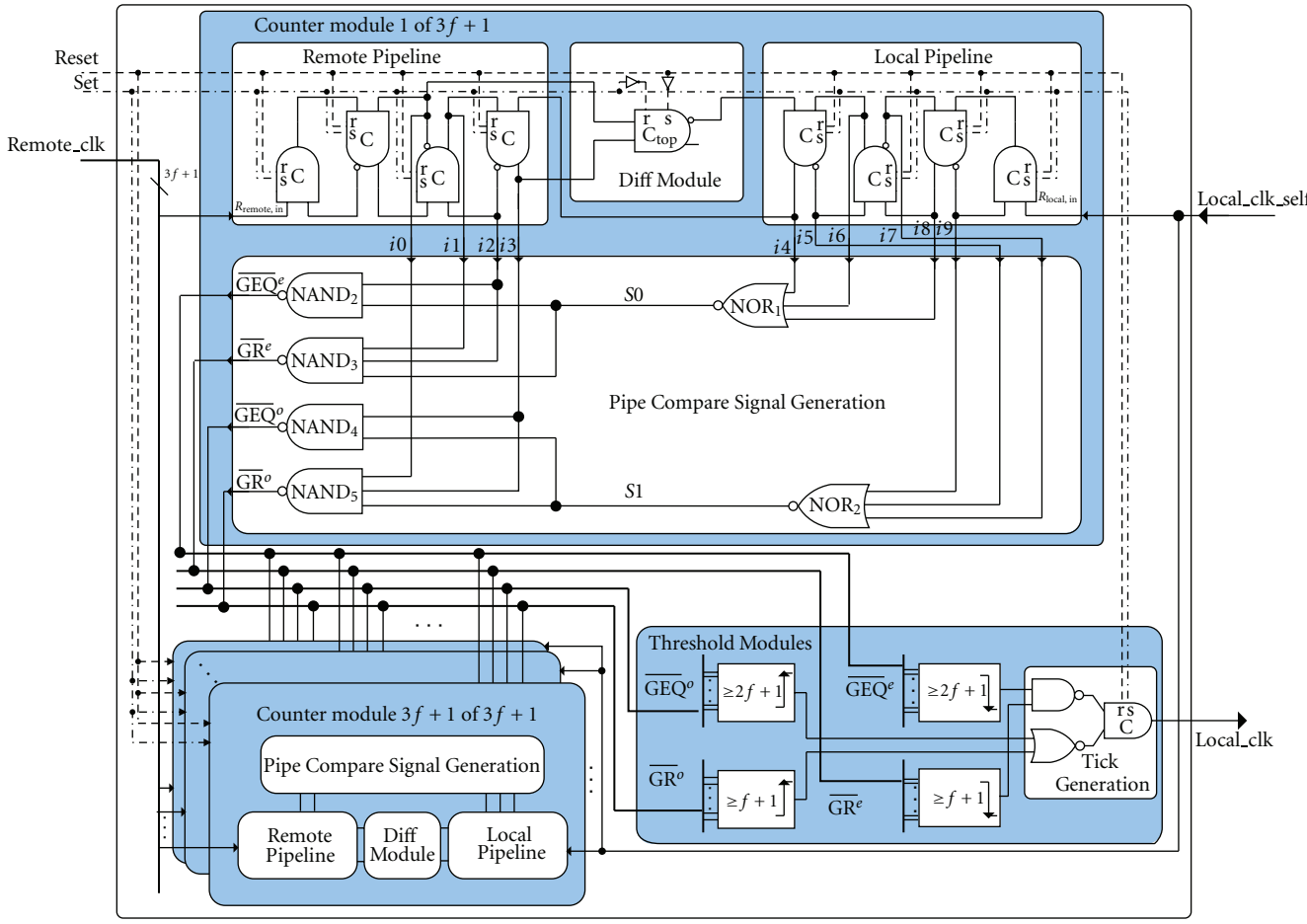


Figure 7: TG-Alg ASIC design architecture.

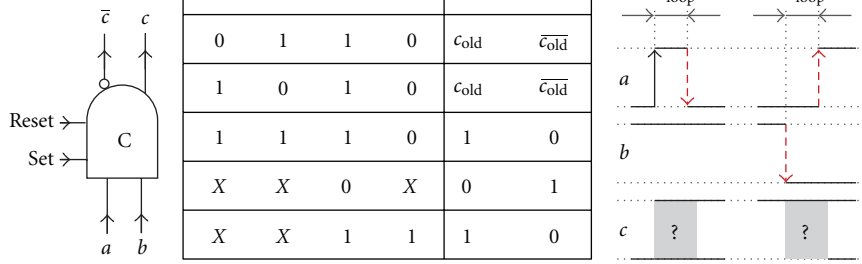
for incoming ticks, the pipelines internally operate fully via handshake, and also the interface to the Diff Module employs handshaking, thus yielding very robust operation. Details on the implementation of these modules will be presented in Sections 5.1 and 5.2.

There it will also become clear that the function of this counter is fully symmetric for TICK(↑) and TICK(↓) transitions. This allows us to use no-return-to-zero (NRZ) encoding for our tick messages, that is, each transition represents a tick, no matter whether it is a rising or a falling one, which perfectly supports our desire for efficient communication (cf. HW-R2). Therefore we are willing to accept the higher implementation efforts usually associated with NRZ coding. By exploiting this symmetry we can furthermore ensure that both half periods of our DARTS clock (HI and LO) have undergone the same treatment through our algorithm, and hence the duty cycle will be very close to 50%. Note that, although in essence the ticks need not be distinguishable, we can separate "odd" and "even" ticks by their polarity (rising corresponds to odd, falling to even). This will become important later on in the context of interlocking.

The "Pipe Compare Signal Generation (PCSG)" module performs the "greater" and "greater or equal" comparisons of Algorithm 3 by inspecting the fill levels of the remote and the local pipelines. In essence this is a conversion from transition signaling to state logic. This move to state logic is inevitable, since a counter value essentially represents a state and not an event, and so does the comparison result between two counter values. Within the state logic implementation we operate without handshaking and rather rely on timing assumptions. This is inevitable for the subsequent stage, the threshold gate, anyway, since, as already outlined, we cannot operate an m-out-of-n threshold in a fully handshake-based manner. In the conversion from transition logic to state logic care must be taken not to incidentally interpret transient states; this issue will be treated in more detail below.

Notice that the PCSG provides two outputs for the "greater" comparison, namely, GR^o and GR^e, and another two for the "greater or equal" comparison, namely, GEQ^o and GEQ^e. This is because the PCSG's operation is *not symmetric* for rising and falling transitions. In order to preserve the distinction between odd and even ticks we therefore generate separate output signals for these.

5.1. Queueing Ticks. As mentioned before elastic pipelines can be viewed as FIFO buffers for transitions. The better part of an elastic pipeline consists of Muller C-Elements



a	b	Reset	Set	c	$\overline{c}$
0	0	1	0	0	1
0	1	1	0	c_{old}	$\overline{c_{old}}$
1	0	1	0	c_{old}	$\overline{c_{old}}$
1	1	1	0	1	0
X	X	0	X	0	1
X	X	1	1	1	0

Figure 8: Basic function of a Muller C-Element.

(cf. Figure 7). The functionality of a two-input Muller C-Element can informally be described as follows: the output `c` is assigned the same logic value as the inputs `a` and `b` whenever both inputs are equal (`c=a=b=0` or `c=a=b=1`), and `c` retains its previous value otherwise `c=c_old`. As a Boolean function this can be expressed as

$$c = c_{old} \cdot (a + b) + a \cdot b = c_{old} \cdot a + c_{old} \cdot b + a \cdot b. \quad (2)$$

The Muller C-Element's ability of retaining the old value of output c clearly demands some sort of storage loop. For this storage loop to operate properly the inputs a and b have to stay stable for at least t_{loop}. This delay is defined by the propagation delay through the logic gates plus some additional wiring delays. More specifically, a Muller C-Element's correct behavior rests upon the assumptions that

(i) a single input does not toggle faster than t_{loop} if the initial transition would cause output c to change its value. For example, if input $a = 0$, $b = 1$, and output $c = 1$, input b is not allowed to toggle faster than t_{loop} since the output preserving feedback loop needs time to settle the new value of $c = 0$ (see left part of Figure 8(c));

(ii) input a and b never change their logic level to the opposite value too close to each other. For instance, again starting with $a = 0$ and $b = 1$ both inputs must not change to the opposite polarity $a = 1$ and $b = 0$ within an interval smaller than t_{loop} (right part of Figure 8(c)).

A storage loop with respective timing restrictions is common to all Muller C-Element designs. In an asynchronous design fully relying on handshaking the REQ/ACK control loop ensures that a further input transition is not applied to the Muller C-Gate before the output transition caused by the previous input transition has been acknowledged. In this way the above timing conditions are always met, at least in the fault-free case. However, in our case we have to be aware of the involved timing constraints.

5.1.1. Elastic Pipeline. Figure 9 shows a four-stage implementation of an elastic pipeline (our theoretical analysis predicted a precision of three ticks, so we considered a pipeline

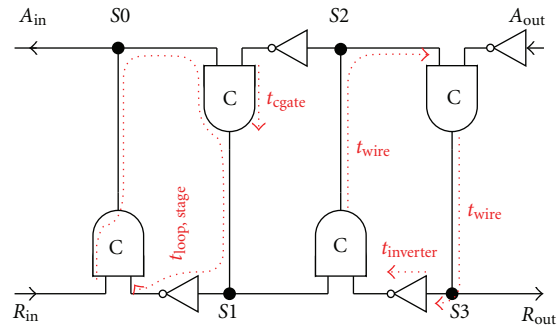


Figure 9: Elastic pipeline design.

depth of four safe) whose regular structure allows for effortless configuration of the FIFO's buffer depth. It is capable of storing up to four transitions applied at R_{in}. The rightmost entry is consumed by a transition on A_{out}, resulting in a right shift of all remaining ones.

In general, the elastic pipeline's way of transition processing provides a very elegant flow control and buffering mechanism as long as some basic timing constraints are maintained. The involved timing paths are depicted in Figure 9. Similar to the Muller C-Element itself, the feedback loops of the elastic pipeline introduce an additional timing condition restricting the input sequence. The path delay $t_{loop,\,stage}$ limits the minimum distance between two subsequent input transitions on R_{in}. The Muller C-Element's input constraint which is characterized by t_{loop} obviously is the less restricitve factor at this point since $t_{loop,\,stage} \approx 2t_{cgate} + 2t_{wire} + t_{inverter}$ with $t_{loop,\,stage} \gg t_{cgate} \approx t_{loop}$.

It is evident that speed, robustness, and area efficiency of a TG-Alg implementation are largely determined by the quality of the available library cell for the Muller C-Element. Therefore we decided to use a customized, transistor level implementation in our ASIC design. It is based on the circuit proposed by van Berkel [44], however, enhanced with several extensions. First of all, dedicated *reset* and *set* inputs (cf. Figure 8(a) for the symbol and Figure 8(b) for the enhanced function) allow to properly initialize the element's state. In the case of our TG-Alg an empty pipeline is used as starting

point. Furthermore, for improved performance two output signals, c and its inverted equivalent $\bar{c}$, are provided, thus saving the extra inverter in the feedback path of the elastic pipeline (cf. Figure 7).

Note that in the TG-Alg's elastic pipeline the output A_{in} is not connected. Again a closer look at Figure 7 reveals that R_{in} corresponds to the clock input signal (remote or local). In turn the feedback output A_{in} would correspond to an acknowledge signal for the incoming clock signal transitions which we omit in accordance with requirements HW-R1 and HW-R7.

In contrast, the far-end interconnection to the Difference Module includes the entire pipeline interface R_{out} and A_{out}. As long $R_{\text{out}} = A_{\text{out}}$, the pipeline is empty and waiting for input transitions and no tick can be removed by the Difference Module. However, as soon as $R_{\text{out}} \neq A_{\text{out}}$ the pipeline holds at least one clock tick which can be consumed by altering A_{out} to the value of R_{out}.

5.2. Counting Ticks. Each of the elastic pipelines presented above manages to buffer incoming clock transitions—four clock transitions in the particular case of the proposed TG-Alg ASIC node design. This buffering scheme is essential because it decouples the time domains between local and remote ticks, thus allowing us to handle them according to a strict, predefined sequence (i.e., without having to consider concurrency), which in turn avoids metastability by design. As already outlined above, the actual +/− counting is implemented by combining a pair of such elastic pipelines, one for the remote ticks and one for the local ones and removing matching ticks from both sides. Note that this removal procedure requires us to have one dedicated instance of the local pipeline per counter.

5.2.1. Difference Module. This module is responsible for an orderly removal of matching ticks. In essence it resembles an asynchronous state machine that first removes a tick from the remote pipeline (as soon as one is available) and only after this removal has been acknowledged enables tick removal at the local side. This procedure ensures that the conditions *remote* ≥ *local* and *remote* > *local*, which directly translate to the fill-level signals $\text{GEQ}^e, \text{GR}^e$, and $\text{GEQ}^o, \text{GR}^o$ are never falsely activated. It turns out from the analysis of the desired behavior that this state machine can be implemented by a Muller C-Element as shown in Figure 10, that is, in contrast to the Muller C-Elements in the elastic pipeline, initialized to 1.

A hardware design optimized for the targeted ASIC manufacturing and implementing the whole +/− Counter Module is presented in Figure 10.

5.2.2. Pipeline Compare Signal Generation. The PCSG module is responsible for generating the comparison results GR^o, GEQ^o and GR^e, GEQ^e based on the fill levels of local and remote pipe. As already mentioned, its function can be partitioned in the processing of odd and even ticks. The PCSG part treating incoming even ticks ultimately triggers the generation of odd ticks by issuing $\text{GEQ}^e, \text{GR}^e$ signals. Similarly, the circuit concerned with odd ticks and controlling GEQ^o and GR^o is responsible for generating even clock ticks.

Generally all output signals of the Pipeline Compare Signal Generation Module ($\text{GEQ}^e, \text{GR}^e$, $\text{GEQ}^o, \text{GR}^o$) as well as all internal logic operations are active *low*. This allowed us to exclusively use inverting basic gates (NAND/NOR instead of AND/OR) within the PCSG design, which contributed to optimizing the speed of the ASIC implementation.

The PCSG performs the conversion from the transition logic used in the elastic pipelines and the Diff Module to state logic. At this point specific care must be taken that the comparison signals *never* switch to the active state before *remote* ≥ *local* and *remote* > *local* conditions, respectively, actually hold. Note that, however, the nature of the tick generation function allows them to stay active for some time even if the respective conditions are no longer fulfilled. From the perspective of the algorithm this means that the early or illegal firing of a rule is disastrous, while the late deactivation of a rule is less critical. In the design of the Diff Module we have already carefully avoided glitches that might be introduced by the removal process. Still, however, we may experience erroneous activations due to a dynamic state of one elastic pipeline, that is, when a transition ripples towards the output of the pipe. For this reason three taps of the local pipeline are combined to ensure that no dynamic effects during tick arrival or removal can compromise the fill level signal, although for the static case only two would be sufficient. In detail, the signals *SLocal*1, *SLocal*2, and *SLocal*3 in conjunction with the NOR_1 gate are used to indicate whether the pipeline holds a single even tick. The fill-level indicators on the remote side (*SRemote*2, *SRemote*3 and *SRemote*2_*N*, *SRemote*3_*N*) are responsible for checking if one or more clock ticks are currently stored in the pipeline. An appropriate combination of local and remote side fill-level signals allows to generate the output signals GEQ^e and GR^e, which represent the conditions *remote* ≥ *local* and *remote* > *local*, respectively. To attain an active fill-level signal GEQ^e, corresponding to *remote* ≥ *local* it has to hold that:

(i) at most one even (tick-↓) clock transition is stored inside the local pipeline which is indicated by NOR_1 (the distinction whether no or one single transition is in the pipeline depends on the state of the ACK, that is, whether the last transition has already been removed by the Diff Module, as this is not relevant for our comparison, involving the ACK signal was not necessary),

(ii) at least one even clock tick is present in the remote pipeline, indicated by signal *SRemote*3_*N*.

These two conditions are combined in a final step via NAND_2, generating the output GEQ^e. Similarly, for the activation of the low active signal GR^e implementing the condition *remote* > *local* the following constraints have to be fulfilled:

(i) again only one even (tick-↓) clock transition is allowed inside the local pipeline, which is assessed by the NOR_1 gate;

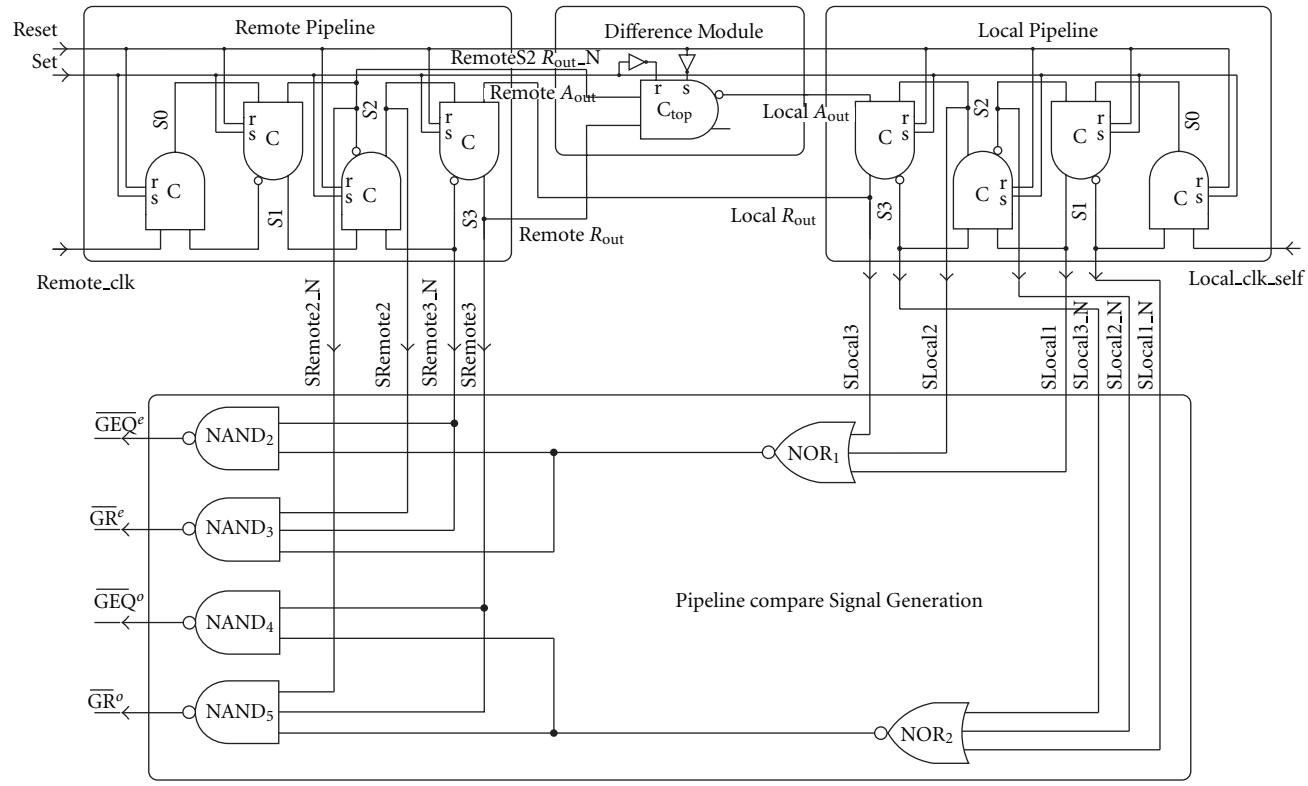


Figure 10: TG-Alg ASIC +/− Counter Module.

(ii) more than one clock tick has to be present in the remote pipeline, an even clock tick in pipeline stage $S3$ and additionally an odd tick in stage $S2$.

These conditions are evaluated by the gate $NAND_3$ via signals *SRemote2_N* and *SRemote3_N* in conjunction with the output of NOR_1. The activation of the signals GEQ^o and GR^o follows by analogous means, simply treating odd instead of even input signals.

We have carefully analyzed the dynamic behavior of the pipe to confirm that our solution can handle all possible dynamic effects caused by transitions rippling through the elastic pipelines.

5.3. Generating Ticks. The final processing step of every TG-Alg node is concerned with the evaluation of the counter fill levels and the generation of new clock ticks according to the "Relay Rule" and "Increment Rule" from Algorithm 3. Here a move back from state logic to transition logic needs to be performed, which requires the careful consideration of possible glitches. After all, in transition signaling every signal change is treated as meaningful data.

5.3.1. Threshold Modules. Four distinct threshold circuits allow to separately evaluate all output signals of a node's $(3f + 1) + /-$ Counter Modules. As depicted in Figure 11, two threshold circuits are responsible for processing the fill-level signals GEQ^e and GR^e for even ticks. This way they implement the tick generation algorithm's "Relay Rule" and "Increment Rule" for falling transitions by virtue of threshold gates with threshold $f + 1$ and $2f + 1$, respectively. In the same way their odd counterparts treat the signals GEQ^o and GR^o.

For the implementation of a single threshold function several possibilities exist. We have evaluated them according to the following criteria.

(i) Low Propagation Jitter: As outlined in Section 3.1 the algorithm's correctness and performance ultimately rely on the ratio Θ of different timing path delays within the TG-Alg design. Therefore we do not want different paths to have substantially different propagation delays (e.g., comprise a different number of logic stages).

(ii) Low Propagation Delay: The threshold module's propagation delay directly adds to the TG-Alg's round-trip time and thus impacts performance.

(iii) Robustness: Since we are heading for a fault-tolerant solution we do not allow a module to compromise the overall robustness. This rules out solutions based on summing up currents or charges in the analogue domain.

(iv) Area Overhead

(v) CMOS Technology: As the approach is targeted for digital CMOS circuits, the threshold gate should as well be implementable in CMOS technology, preferably with standard cells.

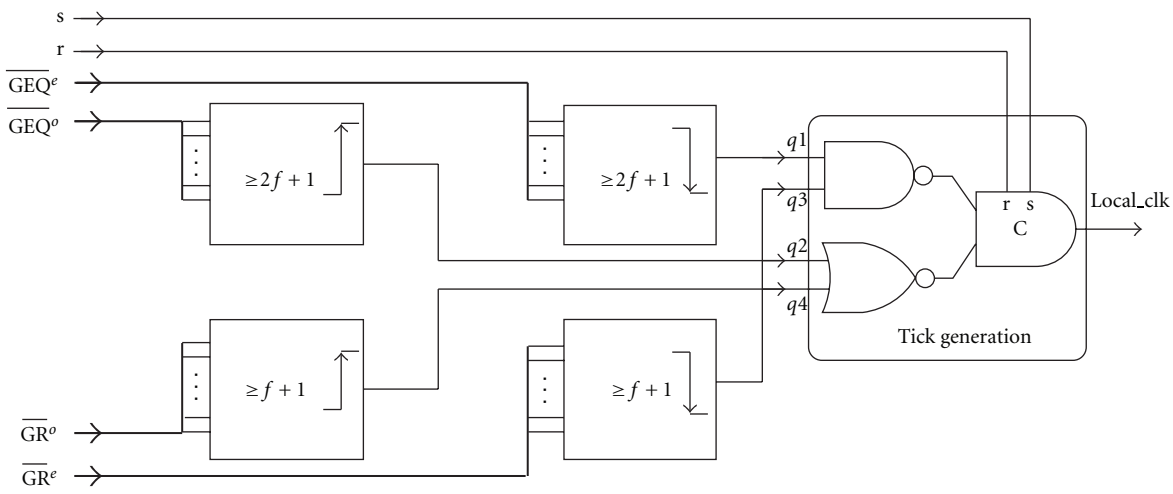


Figure 11: TG-Alg ASIC Threshold Modules and Tick Generation.

