IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 1, 2019, accepted July 19, 2019, date of publication July 24, 2019, date of current version August 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930775

A Ping-Pong Methodology for Boosting the Resilience of Cell-Based Delay-Locked Loop

ZHENG-HONG ZHANG, WEI CHU, AND SHI-YU HUANG, (Senior Member, IEEE)
Electrical Engineering Department, National Tsing Hua University, Hsinchu 30013, Taiwan

Corresponding author: Shi-Yu Huang (syhuang@ee.nthu.edu.tw)

This work was supported in part by the Ministry of Science and Technology (MOST) of Taiwan under Grant MOST-105-2221-E-007-120-MY3.

ABSTRACT In this paper, we present a cell-based delay-locked loop (DLL) with an enhanced continuous tracking range. The main contribution is a novel delay line architecture called ping-pong delay line, making it highly resilient to process and temperature variation. In such a DLL design, two cell-based delay lines are incorporated in a way that they exchange their role of command dynamically like in a ping-pong game, and therefore the joint ping-pong delay line can react to severe environmental changes over a very wide range without disruption to the system's operation. The post-layout simulation using a 90-nm complimentary metal-oxide silicon (CMOS) process technology has demonstrated its advantages. A DLL using such a feature can operate reliably even under an extremely hostile environment when the supply voltage drops from 1 to 0.9 V within a timeframe of 4us.

INDEX TERMS Cell-based DLL, delay line, ping-pong, segment jumping.

I. INTRODUCTION

In today's IC, the delay-locked loop (DLL) circuit is a very common building block for numerous applications, ranging from high-speed multi-phase clock generation, clock synchronization, clock de-skew, timing control for the logic-and-DRAM interface, etc [1]–[3]. Traditionally, analog circuits are used in building a DLL. However, more and more all-digital solutions have emerged as alternatives [4]–[9]. In a number of prior works, DLLs constructed by standard cells only have also been proposed [10]–[13]. Using all-digital DLLs have several benefits. For example, in a system operated with adaptive supply voltages, an all-digital DLL could be more robust than its analog counterpart due to its relatively wider operating range and better resistance to the environmental noise. Also, an all-digital DLL is not only easier to design and verified, but also more easily portable from one process technology to another. Due to their digital nature, compilers for cell-based DLLs or PLLs have been developed [14], [15]. These compilers can generate a cell-based DLL or PLL macro according to the users' requirements within a few minutes.

However, existing DLL designs still face a dilemma of either being too power consuming or having limited continuous tracking range (CTR), as to be detailed later. The most important component in a DLL is the tunable delay line (TDL). *A TDL is a component with the end-to-end delay from its input to its output controllable by control code(s)*. There are two major types of cell-based TDLs, namely the *continuous* TDL, and *segmented* TDL. As illustrated in Fig. 1(a), the delay profile of a continuous TDL is monotonic (i.e., the delay changes monotonically with the increase of the value of a control code). On the other hand, the delay profile of a segmented TDL is divided into several segments, as shown in Fig. 1(b). Note that such a segmented delay profile is more and more popular because a TDL often employs a two-level control code – coarse-tuning code and fine-tuning code – to achieve both wide delay range and fine

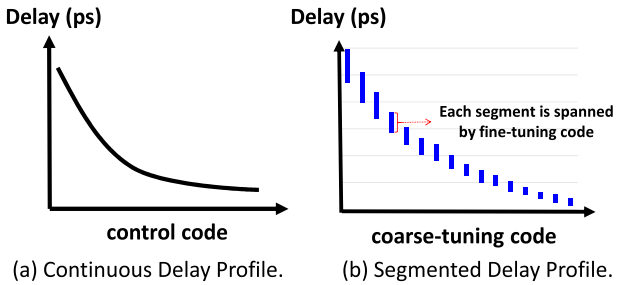


FIGURE 1. The delay profiles of two different types of TDL.

The associate editor coordinating the review of this manuscript and approving it for publication was Gian Domenico Licciardo.

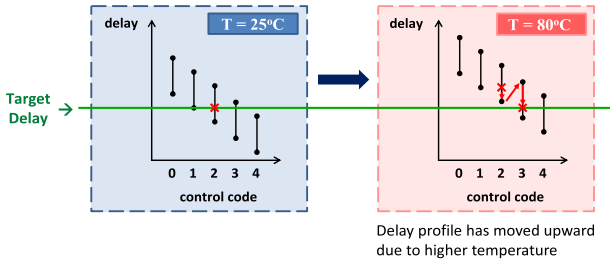


FIGURE 2. Segment-jumping problem–the operating point in the delay profile of a segmented TDL jumps from one segment to its right neighbor during the phase-tracking process of a DLL may introduce significant jitter.

timing resolution. Within each ''segment'', the delay profile is monotonic with the fine-tuning code. However, from one coarse-tuning code to another, the delay profile exhibits ''sudden jumps''.

A segmented TDL is generally superior to continuous TDL in both the area overhead and the power consumption. However, its segmented delay profile is a major weakness, making it unable to adapt to environmental changes robustly. In the following, we elaborate on this issue by considering the operation of a DLL incorporating a segmented TDL.

When a DLL starts to operate, it incorporates a so-called ''phase-locking process'', which changes the delay of its TDL systematically until the delay across the DLL satisfies certain phase-locked condition (e.g., when the phase difference between some clock signal, e.g., the output clock signal, and the input reference clock signal has been reduced to a very small value). Then, DLL enters a ''locked state'' and initiates the ''phase-tracking process''. During the phase-tracking process, the DLL's controller will try to maintain the phase-locked condition by incrementing or decrementing the control code of the TDL. In a segmented TDL, if the control code reaches the end-point of a segment in the delay profile, then the DLL loses the ability to continue to adapt to the environmental changes. In response to this awkward situation, a DLL is forced to perform a ''segment jump'' as illustrated in Fig. 2, in order to continue to maintain the phase-locked condition.

In a segment jump, the operating point of a DLL in the delay profile jumps from one segment to its neighbor segment. Such a jump could lead to significant jitter, creating a segment-jumping problem. The work in [16] has tried to resolve this problem by adopting two mirror TDLs by a sophisticated calibration scheme. However, it cannot easily handle the jitters of segment jumps due to online environmental changes (e.g., VDD scaling and temperature variations), even though it can reduce the jitter of segment jumps due to process calibration.

In this paper, we aim to resolve the segment-jumping problem by a novel *ping-pong TDL*. We have implemented the proposed scheme as a cell-based DLL using a 90nm CMOS process technology, capable of operating from 400MHz to 1.25GHz under severe environmental conditions.

