JOURNAL OF ELECTRONIC TESTING: Theory and Applications 22, 387–398, 2006

DOI: 10.1007/s10836-006-9444-3

On-Chip Random Jitter Testing Using Low Tap-Count Coarse Delay Lines

JIUN-LANG HUANG
Graduate Institute of Electronics Engineering/Department of Electrical Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei 106, Taiwan
jlhuang@cc.ee.ntu.edu.tw

Received September 10, 2005; Revised March 16, 2006; Published Online November 22, 2006

Editor: K.-T. Cheng

Abstract. An on-chip RMS jitter testing technique for design-for-test (DfT) applications is presented in this paper. In addition to utilizing a less complicated low tap-count variable delay line to sample the jitter's cumulative density function (CDF), a sophisticated post-processing algorithm is developed to enhance process variation tolerance. Our simulation results show that using an eight-tap delay line, the probability of making correct pass/fail decisions is higher than 99% in the presence of up to 30% delay line value deviations.

Keywords: jitter measurement, random jitter, design-for-test, analog/mixed-signal testing

1. Introduction

With the advent of IC fabrication technology, the concept of System-on-Chip (SoC) has become a reality. In addition to cost reduction and reliability improvement, the capability of SoC to integrate a whole system onto the same die also enables new multimedia and communication applications all of which require high data transmission bandwidth. As the data bandwidth demand continues elevating, high-speed serial links (HSL) have gradually replaced parallel data transmission of which the achievable bandwidth is severely limited by skew.

This paper is concerned with jitter testing. Jitter is the deviation in a signal's output transitions from their ideal positions. For a communication system, BER is the ultimate figure of merits. However, explicit BER measurement is a time-consuming task and impractical during manufacturing testing—it would take a 1.5 Gbps system 11 min for a single bit error to occur if its BER is 10^{-12}. As a result, characterization of jitter performance becomes essential because of the high correlation between jitter characteristics and BER.

Jitter testing for HSL systems is a difficult task which usually relies on expensive automatic test equipment (ATE) because of the high symbol rate and low voltage swing needed to push for higher bandwidth. For example, the first generation Serialized AT Attachment (SATA) standard [6] features 1.5 Gbps symbol rate and 250 mV voltage swing (single-ended, 500 mV differential). During manufacturing testing, great efforts are made to guarantee the quality of signal paths between chip I/O pins and the test equipment in the existence of parasitic capacitance/inductance and environmental noise. Nevertheless, the challenge of jitter testing gets bigger as the level of IC integration continues growing. For ICs that possess tens or even hundreds of HSL channels, testing all the channels in parallel will require prohibitively expensive test equipment.

One promising solution to reducing jitter testing cost is design-for-test. The idea is to add to the original design dedicated test circuitry that transforms the jitter information of interest into a form (usually digital) that can be easily delivered to the ATE for pass/fail decision or characterization. Since DfT circuitry can be made close to the signal under test, it is not limited by the bandwidth of the chip I/O pins and the signal path quality between the IC and ATE, which substantially lowers the ATE and test fixture performance requirement and enables concurrent testing of multiple HSL channels. Also worth noting is that DfT and functional circuits are always at the same speed and performance level because they are manufactured using the same process technology. The main challenges of DfT approaches include (1) the associated area overhead, (2) the impact on design efforts and performance, and (3) the achievable test resolution and accuracy.

Many DfT approaches to jitter measurement/testing have been reported in the past years. In [1, 2, 9–11], time-to-digital conversion (TDC) circuits are used to explicitly measure the signal periods. After collecting a sufficiently

large number of signal periods, one may apply various DSP techniques [4] to derive the desired jitter characteristics, e.g., periodic jitter amplitudes and frequencies, and the RMS jitter value. To achieve high timing resolution, most TDC techniques are explicitly or implicitly based on the vernier delay line concept. In spite of its high resolution, direct application of vernier delay line in DfT techniques is limited by (1) large hardware overhead, (2) delay line linearity, and (3) long measurement time. To resolve the linearity issue, [1] proposes a component invariant technique, while [9] settles the problem by characterizing the delay line non-linearity for calibration during post-analysis. The VCOBIST technique [11] employs two ring oscillators to measure the signal periods. In this approach, the measurement resolution equals the difference of the two oscillators' periods; therefore, it requires careful tuning and control such that the two oscillation frequencies are close enough to realize the desired resolution.

Compared to TDC approaches, CDF-based jitter extraction techniques incur less hardware overhead and are less sensitive to the delay line non-linearity. Assuming that the jitter's probability density function (PDF) is Gaussian or Gaussian-like, CDF-based jitter measurement approaches ([3, 7]) utilize a variable delay line together with a phase comparator to sample the jitter's CDF curve. Under the Gaussian distribution assumption, two CDF samples are sufficient to derive the RMS jitter value. The major limitation of CDF-based approaches is the Gaussian distribution assumption. In addition, measurement accuracy is affected by (1) the inherent delay line jitter [8], (2) delay line value deviations [12], and (3) the actual RMS jitter value.

In this paper, a CDF-based RMS jitter testing technique is presented. The main contributions of the proposed approach are as follows.