In [45] we have made an elaborate comparison of the available approaches and identified the sum of products scheme as the one that best matches these constraints, although its area overhead is quite substantial.

For the needed implementation of an m-out-of-k design $\binom{k}{m}$ product terms have to be computed and then summed up. The threshold circuits of the ASIC TG-Alg have been designed for 11 input signals. The resulting 4-out-of-11 implementation of the $f + 1$ threshold circuit yields 330 product terms. These product terms have to be summed up in a treelike cascaded architecture since no elementary gates with a fan-in of 330 are available in the ASIC target technology. A notable peculiarity of the sum of products implementation is the fact that for a given configuration of $n - 1$ inputs with $n - 1 = 3f + 1$, the required threshold functions $f+1$ and $2f+1$ can be converted into each other by simply inverting all input and output signals, therefore only one function has to be designed.

Although the chosen sum of products implementation is beneficial in many respects, it cannot be implemented such as to operate completely glitch-free. This is not an implementation deficiency, but a general behavior of asynchronous state logic that tends to produce glitches on the outputs while processing its inputs as long as no strict restriction on the input sequence is ensured [46]. At this point we can take advantage of the separation between odd and even ticks. Having the circuit blocks for odd and even ticks operating in alternation, we can determine a period of inactivity for every threshold gate, during which it may produce glitches without jeopardizing the proper overall operation of the TG-Alg, provided these glitches are orderly masked. It will be the task of the Tick Generation Module (see Section 5.3.2) to provide this masking, and we will have to consider timing constraints for the allowed duration of glitches.

5.3.2. Tick Generation Module. The task of the Tick Generation Module (see Figure 11) is to actually generate and broadcast the next tick, as soon as the rules implemented in the threshold module indicate it is time to do so. It is specifically here where the conversion from state logic back to transition logic takes place.

In the Tick Generation Module the four threshold circuit outputs q_1, q_2, q_3, and q_4 are combined by simple logic gates in a way such that only valid clock ticks are generated, in essence it handles the concurrency of the two rules of the algorithm. Furthermore, the Tick Generation Module has to ensure that after generating a clock tick the TG-Alg's clock output remains stable despite the fact that the outputs of the threshold circuits might toggle due to glitches. This retention of the clock output is enabled mainly by the Muller C-Element at the output which only issues a new tick if *both* inputs indicate to do so. However, since the storage loop of the Muller C-Element needs stable inputs during its settling time the outputs of the threshold circuits have to be stable for a small time interval before and after a new tick is generated. This safety window must be ensured by the timing constraints. Assuming that all implementation constraints are fulfilled and taking the above-mentioned considerations into account, a new tick is generated only if

(i) the threshold circuits responsible for the generation of the last tick issued (by providing enabled input signals GEQ, GR) have become inactive again;

(ii) at least one of the two threshold circuits responsible for the generation of the new tick gets activated.

Note that by these rules the generation of an odd tick $k + 1 \in \mathbb{N}_{\text{odd}} := 2\mathbb{N} + 1$ is triggered only if the last tick generated was even ($k \in \mathbb{N}_{\text{even}} := 2\mathbb{N}$). A thorough analysis of the described tick generation process, conducted in the context of the DARTS project and published in [33], shows that the presented approach is sufficient to avoid that old and new instances of GR^o and GEQ^o get mixed up. The main message of this formal analysis—the derived timing constraints for the hardware implementations—will be given subsequently.

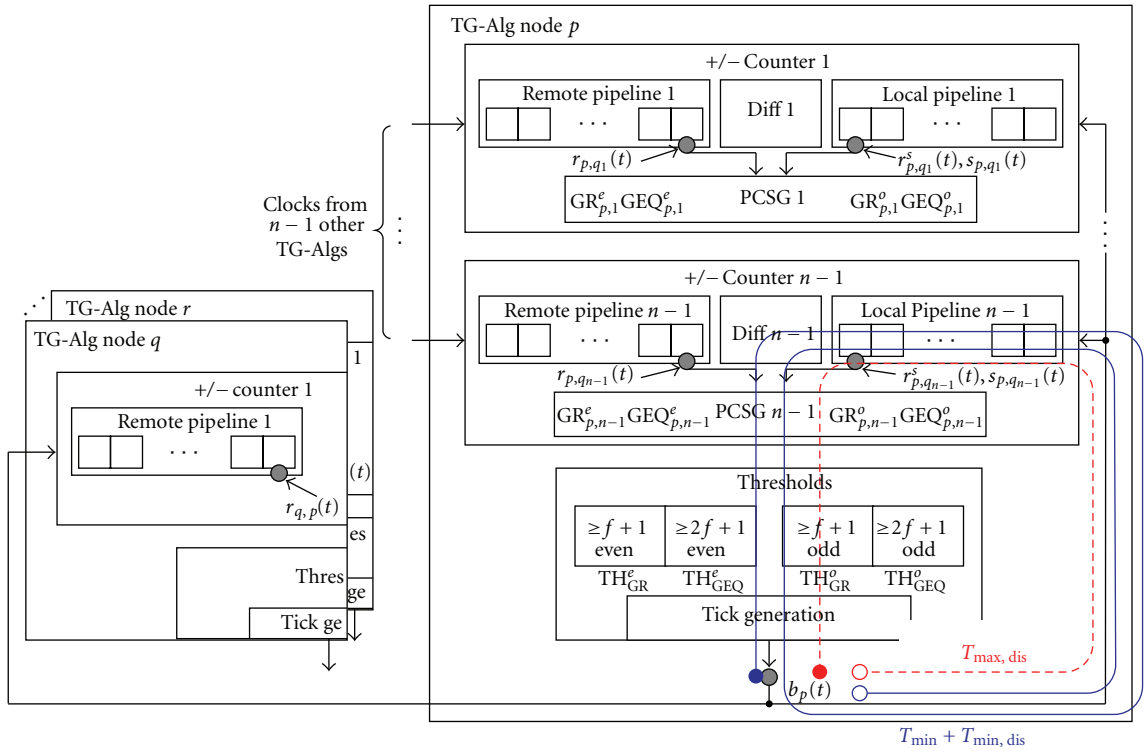


FIGURE 12: Timing paths of the interlocking constraint.

5.4. Timing Constraints. The correct behavior of the tick generation approach of Algorithm 3 relies on some particular timing constraints. In fact, implementation-specific constraints on path delays have to hold. The most important one is given by the *Interlocking Constraint* that ensures that TICK(k) and TICK($k+2$) messages do not interfere with each other.

Constraint 1 (interlocking). $T_{\max,\,\mathrm{dis}} \leq T_{\min} + T_{\min,\,\mathrm{dis}}$ must hold.

With the delay paths.

$$\begin{aligned} T_{\max,\mathrm{dis}} &:= \tau^+_{\mathrm{TH}} + \max\left(\tau^+_{\mathrm{GR}}, \tau^+_{\mathrm{GEQ}}\right) + \tau^+_{\mathrm{loc}} \\ T_{\min} &:= \tau^-_{\mathrm{TH}} + \min\left(\tau^-_{\mathrm{GR}}, \tau^-_{\mathrm{GEQ}}\right) + \tau^-_{\mathrm{loc}} + \tau^-_{\mathrm{Diff}} \\ T_{\min,\mathrm{dis}} &:= \tau^-_{\mathrm{TH}} + \min\left(\tau^-_{\mathrm{GR}}, \tau^-_{\mathrm{GEQ}}\right) + \tau^-_{\mathrm{loc}}, \end{aligned} \tag{3}$$

$T_{\max,\mathrm{dis}}$ represents the slowest disabling path starting and ending at the tick generation output of the respective node. $T_{\min}$ corresponds to the fastest path for generating a clock tick, whereas $T_{\min,\,\mathrm{dis}}$, analogously to $T_{\max,\,\mathrm{dis}}$, accounts for the minimum deactivation time of the previous clock tick which in turn enables the generation of the next clock tick. Figure 12 graphically presents TG-Alg node's opposing interlocking delay paths. Note that the involved paths only include design units at a local node. This locality of Constraint 1 considerably facilitates designing the path delays accordingly.

The interlocking constraint is not the only relevant timing bound. In short, it also has to hold that ticks in the local and remote pipelines get removed fast enough to inhibit excessive queuing of ticks. An additional timing constraint bounds the fastest remotely triggered generation of a new tick in relation to the slowest locally processed one. Moreover, the start-up (booting) of all correct nodes has to be in a certain time interval. A detailed description of these additional constraints can be found in [38].

6. Results

Based on the design presented in the previous section, we have implemented an ASIC prototype in a 0.18 μm technology. We have chosen this relatively large feature size, since it allows a radiation hard implementation, which is an important feature for the spaceborne applications we had in mind. The threshold gate is configurable for a system up to $f = 3$, that is, $n = 11$. This ASIC implementation allows us a first estimation of cost and performance of the DARTS concept in practice. This section reports on our experiences and measurement results.

6.1. TG-Alg Implementation Characteristics. The analysis of the whole TG-Alg design is conducted by putting together the characteristics of all subunits. For this purpose exact numbers for the hardware effort in terms of gate equivalents and die size will be given. The fully connected network topology clearly implies a quadratic growth of the TG-Net's

Table 1: Hardware effort of a single TG-Alg and its components.

	No. of basic gates	No. of C-Elements	area in [μm^2]	area in %
Remote Pipeline	—	4	944	0.20
Local Pipeline	—	4	944	0.20
Difference Module	2	1	270	0.06
PCSG	6	—	395	0.08
+/− Counter	8	9	2,553	0.05
11 +/− Counters	88	99	28,083	5.80
$f + 1$ circuit	550	—	51,641	10.67
$2f + 1$ circuit	1,013	—	176,083	36.39
Tick Generation	2	1	303	0.06
Threshold Modules	3,128	1	455,751	94.19
single TG-Alg	3,218	100	483,862	100.00

number of links with node count n and thus also with f, that is, $\mathcal{O}(f^2)$, and has direct impact on the complexity of a TG-Alg's implementation. Notice, however, that we do not advocate building a system with high n; even a value of $f = 3$, as it is available for our experiments, is much better than the "single fault assumption" usually applied for hardware.

Staying with the flow of the previously presented subblocks the tick queueing and tick counting mechanisms are treated first. The hardware effort for building a TG-Alg's queueing and counting blocks is for a considerable part determined by the amount of incorporated Muller C-Elements. Considering the remote and local elastic pipelines as well as the Difference Module, the Muller C-Element presents the only relevant building block, whereas the Pipeline Compare Signal Generation Module is assembled using a few basic gates with two and three inputs, respectively. Recall from Figure 7 that $n - 1 = 11$ individual +/− Counter Modules are required—one for each remote TG-Alg. The upper part of Table 1 presents numbers for gate count and silicon area (in the 0.18 μm ASIC target technology) treating submodules as well as the whole design of a +/− Counter. Furthermore, the hardware effort is added up to account for the 11 +/− Counters of the actual TG-Alg implementation. It can be observed that the elastic pipelines are the main contributors to the chip area of each +/− Counter. This is due to the relatively complex structure of the library cell employed for the Muller C-Element with its extra inputs for direct set and reset. Note, however, that we used custom cells designed on transistor level. Had we chosen a gate-level implementation (e.g., a design based on NAND gates), we would have experienced an even higher area overhead and a notable loss of performance.

In contrast to the queueing and counting blocks (+/− Counter) every TG-Alg holds only one Threshold Module—incorporating four threshold circuit units and the Tick Generation Module. As thoroughly described in Section 5.3.1, threshold circuits are purely combinatorial blocks following a sum of products implementation. Given an input width of $n - 1 = 11$, the presented complexity growth with the number of inputs yields 330 and 462 product terms for each of the $f + 1$ and $2f + 1$ threshold circuits, respectively. Therefore the exponential increase with approximately $\binom{3f+1}{f+1}$ is one of the prominent cost driving factors when scaling the tick generation system's resilience $f \leq \lfloor (n-2)/3 \rfloor$ and hence the number of nodes n. Due to the fact that basic standard cell gates like NAND and NOR, which are used in the sum of products implementation, are typically available only with two and three inputs, hardware effort is additionally increased with increasing number of n. This is true for the product terms as well as for the terminal sum term because increasing numbers n and m result in the need for cascading basic gates.

In contrast to the threshold circuits the Tick Generation Module does not suffer from scaling effects since it consists of two basic gates and a single Muller C-Element only. Similarly to the elastic pipelines it benefits from the transistor-level implementation of the Muller C-Element. The lower part of Table 1 lists gate count and area numbers for the involved design units and the Threshold Module block overall.

The comparison of TG-Alg's components in terms of hardware effort, shown in Table 1, reveals that the sum of product threshold circuit implementation accounts for a substantial part of the entire design. Almost 95% of a TG-Alg's chip area is devoted to the Threshold Modules. The enormous hardware effort reflects the threshold circuits' unfavorable scaling with f and n. In general, the Threshold Modules' predominance in hardware effort allows to give an estimate for the scaling of a TG-Alg's chip area following $\approx \binom{3f+1}{f+1}$. This scaling obviously only applies for the used sum of products approach and would be completely different for other implementation technologies. Analogously to the customized Muller C-Element, an applicable enhancement to reduce the sum of products area effort might be given by an optimized transistor-level implementation. Furthermore, the design alternatives presented in [45] might provide reasonable options for improvement.

6.2. Performance Assessment. The following assessment aims at thoroughly characterizing the properties of a running tick generation system. These evaluations give insight on tick generation under varying operating conditions and validate the fault tolerance properties (worst-case scenarios) of the DARTS approach. A tick generation system composed of

$n = 8$ (and $f = 2$) fully interconnected ASIC nodes ($U1$ to $U8$) was assembled for all evaluations of the cluster.

In the context of average-case experiments the assessment of implementation and operation characteristics is certainly of interest. In particular, stability considerations arise when recalling that the primary goal of the DARTS clocking scheme is to provide conventional synchronous circuits with a fault-tolerant clock. On the one hand the asynchronous nature of the TG-Alg implementation allows the design to adapt its operation to varying conditions, thus increasing its robustness. On the other hand this flexibility might be problematic from the synchronous unit's point of view since it is controlled by the adaptive, thus varying, TG-Alg clock. Due to the TG-Algs' asynchronous implementation a certain degree of operation parameter sensitivity can be expected.

6.2.1. Frequency. The attainable clock frequency solely relies on switching delays of the asynchronous circuits and interconnection delays of the remote and local clock lines. Using predictions from theory the tick generation scheme's frequency can be bounded by the synchronization property Progress (P) together with the tick generation path T^{-}_{first}:

$$f_{\text{average}} = \left[\frac{1}{2}T_P, \frac{1}{2}T^{-}_{\text{first}}\right]. \tag{4}$$

The path given by T_P denotes the slowest possible generation of a subsequent tick, while T^{-}_{first}, and $T_{\min}$ represent the fastest remotely, and locally triggered tick generation, respectively. The required delay parameters can be extracted from the ASIC design files. Together with delays for the chip interconnect this is sufficient to give a sound estimation of the achievable clock frequency. For the DARTS design, $T^{-}_{\text{first}} \approx 6\,\text{ns}$ and $T_{\min} \approx 6\,\text{ns}$ with an assumed interconnect delay of 1 ns lead to an expected $f_{\max} \approx 71\,\text{MHz}$.

Measurements of a similar path for T^{-}_{first} showed a delay of 7 ns. This path involves `remote_clk[·]` input pin, next 6 Muller C-Elements, the PCSG unit, the Threshold Modules including tick generation, and finally `local_clk` output pin. Analogously to the above examination the delay for $T_{\min}$ is measured as 7 ns. The path is also quite similar comprising the `local_clk_self` input pin followed by 5 Muller C-Elements, the PCSG unit, the Threshold Modules, and Tick Generation block, ending at the `local_clk` output pin.

The difference of 1 ns between measurements and design files is mainly due to the fact that the measurement setup does not—unlike the evaluation of the design files—use the shortest path through the threshold circuits. (This is due to the fact that a system of 8 nodes with $f = 2$ was used for the measurements; however, the paths considered in the design files use the faster paths of $n = 5$ and $f = 1$). From theory, however, it is clear that the rate based on the fastest paths (T^{-}_{first} and $T_{\min}$, resp.) can only be maintained for a short period, that is, either the remote or the local pipeline has to be full while ticks at the opposite side arrive at T^{-}_{first} or $T_{\min}$, respectively. As soon as the previously filled pipeline gets emptied the clock rate will notably slow down which leads to the observed average frequency of $f_{\text{average}} = 54\,\text{MHz}$.

Table 2: Cluster of 8 standard nodes: voltage scaling.

Core voltage in (V)	Avg. frequency in (MHz)	Current ASIC U6 in (mA)	Current all in (mA)
1.3	38	11.7	100
1.4	43	15.1	126
1.5	47	17.6	150
1.6	50	20.6	178
1.7	52	23.8	204
1.8	54	27.2	233

6.2.2. Operation Condition Dependency. As mentioned above, the switching speed of the circuits is likely to be a function of supply voltage. Moreover, digital CMOS circuits are also known to be sensitive to temperature variations. Both effects are typically encountered in normal operation modes. The mentioned voltage dependence of a CMOS circuit can be approximated by deriving the delay times for a single gate and essentially boils down to

$$t_{\text{gate}} \approx \frac{C_L}{\beta V_{\text{DD}}}, \tag{5}$$

with C_L being the load capacitance, β and V_{DD} representing the CMOS transistors' gain and supply voltage, respectively [47]. Note that the above mentioned temperature dependence of CMOS circuits is hidden inside β. The carrier mobility (electrons and holes) decreases with increasing temperature, thus β decreases, yielding a slowdown of the circuit as temperature rises. Table 2 shows the measured average frequency and the corresponding current drawn by the design.

As expected from (5) the achieved clock frequency scales proportional to the supplied core voltage. Figure 13(a) presents results of detailed measurements in which the applied core voltage has been changed in 10 mV steps in an interval starting with 1.30 V and ending at the nominal voltage of 1.80 V. An improved illustration of the measurement data which makes the correlation of voltage and clock frequency more evident is given in Figure 13(b). Core voltage and frequency are given in percentages of their respective maximum value. This way it can be observed that a voltage change of 1% yields approximately 1% variation in clock frequency (red line in Figure 13(b)). The strong impact of the core voltage on the operating frequency of the asynchronous tick generation implementation meets the expectation. A second important factor of influence is temperature. Again, according to (5) the switching speed and propagation delay of CMOS circuits scale indirectly proportional to temperature. This expectation has also been confirmed by the measurements.

6.2.3. Short-Term Jitter. Short-term fluctuations of the frequency and discontinuities in the clock periods during fault-free operation are expected from at least two sources. Firstly, the above-mentioned supply voltage dependence will project its voltage jitter to a frequency jitter. Temperature changes

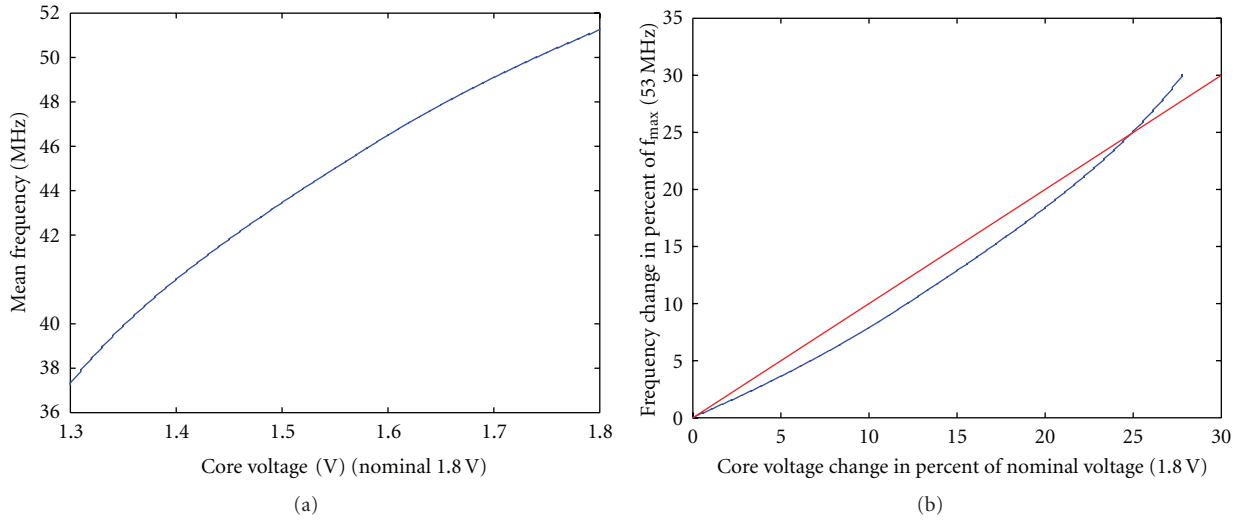
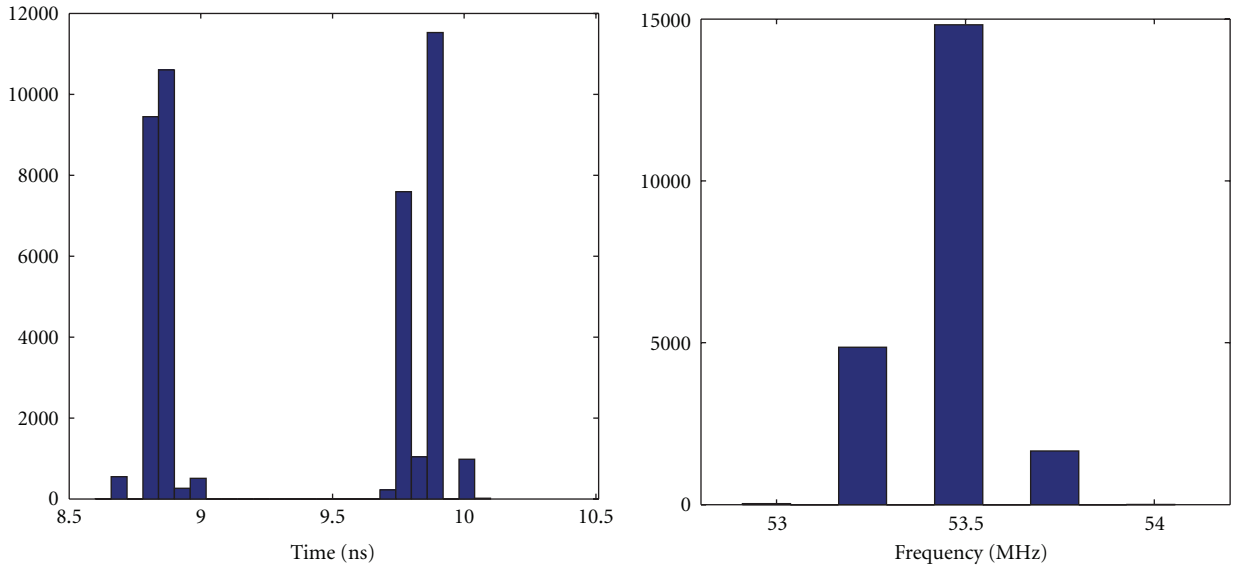


Figure 13: DARTS cluster's mean clock frequency core voltage dependence.