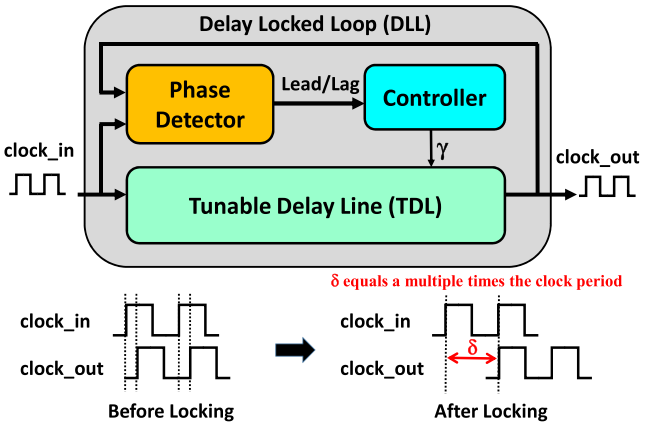


FIGURE 3. General architecture of a DLL and its function.

The contributions of this work include the following.

(1) Unlike [16], a fabricated ping-pong DLL macro in a chip can be put into use immediately without any process calibration, and is able to track significant online environmental changes without creating significant jitters.
(2) As compared to a DLL incorporating continuous TDL, the proposed ping-pong DLL has a much smaller area overhead and much lower power consumption.
(3) As compared to a DLL incorporating segmented TDL, the CTR can be enlarged tremendously, and thereby removing the significant jitters arising from potential segment jumps.

In some sense, the proposed ping-pong scheme has achieved one important property – it transforms a segmented TDL into a pseudo-continuous TDL and so the entire delay range of a TDL can become continuous to support full-range phase-tracking operation. We believe that this improvement is significant since a segmented TDL based DLL will become much more useful in the future after overcoming this major limitation with a reasonable area overhead.

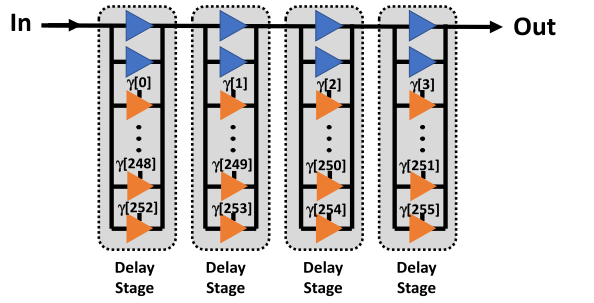
The rest of this paper is organized as follows. Section II reviews the general block diagram of an all-digital DLL and its operation. Section III presents the proposed ping-pong TDL, including its basic concept, architecture, circuitry, and operations. Section V presents the post-layout simulation results of a DLL, and Section VII concludes. It is notable that even though the rough idea has ever appeared in a brief Late-Breaking-Result conference paper [17], some detailed circuits and the most important technique that enables the ping-pong operation, referred to as *wake-up-and-get-ready* (WUGR) protocol, are only presented in this paper for the first time.

II. BACKGROUNDS

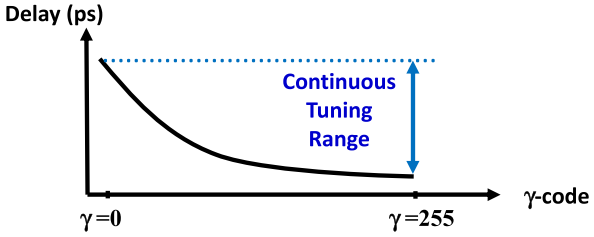
A. DELAY-LOCKED LOOP (DLL)

The general architecture of a DLL is shown in Fig. 3.

It consists of three major blocks: (1) a phase detector, (2) a TDL, and (3) an overall controller. For simplicity without

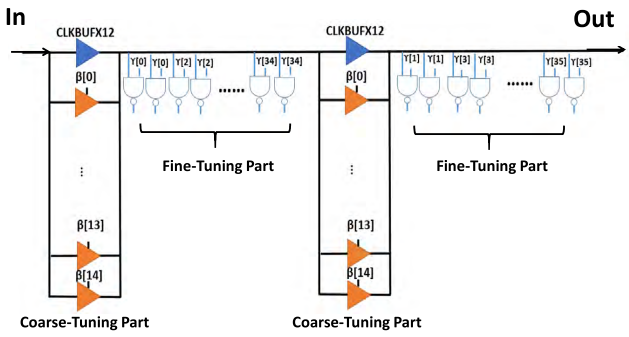


(a) Parallel tri-state buffers based TDL, using 4 delay stages.

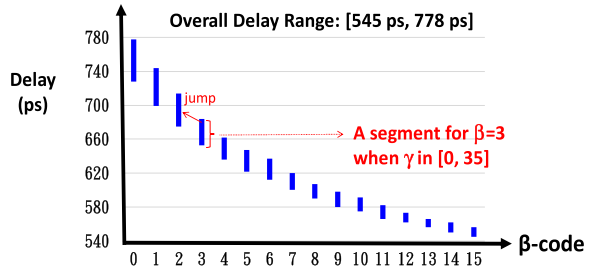


(b) Delay profile of a continuous TDL.

FIGURE 4. A cell-based continuous tunable delay line [10], [11].



(a) Circuit of Segmented Tunable Delay Line, using 98 standard cells.



(b) Delay Profile of a Segmented TDL.

FIGURE 5. A segmented tunable delay line [12], [14].

losing generality, we assume that the phase-locked condition in this case is to make the output clock signal in-phase with the input clock signal.

Initially, the output clock signal (namely *clock_out*) is not in-phase with the input clock signal (namely *clock_in*). But after the DLL is locked, their phase difference will almost disappear as shown in the figure. During the phase-locking process, the output clock is successively compared with the input clock to decide the result of a one-bit signal, called *lead/lag* via a *phase detector*. When *lead/lag* signal is '1', it means the output clock signal is ahead of the input clock signal in the timings of their rising edges. On the other hand, when *lead/lag* signal is '0', it means the output clock signal is behind of the input clock signal. Then this *lead/lag* signal will guide the controller to update the value of the control code of the TDL so that the phase difference between the output clock and the input clock will gradually diminish. After phase-locking, the delay across the TDL will equal the clock period of the input clock signal in the ideal case.

B. TUNABLE DELAY LINE (TDL)

In a DLL, the most important part is the TDL with its end-to-end delay controlled by a digital code or several levels of digital codes. Such a tunable delay is often characterized by two factors - time resolution and operating range. Here, the time resolution is referred to the *change of a TDL's end-to-end delay when the value of the digital control code is incremented or decremented by 1*, while the operating range is referred to the *range defined by the maximum delay and minimum delay of the TDL*. It is often desirable that a TDL has a very wide operating range and a very fine time resolution (e.g., 1ps).