(a) Half periods (b) Clock frequency

Figure 14: Single clock evaluation of a running standard node cluster.

are considered to be too slow to yield perceivable short-time effects, but since propagation delay is known to be affected by thermal noise, so will be the frequency produced by our system. The evaluations are based on short-time measurements with very high resolution (up to 10 GS/s) including approximately 40,000 clock transitions. These measurements aim at characterizing the clock of a single node running within a DARTS cluster.

The measured half periods are presented in the histogram plot shown in Figure 14(a). Two cluster points can be identified, one at $\approx$ 8.7 ns and the other at $\approx$ 9.8 ns. A separate examination (not shown) revealed that these accumulation points correspond with the distributions of the *HI* and *LO* periods. This observation can be explained by the slightly different processing speed of the respective clock signals, in particular, (a) the separate processing of rising and falling transitions in our implementation yielding different path delays and (b) the different timing behavior (rise time/fall time, e.g.,) of Muller C-Elements for rising and falling transitions. Figure 14(b) presents the distribution of the clock frequency with a mean frequency of 53.4 MHz and a standard deviation of 0.153 MHz.

6.2.4. Long-Term Jitter. In the tick generation system's long-term operation especially the effect of varying temperature is expected to be noticeable. Self-heating of the TG-Alg chips is anticipated to continuously slow down the tick generation process. The numbers presented in Table 2 show that the stability of the clock frequency heavily depends on

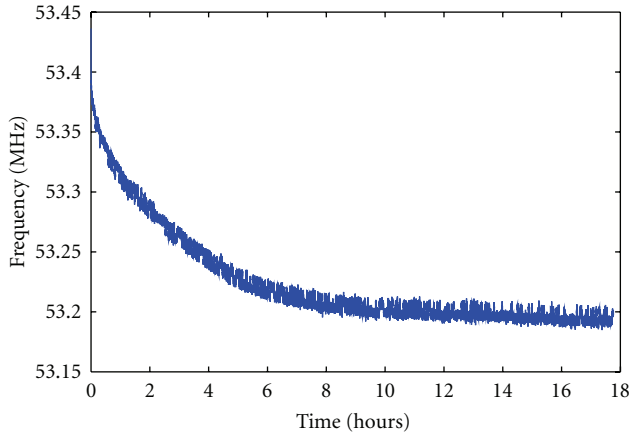


Figure 15: Long-term clock stability: 17-hour run.

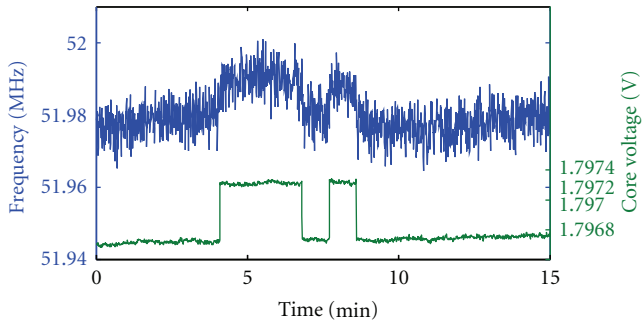


Figure 16: Frequency and voltage trace showing power supply variations.

the stability of the operating conditions. Figure 15 presents a long-term assessment of a node's mean clock frequency with an evaluation interval of more than 17 hours. It can be observed that clock frequency noticeably decreases over time by about 250 kHz. The operating conditions, that is, core voltage and ambient temperature, were not varied in this experiment setup. The measurements start with all nodes in reset state with no activity. Thus no mentionable current is drawn by the chips. As soon as the reset gets deactivated the designs start to draw substantial current that contributes to self-heating of the running chips and causes the asymptotic decrease of the mean clock frequency depicted in Figure 15.

A 15-minute snapshot of another frequency measurement including a high resolution trace of the core voltage is presented in Figure 16. In this figure it can be observed that a discrete jump of the core voltage is directly followed by a frequency jump. This behavior perfectly fits into the design's voltage dependence presented earlier. In the depicted measurement the voltage changed by $\approx 0.5\,\mathrm{mV_{rms}}$ which led to the aforementioned shift of average frequency by 10–20 kHz. The initial cause for the observed minor voltage change is hidden in the used digital power supply which has quantization steps of $0.5\,\mathrm{mV_{rms}}$. The correctness of this interpretation for the frequency jumps has additionally been confirmed by crosscheck measurements with analog power supplies. In these evaluations overall increased frequency

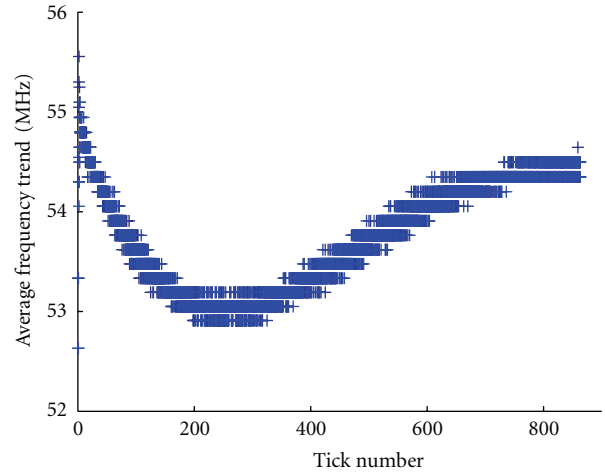


Figure 17: Mean frequency trend over all 8 nodes.

jitter was observed due to the higher level of voltage noise. However, neither discrete steps in the voltage level nor in the mean frequency were encountered.

In addition to the presented assessment of a single clock signal, the clock signals of the whole ensemble were evaluated as well, albeit with a lower precision, since these measurements had to be performed with a logic analyzer, whose time resolution was limited to 250 ps. The main interest clearly resides in the synchronization of the clock ensemble. Detailed short-term measurement showed that for the fault-free case the ensemble starts with tight synchrony and remains closely synchronized (the small initial offset is due to differences in the propagation of the reset signal). Under normal conditions, that is, nominal core voltage and room temperature, evaluations yielded initial offsets in the range of 1 ns to 1.5 ns. In these short-term measurements the maximum skew among any two TICK(k) clock transitions never exceeded its initial offset of 1.5 ns. Hence, a fault-free clock ensemble running under nominal operating conditions has precision $\pi = 1$. In Figure 17 all 8 nodes' frequencies of a DARTS cluster (starting from reset state) are depicted. (To enhance the expressiveness of the graph the data values actually have been smoothed to compensate for the limited resolution of the logic analyzer. Note that this did *not* affect the general trend but only the magnitude of the frequency changes). It can be observed that the frequencies of all DARTS clocks change jointly, thus yielding close synchronization.

6.3. Fault Tolerance Properties. Up to now all evaluations have assumed TG-Alg nodes operating fault-free. In contrast, the evaluations presented in this paragraph consider scenarios with faults artificially introduced into a running cluster of 8 nodes (which by design should be resilient to $f = 2$ Byzantine faults). In the conducted experiments the consequences of crashing TG-Alg nodes are examined in particular. The node crash scenarios are implemented by resetting one or two nodes of the DARTS cluster. Note that these scenarios do not necessarily have the benign properties

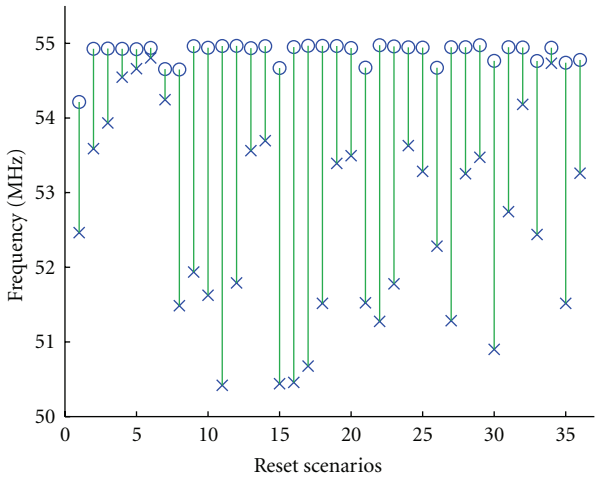


Figure 18: Mean Frequency (o) before and (x) after reset of 1 or 2 nodes.

of crashes as they are assumed in distributed systems. Even stuck-at faults can be outside the scope of the crash fault scenario. For instance, an early clock transition, that is, changing a clock rail from *HI* to *LO* (stuck-at-0) through the activation of a node's reset is already within the class of malicious/Byzantine failures. All combinations of scenarios with one or two nodes crashing yielded Figure 18. For each of the reset scenarios the mean frequency before and after the crash has been derived from measurement data. The lines interconnecting these two mean frequency values illustrate the actual drop of the clock frequency. As anticipated, in all 36 reset scenarios the deactivation of nodes leads to a decrease of the mean clock frequency. This slowdown is quite natural since for the remaining nonfaulty nodes of the cluster the crashing of nodes implies that one or two of the previously $2f + 1$ fastest node(s) has/have been deactivated. Hence the correct nodes have to wait until tick messages are received from slower nodes which are still up and running, consequently leading to additional delay before the next tick can be generated. Note that due to small differences in propagation delays and the close synchronization of all clocks, each node's set of $2f + 1$ fastest neighbors might be different. This leads to the effect that the reset of each node at least slightly influences the clock overall clock frequency.

The synchronization precision π represents one of the most important synchronization properties of the DARTS tick generation system. It may simply be assessed by measuring the clocks' relative offset. However, this evaluation is unlikely to reflect worst-case conditions. As already pointed out in Section 6.2.4 the computation and interconnect delays of the DARTS cluster are almost perfectly matched—which yields a precision $\pi = 1$. Thus an appropriate scenario has to be derived and established for worst-case measurements. Figure 19(a) shows the generic setup to statically force a system of $3f + 2$ TG-Alg nodes into an operation mode with worst-case precision. The only relevant parameters for this scenario are given by the interconnecting remote delays τ_{rem}. As depicted, a set of f nodes have to be *faulty* in a way that no TICK(k) messages are delivered to a second set of f *slow* TG-Algs. It further has to be ensured that ticks sent among the set of slow nodes as well as those received from the group of $f + 2$ fast TG-Algs are issued with the maximum remote delay τ^{+}_{rem}. Connections not explicitly shown in Figure 19(a) can be assumed to have delay τ^{-}_{rem}. More formally speaking, in a system where P denotes the set of all nodes there are three distinct sets of TG-Algs with A comprising the fast nodes, B the slow, and the F faulty ones. The remote delays from p to q in this setup are given by

$$\begin{aligned}
\tau_{\text{rem}}(p \in A, q \in B) &= \tau^{+}_{\text{rem}} \\
\tau_{\text{rem}}(p \in F, q \in B) &= +\infty \\
\tau_{\text{rem}}(p \in B, q \in B) &= \tau^{+}_{\text{rem}} \\
\tau_{\text{rem}}(p \in P, q \in A) &= \tau^{-}_{\text{rem}} \\
\tau_{\text{rem}}(p \in P, q \in F) &= \tau^{-}_{\text{rem}}.
\end{aligned} \tag{6}$$

To get a better understanding for the reasons why this static evaluation setup represents a valid worst-case scenario for the tick generation system, Figure 19(b) depicts an execution trace of the relevant (non-faulty) nodes. As indicated in the trace, it is assumed that all nodes start at approximately the same time by issuing TICK(0). For the example, it is assumed that τ_{rem} alone determines the processing speed of the tick generation system. (Recall from Section 3.1 that only the ratio Θ of fastest to slowest path determines the algorithm's properties, thus it makes no difference if τ_{rem} or the whole delay of the tick generation path is considered in the experiment scenarios.) In the given setup, set A comprises $f + 2$ fast TG-Algs. Together with f fast, but faulty nodes $\in F$, ticks are generated continuously at a rate determined by τ^{-}_{rem} and according to the algorithm's "Increment Rule" ($= 2f + 1$ threshold). Analogously, the f slow TG-Algs $\in B$ also start to issue clock ticks triggered by the "Increment Rule" (I), however, at a period determined by τ^{+}_{rem}. Thus, group A starts "running away" with τ^{-}_{rem} while the slow group B "runs behind" with period τ^{+}_{rem}. At some point the slow nodes' flow of issuing ticks changes to the operation mode where the "Relay Rule" (R) takes over tick generation. This switching point is reached when TICK(k) messages arrive at the slow nodes, indicating that the fast remote nodes are ahead by at least one tick, that is, $k > l$, with ℓ being the current local tick number. This way the "Relay Rule" ensures that the system stays in a synchronized state. The maximum offset in time between the first sending of TICK(k) at t_k and the last sending of TICK(k) at t'_k for any pair of correct nodes p, q can be used to derive the cluster's precision π (for more precise calculations of the maximum clock skew of correct nodes refer to the detailed formal analysis presented in [33, 48]).

The implementation of the above described evaluation scenario confirmed the predictions from theory. As expected the measurements showed that the synchronization of fault-free nodes only depends on the ratio of fastest to slowest path Θ. Via these artificially introduced path delays the worst case ratios of fastest to slowest path could be set to achieve precision $\pi = 3$ which still poses no threat to the clocking

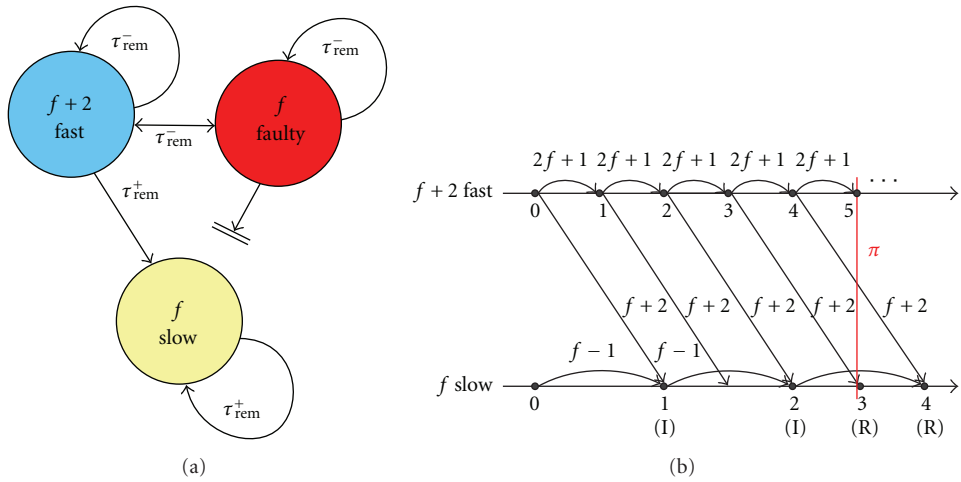


Figure 19: Evaluation scenario to attain worst-case precision π.

approach and can be handled by the buffering of the four-stage elastic pipelines.

7. Conclusions

Considering the increasing need for fault tolerance in general, and the lack of fault-tolerant clocking schemes that allow globally synchronized operation even in the presence of significant skew in particular, we have proposed a novel clock generation approach that closes this gap. DARTS provides a clock with scalable fault tolerance for both, clock generation units as well as interconnect, that is globally synchronized with a bounded precision.

The key idea behind DARTS is to use a tick synchronization algorithm from the distributed systems community whose performance and fault-tolerance features can be formally proven. By moving this originally software-based algorithm to a hardware implementation, the attainable precision can be improved to a level that is suitable for clocking hardware units with reasonable synchrony. When doing so, however, two crucial issues had to be mastered. Firstly, the algorithm had to be appropriately chosen and modified so as to suit to a hardware implementation, and secondly the hardware implementation as such raised considerable challenges. Many of these challenges originated not only from the desire to attain a fast and area-efficient solution, but also from the fact that many facets of the abstract algorithm turned out to be difficult to project to hardware, such as unbounded count values or atomicity of actions.

We have presented a solution that is based on asynchronous logic design, partly based on the original indication principle with handshaking and partly on timing assumptions. The latter turned out to be necessary to attain the desired fault tolerance and also for keeping the clock distribution in a single-rail fashion. Among the key measures for achieving a robust and efficient solution were the reduction of the problem from an absolute to a relative comparison, the differential transmission of the counter values, the realization of the required +/− Counter by means of elastic pipelines, the separated treatment of rising and falling ticks to facilitate the interlocking, and the masking of glitches during idle phases of the threshold gates.

We have reported on the implementation and measurement results obtained with a CMOS ASIC design of the DARTS concept. In this context we have identified the threshold gates as the major contributors to area consumption. Beyond serving as a proof of concept, our experiments have investigated the clocking scheme's behavior in terms of clock stability, clock jitter, precision of synchronization, and fault tolerance. Overall the expectations from the theoretical models could be confirmed.

Although the DARTS approach as it is already represents an attractive solution for fault-tolerant clock generation, we are still considering a lot of future improvements. One of these is the use of a weaker fault model in order to reduce the area and improve the scaling with the number n of nodes. Other ideas include the pipelining of clock ticks to speed up the clock frequency or to build an efficient global communication scheme on top of the DARTS clocks that is metastability-free by construction.

Acknowledgments

This work originates in the *DARTS* (Distributed Algorithms for Robust Tick-Synchronization) project, which has been a joint effort of Vienna University of Technology and RUAG Space, see http://ti.tuwien.ac.at/darts for details. The authors would like to thank all DARTS projects members from RUAG Space GmbH as well as the Vienna University of Technology, specifically Ulrich Schmid for initiating the DARTS project and pushing it forward all the time. Many thanks also go to Matthias Függer for his never ending enthusiasm and his great job on the formal aspects of the novel tick generation approach. Last but not least they are grateful for the support by the Austrian bm:vit via FIT-IT project grant *DARTS* (809456-SCK/SAI) and the Austrian FWF project Theta (P17757).

References

[1] G. Moore, "Progress in digital integrated electronics," in *Proceedings of the Technical Digest of IEEE International Electron Devices Meeting (IEDM '75)*, pp. 11–13, 1975.

[2] "International technology roadmap for semiconductors," 2009, http://www.itrs.net/.

[3] L. Wissel, S. Pheasant, R. Loughran, C. LeBlanc, and B. Klaasen, "Managing soft errors in ASICs," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 85–88, May 2002.

[4] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design and Test of Computers*, vol. 22, no. 3, pp. 258–266, 2005.

[5] D. Rossi, M. Omaña, F. Toma, and C. Metra, "Multiple transient faults in logic: an issue for next generation ICs?" in *Proceedings of the 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '05)*, pp. 352–360, October 2005.

[6] A. Narasimhan and R. Sridhar, "Impact of variability on clock skew in H-tree clock networks," in *Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED '07)*, pp. 458–463, March 2007.

[7] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 665–692, 2001.

[8] P. J. Restle, C. A. Carter, J. P. Eckhardt et al., "The clock distribution of the power4 microprocessor," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '02)*, vol. 2, pp. 144–137, February 2002.

[9] M. Omaña, D. Rossi, and C. Metra, "Fast and low-cost clock deskew buffer," in *Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '04)*, pp. 202–210, October 2004.

[10] D. W. Dobberpuhl, R. T. Witek, R. Allmon et al., "A 200-MHz 64-b dual-issue CMOS microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, pp. 1555–1567, 1992.

[11] S. D. Naffziger, G. Colon-Bonet, T. Fischer, R. Riedlinger, T. J. Sullivan, and T. Grutkowski, "The implementation of the itanium 2 microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1448–1460, 2002.

[12] W. Hu, R. Wang, Y. Chen et al., "Godson-3B: a 1 GHz 40W 8-core 128GFLOPS processor in 65 nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC '11)*, pp. 76–77, 2011.

[13] J. Calamia, "China's godson gamble," *IEEE of Spectrum*, vol. 48, no. 5, pp. 14–16, 2011.

[14] N. Seifert, P. Shipley, M. D. Pant, V. Ambrose, and B. Gill, "Radiation-induced clock jitter and race," in *Proceedings of the 43rd Annual IEEE International Reliability Physics Symposium*, pp. 215–222, April 2005.

[15] D. M. Chapiro, *Globally-asynchronous locally-synchronous systems*, Ph.D. thesis, Stanford University, Palo Alto, Calif, USA, 1984.

[16] R. Ginosar, "Fourteen ways to fool your synchronizer," in *Proceedings of the 9th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC '03)*, pp. 89–96, May 2003.

[17] J. Muttersbach, T. Villiger, and W. Fichtner, "Practical design of globally-asynchronous locally-synchronous systems," in *Proceedings of the 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC '00)*, pp. 52–59, April 2000.

[18] R. Dobkin, R. Ginosar, and C. P. Sotiriou, "Data synchronization issues in GALS SoCs," *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, vol. 10, pp. 170–179, 2004.

[19] M. S. Maza and M. L. Aranda, "Interconnected rings and oscillators as gigahertz clock distribution nets," in *Proceedings of the 13th ACM Great Lakes Symposium on VLSI (GLSVLSI '03)*, pp. 41–44, 2003.

[20] M. S. Maza and M. L. Aranda, "Analysis and verification of interconnected rings as clock distribution networks," in *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI '04)*, pp. 312–315, 2004.

[21] S. Fairbanks, "Method and apparatus for a distributed clock generator," 2004, US patent no. US2004108876, http://v3.espacenet.com/textdoc?DB=EPODOCn&IDX=US2004108876

[22] S. Fairbanks and S. Moore, "Self-timed circuitry for global clocking," in *Proceedings of the 7th International Symposium on Asynchronous Circuits and Systems*, pp. 86–96, March 2005.

[23] S. Hauck, "Asynchronous design methodologies: an overview," *Proceedings of the IEEE*, vol. 83, no. 1, pp. 69–93, 1995.

[24] A. J. Martin, M. Nyström, K. Papadantonakis et al., "The Lutonium:a sub-nanojoule asynchronous 8051 microcontroller," in *Proceedings of the 9th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC '03)*, pp. 14–23, May 2003.

[25] C. Uri, "Terabit crossbar switch core for multi-clock-domain SoCs," in *Proceedings of the 15th Symposium on High Performance Chips (HOT CHIPS '03)*, p. 102ff, 2003.