There are two major types of TDLs – namely the continuous TDL and segmented TDL.

A widely adopted Continuous TDL is shown in Fig. 4, first proposed in [10], [11]. Its end-to-end delay represents the time needed for a signal propagating from input signal *IN* to output signal *OUT*. In this example, it contains four delay stages, each consisting of 64 parallel tri-state buffers. Overall, there are $64 \times 4 = 256$ tri-state buffers, each controlled by one bit of a 256-bit thermometer code $\gamma[0{:}255]$. Note that when the value of this thermometer γ-code is larger, more tri-state buffers will be turned on, leading to a shorter propagation delay from input signal *IN* to output signal *OUT*. Therefore, the delay of this TDL decreases monotonically with the increase of the value of the thermometer γ-code.

The delay profile of the above TDL is smooth and its entire operating range is also the CTR, during which the DLL can move around without creating significant jitter. However, it often requires a large number of tri-state buffers (e.g., 256 in the example) and thus consuming relatively larger power.

The segmented TDL is another type of TDLs. One such example is shown in Fig. 5. The delays from the input signal *IN* to the output signal *OUT*can be tuned by two different schemes, namely (1) *tunable driving strength* by parallel tri-state buffers, controlled by thermometer β-code with a value in [0, 14], and (2) *tunable output capacitances* at certain nodes, controlled by thermometer γ-code with a value in [0, 34]. The tuning of the driving strength is the same as that used in the continuous TDL just discussed, while the tunable output capacitances are realized by adding parallel NAND gates at the nodes where the capacitances need to

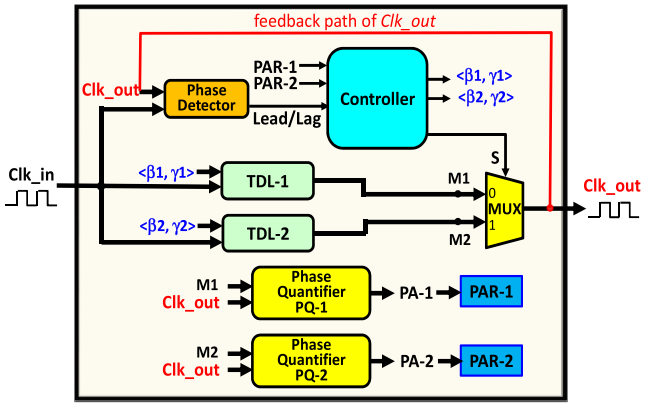


FIGURE 6. Architecture of the proposed Ping-Pong DLL. Two "phase quantifiers" are inserted to facilitate "instant handover" between the two primitive TDLs.

be tuned [12], [14]. The control code for this TDL is a 2-level code, characterized by $<$ β-code, γ-code$>$. Here, β-code is called the coarse-tuning code and γ-code is called the fine-tuning code. When the fine-tuning code,, i.e., γ-code, is incremented or decremented, there is only a small amount of change in the TDL delay. However, when the coarse-tuning code, i.e., β-code, is incremented or decrement, the TDL delay will change with a large stride. Clearly, this Segmented TDL has a much smaller area than the Continuous TDL aforementioned. However, a delay segment could have a relatively small range. As shown in the figure, the delay segment for $\beta = 3$ has a span of roughly only 30ps. During the phase-tracking process, it is very likely that the operating point in the delay profile may need to jump to its neighbor segment (as illustrated in the figure), and thereby causing a significant jitter.

III. PROPOSED PING-PONG DLL

A. BASIC ARCHITECTURE

Fig. 6 shows the overall architecture of our proposed ping-pong DLL. We have made some modifications beyond a traditional DLL, as the following:

(1) We have replaced the TDL with two parallel primitive TDLs, jointly forming a cooperating ping-pong TDL. Since these two primitive TDLs take turn to produce the final output clock signal, *Clk_out*, a MUX is inserted at their outputs to decide who is in command.

(2) In order to facilitate ''instant handover'' between the two primitive TDLs, two so-called ''phase quantifiers'' (*PQ*-1 and *PQ*-2) are further added. Their functions are mainly to quantify the delay across the MUX, i.e., from signal labeled as $M1$ to *Clk_out*, and $M2$ to *Clk_out*, respectively. The scheme of the instant handover and the details of the phase quantifiers will be further explained in latter sections.

(3) The controller is more sophisticated. Not only it will take the *lead/lag* signal produced by the Phase Detector, but also the input clock signal, *Clk_in*, and the signals produced by the two ''phase quantifiers'', i.e., *PA*-1 and *PA*-2, to make necessary decision and thereby regulating the entire operation of the DLL. It produces the 2-level control codes for the two primitive TDLs, i.e., $< \beta_1, \gamma_1 >$ for TDL-1, and $< \beta_2, \gamma_2 >$ for TDL-2. And it also produces two auxiliary 2-level control codes for the two ''phase quantifiers'', namely $< x_1, y_1 >$ for *PQ*-1, and $< x_2, y_2 >$ for *PQ*-2. The meanings of some key signals are summarized in Table 1 for easier reference.

TABLE 1. The meanings of key signals in the our ping-pong DLL.

Signal Name	Meaning
Clk_in	Input Clock Signal
Clk_out	Output Clock Signal (at the output of MUX, and entire DLL)
S	Control signal to select the TDL in command, (TDL-1 when SEL=='0', and TDL-2 when SEL=='1')
M1 and M2	Output Signals of TDL-1 and TDL-2, respectively
PA-1 and PA-2	Output Signals of PQ-1 and PQ-2, respectively (PA-1 characterizes the delay amount from M1 to Clk_out) (PA-2 characterizes the delay amount from M2 to Clk_out)
PAR-1 and PAR-2	Registers holding PA-1 and PA-2, respectively
$<\beta_1, \gamma_1>$ and $<\beta_2, \gamma_2>$	2-Level Control Codes for TDL-1 and TDL-2, respectively

Acronyms:
TDL-1 and TDL-2 are the two primitive Tunable Delay Lines
PQ-1 and PQ-2 are the two Phase Quantifiers

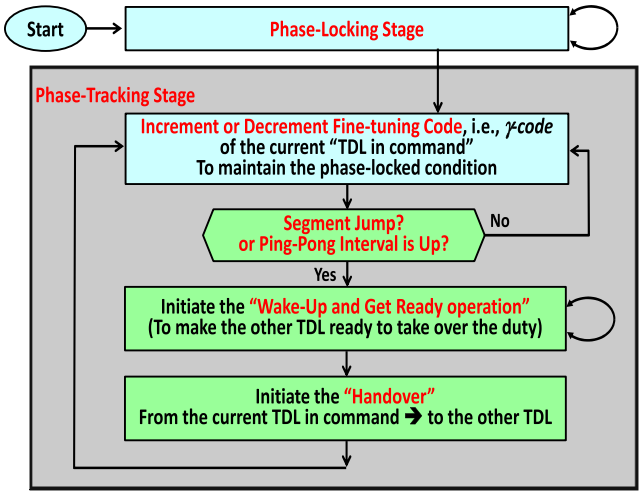


FIGURE 7. The overall operation of our ping-pong DLL. The phase-tracking stage is now more involved with the need to perform ping-pong operation in which the "role of command" is transferred from one primitive TDL to the other on a regular basis.