[26] J. Dama and A. Lines, "GHz asynchronous SRAM in 65nm," in *Proceedings of the 15th International Symposium on Asynchronous Circuits and Systems (ASYNC '09)*, pp. 85–94, May 2009.

[27] A. J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," in *AUSCRYPT '90 Proceedings of the sixth MIT Conference on Advanced Research in VLSI*, pp. 263–278, MIT Press, Cambridge, Mass, USA, 1990.

[28] W. Jang and A. J. Martin, "SEU-tolerant QDI circuits," in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC '05)*, pp. 156–165, March 2005.

[29] W. Friesenbichler, T. Panhofer, and M. Delvai, "Improving fault tolerance by using reconfigurable asynchronous circuits," in *Proceedings of the IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS '08)*, pp. 267–270, April 2008.

[30] M. Delvai, *Design of an asynchronous processor based on code alternation logic—treatment of non-linear data paths*, Ph.D. thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2005.

[31] D. Dolev, J. Y. Halpern, and H. R. Strong, "On the possibility and impossibility of achieving clock synchronization," *Journal of Computer and System Sciences*, vol. 32, no. 2, pp. 230–250, 1986.

[32] T. Polzer, T. Handl, and A. Steininger, "A metastability-free multi-synchronous communication scheme for SoCs," in *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS '09)*, vol. 5873 of *Lecture Notes in Computer Science*, pp. 578–592, 2009.

[33] M. Függer, U. Schmid, G. Fuchs, and G. Kempf, "Fault-tolerant distributed clock generation in VLSI systems-on-chip," in *Proceedings of the 6th European Dependable Computing Conference (EDCC '06)*, pp. 87–96, IEEE Computer Society Press, October 2006.

[34] L. Lamport, "The mutual exclusion problem: part I–the theory of interprocess communication," *Journal of the ACM*, vol. 33, no. 2, pp. 313–326, 1986.
[35] L. Lamport, "Arbitration-free synchronization," *Distributed Computing*, vol. 16, no. 2-3, pp. 219–237, 2003.
[36] T. K. Srikanth and S. Toueg, "Optimal clock synchronization," *Journal of the ACM*, vol. 34, no. 3, pp. 626–645, 1987.
[37] J. Widder and U. Schmid, "The Theta-Model: achieving synchrony without clocks," *Distributed Computing*, vol. 22, no. 1, pp. 29–47, 2009.
[38] G. Fuchs, *Fault-tolerant distributed algorithms for on-chip tick generation: concepts, implementations and evaluations*, Ph.D. thesis, Vienna University of Technology, Fakultät für Informatik, Vienna, Austria, August 2009.
[39] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design*, DIMES, 2001.
[40] M. Ferringer, G. Fuchs, A. Steininger, and G. Kempf, "VLSI implementation of a fault-tolerant distributed clock generation," in *Proceedings of the 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '06)*, pp. 563–571, October 2006.
[41] D. L. Black, "On the existence of delay-insensitive fair arbiters: trace theory and its limitations," *Distributed Computing*, vol. 1, no. 4, pp. 205–225, 1986.
[42] A. J. Martin, "Compiling communicating processes into delay-insensitive VLSI circuits," *Distributed Computing*, vol. 1, no. 4, pp. 226–234, 1986.
[43] I. E. Sutherland, "Micropipelines," *Communications of the ACM, Turing Award*, vol. 32, no. 6, pp. 720–738, 1989.
[44] K. van Berkel, "Beware the isochronic fork," *Integration, the VLSI Journal*, vol. 13, no. 2, pp. 103–128, 1992.
[45] G. Fuchs, J. Grahsl, U. Schmid, A. Steininger, and G. Kempf, "Threshold Modules—Die Schlüsselelemente zur Verteilten Generierung eines Fehlertoleranten Taktes," in *Proceedings of the Austrian National Conference on the Design of Integrated Circuits and Systems (Austrochip '06)*, pp. 149–156, Vienna, Austria, October 2006.
[46] S. M. Nowick and C. W. O'Donnell, "On the existence of hazard-free multi-level logic," in *Proceedingsof the International Symposium on Asynchronous Circuits and Systems*, pp. 109–120, May 2003.
[47] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison Wesley Longman, Boston, Mass, USA, 1985.
[48] M. Fuegger, U. Schmid, G. Fuchs, A. Steininger, G. Kempf, and M. Sust, "Fault-tolerant distributed tick generation in VLSI systemson-chip," Tech. Rep. 53/2009, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2009.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 139183, 9 pages
doi:10.1155/2011/139183

Review Article

Design Considerations for Autocalibrations of Wide-Band ΔΣ Fractional-N PLL Synthesizers

Jaewook Shin and Hyunchol Shin

High-Speed Integrated Circuits and Systems Laboratory, Kwangwoon University, Seoul 139-701, Republic of Korea

Correspondence should be addressed to Hyunchol Shin, hshin@kw.ac.kr

Received 27 April 2011; Accepted 5 July 2011

Academic Editor: Kenichi Okada

Autocalibration of VCO frequency and loop gain is an essential process in PLL frequency synthesizers. In a wide tuning-range fractional-N PLL frequency synthesizer, high-speed and high-precision automatic calibration is especially important for shortening the lock time and improving the phase noise. This paper reviews the design issues of the PLL auto-calibration and discusses on the limitations of the previous techniques. A very simple and efficient auto-calibration method based on a high-speed frequency-to-digital converter (FDC) is proposed and verified through simulations. The proposed method is highly suited for a very wide-band ΔΣ fractional-N PLL.

1. Introduction

ΔΣ fractional-N RF frequency synthesizer is an essential building block in modern wireless communication systems. Achieving wide frequency tuning range, low phase noise, and fast locking time is challenging especially as the required tuning range becomes wider. If a single tuning curve is used to cover total tuning range, the VCO gain K_{VCO} of an LC VCO needs to be extremely large as illustrated in Figure 1(a). As a result, the phase noise and spur performances will be unacceptably poor. This is why multiple subband tuning curves via a switched capacitor bank are usually needed to cover the wide tuning range as well as keep K_{VCO} low, as shown in Figure 1(b) [1]. In this case, if the total capacitance of the switched capacitor bank varies in linear proportion to the cap bank code n, the K_{VCO} and the sub-band spacing f_{step} will vary in a cubic power of the f_{VCO} variation [2]. For example, K_{VCO} and f_{step} vary eight times if f_{VCO} varies two times. Due to such a wide variation of the VCO tuning characteristics, the PLL's essential performance parameters such as loop gain, lock time, and phase noise vary much, which is usually undesirable. Therefore, a fast and accurate autocalibration is needed for a PLL frequency synthesizer.

Finding a sub-band tuning curve that is the closest to a target frequency is referred to as the VCO frequency calibration. There had been many methods reported in the literature [3–10]. However, they all showed severe speed-resolution limitation. Meanwhile, maintaining the loop gain constant over the entire tuning range is referred to as the loop gain calibration. It is required in order to have a constant loop bandwidth and low phase noise. Previous techniques of this also showed slow calibration time due to the closed loop operation [11, 12].

This paper reviews the previous calibration techniques and examines their limitations. Based on the considerations, a new autocalibration technique based on a high-speed FDC is presented, which is highly suited to a wide tuning range ΔΣ fractional-N synthesizer.

2. Previous Techniques and Design Considerations

2.1. VCO Frequency Calibration. The most critical issue for the VCO frequency calibration is the accuracy and speed, that is, the calibration resolution and time. The calibration time should be fast enough because it will enlarge the total lock time, and the calibration resolution should be at least better than $f_{step}/2$ in order to be able to resolve two adjacent sub-band tuning curves. Note that f_{step} can be easily smaller than f_{REF} in a fractional-N PLL. Thus, the calibration circuit should be able to provide sub-f_{REF}

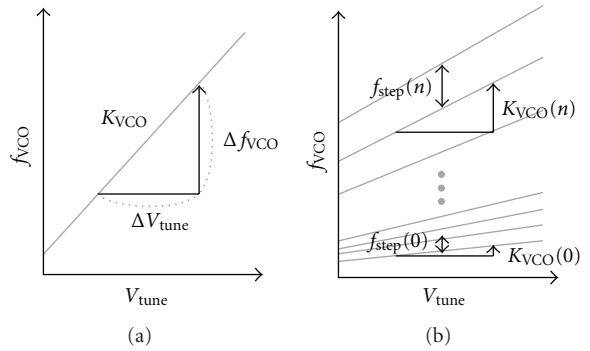


Figure 1: Frequency tuning characteristics in a wide band VCO. (a) Single tuning curve. (b) Multiple tuning curves with $K_{VCO}(n)$ and $f_{step}(n)$ variation.

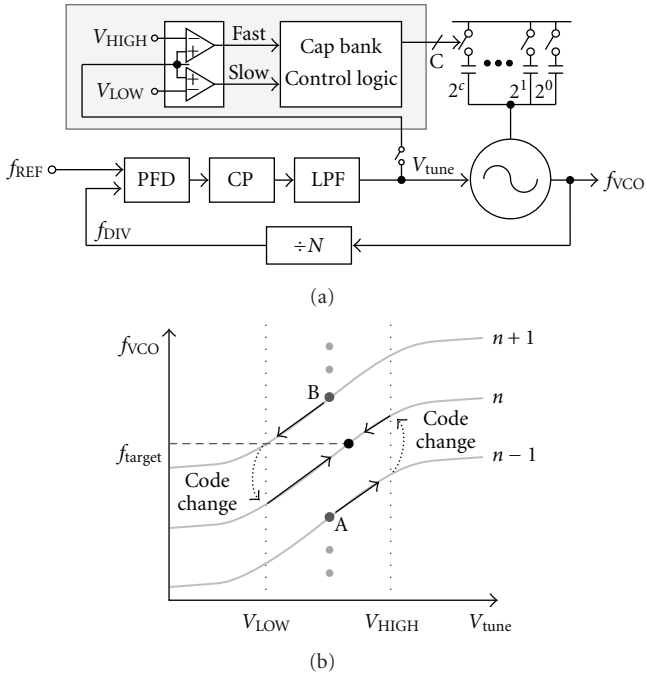


Figure 2: VCO frequency calibration by V_{tune} monitoring. (a) Architecture. (b) Operation principle [4–6].

calibration resolution while the calibration time is as fast as possible.

The tuning voltage monitoring technique as shown in Figure 2(a) was the oldest technique [4–6]. It monitors the VCO tuning voltage V_{tune} to find if it goes out of the predefined range between V_{HIGH} and V_{LOW}. If it goes out of the range, the cap bank code is automatically adjusted. Figure 2(b) illustrates the operation procedure. First, the calibration starts at point A on code $n-1$. Then, V_{tune} goes up toward V_{HIGH} due to the high f_{target}. When V_{tune} exceeds V_{HIGH}, the calibration circuit changes the cap bank code from $n-1$ to n. Then, PLL settles into the lock state with f_{VCO} approaching f_{target}. If the calibration process starts from point B on code $n+1$, similar process is carried out to reach the cap bank code n. This method always maintains the closed-loop lock state during the calibration process. Because of the closed-loop operation, this method usually requires several hundreds of μsec for the calibration, which is the major drawback of this method. This method, however, can be useful when it is used as an auxiliary lock maintaining tool to deal with the unwanted VCO tuning characteristic variation during the normal closed-loop operation of PLL [6].

Next the most popular VCO calibration technique is the "relative" frequency comparison method [7, 8]. The structure is shown in Figure 3(a). It compares the counted values of two clock signals f_{REF} and f_{VFC} during $k \cdot T_{REF}$ counting period. As illustrated in Figure 3(b), f_{VFC} ($= f_{VCO}/M$) is counted during $k \cdot T_{REF}$. Then, the comparator simply compares the two values and generates Fast/Slow flag signal, and then the control logic changes cap bank code accordingly. During the calibration, PLL stays in the open-loop state and V_{tune} is fixed at $V_{DD}/2$. The frequency resolution of this technique is given by $M \cdot f_{REF}/k$. As in the usual implementation, if M is the PLL's total division ratio N, the frequency resolution will be f_{VCO}/k. Thus, k must be higher than N in order to have sub-f_{REF} resolution, and higher k leads to a longer calibration. Thus, the typical calibration time of this method usually reaches several tens of μsec or even hundreds of μsec, which is also a major drawback of this method.

Another very fast but more analog intensive method is the time-to-voltage converter- (TVC-) based "relative" period comparison technique [9, 10]. The schematic structure is shown in Figure 4(a). It compares the voltages V_{C1} and V_{C2} that are proportional to the periods of the two incoming clock signals f_{REF} and f_{VCO}/N (Figure 4(b)). This method is usually very fast and only requires a few μsec or even sub-μsec. But due to the analog circuitry involved in the comparison process, its accuracy can be easily affected by the circuit mismatches and PVT variations. In addition, it is not suitable to a $\Delta\Sigma$ fractional-N PLL. Figure 5 illustrates how the f_{DIV} clock edges can vary when $\Delta\Sigma$ modulator is activated to provide a fractional division ratio $N \cdot f$. Assuming an MASH-111 is used for the $\Delta\Sigma$ modulator, f_{DIV} clock edge can be anywhere between $(N-3) \cdot T_{VCO}$ and $(N+4) \cdot T_{VCO}$ depending on the $\Delta\Sigma$ modulator output values, where T_{VCO} is a VCO signal period in lock state. Therefore, if the TVC-based technique is applied to a $\Delta\Sigma$ fractional-N PLL, a multiple period integration of the TVC output is needed, which will not only make the calibration time slow but also degrade the calibration accuracy.

Another issue in the VCO frequency calibration methods is about the code selection algorithm. Note that the conventional simple binary search process only gives an odd numbered code as a final result [7]. In Figure 6, code 9 would be selected as a final code by the conventional simple binary search process although even-numbered code 10 is the closest to the target frequency f_1. In order to overcome this, an improved method in [10] compares the last two searched codes. But this algorithm will also fail if the closest code is in multiples of 4. This limitation is explained in Figure 6. This method is not a problem for the target frequency f_2, for which the closest code 2 will be selected. But for the target

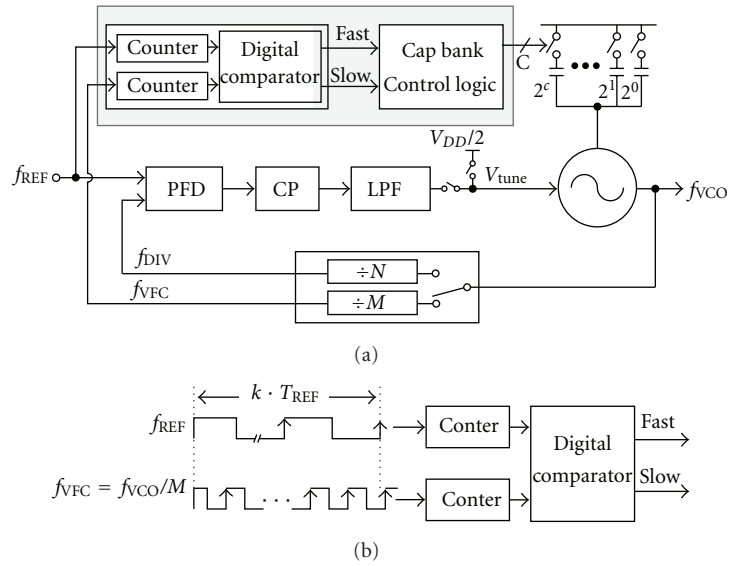


Figure 3: Relative frequency comparison VCO frequency calibration. (a) Architecture. (b) Operation principle [7, 8].

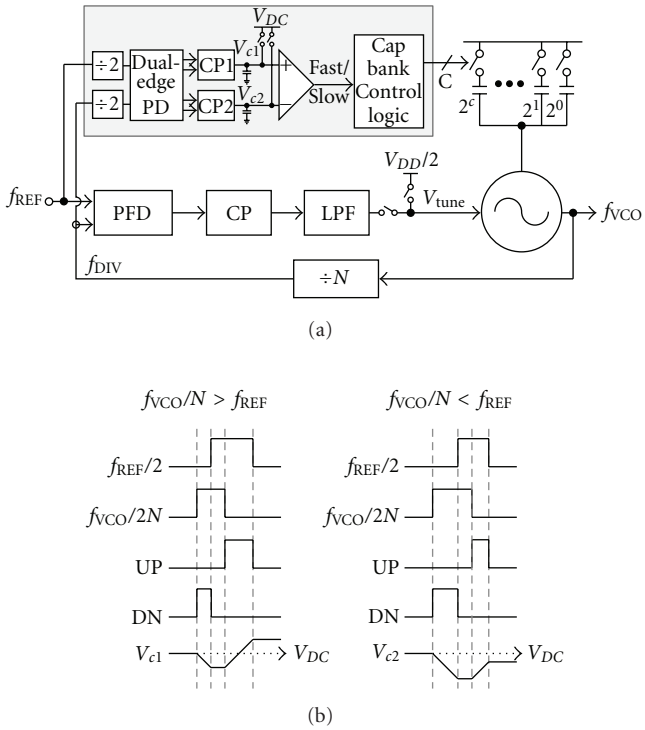


Figure 4: Relative period comparison VCO frequency calibration. (a) Architecture. (b) Operation principle [9, 10].

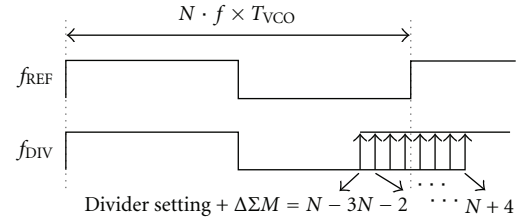


Figure 5: f_{DIV} clock period jittering at the lock state with a $\Delta\Sigma$ modulator activated in a $\Delta\Sigma$ fractional-N PLL.

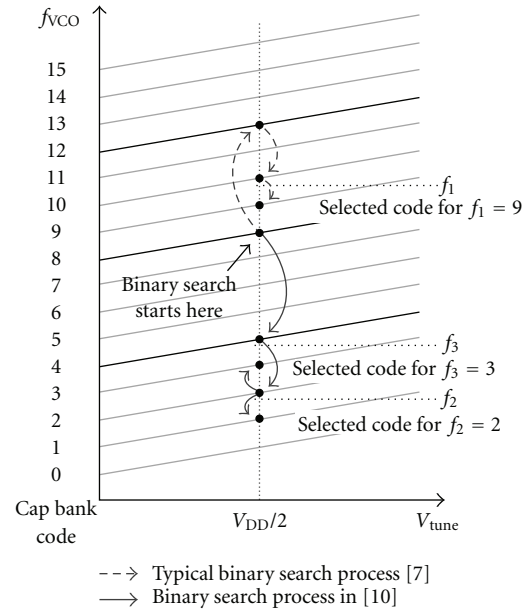


Figure 6: Final code selection of binary search process for various target frequencies in previous techniques. For f_1, typical binary search process is used [7], and improved algorithm in [10] is applied for f_2 and f_3.

frequency f_3, the method in [10] will provide the less optimal code 3 rather than the optimal code 4. Therefore, a truly closest code selection algorithm must be able to store the frequency error for all searched codes during the calibration process and finally produce the closest code to the target frequency.

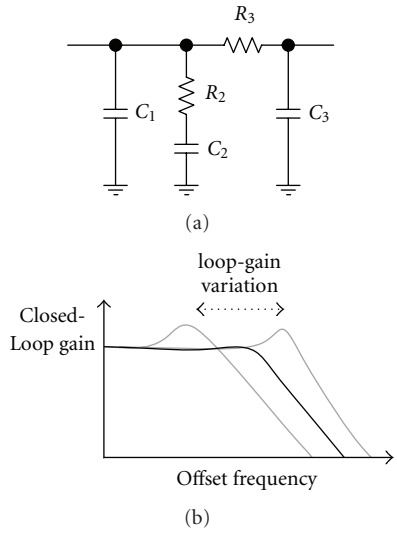


Figure 7: (a) A third-order passive loop filter. (b) Closed-loop gain and bandwidth variation in PLL.

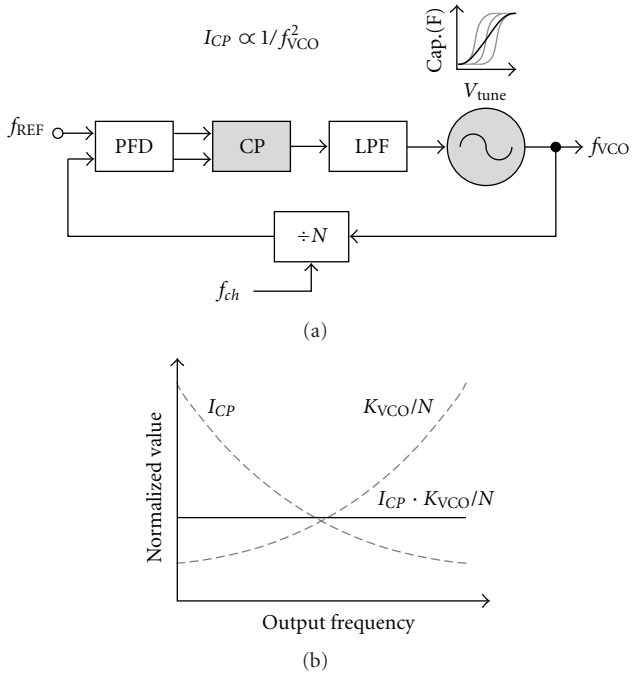


Figure 8: Loop gain compensation by preset pattern of charge pump gain I_{CP}. (a) Architecture. (b) Operation principle [13].

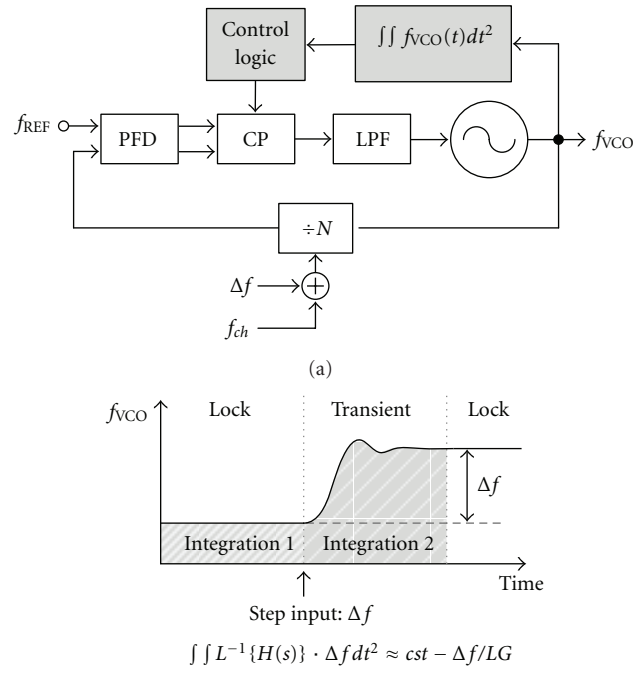


Figure 9: Loop gain calibration by measuring step-response of PLL (a) Architecture. (b) Operation principle [10, 11].

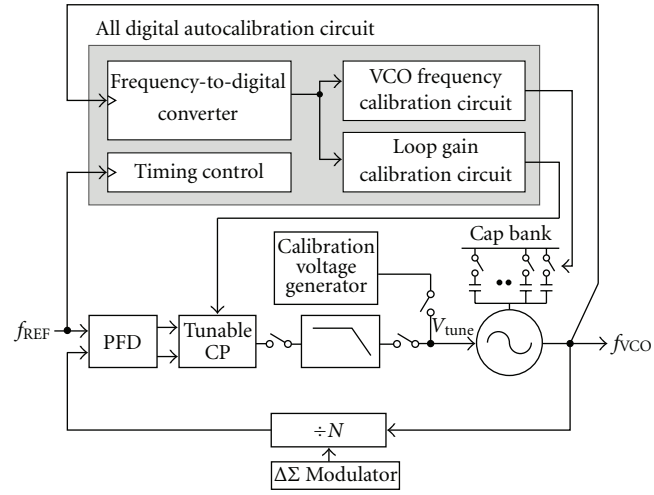


Figure 10: PLL block diagram with the FDC-based autocalibration circuit.

2.2. Loop Gain Calibration. A charge pump PLL generally uses the third-order passive loop filter as shown in Figure 7(a). It is known that the loop gain is proportional to $I_{CP} \cdot K_{VCO} \cdot R_2/N$, the lock time is proportional to $\sqrt{(I_{CP} \cdot K_{VCO} \cdot R_2/N)}$, and the integrated phase noise is inversely proportional to $I_{CP} \cdot K_{VCO} \cdot R_2/N$. Thus, the variation of each parameter will cause significant variation of the PLL closed-loop performances such as the loop gain, the lock time, and the phase noise. Figure 7(b) depicts the loop bandwidth variation due to the loop gain variation. The effect of I_{CP} variation on the loop gain can be minimized by using the same-type on-chip resistor for the loop filter R_2 and the I_{CP} reference voltage generation [15]. The division ratio N is a digital value, so its impact on the loop gain can be precisely predicted and accurately compensated. Meanwhile, the variation of K_{VCO} due to PVT variation cannot be accurately predicted. Therefore, the loop gain calibration circuit must be able to gauge the K_{VCO} variation

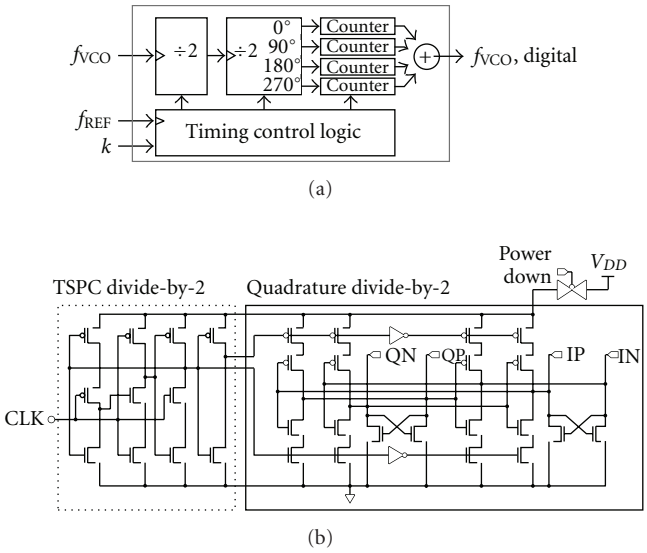


Figure 11: (a) Detail block diagram of the FDC. (b) Circuit schematic of quadrature-phase predivider having division ratio of 4.

accurately, which should be equivalent to the loop gain variation.

Figure 8(a) shows the previous loop gain compensation method [13]. Note that the term "compensation" is used here instead of the preferred "calibration." The basic idea of this compensation is to make the charge pump gain follow a preset way such that it is inversely proportional to $f_{VCO}{}^2$. Figure 8(b) illustrates the operation principle of this compensation. To compensate K_{VCO}/N variation that is proportional to $f_{VCO}{}^2$ in an LC tuned VCO, the charge pump gain is controlled to be inversely proportionally to $f_{VCO}{}^2$. Then, the loop gain that is proportional to $I_{CP} \cdot K_{VCO}/N$ will stay at a constant value, resulting in the constant loop bandwidth. But since this method cannot deal with any unexpected PVT variation of K_{VCO}, its effect would be limited after the chip fabrication.

The limitation of the previous method in [13] comes from the absence of the on-chip gauge of the loop gain. The on-chip measurement of the loop gain and the use of this value for loop gain calibration should be a very effective approach. It is known that PLL's closed-loop time-domain step response is closely related to the PLL's loop bandwidth. Previously, using the time-domain step-response to measure the loop gain and calibrate the charge pump gain was reported [11, 12]. Figure 9(a) shows the block diagram and Figure 9(b) shows the operating procedure. To calibrate loop gain, the two-step integrator integrates the VCO output signal in the locked state. After that, the step input Δf is triggered, and the integrator integrates the PLL's step-response. In [12], only the step-response is integrated to reduce the calibration time. The drawback of this calibration method is a long calibration time, that is almost up to several tens of μsec due to the closed-loop operation. In addition, the accuracy might be affected by the K_{VCO} variation during the transient state. Thus, we need

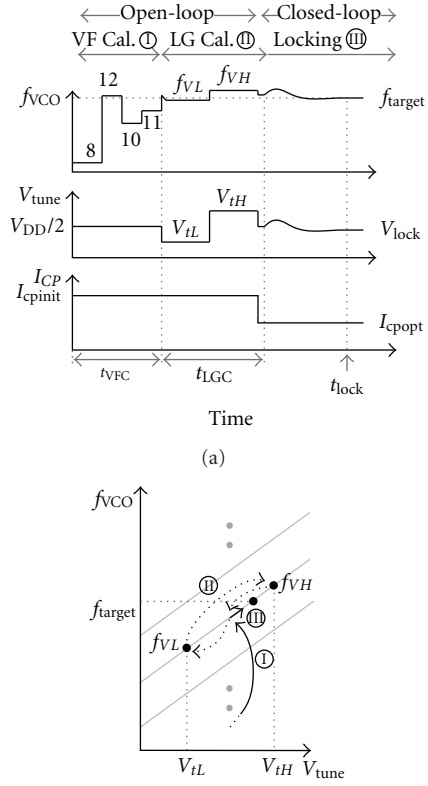


Figure 12: (a) Transitions of f_{VCO}, V_{tune}, and I_{CP} during calibration and locking process. (b) Transition of f_{VCO} during calibration.

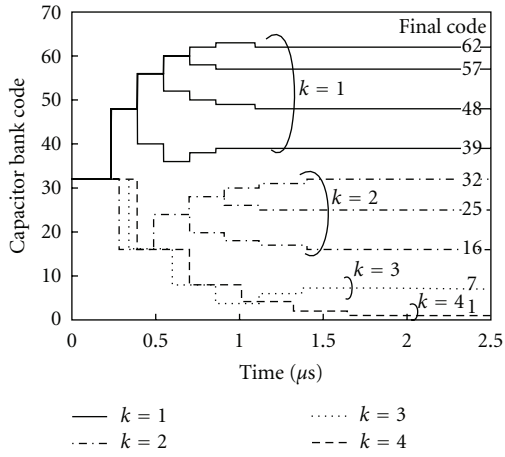


Figure 13: Behavioral simulation result of the VCO frequency calibration with k-value varying from 1 to 4 according to the required resolution.

more accurate and fast on-chip K_{VCO} measurement tool for the loop gain calibration.

2.3. Linearization of VCO Coarse Tuning Characteristics. The variation range of K_{VCO} and f_{step} are known to be

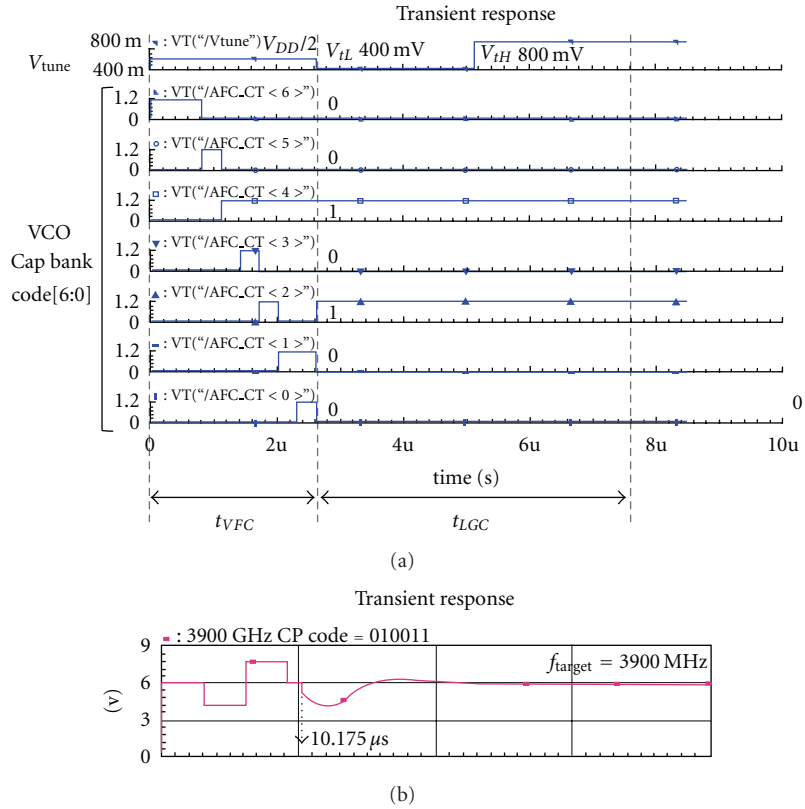


FIGURE 14: Behavioral simulation results for the proposed autocalibration. (a) V_{tune} and VCO cap bank code during autocalibration. (b) V_{tune} for the total locking process for 3900 MHz target frequency.

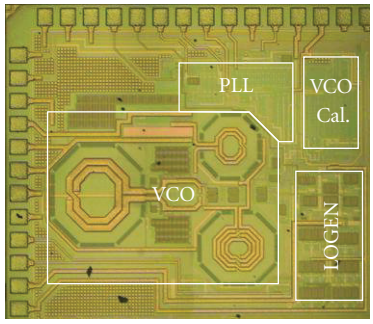


FIGURE 15: Chip micrograph of the ΔΣ fractional-N PLL synthesizer including only the VCO frequency calibration circuit [14].

proportional to the cubic power of the f_{VCO} total tuning range as the following equation:

$$\frac{K_{VCO,max}}{K_{VCO,min}} = \frac{f_{step,max}}{f_{step,min}} = \left(\frac{f_{VCO,max}}{f_{VCO,min}}\right)^3. \quad (1)$$

The large variation of K_{VCO} and f_{step} degrades the speed and accuracy of the loop gain and VCO frequency calibration processes. Therefore, reducing the total variation of K_{VCO} and f_{step} across the entire tuning range is desirable for improving the calibration result. Several studies were reported to reduce the K_{VCO} and f_{step} variation in LC VCO: authors of [16, 17] only addressed the K_{VCO} issue, and authors of [18] proposed a new cap bank structure to reduce the K_{VCO} and f_{step} variations together, but it relied on arbitrary fractional scaling of the capacitance value over the sixteen tuning curves, which made it difficult to apply the method to an even wider tuning range and more tuning curves. Thus, we need more simple and systematic design method of the capacitor bank structure for reducing the K_{VCO} and f_{step} variations across the total tuning range.

3. FDC-Based Autocalibration Technique

The limitations of the previous calibration technique can be overcome by the proposed FDC-based autocalibration technique, which is described in this section. Figure 10 shows the architecture of the fractional-N PLL with the proposed FDC-based all-digital autocalibration scheme. The calibration circuit comprises a high-speed FDC, the VCO frequency calibration circuit, the loop gain calibration circuit and the auxiliary timing control logic, and calibration voltage generator.

The high-speed FDC enables the fast calibration of VCO frequency and loop gain. The FDC operates in the time period of $k \cdot T_{REF}$ to convert the VCO frequency to a digital value with a conversion resolution of f_{REF}/k whereas the conventional technique in Figure 3 requires N times longer time $N \cdot k \cdot T_{REF}$ for acquiring the same

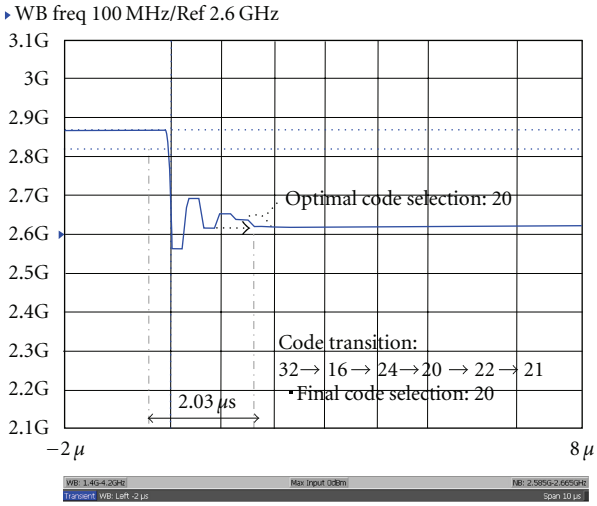


Figure 16: Measured VCO frequency during the VCO frequency calibration.

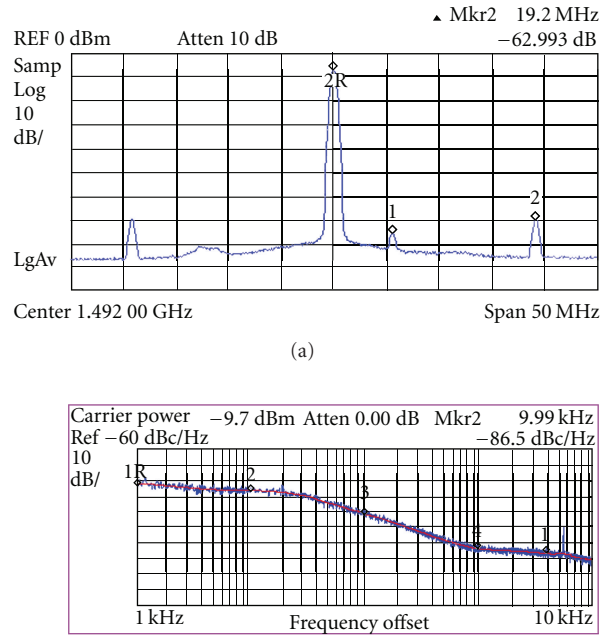


Figure 17: Measured PLL results at 1492 MHz. (a) Output spectrum. (b) Phase noise.

resolution. The detailed block diagram of the FDC is shown in Figure 11(a). The FDC accepts the VCO signal and divides it down to $f_{VCO}/4$ by using a quadrature-phase predivider, which is composed of a true single phase clock (TSPC) divide-by-2 and a quadrature-phase divide-by-2 as shown in Figure 11(b). After the predivider, the subsequent four counters that is connected to each of the quad-phase output signals count the VCO frequency. As a result, each counter can operate at a reduced speed of $f_{VCO}/4$ while the overall frequency resolution is not diminished. Finally the sum of the four counter outputs represents the VCO frequency.

The frequency error is generated by subtracting the target f_{target} digital code from the current f_{VCO} digital code. The resulting sign bit is used as Fast/Slow flag for the binary search process. The absolute value of the frequency error is utilized for the final optimal code selection process to find the final code that has shown the minimum frequency error. The more detailed description of the proposed VCO frequency calibration technique can be found in [14]. After the VCO frequency calibration is completed, the loop gain calibration process begins. It is basically based on the on-chip measurement of K_{VCO}. The FDC extracts two VCO frequencies f_{VH} and f_{VL} at two different tuning voltages V_{tH} and V_{tL}. Then, K_{VCO} is computed digitally. With the computed K_{VCO}, the optimal charge pump gain is generated according to the relation $I_{CP} \sim N/K_{VCO}$. After these two autocalibration processes, the normal PLL locking process begins. Proper selection of V_{tL} and V_{tH} is important in the proposed calibration technique. In the chosen range, the charge pump current must be sufficiently constant for accurate loop gain calibration. At the same time, the VCO sub-band tuning curves must be sufficiently overlapping to ensure V_{tune} to remain in this range after the PLL locking. Considering these aspects, V_{tL} and V_{tH} are chosen to be 0.4 and 0.8 V, respectively, which is 0.4 V off from the 1.2-V supply voltage.

Figure 12(a) shows the calibration procedure by illustrating the waveforms of f_{VCO}, V_{tune}, and I_{CP}, and Figure 12(b) illustrates how the VCO output frequency wanders on the VCO tuning curve plane during the calibration process. Note that this illustration is only for 4-bit cap bank case. In the first VCO frequency calibration mode, the binary search is performed with V_{tune} fixed at $V_{DD}/2$. The final optimal code is fed to VCO at t_{VFC}. In the subsequent loop gain calibration mode, V_{tune} is sequentially switched from V_{tL} to V_{tH} to extract f_{VL} and f_{VH}, and finally producing K_{VCO}. After t_{LGC}, the optimal charge pump code I_{cpopt} is generated to control the loop bandwidth. After the two-step open-loop calibration is completed, the PLL loop is closed and a normal locking process starts.

In order to verify the proposed autocalibration method, various behavioral simulations were performed for a PLL built in a mixed-signal circuit environment including verilog codes and circuit elements. Figure 13 shows a simulation result of the VCO calibration with a 6-bit cap bank. Due to the widely varying sub-band spacing f_{step}, the frequency resolution is adjusted by varying k from 1 to 4. Thus, the calibration time varies from 1.05 to 2.03 μs. Figure 14(a) shows a simulated result of V_{tune} and VCO cap bank code during the full autocalibration process. V_{tune} is fixed at $V_{DD}/2$ during the first VCO frequency calibration, and the final code found is 0010100. At the subsequent loop gain calibration, V_{tune} is switched from 400 to 800 mV. In this time, the FDC extracts two frequencies to calculate K_{VCO}. The simulated loop gain calibration time is 5 μs. Figure 14(b) shows the V_{tune} waveform during the total locking process at the target frequency of 3900 MHz. All of the above

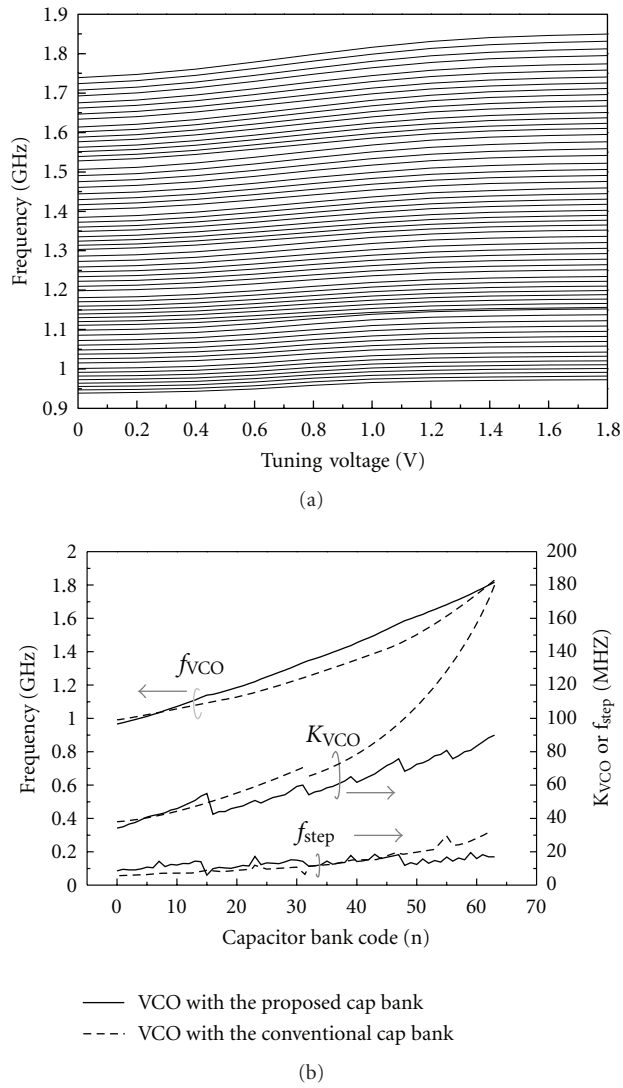


FIGURE 18: Measured VCO tuning characteristics. (a) Frequency tuning curves. (b) VCO output frequency, K_{VCO}, and f_{step} over the 64 tuning curves.

behavioral simulation results prove that the proposed FDC-based autocalibration method works successfully for the VCO frequency and loop gain calibrations.

A prototype chip including only the VCO frequency calibration circuit is fabricated in 0.13-μm CMOS technology [14]. The chip micrograph is shown in Figure 15. The die size including the pad frame is $1.33 \times 1.58\,\text{mm}^2$, and the active area of the VCO calibration circuit is $240 \times 400\,\mu\text{m}^2$. It dissipates 15.8 mA from a 1.2 V supply.

Figure 16 shows the measured f_{VCO} waveform during the VCO frequency calibration process at 2620 MHz. The measured calibration time is 2.03 μs, and the finally selected optimal code is 20, which show successful operation of the proposed optimal code selection algorithm. The PLL output spectrum at the output frequency of 1492 MHz is shown in Figure 17(a). The reference and fractional spurs are −63 and −68 dBc, respectively. The reference spur appears at the reference frequency of 19.2 MHz. Figure 17(b) shows the measured phase noise, which is −102.1 and −124.1 dBc/Hz at 100 kHz and 1 MHz offset, respectively. The in-band phase noise is −86.5 dBc/Hz at 10 kHz offset. And another prototype PLL chip including the full autocalibration capability has been fabricated and planned to be characterized.

The linearized coarse tuning characteristics are obtained by proposing a pseudoexponential capacitor bank structure [2] while the conventional cap bank produces a linear capacitance variation. Figure 18(a) shows the measured frequency tuning characteristics of the VCO with the proposed cap bank. Notice that the 64 sub-band tuning curves are evenly distributed. Figure 18(b) gives the plots for the f_{VCO}, K_{VCO}, and f_{step} over the 64 cap bank codes. For the sake of comparison, the measured results for a test VCO employing a conventional binary-weighted cap bank structure are shown together. It can be observed that the proposed cap bank structure substantially reduce the K_{VCO} variation from 38–180 (473%) to 34–90 (264%) MHz/V, and the f_{step} variation from 5.6–31 (553%) to 6–18 (300%) MHz/code. The results prove that the proposed pseudoexponential cap bank structure effectively linearizes the coarse tuning characteristics and reduces the K_{VCO} and f_{step} variations. It is found to be very instrumental for the autocalibration of a PLL having a wide-tuning range LC VCO.

4. Conclusion

Design considerations for the autocalibration of VCO frequency and loop gain are discussed for wide-band ΔΣ fractional-N PLL synthesizers. To overcome the limitations of the conventional techniques, the FDC-based all-digital autocalibration circuit is proposed. The FDC-based method remarkably shortens calibration time and improves calibration accuracy, and was found to be highly suitable for a wide band ΔΣ fractional-N PLL.