B. PING-PONG PROTOCOL

The overall operation of the proposed ping-pong DLL is illustrated in Fig. 7. Similar to a traditional DLL, it has two stages – phase-locking and phase-tracking.

Initially, the DLL performs phase-locking which finds a proper 2-level control code in a binary search manner for a designated TDL to reach a phase-locked condition (e.g., signal *Clk_out* is in phase with signal *Clk_in*). In our implementation, if the *lead/lag* signal changes its polarity

for more than 4 times within a time frame of 16 consecutive cycles, the DLL enters the ''locked state'' and phase-tracking stage begins.

In the phase-tracking stage, DLL increments or decrements the fine-tuning code, i.e., the γ-code, to maintain the phase-locked condition. Nevertheless, the phase-tracking stage is now more involved. In addition to the tuning of the fine-tuning code of the TDL for maintaining the phase-locked condition, we also need to perform a so-called ''ping-pong operation'' in which *the responsibility of driving the output clock signal,Clk_out,is transferred from one TDL to the other*. The ping-pong operation involves the following 3 steps:

(*Step 1:*) When the fine-tuning code of the ''TDL in command'' is already very close to its boundary of the current segment in the TDL's delay profile, then a segment jump for the current ''TDL in command'' is imminent. Therefore, a ping-pong operation is initiated and the command is to be transferred to the other primitive TDL. Also, our DLL is designed in a way that the two primitive TDLs will take turn to become the ''TDL in command'' regularly for every designated amount of time, e.g., 10,000 clock cycles of the input reference clock. This amount of time is referred to as ping-pong interval. When a ping-pong interval expires, a ping-pong operation will be initiated even though the ''TDL in command'' does not have to perform segment jumping yet. The reason of such regular ping-pong operations will become clear in our latter discussion. But in a nutshell, it is a mechanism to facilitate ''regular delay calibration for each primitive TDL to keep track of the environmental changes''.

(*Step 2:*) Once the ping-pong operation is initiated, some preparation work is needed before the other TDL can actually take over the ''role of command''. This preparation work is called wake-up and get ready (WUGR) operation. In a nutshell, the other TDL needs to wake up if it is in the sleep state and start to perform ''open-loop delay calibration'' with the assistance of the ''phase quantifier'' aforementioned in the subsection III.A on ''Basic Architecture''. More details will be revealed later. Once this WUGR operation is completed, the delay of the other TDL (which is not in command yet) will approximately match that of the current ''TDL in command'', and therefore, the jitter amount when the actual ping-pong handover occurs is mitigated.

(*Step 3:*) Once the other primitive TDL has completed the WUGR operation, we can flip the value of the control signal of the MUX, i.e., signal S, to actually switch the ''TDL in command'' from the current one to the other. This ''handover action'' should occur at a time instant with ample timing margins away from the clock edges of *Clk_out* in order to avoid creating glitches in the final output clock signal, *Clk_out*.

C. WAKE-UP AND GET READY (WUGR) OPERATION

The WUGR operation is the most challenging part of this work, in which we rely on ''phase quantifiers'' (PQ-1 and PQ-2) to perform online MUX delay characterization.

Fig. 8 shows the micro-architecture to support the WUGR operation, including the following key issues.

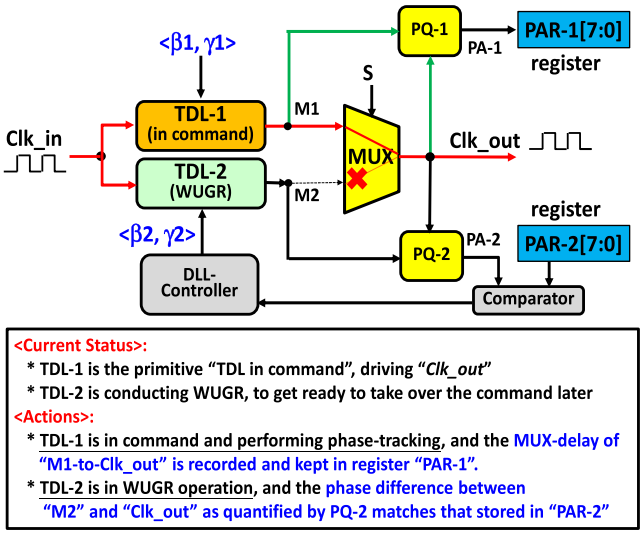


FIGURE 8. Micro-architecture to support the WUGR operation. This example assumes that TDL-1 is in command and TDL-2 is conducting the WUGR operation.

(1) During the phase-locking stage, the MUX delays (including ''M1-to-Clk_out'' and ''M2-to-Clk_out'') have both quantified by the two phase quantifiers (PQ-1 and PQ-2) as two digital codes, PA-1 and PA-2, recorded in two registers PAR-1 and PAR-2 (where PAR stands for *phase amount register*).

(2) For simplicity without losing generality, we assume that TDL-1 is now in command, driving the output clock signal *Clk_out*. On one hand, it continues to update the fine-tuning code so as to maintain the phase-locked condition. On the other hand, the MUX delay (''M1-to-Clk_out'') will be quantified regularly and recorded in register PAR-1.

(3) When the fine-tuning code of TDL-1, i.e., $\gamma 1$, has entered a warning zone near the boundary of its segment in the delay profile, the WUGR operation of the other TDL, i.e., TDL-2, will be triggered. The WUGR operation of TDL-2 is conducted in conjunction with the above operation of TDL-1.

(4) *The WUGR operation of TDL-2, as controlled by DLL-controller, is conducted in an open-loop configuration*. It is mainly a process that repeatedly updates the control code of TDL-2, including $\gamma 2$ and possibly $\beta 2$, so that it will produce a signal at *M2* such that the following WUGR condition is satisfied:

The phase difference of signal *M2* and signal *Clk_out* as quantified by PQ-2 produces a digital code matching that previously stored in register PAR-2.

It is notable that the digital code stored in register PAR-2 is the MUX-delay (from *M2* to *Clk_out*) recorded last time when TDL-2 is still ''in command''. If the above WUGR condition is satisfied, then it implies the following condition:

The current TDL-2 delay plus the recently recorded MUX delay (from M2 to Clk_out) can produce at the DLL's output

TABLE 2. The meanings of flags used in our DLL controller for regulating the ping-ping procedure.

Name of Flag	Meaning
S	Set to '0' when TDL-1 is in command Set to '1' when TDL-2 is in command
Zone	Set to '1' when (1) The current "TDL in command" is close to the boundary of segment (e.g., γ-code is less than 4 or larger than 31), or (2) the time elapsed since the last "ping-pong handover action" has reached 10,000.
Action_now	Set to '1' when (1) The current "TDL in command" has reached boundary of the segment (e.g., γ-code is 0 or 35), or (2) the time elapsed since the last "ping-pong handover action" has reached 11,000.

a signal very close to the current output clock signal Clk_out (driven by TDL-1) in their phases. As a result, when we switch the command of the DLL from TDL-1 to TDL-2, the new clock output signal will match that of the old clock output signal in their phases and therefore a low jitter can be ensured during the ping-pong handover action.

There is one subtle point regarding why the above WUGR operation is needed. An idle TDL may have experienced a segment jump a while ago when it relinquished the command to the other TDL. Now it may have been waken up in a new segment in the delay profile, and thus it needs the WUGR operation to adjust itself in the new segment of the delay profile until it can reach a new phase-locked condition according to the latest operating environment.

Table 2 summarizes several binary flags and their meanings, used in our DLL controller for regulating the ping-pong procedure. For example, flag *Zone* is used to indicate if the TDL in command has been operating in a warning zone (e.g., when γ-code is less than 4 or larger than 31, or the number of clock cycles elapsed since the last "handover action" has reached a pre-designated value (e.g., 10,000). When this flag is set to '1', the WUGR operation needs to be initiated for the other TDL. Also, a flag "*Action_now*" is used to indicate that a "handover action" needs to be executed immediately (e.g., when γ-code is now 0 or 35, or the number of clock cycles elapsed since the last "handover action" has reached a pre-designated threshold value (e.g., 11,000).

D. CIRCUIT AND OPERATION OF PHASE QUANTIFIER

To support our ping-pong operation, two phase quantifiers with high resolutions (e.g., with a time resolution of 3ps) are needed. Recall that a *phase quantifier is*used to calibrate the delay across the primary MUX in our ping-pong architecture at a particular moment. So, one of its input is either M1 or M2, the other input is *Clk_out*. The typical delay across a MUX in the 90nm CMOS process is shown in Fig. 9(a), ranging from 109ps to 185ps, under 5 process corners denoted as {TT, FF, SS, SF, FS}. Therefore, our phase quantifier needs to cover a delay range larger than [109ps, 185ps] to ensure robust operation under process variations. In order to minimize the area overhead, we incorporate a 2-stage micro-architecture

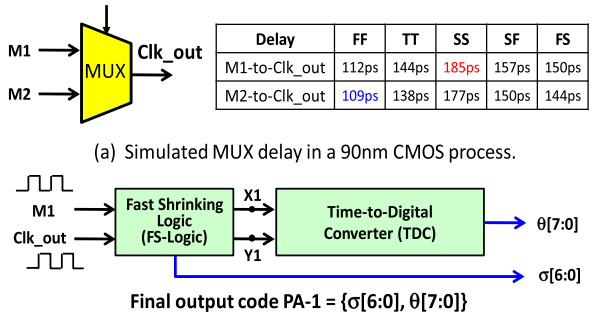


FIGURE 9. The phase quantifiers and the target MUX delay to be calibrated. (a) Simulated MUX delay in a 90 nm CMOS process and (b) the two stages of a phase quantifier.

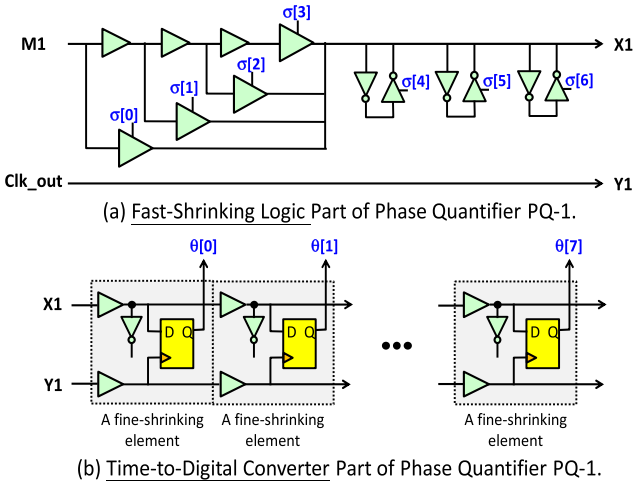


FIGURE 10. Detailed circuitry of a phase quantifier.

as shown in Fig. 9(b), including a fast-shrinking logic (FS-Logic) as the first stage, and a time-to-digital converter (TDC) as the second stage.

The function of the FS-Logic is to quickly reduce the amount of the input timing signal (i.e., the phase difference between M1 and *Clk_out*, assuming for phase quantifier PQ-1) to a smaller range (e.g., less than 30ps). Then, this timing signal (defined by the phase difference between the two signals at nodes X1 and Y1) is further provided to a higher-resolution TDC in the second stage for encoding. The produced output code for PQ-1 is denoted as PA-1, further consisting of two digital codes - Fast-Shrinking part, σ[6:0], and TDC part, θ[7:0]. Both of these two digital codes are recorded to assist the subsequent WUGR operation. Fig. 10 shows the detailed circuitry of a phase quantifier.

Fig. 10(a) is the Fast-Shrinking part. The 4 LSB bits of σ-code, i.e., σ[3:0], is one-hot. Each pin when turned ON selects a unique path in the upper path. When considered together with the fixed delay along the lower path, it creates a unique amount of phase shrinking. The 3 LSB bits of σ-code, i.e., σ[6:4], is thermometer like. More pins when turned ON creates larger capacitance at the output signal of

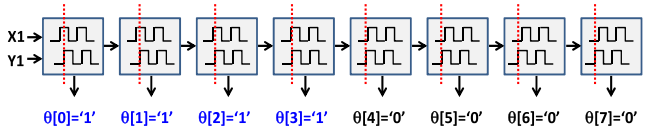


FIGURE 11. A illustration of a "differential timing signal at (X1, Y1)" passing through the TDC, while producing a thermometer code θ[7:0].