Acknowledgment

This paper has been supported by the ITRC Program (NIPA-2011-C1090-1111-0006).

References

[1] A. Kral, F. Behbahani, and A. A. Abidi, "RF-CMOS oscillators with switched tuning," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 555–558, May 1998.

[2] J. Kim, J. Shin, S. Kim, and H. Shin, "A wide-band CMOS *LC VCO* with linearized coarse tuning characteristics," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 5, pp. 399–403, 2008.

[3] W. B. Wilson, U. K. Moon, K. R. Lakshmikumar, and L. Dai, "A CMOS self-calibrating frequency synthesizer," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 10, pp. 1437–1444, 2000.

[4] T. H. Lin and W. J. Kaiser, "A 900 MHz 2.5 mA CMOS frequency synthesizer with an automatic SC tuning loop," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 3, pp. 424–431, 2001.

[5] A. Aktas and M. Ismail, "CMOS PLL calibration techniques," *IEEE Circuits and Devices Magazine*, vol. 20, no. 5, pp. 6–11, 2004.

[6] H. R. Lee, M. S. Hwang, B. J. Lee et al., "A 1.2 V-Only 900 mW 10 Gb ethernet transceiver and XAUI interface with robust VCO tuning technique," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 11, pp. 2148–2157, 2005.

[7] H. I. Lee, J. K. Cho, K. S. Lee et al., "A Σ-Δ fractional-N frequency synthesizer using a wide-band integrated VCO and a fast AFC technique for GSM/GPRS/WCDMA applications," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 7, pp. 1164–1169, 2004.

[8] M. Marutani, H. Anbutsu, M. Kondo, N. Shirai, H. Yamazaki, and Y. Watanabe, "An 18mW 90 to 770MHz synthesizer with agile auto-tuning for digital TV-tuners," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '06)*, pp. 191–192, February 2006.

[9] T. H. Lin and Y. J. Lai, "An agile VCO frequency calibration technique for a 10 GHz CMOS PLL," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 2, pp. 340–349, 2007.

[10] J. Lee, K. Kim, J. Lee, T. Jang, and S. Cho, "A 480-MHz to 1-GHz sub-picosecond clock generator with a fast and accurate automatic frequency calibration in 0.13-μm CMOS," in *Proceedings of the IEEE Asian Solid-State Circuits Conference, (A-SSCC '07)*, pp. 67–70, November 2007.

[11] Y. Akamine, M. Kawabe, K. Hori, T. Okazaki, M. Kasahara, and S. Tanaka, "ΔΣ PLL transmitter with a loop-bandwidth calibration system," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 497–506, 2008.

[12] H. Shanan, G. Retz, K. Mulvaney, and P. Quinlan, "A 2.4 GHz 2 Mb/s versatile PLL-based transmitter using digital pre-emphasis and auto calibration in 0.18μm CMOS for WPAN," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '09)*, February 2009.

[13] T. Wu, P. K. Hanumolu, K. Mayaram, and U. K. Moon, "Method for a constant loop bandwidth in LC-VCO PLL frequency synthesizers," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 2, Article ID 4768902, pp. 427–435, 2009.

[14] J. Shin and H. Shin, "A fast and high-precision VCO frequency calibration technique for wideband ΔΣ fractional-N frequency synthesizers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 7, Article ID 5371871, pp. 1573–1582, 2010.

[15] W. Rhee, H. Ainspan, D. J. Friedman, T. Rasmus, S. Garvin, and C. Cranford, "A uniform bandwidth PLL using a continuously tunable single-input dual-path LC VCO for 5 Gb/s PCI express Gen 2 application," in *Proceedings of the IEEE Asian Solid-State Circuits Conference, (A-SSCC '07)*, pp. 63–66, November 2007.

[16] D. Hauspie, E. C. Park, and J. Craninckx, "Wideband VCO with simultaneous switching of frequency band, active core, and varactor size," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 7, pp. 1472–1480, 2007.

[17] P. Väänänen, N. Mikkola, and P. Heliö, "VCO design with on-chip calibration system," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 10, pp. 2157–2166, 2006.

[18] L. Lu, J. Chen, L. Yuan, H. Min, and Z. Tang, "An 18-mW 1.1752-GHz frequency synthesizer with constant bandwidth for DVB-T tuners," *IEEE Transactions on Microwave Theory and Techniques*, vol. 57, no. 7, Article ID 4799232, pp. 928–937, 2009.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 361910, 7 pages
doi:10.1155/2011/361910

Research Article

A Tunable Wideband Frequency Synthesizer Using LC-VCO and Mixer for Reconfigurable Radio Transceivers

Yusaku Ito,[1] Kenichi Okada,[2] and Kazuya Masu[1]

[1] *ICE Cube Center, Tokyo Institute of Technology, Tokyo 226-8503, Japan*
[2] *Department of Physical Electronics, Tokyo Institute of Technology, Tokyo 152-8552, Japan*

Correspondence should be addressed to Kenichi Okada, okada@ssc.pe.titech.ac.jp

Received 2 May 2011; Accepted 6 June 2011

Academic Editor: Antonio Liscidini

This paper proposes a novel wideband LC-based voltage-controlled oscillator (VCO) for multistandard transceivers. The proposed VCO has a core LC-VCO and a tuning-range extension circuit, which consists of switches, a mixer, dividers, and variable gain combiners with a spurious rejection technique. The experimental results exhibit 0.98 to 6.6 GHz continuous frequency tuning with −206 dBc/Hz of $\mathrm{FoM_T}$, which is fabricated by using a 0.18 μm CMOS process. The frequency tuning range (FTR) is 149%, and the chip area is 800 μm × 540 μm.

1. Introduction

Recently, dozens of wireless communication standards have been used for small mobile terminals, for example, GSM, UMTS, LTE, WiMAX, WLAN, Bluetooth, UWB, GPS, DTV, and RFID, and the standards use several frequency bands spreading in a quite wide range such as 800 MHz to 6 GHz. The mobile terminals have been obtaining multistandard operations, smaller size, and lower power operation [12]. However, the present multistandard RF front end consists of several LNAs, VCOs, mixers, and PAs for each frequency band (Figure 1). A multistandard RF front end implemented in a single chip is required for smaller size, lower power, and more flexible wireless communication terminals such as 800 MHz to 6 GHz. The software defined radio (SDR) has been studied [9, 13], and the multistandard RF front end is also needed to realize the SDR with feasible power consumption. Several multistandard RF front ends have been proposed. Digital-assist architectures are suitable for Si CMOS chips [14, 15]. As a common component for the multistandard RF front ends, this paper proposes a wideband frequency synthesizer covering 0.98 GHz to 6.6 GHz [20].

2. Previous Work

Ring-oscillator-based VCOs have unacceptably large phase noise for the wireless communication while it has very wide frequency tuning range. Thus, LC-based VCOs are required for the application due to the phase noise requirement. However, the tuning range of LC-based VCOs is usually very narrow such as 2 to 3 GHz even through the 800 MHz-to-6 GHz tuning range is required for the multistandard RF front ends. The conventional LC-VCO cannot overcome the trade-off, so a new wideband LC-based VCO architecture has to be developed.

A VCO using switched capacitors is a well-known topology to extend the tuning range [7, 21], and a switched inductor and a variable active inductor are also utilized [8, 16]. However, these circuits have a trade-off between the phase noise and the tuning range. The VCO using a variable MEMS inductor achieves wide-tuning range with superior phase noise characteristics [18]. However, it is difficult for these pure CMOS VCOs to obtain wide-tuning range with adequate phase noise.

Recently, wideband VCOs for MB-OFDM UWB have been reported [1, 2, 4, 17, 22, 26], which use a tuning range extension technique using QVCO, dividers, and single-sideband mixer (SSBM). These VCOs achieve quite wide tuning range and high spurious rejection using SSBM with I/Q signals. However, the VCOs in [1, 2] use two oscillators and have large layout area and larger power consumption. Although the VCOs in [10, 22, 26] use only one QVCO, these VCOs also have larger phase noise and larger power consumption.

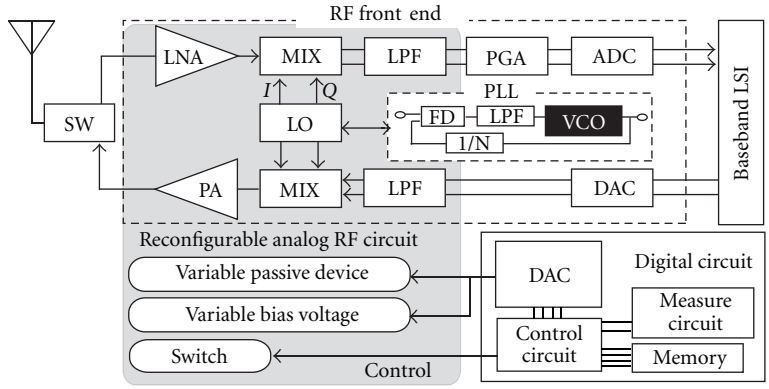


Figure 1: Concept of the reconfigurable RF circuit design.

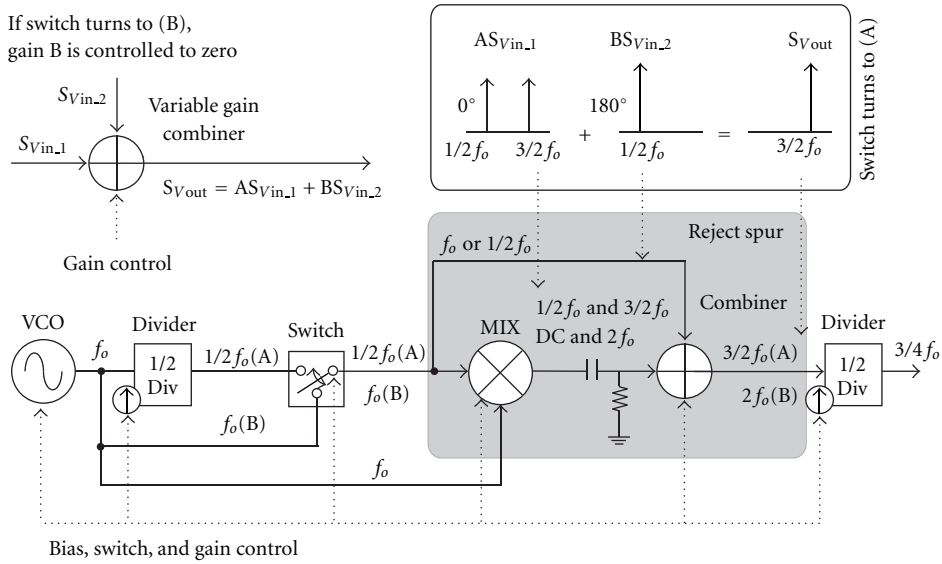


Figure 2: The proposed wideband VCO architecture.

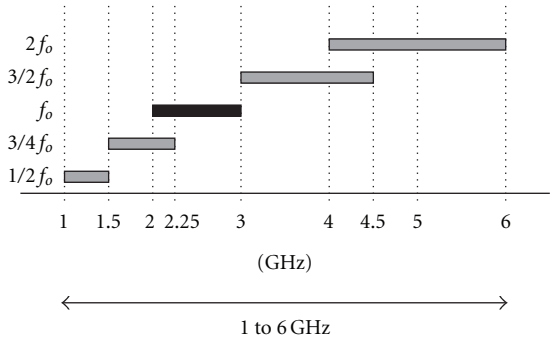


Figure 3: Frequency plan from 1 GHz to 6 GHz.

Wideband VCOs for multistandard transceivers are also reported [10, 13, 23]. The VCO in [10] use a QVCO and SSBMs, which also has larger phase noise and larger power consumption. The VCOs in [13, 23] use differential oscillators and 1/2 frequency dividers to avoid utilizing SSBM and the quadrature generation. The VCO in [13] uses two oscillators, and it requires, moreover, three oscillators for continuous frequency tuning. The VCO in [23] still requires two oscillators.

The wideband VCO proposed in [19] uses divide-by-2, divide-by-3, divide-by-4, divide-by-5, divide-by-6, divide-by-8, and divide-by-10 frequency dividers for the tuning range extension. This architecture requires a wideband QVCO, and continuous tuning cannot be realized in the measurement [19] because ±20% tuning range is difficult for QVCOs.

Various topologies for tuning range extension can be utilized depending on the required performances. In this paper, we propose a novel extension architecture to achieve wider tuning range with lower power, smaller layout area, and lower phase noise, which achieves ±71% of tuning range from a ±20%-range core VCO [20]. The proposed architecture utilizes a differential VCO to generate the

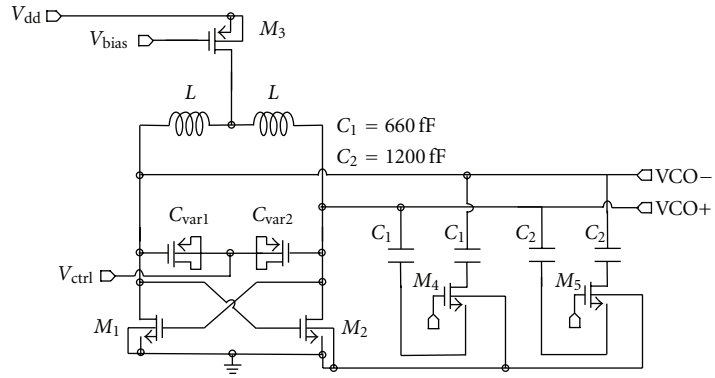
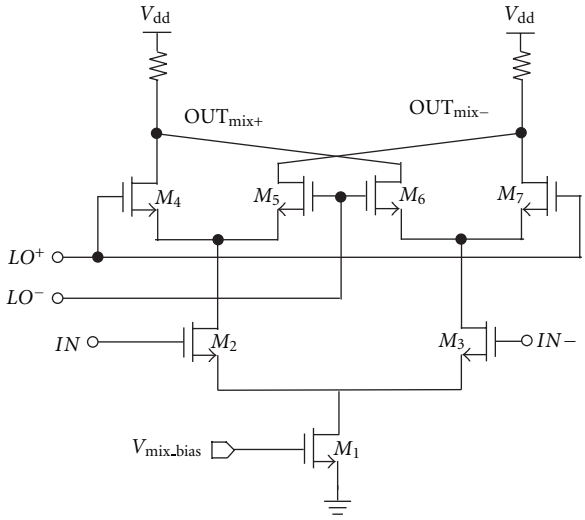
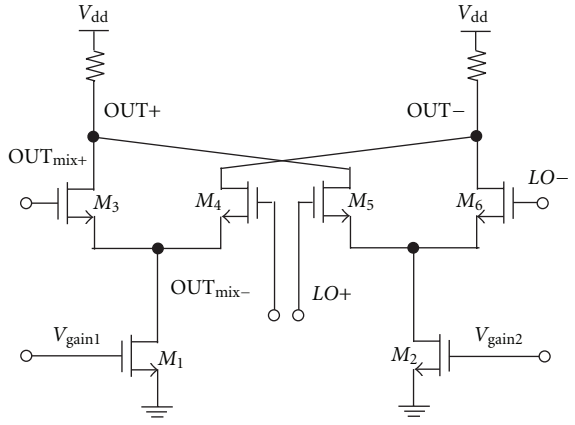


Figure 4: Schematics of core VCO using switched capacitors.



(a) wideband mixer



(b) variable gain combiner

Figure 5: Circuit schematics used in the proposed wideband VCO.

fundamental frequency with smaller layout area, lower power consumption, and lower phase noise characteristics than quadrature VCOs. A variable gain combiner is employed to reject spur instead of SSBM.

3. Wideband VCO Architecture

Figure 2 shows the proposed VCO architecture, which consists of a core VCO, two dividers, a switch, a mixer, a high-pass filter, and a combiner [20]. The proposed architecture aims to achieve wider tuning range with lower power and lower phase noise, so a differential VCO and a novel compact frequency extension circuit are introduced. Figure 3 shows frequency plan of the proposed architecture, and $2f_0$, $3/2f_0$, $3/4f_0$, and $1/2f_0$ are generated from the fundamental frequency f_0 of the core VCO. $2f_0$ is generated by the mixer, and $1/2f_0$ is divided from the fundamental frequency f_0. $3/2f_0$ is generated from f_0 and $1/2f_0$, and $1/2f_0$ is also generated as a spurious signal. $3/4f_0$ is divided from $3/2f_0$. The core VCO is required to have frequency tuning range of ±20%, and the total tuning range of ±71% can be realized by the frequency extension circuit. For example, tuning range of 2-3 GHz can be extended to 1–6 GHz as shown in Figure 3. Lower frequency can also be generated by a divide-by-2 frequency divider chain [3].

A differential VCO is employed as the core VCO. Figure 4 shows the schematic of the core VCO, and switched capacitors are utilized for coarse tuning. The differential VCO has better phase noise characteristic than the quadrature VCO, and smaller layout area and lower power consumption can also be achieved. The core VCO has frequency tuning range of more than ±20%. At higher frequencies, it is difficult to achieve wide tuning range due to parasitic capacitances, so the lower fundamental frequency is chosen and upconverted to higher frequencies by the mixer.

A CML divider is used as a wideband frequency divider to obtain 1/2 of input frequency, and a wideband mixer shown in Figure 5(a) is used as a frequency multiplier. The mixer is shared to generate $2f_0$ and $3/2f_0$, and input signal of mixer is switched as shown in Figure 2. In case (A) shown in Figure 2, mixer input signals are f_0 and $1/2f_0$, and $3/2f_0$ and $1/2f_0$ are generated. In case (B) shown in Figure 2, both mixer input signals are f_0, and DC and $2f_0$ are generated.

In case (A), $3/2f_0$ is the desired frequency and $1/2f_0$ is spurious frequency. The tuning range extension using SSBM requires I/Q phases to reject the spurious frequency. In the proposed architecture, output of the first divider has

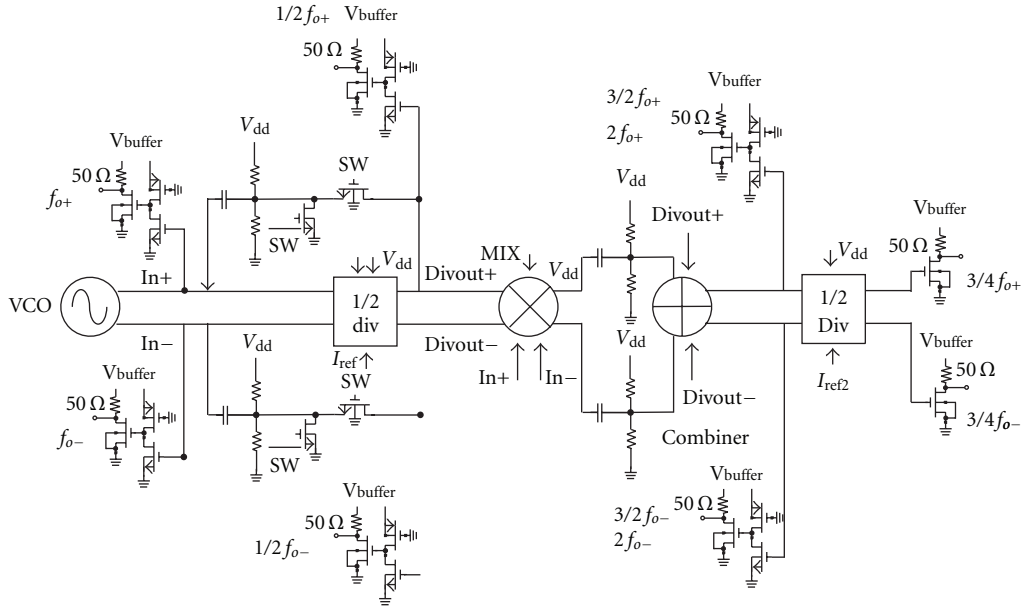


Figure 6: Block diagram of the proposed wideband VCO.

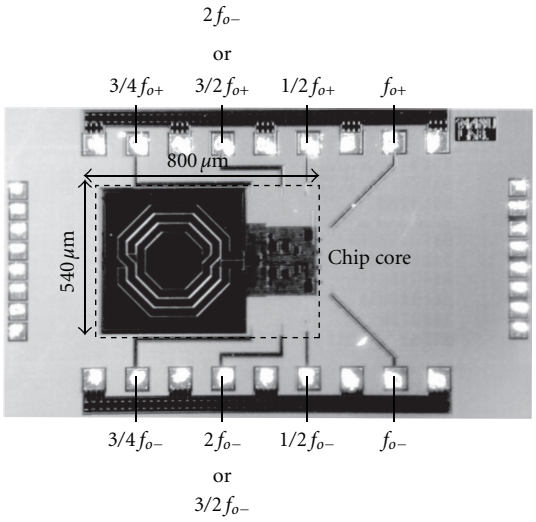


Figure 7: Chip micrograph of fabricated wideband VCO. Core size is 540 μm × 800 μm.

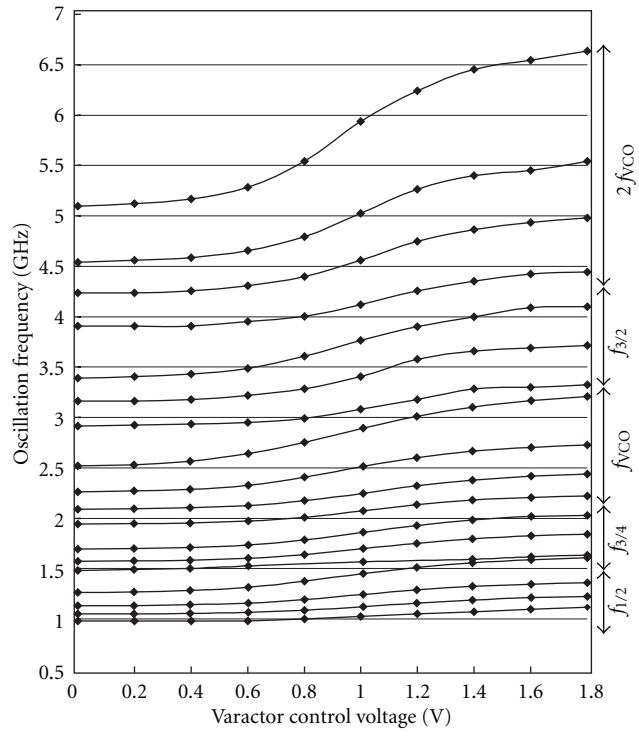


Figure 8: Measured tuning characteristics of the proposed VCO, which exhibits from 0.98 GHz to 6.6 GHz oscillation (149%).

the same frequency as $1/2\,f_0$ of spur, and it can be used for the spurious rejection instead of the SSBM technique. Therefore, the proposed architecture does not need QVCO and SSBM, and small-size synthesizer can be realized. First, the spurious frequency is rejected by the high-pass filter shown in Figure 2. Second, the remaining spur in the output of filter is rejected by a variable gain combiner shown in Figure 5(b). The gains of combiner are adjusted by bias voltages V_{gain1} and V_{gain2}. The high-pass filter is also used for phase adjustment, and the filter should be carefully designed to reduce phase mismatch in wide frequency range.