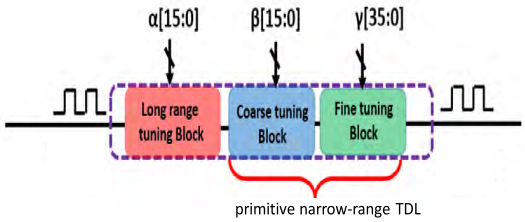
the upper path, and thus longer upper path delay. Again, when considered together with the fixed delay along the lower path, it creates a larger amount of phase shrinking as observed as (X1, Y1). In our implementation, we use CLKBUFX2 for the regular buffer and TBUFX4 for those tri-state buffers. Typically, the phase shrinking amount of our Fast-Shrinking Logic is approximately in a range from [101ps, 247ps] in a 90nm CMOS process, adequate to cover the MUX delay under calibration.

Fig. 10(b) is the TDC part. We follow the Vernier delay line based TDC concept. The two input signals propagates through two different paths of delay - the upper path with slightly longer delay, and the lower path with slightly shorter delay. Note that their delay difference is created by adding buffers as extra loading at the upper path. The structure can be viewed as the cascade of 8 fine-shrinking elements, each having a shrinking ability of about 3ps. After each fine-shrinking element, the differential signal is sampled by a D-type Flip-Flop, to produce a bit of the TDC's output code, from θ[0] to θ[7], respectively.

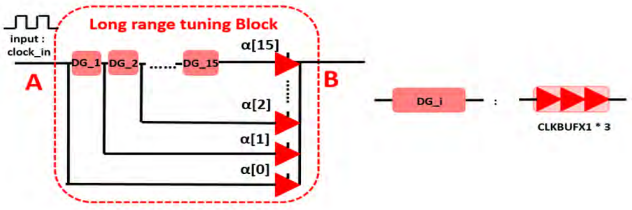
(Example 1): Consider a differential timing signal at (X1, Y1) at the end of the Fast-Shrinking Logic in Fig. 11. This differential timing signal then passes through the TDC part, with their value sampled by the D-type flip-flops changing from positive to negative. Thus, we will have an output of thermometer code θ[7:0] = [00001111].

Before we conclude this subsection, we discuss one detail of how phase quantifiers are utilized in the ping-pong procedure operation. In this discussion, we assume that TDL-1 is in command, and TDL-2 is in WUGR operation.

(1) For TDL-1, the PQ-1 is operated from time to time, and the resulting output code PA-1 (including the σ1[6:0] part and θ1[6:0] part) is recorded into register PAR-1.
(2) For TDL-2 in WUGR operation, the previously recorded digital code of the FS-Logic part in register PAR-2, i.e., σ2[6:0], is used to configure the FS-Logic part of PQ-2. Then, an iterative process is employed to search for a control code $< \beta 2, \gamma 2 >$ for the TDL-2 so that the produced output signal at M2 will have a right timing to drive the PQ-2 and produce a new digital code for the TDC part, i.e., θ2[6:0], which matches the old digital code θ2[6:0] recorded in the register PAR-2 previously. When this match is complete, the open-loop calibration is completed and the TDL-2 is synchronized to TDL-1 and become ready for the imminent handover action.



(a) The architecture of a wide-range TDL.



(b) The architecture of Long-Range Tuning Block.

FIGURE 12. The architecture of wide-range TDL.

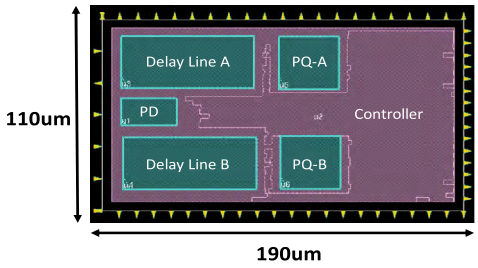


FIGURE 13. Layout of proposed DLL.

E. WIDE-RANGE DELAY TUNING

As illustrated previously, we can construct a Segmented TDL using only 102 standard cells, covering a tunable delay range from 545ps to 778ps, controlled by two-level control codes - a 16-bit coarse-tuning thermometer code β[15:0] and a 36-bit fine-tuning thermometer code γ[35:0]. However, the delay range is certainly too narrow. Thus, we need to add another Long-Range Tuning Block, controlled by another 16-bit one-hot code, namely α[15:0], as illustrated in Fig. 12. Note that the time resolution of the fine-tuning γ-code can be as small as 1ps in a 90nm CMOS process, while the time resolution of the long-range tuning α-code is roughly 200ps in our design. Here the DG means the delay group, made up of three clock buffer cells of CLKBUFX1, representing the time resolution of the Delay Block. After adding this Delay, our design can support the DLL operation for input clock signals from 400MHz to 1GHz over three process corners by post-layout simulation.

IV. SIMULATION RESULTS

A. LAYOUT AND SIMULATION

We have realized the proposed cell-based DLL design in a 90nm CMOS process with the layout show in Fig. 13. The layout size is 0.023mm^2 (with a configuration of 210μm $\times$ 110μ m). It has been divided into six hard macros - two

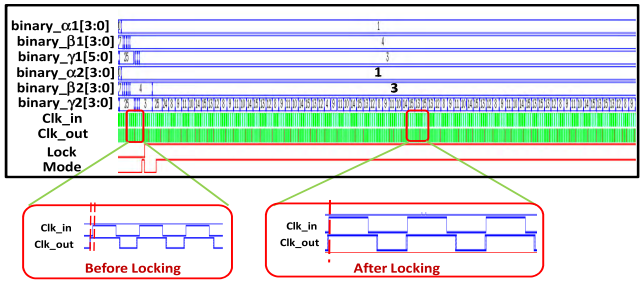


FIGURE 14. Post-layout simulation waveforms of our DLLs under 1GHz input clock signal and stable operating conditions (VDD = 1V, 25°C).

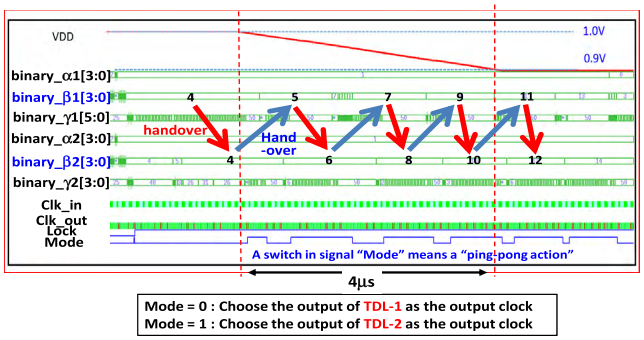


FIGURE 15. Post-layout simulation waveforms of our DLLs under 1GHz input clock signal under more changing operating conditions (VDD changes from 1V to 0.9V gradually in 4μs, 25°C).

long-range TDLs, two phase quantifiers, one phase detector, and a controller.

Fig. 14 shows the post-layout simulation waveform of our DLL operated with 1GHz input clock signal with stable operating conditions (at 1V VDD and 25°C). The phase differences between input and output clock signals, i.e., *Clk_in* and *Clk_out*, before and after the phase-locking has been highlighted to demonstrate its correct function. Note that the operating conditions (such as *VDD* and temperature) have been kept stable during this simulation and thus the α-code and β-code of the "TDL in command - which is TDL2" have been stable at $\alpha 2 = 1$ and $\beta 2 = 3$ (in their binary versions instead of thermometer versions).

To demonstrate the resilience of our DLL, we perform another set of post-layout simulation with more varied operating conditions in which the VDD drops gradually from 1V to 0.9V in 4μs as shown in Fig. 15. Since temperature change is not easy to simulate and thus we attempt to use more supply voltage change to signify the effects of both VDD and temperature changes.

It is also notable that that even though the ping-pong interval (which is set to 10,000 cycles of the input clock signal) has not been reached yet during the simulation, segment jumps have occurred, and thus triggering the ping-pong actions for several times. This can be evidenced in two aspects. First, the control signal of the MUX, i.e., *Mode*, has been changing frequently, with each switch in its value from 0→1 or 1→0 signifies a ping-pong action (or handover in command

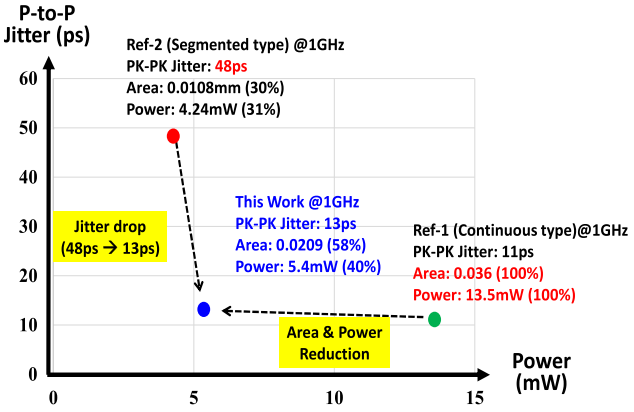


FIGURE 16. Comparisons with two previous cell-based DLL designs.

between the two primitive TDLs). Second, in order to catch up with the environmental changes in the operating conditions while remaining in lock, a Segmented TDL would have to keep increasing its coarse-tuning code, i.e., β-code, in this case from 4, 5, 6, …, etc. Such an increase has been jointly achieved by the two primitive TDLs taking turn to drive the output clock signals as demonstrated in their β-codes.

B. PERFORMANCE COMPARISON

In this subsection, we highlight the contributions of this work by comparing with two reference works in cell-based DLLs - (Ref-1) continuous type of DLL using parallel tri-state buffers, first proposed in [10], [11] and automated in [15], and (Ref-2) segmented type of DLL using both parallel tri-state buffers as well as "cell-based varied capacitance" first proposed in [12] and automated in [14] shown in Fig. 16.

We focus on three criteria, including area, power consumption, and Peak-to-Peak jitters (*Pk-Pk* jitters). Note that the *Pk-Pk* jitters are measured in simulation over 1000 clock cycles after locking.

(Comparison #1): We have achieved smaller area and less power consumption than Ref-1 (Continuous type) design. If the area and power consumption of the Ref-1 design are denoted as 100%, then the area and the power consumption of our ping-pong DLL is 61% and 53%, respectively, while the increase of the Pk-Pk jitter is only moderate from 11ps to 13ps.

(Comparison #2): We have solved the robust problem that have long plagued the Ref-2 (Segmented type) design due to the potential segment jumping under hostile operating environment. Even though a Ref-2 design could have a smaller area (30%) and a lower power consumption (31%) as compared to Ref-1 design, its Pk-Pk jitter could be as high as 48ps, mainly due to the segment jumps. On the other hand, if we adopt the proposed pong-pong scheme, the Pk-Pk jitter can be dramatically reduced significantly from 48ps down to only 13ps. In Fig. 17, we plot the histograms of the jitters observed by post-layout simulation within a time frame of 1000 clock period samples for both Ref-2 design and this work, for easier visual comparison.

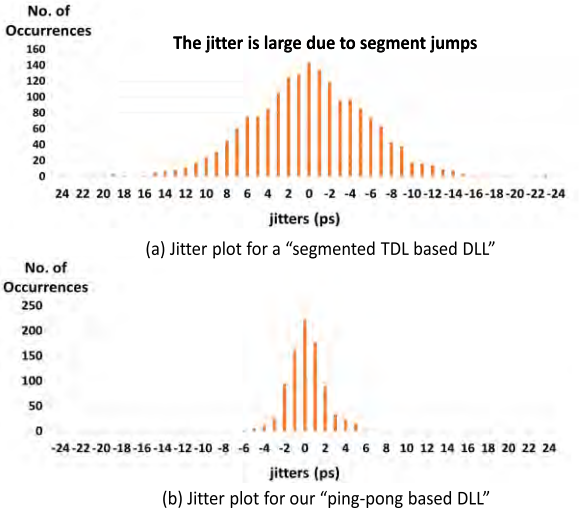


FIGURE 17. Simulated jitter plots of traditional segmented TDL-based DLL and the proposed ping-pong-based DLL, under a VDD-changing scenario.

We want to further point out that this segment-jumping problem has been the "Achilles' heel" of synthesizable PLLs or DLLs as evidenced by the following quote published very recently by a state-of-the-art commercial IP provider in May 2019 [18]:

"*Unfortunately, most (synthesiable) designs proposed do not (have wide linear continuous tuning ranges with high resolution on the clock frequency). They have a large number of frequency bands with highly non-linear frequency control. What this means for your PLL is that once locked, where a locking assist circuit has found the center of one of the bands, very little change in input frequency, voltage, or temperature, can be tolerated without the design producing large amounts of jitter and glitching.*"

This paper has given the synthesable solutions a boost by solving the above nasty segment-jumping problem that have plagued the cell-based PLLs or DLLs for a long time.