In case (B), $2\,f_0$ is the desired frequency and DC signal is spurious. The DC signal can be suppressed by the high-pass filter. In the proposed architecture, distance to spur is large, which is a desirable feature for spurious rejection in both cases (A) and (B). The proposed architecture is also expected to be robust for LO leak.

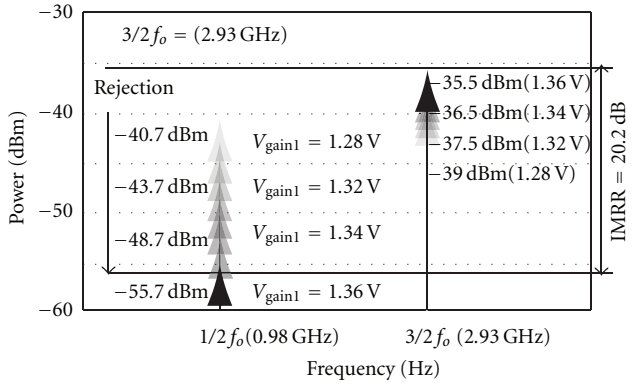


Figure 9: Spectrum of combiner output including $3/2\,f_0$ and $1/2\,f_0$ frequencies. The spurious rejection is performed by the high-pass filter and the variable gain combiner.

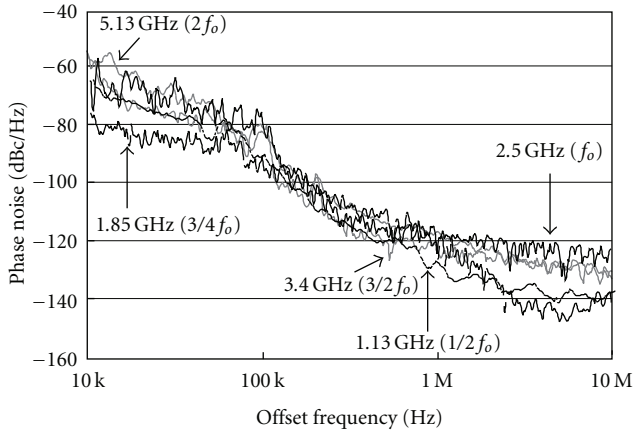


Figure 10: Phase noise at f_0 (2.50 GHz) and $3/4\,f_0$ (1.85 GHz).

Table 1: VCO performance summary.

Technology	TSMC 0.18 μm CMOS process with RF option
Supply voltage V_{DD}	1.8 V
VCO core current	2.45~14.9 mA
Power consumption	4.41~26.9 mW
Center frequency	3.81 GHz
Tuning range	0.98 GHz~6.64 GHz 149%
Chip area	800 μm × 540 μm

Figure 6 shows the detailed block diagram of the proposed wideband VCO. $2\,f_0$, $3/2\,f_0$, $3/4\,f_0$, and $1/2\,f_0$ are output from each node as shown in Figure 6. In the measurement, *I*/*O* buffers are utilized for each output. For an actual use, a selector is required, and some switching time is required for the frequency selection.

4. Measurement Result

Figure 7 shows a chip micrograph of the proposed wideband VCO, which is fabricated by using a 0.18 μm CMOS process.

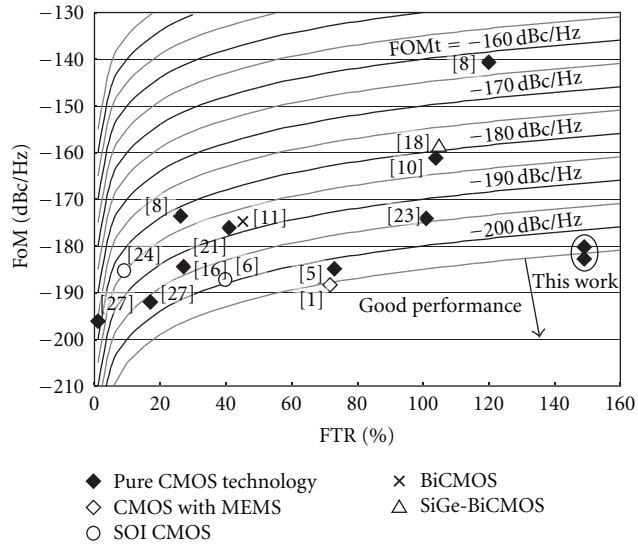


Figure 11: VCO performance comparison using FoM and frequency tuning range (FTR) [5–8, 10, 11, 16, 18, 21, 24–26].

Table 2: Phase noise performances.

Oscillation frequency	Phase noise @1 MHz offset	FoM	FoM_T
5.12 GHz ($2\,f_0$)	−117 dBc/Hz	−179 dBc/Hz	−203 dBc/Hz
3.40 GHz ($3/2\,f_0$)	−122 dBc/Hz	−179 dBc/Hz	−203 dBc/Hz
2.50 GHz (f_0)	−125 dBc/Hz	−183 dBc/Hz	−206 dBc/Hz
1.85 GHz ($3/4\,f_0$)	−128 dBc/Hz	−180 dBc/Hz	−203 dBc/Hz
1.13 GHz ($1/2\,f_0$)	−130 dBc/Hz	−179 dBc/Hz	−202 dBc/Hz

Core size is 800 μm × 540 μm. Depicted in Figure 7, the core area is dominated by the spiral inductor for LC-VCO. Signal Source Analyzer (Agilent E5052A) and Spectrum Analyzer (Agilent 8563EC) were used for measurement. GSG probes were also used to obtain on-chip signals. Figure 8 shows the tuning characteristics of the VCO, which exhibits 0.98 GHz to 6.6 GHz oscillation. The right *y* axis shows the frequency coverage of each output path. The tuning range is found to be 149%. Table 1 summarizes the measured results.

Figure 9 shows spectrum of the combiner output, which contains $3/2\,f_0$ and $1/2\,f_0$ of frequency. In this case, $3/2\,f_0$ is 2.93 GHz. The spurious frequency of $1/2\,f_0$ is rejected by both the high-pass filter and the variable gain combiner. The total image rejection ratio (IMRR) is 20.2 dB. In this measurement, the bias voltages in the variable gain combiner were manually adjusted.

Figure 10 shows measured phase noise characteristics for f_0 as the fundamental frequency and $3/4\,f_0$ as the final output. The $3/4\,f_0$ signal is generated through all the circuit blocks shown in Figure 2. This result shows that the proposed wideband VCO operates with the wideband and the low phase noise. Table 2 summarizes the measured phase noise and FoM. The proposed VCO achieves −183 dBc/Hz of FoM for 2.50 GHz oscillation. In this paper, FoM_T is also

employed to evaluate tuning range in addition to the phase noise. $\mathrm{FoM_T}$ is defined as the following equation [22]:

$$\begin{aligned}\mathrm{FoM_T} &= \mathrm{FoM} - 20\log\left(\frac{\mathrm{FTR}}{10}\right),\\ &= \mathcal{L}\{f_{\mathrm{offset}}\} - 20\log\left\{\frac{f_0}{f_{\mathrm{offset}}}\cdot\frac{\mathrm{FTR}}{10}\right\}\\ &\quad + 10\log\left(\frac{P_{\mathrm{DC}}}{1\mathrm{m}W}\right),\end{aligned} \tag{1}$$

where $\mathcal{L}\{f_{\mathrm{offset}}\}$ is phase noise, f_{offset} is certain frequency offset, f_o is center frequency, and P_{DC} is power consumption. FTR is frequency tuning range, which is defined as $(f_{\max} - f_{\min})/f_0$. Table 2 also shows $\mathrm{FoM_T}$, and the proposed VCO achieves −206 dBc/Hz of $\mathrm{FoM_T}$ for 2.50 GHz oscillation.

Figure 11 plots performances of wideband LC-VCO reported in the literature [5–8, 10, 11, 16, 18, 21, 24–26], which includes low phase noise VCOs using SOI [24, 25] and BiCMOS processes [6] and CMOS VCOs using phase noise improvement techniques [5, 11]. The proposed VCO achieves the widest tuning range and the best $\mathrm{FoM_T}$ simultaneously.

5. Conclusion

This paper has proposed a novel wideband LC-VCO for multiband applications. The VCO has the core VCO and the tuning range extension circuit. A differential LC-VCO and a double-balanced mixer are utilized instead of a quadrature VCO and a single-sideband mixer for the spurious rejection. In measurement results, the proposed VCO performs 0.98 to 6.6 GHz continuous frequency tuning with −206 dBc/Hz of $\mathrm{FoM_T}$, which is fabricated by using a 0.18 μm CMOS process. The frequency tuning range (FTR) is 149%, and the chip area is 800 μm × 540 μm. The proposed VCO achieves the widest tuning range and the best $\mathrm{FoM_T}$.

Acknowledgments

This work was partially supported by JSPS.KAKENHI, STARC, MIC.SCOPE, and VDEC in collaboration with Cadence Design Systems, Inc.

References

[1] D. Leenaerts, R. van de Beek, G. van der Weide et al., "A SiGe BiCMOS 1ns fast hopping frequency synthesizer for UWB radio," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '05)*, pp. 202–593, February 2005.

[2] J. Lee, "A 3-to-8 GHz fast-hopping frequency synthesizer in 0.18 xμm CMOS technology," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 566–573, 2006.

[3] Y. Kobayashi, K. Ohashi, Y. Ito, H. Ito, K. Okada, and K. Masu, "A 0.49–6.50 GHz wideband LC-VCO with high-IRR in a 180 nm CMOS technology," in *Proceedings of the International Conference on Solid State Devices and Materials*, pp. 268–269, September 2007.

[4] B. Razavi, T. Aytur, C. Lam et al., "Multiband UWB transceivers," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 140–147, September 2005.

[5] C. Yao and A. Willson, "A phase-noise reduction technique for quadrature LC-VCO with phase-to-amplitude noise conversion," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '06)*, pp. 196–191, February 2006.

[6] P. Vaananen, M. Metsanvirta, and N. T. Tchamov, "4.3 GHz VCO with 2 GHz tuning range and low phase noise," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 1, pp. 142–146, 2001.

[7] A. Fard, T. Johnson, and D. Aberg, "A low power wide band CMOS VCO for multi-standard radios," in *Proceedings of IEEE Radio and Wireless Conference*, pp. 79–82, September 2004.

[8] R. Mukhopadhyay, Y. Park, P. Sen et al., "Reconfigurable RFICs in Si-based technologies for a compact intelligent RF front-end," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 1, pp. 81–91, 2005.

[9] H. Tsurumi and Y. Suzuki, "Broadband RF stage architecture for software-defined radio in handheld terminal applications," *IEEE Communications Magazine*, vol. 37, no. 2, pp. 90–95, 1999.

[10] D. Guermandi, P. Tortori, E. Franchi, and A. Gnudi, "A 0.83-2.5 GHz continuously tunable quadrature VCO," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2620–2627, 2005.

[11] A. D. Berny, A. M. Niknejad, and R. G. Meyer, "A 1.8 GHz LC VCO with 1.3 GHz tuning range and digital amplitude calibration," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 4, pp. 909–916, 2005.

[12] H. Eul, "ICs for mobile multimedia communications," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '06)*, pp. 31–20, February 2006.

[13] R. Bagheri, A. Mirzaei, S. Chehrazi et al., "An 800MHz to 5GHz software-defined radio receiver in 90nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '06)*, pp. 469–480, February 2006.

[14] Y. Yoshihara, H. Sugawara, H. Ito, K. Okada, and K. Masu, "Reconfigurable RF circuit design for multi-band wireless chip," in *Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, pp. 418–419, jpn, August 2004.

[15] K. Okada, Y. Yoshihara, H. Sugawara, and K. Masu, "A dynamic reconfigurable RF circuit architecture," in *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference*, pp. 683–686, 2005.

[16] A. Kral, F. Behbahani, and A. A. Abidi, "RF-CMOS oscillators with switched tuning," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 555–557, May 1998.

[17] C. Liang, S. Liu, Y. Chen, T. Yang, and G. Ma, "A 14-band frequency synthesizer for MB-OFDM UWB application," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '06)*, pp. 113–126, February 2006.

[18] Y. Ito, Y. Yoshihara, H. Sugawara, K. Okada, and K. Masu, "A 1.3–2.8 GHz wide range CMOS LC-VCO using variable inductor," in *Proceedings of the IEEE Asian Solid-State Circuits Conference Digest of Technical Papers*, pp. 265–268, November 2005.

[19] B. Razavi, "Multi-decade carrier generation for cognitive radios," in *Proceedings of the Symposium on VLSI Circuits*, pp. 120–121, June 2009.

[20] Y. Ito, H. Sugawara, K. Okada, and K. Masu, "A 0.98 to 6.6 GHz tunable wideband VCO in a 180 nm CMOS technology for reconfigurable radio transceiver," in *Proceedings of the IEEE Asian Solid-State Circuits Conference Digest of Technical Papers*, pp. 359–362, November 2006.

[21] A. D. Berny, A. M. Niknejad, and R. G. Meyer, "A wideband low-phase-noise CMOS VCO," in *Proceedings of the IEEE*

Custom Integrated Circuits Conference, pp. 555–558, September 2003.

[22] A. Ismail and A. Abidi, "A 3.1- To 8.2 GHz zero-IF receiver and direct frequency synthesizer in 0.18 μm SiGe BiCMOS for mode-2 MB-OFDM UWB communication," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2573–2582, 2005.

[23] P. Nuzzo, K. Vengattaramane, M. Ingels, V. Giannini, M. Steyaert, and J. Craninckx, "A 0.1-5 GHz dual-VCO software-defined $\Sigma\Delta$ frequency synthesizer in 45 nm digital CMOS," in *Proceedings of the IEEE Radio Frequency Integrated Circuits Symposium, (RFIC '09)*, pp. 321–324, June 2009.

[24] N. Fong, J. Plouchart, N. Zamdmer et al., "A 1V 3.8-5.7 GHz differentially-tuned VCO in SOI CMOS," in *Procedings of the IEEE Radio Frequency Integrated Circuits Symposium, Digest of Papers*, pp. 75–78, June 2002.

[25] J. Kim, J. O. Plouchart, N. Zamdmer et al., "A 44 GHz differentially tuned VCO with 4 GHz tuning range in 0.12 μm SOI CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '05)*, pp. 416–607, February 2005.

[26] A. Tanaka, H. Okada, H. Kodama, and H. Ishikawa, "A 1.1V 3.1-to-9.5 GHz MB-OFDM UWB transceiver in 90 nm CMOS," in *Proceedings of the IEEE International Solid-State Circuits Conference, (ISSCC '06)*, pp. 120–113, February 2006.

Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2011, Article ID 701730, 7 pages
doi:10.1155/2011/701730

Research Article

A Wide Lock-Range Referenceless CDR with Automatic Frequency Acquisition

Seon-Kyoo Lee, Young-Sang Kim, Hong-June Park, and Jae-Yoon Sim

Department of Electronic and Electrical Engineering, Pohang University of Science and Technology (POSTECH), San 31, Hyojadong, Pohang 790-784, Republic of Korea

Correspondence should be addressed to Seon-Kyoo Lee, leesk@postech.ac.kr

Received 26 April 2011; Accepted 16 June 2011

Academic Editor: Woogeun Rhee

A wide lock-range referenceless CDR circuit is proposed with an automatic tracking of data rate. For efficient frequency acquisition, a DLL-based loop is used with a simple phase/frequency detector to extract 1-bit period of input data stream. The CDR, implemented in a 65 nm CMOS, shows a lock range of 650 Mb/s-to-8 Gb/s and BER of less than 10^{-12} at 8 Gb/s with low power consumption.

1. Introduction

Performance of a digital system is determined by the data rate of interchip communication as well as on-chip operating speed. As the development in process technology has successfully driven ever-increasing on-chip operating frequency, the off-chip interface is becoming the bottleneck in further improvement of system performance. For high-speed chip-to-chip communication, serial link protocol has been widely adopted in various computer-to-peripheral interfaces and has achieved data rates of over 10 Gb/s using differential signaling through a well-defined optical channel [1]. The widespread use of serial links for multipurpose, however, still presents some challenges which must be overcome by circuit design.

For wide-range CDR, two kinds of circuit schemes have been researched. One is the multirate CDR circuit with multiple reference clocks [2] or single reference clock with a programmable divider [3] or without reference clock [4, 5]. The other is the continuous-rate CDR circuit with fractional-N divider [6] or without an external reference clock [7–10]. The latter CDR scheme detects a change in the bit rate of the incoming data and adaptively controls the internal wide-range VCO to track the bit rate without harmonic-lock issue. To extract the data frequency directly from an input data stream, several techniques have been presented with complicated state-machine-based frequency detectors [7–9] or using limited run length of 8B10B coding [10, 11]. The previous frequency acquisition circuits, however, cause large power consumption and area overhead. Therefore an efficient frequency acquisition algorithm is required to reduce complexity and power consumption.

This paper presents a 650 Mb/s to 8 Gb/s referenceless CDR with an automatic tracking of data rate [12]. With a novel DLL-based frequency acquisition, the proposed dual-loop CDR shows the highest performance in lock-range, power consumption, and size compared with previously reported continuous-rate CDRs.

2. Circuit Description

Figure 1 shows the proposed CDR which consists of a DLL-based frequency acquisition loop and a PLL-based loop for the clock and data recovery. In the frequency loop, the voltage-controlled delay line (VCDL) is automatically biased so that the delay of VCDL, T, would be equal to one bit duration, Tb. This frequency loop performs a two-step acquisition procedure which is a coarse lock with the coarse delay tracking (CDT) followed by a fine lock with the fine delay tracking (FDT). When CDT ends, FDT loop is enabled with the phase loop. A loss of lock detector (LLD) is included in the frequency loop to monitor a change in the data rate

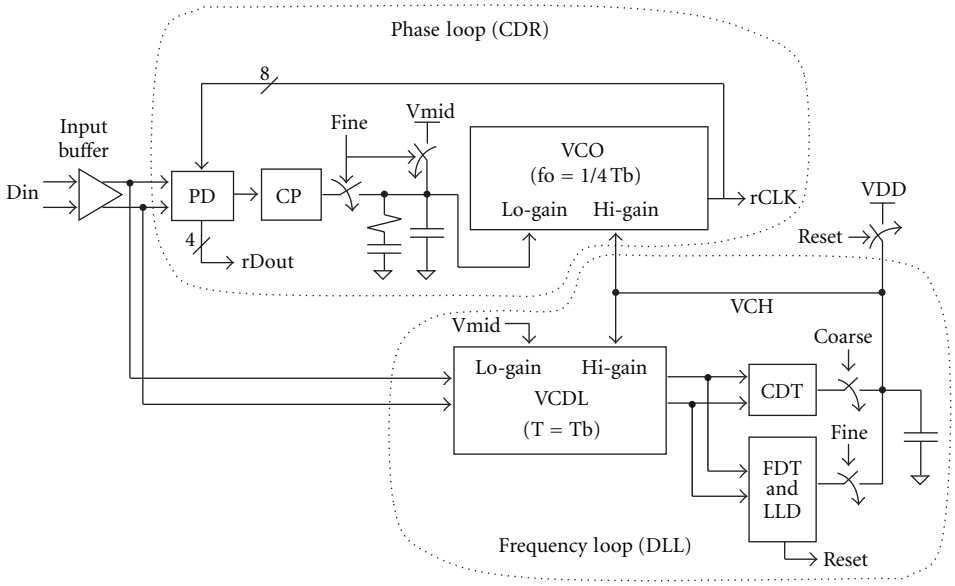


Figure 1: Architecture of proposed CDR.

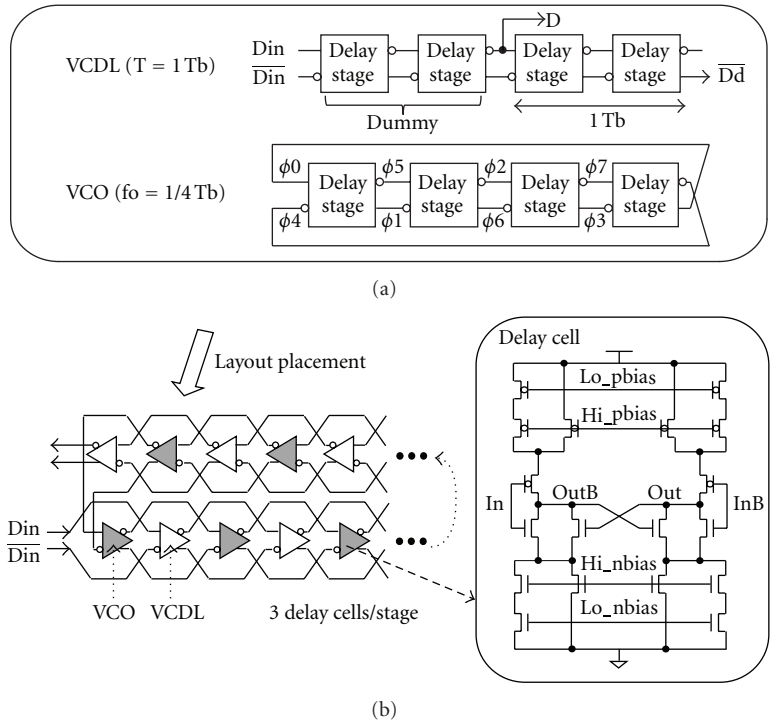


Figure 2: VCDL and VCO.

during the fine lock state. If LLD detects a change in the data rate, *Reset* signal is generated and it forces to restart from the coarse frequency lock again for automatic frequency acquisition. In the phase loop, a quarter-rate binary phase detector (PD) [13] is used with an 8-phase VCO.

Since matching between VCDL and VCO is an important assumption in the proposed frequency acquisition, identical delay circuit is used for both VCDL and VCO. The delay cell has hi-gain and lo-gain paths for the frequency lock and the phase lock, respectively.

2.1. VCO and VCDL. Figure 2 shows the circuit diagram of the delay cell. Hi_pbias and Lo_pbias are PMOS gate bias voltages generated by current-mirrored transformations from Hi_nbias and Lo_nbias, respectively. With the control range of from 0.5 V to 1.1 V, the gain of the hi-gain path is

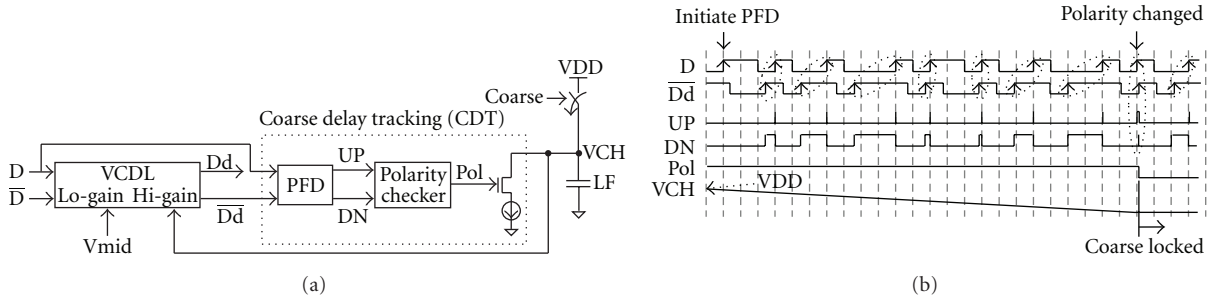


Figure 3: Coarse delay tracking.