V. CONCLUSION

Significant jitter due to segment jumping in a cell-based DLL using segmented TDL is one major limiting factor that prevents it from widespread adoption. If this problem is not solved, then a segmented TDL based DLL could have reliability problem in an application in hostile operation conditions. In this work, we have resolved this problem by a ping-pong protocol, and thereby reducing the peak-to-peak jitter from 48ps to 13ps when operating in 1GHz. In this novel DLL architecture, two primitive Segmented TDLs are incorporated to take turn to produce the output clock signal. Through a sophisticated WUGR operation, instant handover can be achieved, while not creating noticeable jitter. Our implementation of a 400MHz-1GHz DLL using a 90nm CMOS process shows that, it also enjoys an area reduction of 35%, and a power reduction of 47% as compared to the traditional continuous type of DLLs, while keeping the increase of the jitter only marignally from 11ps to 13ps. On the othe other hand, if compared to a segmented DLL, then we enjoy a huge jitter reduction from 48ps to only 13ps. Furthermore, the proposed ping-pong protocol can be applied to any other timing circuits (such as Phased-Locked Loop) that incorporates a cell-based Tunable Delay Line.

ACKNOWLEDGMENT

The authors would like to thank the help of Chip Implementation Center, Taiwan for their assistance in providing the access to EDA tools.

REFERENCES

[1] Y.-H. Tu, K.-H. Cheng, H.-Y. Wei, and H.-Y. Huang, "A low jitter delay-locked-loop applied for DDR4," in *Proc. IEEE 16th Int. Symp. Design Diagnostics Electron. Circuits Syst. (DDECS)*, Apr. 2013, pp. 98–101.
[2] S.-L. Chen, M.-J. Ho, Y.-M. Sun, M. W. Lin, and J.-C. Lai, "An all-digital delay-locked loop for high-speed memory interface applications," in *Proc. Int. Symp. VLSI Design, Automat. Test (VLSI-DAT)*, Apr. 2014, pp. 1–4.
[3] C.-Y. Cheng, S.-Y. Huang, D.-M. Kwai, and Y.-F. Chou, "DLL-assisted clock synchronization method for multi-die ICs," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Boston, MA, USA, Nov. 2017, pp. 473–476.
[4] H.-H. Chang and S.-I. Liu, "A wide-range and fast-locking all-digital cycle-controlled delay-locked loop," *IEEE J. Solid-State Circuits*, vol. 40, no. 3, pp. 661–670, Mar. 2005.
[5] R.-J. Yang and S.-I. Liu, "A 40–550 MHz harmonic-free all-digital delay-locked loop using a variable SAR algorithm," *IEEE J. Solid-State Circuits*, vol. 42, no. 2, pp. 361–373, Feb. 2007.
[6] J. A. Tierno, A. V. Rylyakov, and D. J. Friedman, "A wide power supply range, wide tuning range, all static CMOS all digital PLL in 65 nm SOI," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 42–51, Jan. 2008.
[7] Y.-S. Kim, S.-K. Lee, H.-J. Park, and J.-Y. Sim, "A 110 MHz to 1.4 GHz locking 40-phase all-digital DLL," *IEEE J. Solid-State Circuits*, vol. 46, no. 2, pp. 435–444, Feb. 2011.
[8] M.-H. Hsieh, L.-H. Chen, S.-I. Liu, and C. C.-P. Chen, "A 6.7 MHz-to-1.24 GHz 0.0318 mm^2 fast-locking all-digital DLL in 90 nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2012, pp. 244–245.
[9] C.-Y. Yao, Y.-H. Ho, Y.-Y. Chiu, and R.-J. Yang, "Designing a SAR-based all-digital delay-locked loop with constant acquisition cycles using a resettable delay line," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 3, pp. 567–574, Mar. 2014.
[10] T. Olsson and P. Nilsson, "Fully integrated standard cell digital PLL," *IEE Electron. Lett.*, vol. 37, pp. 211–212, Feb. 2001.
[11] T. Olsson and P. Nilsson, "A digitally controlled PLL for SoC applications," *IEEE J. Solid-State Circuits*, vol. 39, no. 5, pp. 751–760, May 2004.
[12] D. Sheng, C.-C. Chung, and C.-Y. Lee, "An ultra-low-power and portable digitally controlled oscillator for SoC applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 11, pp. 954–958, Nov. 2007.
[13] J.-Y. Liu, S.-Y. Huang, and T.-S. Chu, "Cell-based programmable phase shifter design for pulsed radar SoC," in *Proc. IEEE 11th Int. Conf. ASIC (ASICON)*, Nov. 2015, pp. 1–4.
[14] C.-W. Tzeng, S.-Y. Huang, and P.-Y. Chao, "Parameterized all-digital PLL architecture and its compiler to support easy process migration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 3, pp. 621–630, Mar. 2014.
[15] P.-C. Huang and S.-Y. Huang, "Cell-based delay locked loop compiler," in *Proc. Int. SoC Design Conf. (ISOCC)*, Oct. 2016, pp. 91–92.
[16] P.-Y. Chao, C.-W. Tzeng, S.-Y. Huang, C.-C. Weng, and S.-C. Fang, "Process-resilient low-jitter all-digital PLL via smooth code-jumping," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 12, pp. 2240–2249, Dec. 2013.
[17] Z.-H. Zhang, W. Chu, and S.-Y. Huang, "The ping-pong tunable delay line in a super-resilient delay-locked loop," in *Proc. 56th Annu. Design Automat. Conf. (DAC)*, Jun. 2019, p. 231.
[18] J. G. Maneatis. (May 22, 2019). *Why Synthesizable-Digital PLLs Are No Substitute for Hardened Mixed-Signal PLLs*. [Online]. Available: https://www.chipestimate.com/Why-Synthesizable-digital-PLLs-Are-No-Substitute-for-Hardened-Mixed-signal-PLLs/True-Circuits/blogs/3175

ZHENG-HONG ZHANG received the B.S. degree in electrical engineering from the National University of Kaohsiung, Taiwan, in 2015, and the M.S. degree from National Tsing Hua University, in 2018. In 2017, he mainly studied how to make the delay locked loop has both the wide tuning delay range and a little jitter in code jumping, which is the prototype of the ping-pong DLL. His research interests mainly include IC design and the test methods for multi-die integrated ICs.

WEI CHU received the B.S. degree in electrical engineering from National Central University, Taiwan, in 2017, and the M.S. degree from National Tsing Hua University. Her research interest mainly includes the test methods for multi-die integrated ICs.

SHI-YU HUANG (S'94–M'97–SM'12) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, in 1988 and 1992, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1997, respectively. He has been a Faculty Member of the Electrical Engineering Department, National Tsing Hua University, Taiwan, since 1999. His research interests include VLSI design, automation, and testing, with a current emphasis on all-digital phase-locked loop (ADPLL) design and its application in parametric fault testing and monitoring in 3D ICs. He has served as the Program Co-Chair or the General Co-Chair for a number of IEEE conference/symposia/workshops including 2004 and 2009 ATS, 2005 and 2006 MTDT, 2014 and 2015 VLSI-DAT, and 2017 ITC-Asia. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS, since 2015.