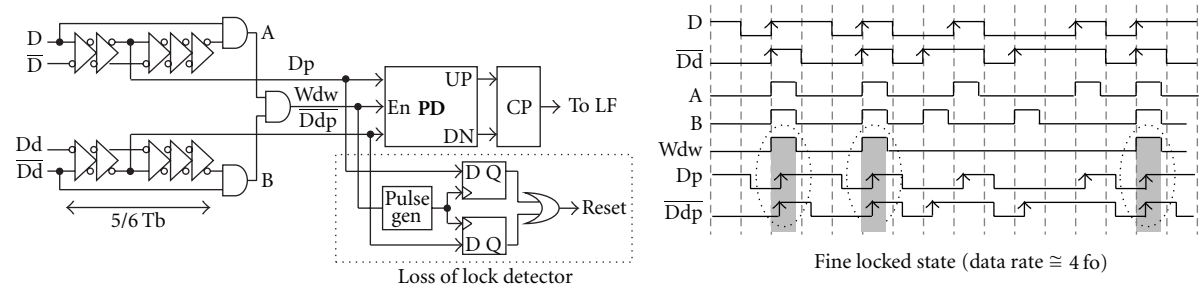


Figure 4: Fine delay tracking.

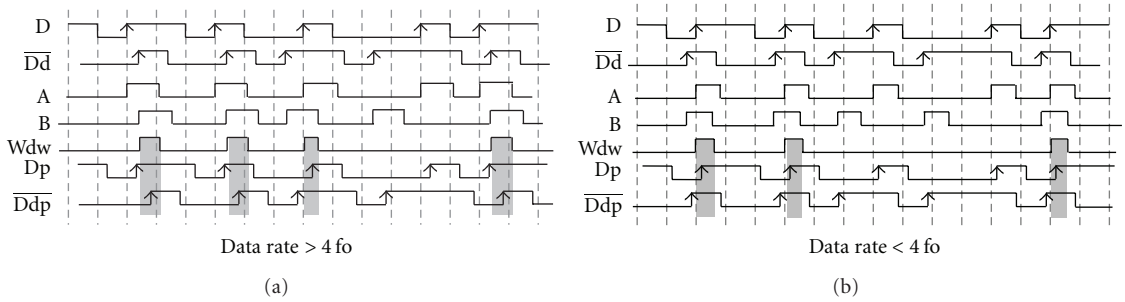


Figure 5: Cases of loss of lock.

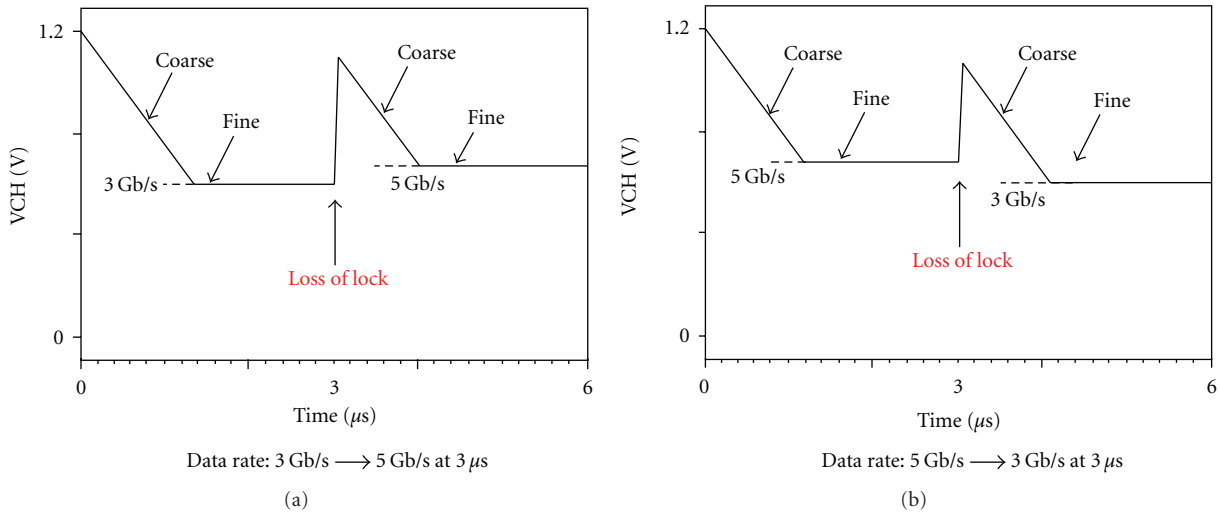
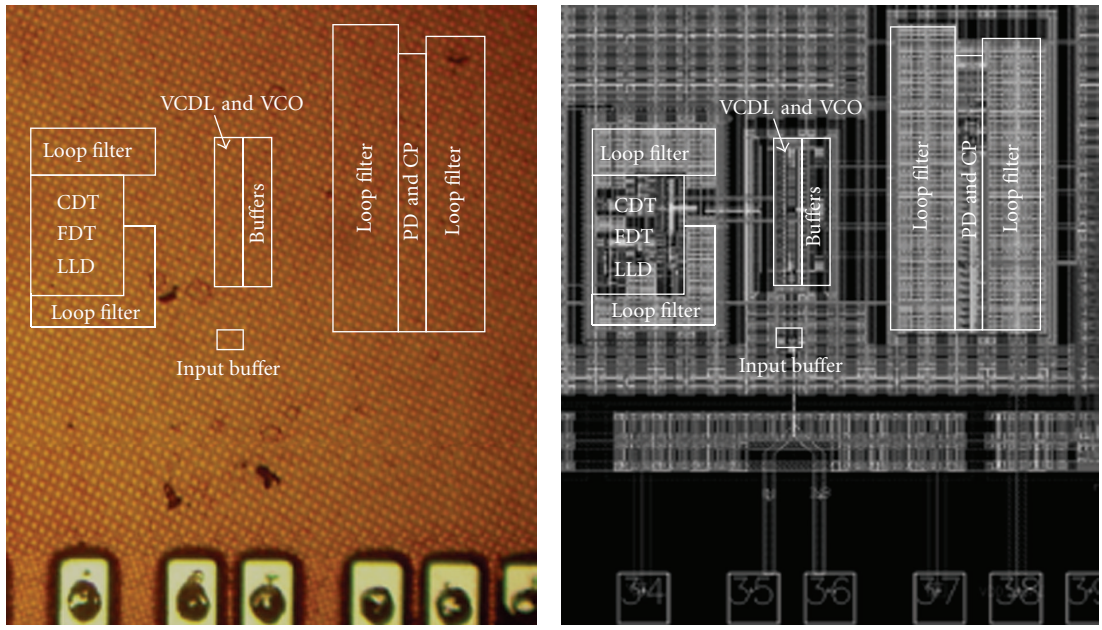
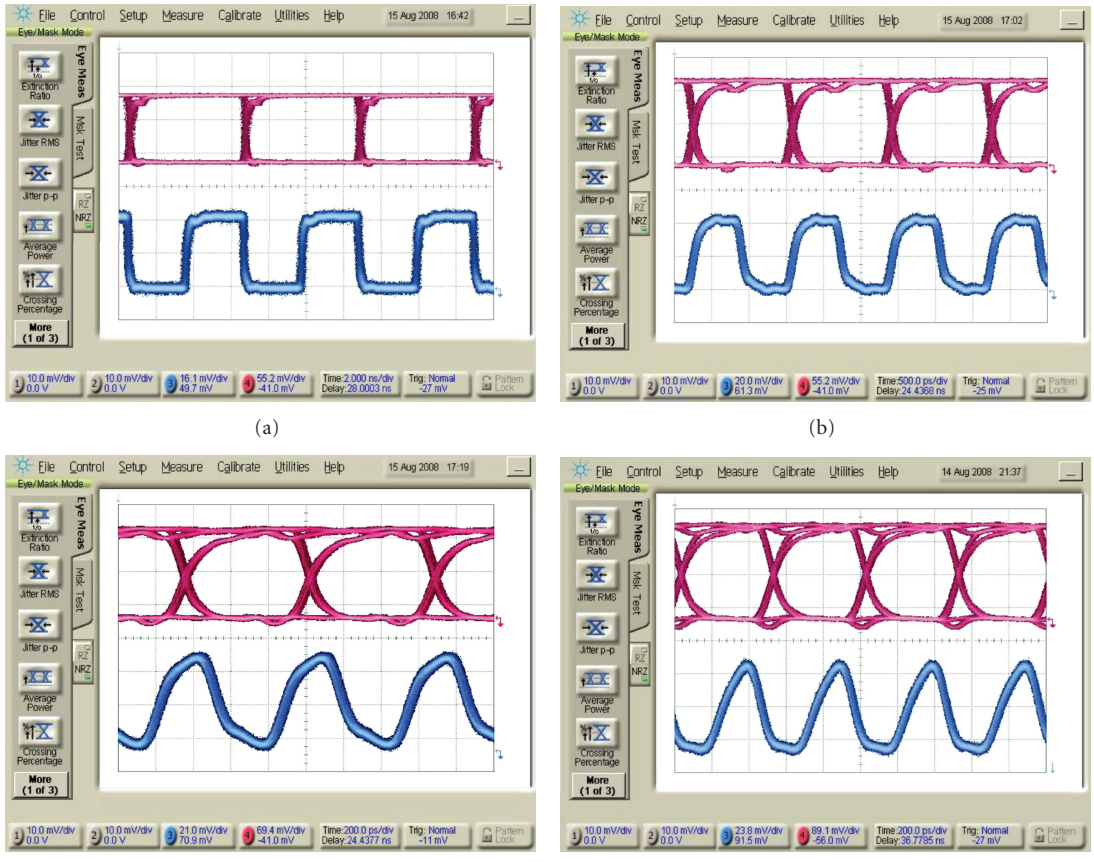


Figure 6: Simulated automatic frequency acquisition in case of data rate changes.



(a) (b)

Figure 7: Photomicrograph and layout of the test chip.



(a) (b)

(c) (d)

Figure 8: Measured eye diagrams and clock @650 Mb/s (a), 3 Gb/s (b), 6 Gb/s (c) and 8 Gb/s (d).

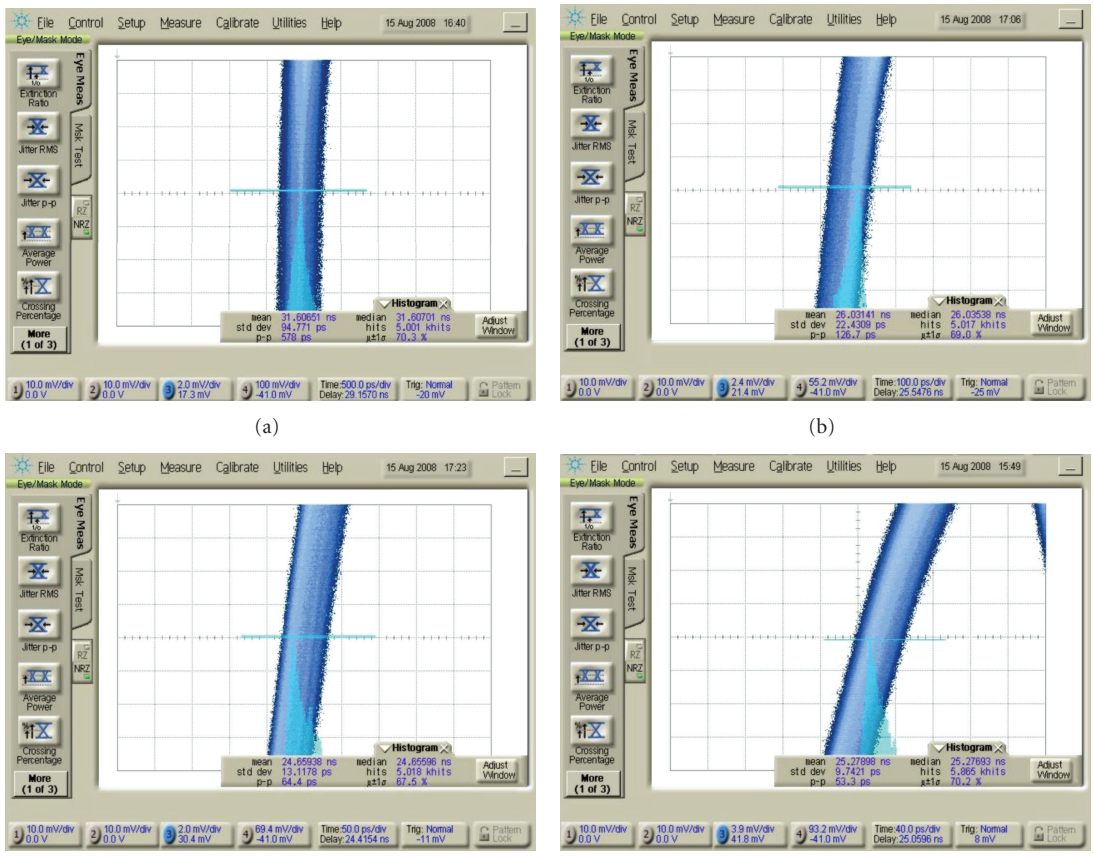


Figure 9: Measured clock jitter @650 Mb/s (a), 3 Gb/s (b), 6 Gb/s(c) and 8 Gb/s (d).

designed to be 3.5 GHz/V for wide lock-range while the lo-gain path is 150 MHz/V for better jitter performance. One delay stage is implemented by the cascade of three delay cells. But in actual layout placement, the total 24 delay cells for VCO and VCDL are alternated for improved matching.

2.2. Coarse Frequency Tracking Loop. Figure 3 illustrates the operation of CDT and how the delay of VCDL can be set to Tb, which is performed by successful phase detection from a random NRZ bit stream. Before the coarse lock operation, the loop filter (LF) is initially charged to VDD so that VCDL would experience the minimum delay. When the coarse lock is started, the first coming rising edge of the input data D initiates the phase detection between the rising edges of D and the inverse of the delayed input, $\overline{\text{Dd}}$. The phase detection can be performed by a typical phase/frequency detector (PFD). Since the PFD is a sequential logic based on flip-flops, the initial value of PFD determines the pulse width of up/dn signal. So, the initialization by the first coming rising edge of D makes the desired initial value for the coarse operation. Since the initial delay of VCDL is set to be the minimum, PFD generates more DN pulse in the beginning. Then a pull-down current source decreases *VCH* until the polarity of PFD output changes to UP. Once the UP pulse becomes greater than DN pulse, the output of the polarity checker, Pol, is latched to low. It stops discharging *VCH*, which is the end of the coarse lock. This coarse delay tracking is performed through a hi-gain path, while lo-gain bias is fixed to the center of the control voltage range.

2.3. Fine Frequency and Phase Tracking Loop. After the coarse lock, FDT takes over the frequency loop. As shown in Figure 4, the rising edges of D and $\overline{\text{Dd}}$ generate autopulses on A and B with a pulse width of 5/6Tb which is implemented by five-stage replica delay cells. An AND gate is used to generate a window signal, Wdw, to select appropriate rising edges for the phase detection. Dp and $\overline{\text{Dpd}}$ are delayed signals of D and $\overline{\text{Dd}}$, respectively, to make sure the rising edges be in the middle of Wdw when locked. With the replica delay cells, it is guaranteed that the rising edges of Dp and $\overline{\text{Dpd}}$ are placed in the middle of Wdw pulse regardless of the change in bit rate. By accepting the rising edges of Dp and $\overline{\text{Dpd}}$ only when Wdw is high, valid phase detection is achieved with a simple binary phase detector, and the output of PD drives a charge pump to perform the fine lock. Wdw, Dp, and $\overline{\text{Dpd}}$ are also applied to LLD to detect a change of bit rate during the fine lock state. If LLD detects a change, the coarse lock procedure starts again for automatic tracking. If both rising edges of Dp and $\overline{\text{Dpd}}$ are not in the Wdw pulse, it represents the loss of lock and reset signal is generated. Figure 5 shows the two cases of the loss of lock condition.

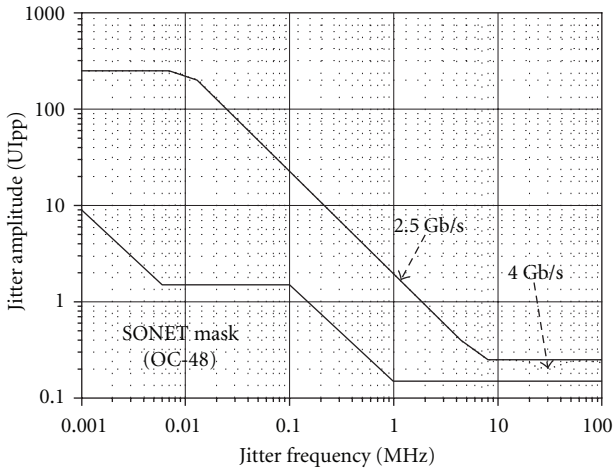


Figure 10: Measured jitter tolerance.

Figure 6 shows the simulated VCH when loss of lock occurs. When the input data rate is changed, the frequency loop detects it and automatically tracks the new data rate by starting the frequency acquisition again.

3. Mesurements

For verification, the proposed CDR circuit was implemented with a 65 nm CMOS technology as shown in the Figure 7. Active area was 0.108 mm^2 including LF capacitors of 80 pF in frequency tracking loop and 200 pF in phase tracking loop. With a BER of less than 10^{-12}, CDR operates at a lock range from 650 Mb/s to 8 Gb/s. Figure 8 shows measured eye diagrams of the quarter-rate recovered data and clock at different data rate.

Figure 9 shows measured quarter-rate recovered clock jitter. It was measured to 9.7 ps_{rms} and 53.3 $ps_{p\text{-}p}$ at the data rate of 8 Gb/s. The measured jitter can be decomposed into a pattern-dependent deterministic jitter of 20 $ps_{p\text{-}p}$ and a random jitter of 2.8 ps_{rms}, respectively. As shown in Figure 10, the CDR also passed the OC-48 jitter tolerance specification at 2.5 Gb/s.

The CDR consumes power of 20.6 mW and 88.6 mW at 650 Mb/s and 8 Gb/s, respectively. Table 1 summarizes the performance of designed CDR. The proposed DLL-based frequency acquisition scheme achieved an efficient circuit implementation and shows suitability for the low-power and wide lock-range referenceless CDR.

4. Conclusion

A wide lock-range of 650 Mb/s-to-8 Gb/s referenceless CDR circuit is proposed with an automatic tracking of data rate. For an efficient frequency acquisition in case of continuous data rate changes, a DLL-based loop is used with a simple phase/frequency detector. The CDR, implemented in a 65 nm CMOS, shows a BER of less than 10^{-12} with the best performance in lock-range, power consumption. The proposed DLL-based frequency acquisition scheme achieved a simplified circuit realization and shows suitability for the low-power and wide lock-range referenceless CDR.

Table 1: Performance summary.

Performance metric	Value
Technology	65 nm 1P6M CMOS
Supply voltage	1.2 V
Active area	0.108 mm^2 (including LF)
Data rate	650 Mb/s–8 Gb/s
Power consumption	88.6 mW @ 8 Gb/s 20.6 mW @ 650 Mb/s
Jitter (rms)	9.7 ps @ 8 Gb/s 94.7 ps @ 650 Mb/s
Jitter (p-p)	53.3 ps @ 8 Gb/s 578 ps @ 650 Mb/s

Acknowledgment

This paper was supported by Mid-Career Researcher Program through NRF Grant funded by the MEST (2011-0010685) and BK21 program.

References

[1] B. Razavi, "Design of high-speed circuits for optical communication systems," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 315–322, San Diego, Calif, USA, May 2001.

[2] J. C. Scheytt, G. Hanke, and U. Langmann, "A 0.155, 0.622 and 2.488-Gb/s automatic bit-rate selecting clock and data recovery IC for bit-rate transparent SDH systems," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 12, pp. 1935–1943, 1999.

[3] D. Belot, L. Dugoujon, and S. Dedieu, "A 3.3-V power adaptive 1244/622/155 Mbit/s transceiver for ATM, SONET/SDH," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 7, pp. 1047–1058, 1998.

[4] M. H. Perrott, Y. Huang, R. T. Baird et al., "A 2.5-Gb/s multi-rate 0.25-μm CMOS clock and data recovery circuit utilizing a hybrid analog/digital loop filter and all-digital referenceless frequency acquisition," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 12, pp. 2930–2942, 2006.

[5] H. Nosaka, E. Sano, K. Ishii et al., "A 39-to-45-Gbit/s multi-data-rate clock and data recovery circuit with a robust lock detector," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1361–1365, 2004.

[6] J. P. Frambach, R. Heijna, and R. Krösschell, "Single reference continuous rate clock and data recovery from 30Mbit/s to 3.2Gbit/s," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 375–378, Scottsdale, Ariz, USA, May 2002.

[7] R.-J. Yang et al., "A 200-Mbps~2-Gbps continuous-rate clock-and-data-recovery circuit," *IEEE Transactions on Circuits and Systems I*, vol. 53, no. 4, pp. 842–847, 2006.

[8] R.-J. Yang, K. H. Chao, S. C. Hwu, C. K. Liang, and S. I. Liu, "A 155.52 Mbps-3.125 gbps continuous-rate clock and data recovery circuit," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 6, Article ID 1637602, pp. 1380–1390, 2006.

[9] D. Dalton, K. Chai, E. Evans et al., "A 12.5-Mb/s to 2.7-Gb/s continuous-rate CDR with automatic frequency acquisition and data-rate readback," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2713–2724, 2005.

[10] M. S. Hwang, S. Y. Lee, J. K. Kim, S. Kim, and D. K. Jeong, "A 180-Mb/s to 3.2-Gb/s, continuous-rate, fast-locking CDR without using external reference clock," in *Proceedings of the IEEE Asian Solid-State Circuits Conference*, pp. 144–147, Jeju, korea, November 2007.

[11] A. X. Widmer and P. A. Franaszek, "A DC-balanced, partitioned-block, 8B/10B transmission code," *IBM Journal of Research and Development*, vol. 27, no. 5, pp. 440–451, 1983.

[12] S.-K. Lee, Y. S. Kim, H. Ha, Y. Seo, H. J. Park, and J. Y. Sim, "A 650Mb/s-to-8Gb/s referenceless CDR circuit with automatic acquisition of data rate," in *Proceedings of the IEEE International Solid-State Circuits Conference ISSCC 2009*, vol. 52, pp. 184–185, San Francisco, Calif, USA, February 2009.

[13] J. D. H. Alexander, "Clock recovery from random binary signals," *Electronics Letters*, vol. 11, no. 22, pp. 541–542, 1975.