1.1. Lower DfT Hardware Complexity

The core DfT circuitry consists of a variable delay line and a phase comparator. In practice, the phase comparator can be realized with a D-type flip-flop (DFF); thus, we focus on relaxing the delay line resolution requirement and reducing the number of delay taps. As will be shown, the delay line resolution does not have to be a fraction of the RMS jitter value. In fact, with the proposed post-analysis algorithm, a relatively coarse resolution—about one half the *RMS jitter specification*—is sufficient regardless of the actual RMS jitter value. As a result of the coarse resolution, eight taps are enough to tolerate reasonable delay line value variations.

1.2. High Process Variation Tolerance

Both the delay line configuration and the post-processing algorithm parameters are selected with process variations in mind. As a result, even in the existence of up to 30% delay line deviations, the obtained CDF samples are still sufficient for the post-processing algorithm to make correct pass/fail decisions.

Simulation results show that the proposed DfT hardware together with the post-processing algorithm is a promising solution: in the existence of up to 30% delay line deviations, the probability of making correct pass/fail decisions is 99%.

This paper is organized as follows. In Section 2, CDF-based jitter measurement methods are briefly reviewed. Then, the proposed RMS jitter measurement technique is presented in Section 3. Section 4 details the simulated annealing (SA) based heuristic that determines the delay line and post-analysis parameters. In Section 5, numerical simulation results are shown and discussed to validate our technique. Finally, we conclude this work in Section 6.

2. CDF-Based RMS Jitter Measurement Techniques

In this section, we will review the CDF-based jitter measurement techniques in [3, 7]. The key steps are CDF sampling, delay line calibration, and RMS jitter computation.

2.1. Jitter CDF Sampling

One simple approach to sample the jitter CDF curve is to utilize an adjustable delay line and a phase comparator. Fig. 1(a) depicts the CDF sampling method in [7]. It employs two delay lines to adjust the time interval between the edges of the signal under test (SUT) and the *ideal* reference clock $\left(CLK_{ref}\right)$. The DFF functions as a phase comparator to determine the phase relationship (lead or lag) between CLK'_{ref} and SUT'. In Fig. 1(b), the selected adjustable delay value causes the rising edge of SUT', B, to lag that of CLK'_{ref}, A, by Δd. If Δd is positive, the DFF output should remain zero. However, in the existence of jitter, B deviates from its nominal position and may lead A which causes the DFF output to become one. Let PDF_{edge} be the jitter PDF of SUT with the origin at the ideal position of B. The probability that the DFF output is one, called the one probability, is as follows.

$$p = \int_{-\infty}^{-\Delta d} PDF_{edge}(t)dt \tag{1}$$

$$= CDF_{edge}(-\Delta\, d) \tag{2}$$

where CDF_{edge} is the CDF of B's actual position.

Another delay line based CDF sampling method is shown in Fig. 2(a) [3]. Without the ideal clock as the timing reference, phase comparisons are made between the rising edges of SUT and a delayed version of itself, denoted by SUT'. The phase comparator output is one/zero if B leads/lags A'. Figure 2(b) illustrates the under-

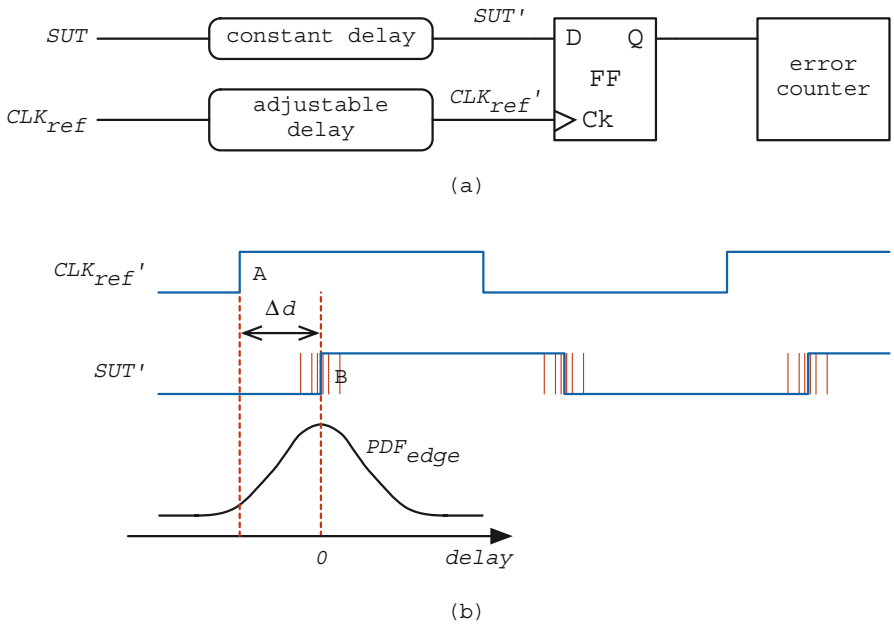

Fig. 1. The jitter CDF sampling method in [7].

lying idea. Let's use the rising edge A of SUT as the reference point and assume that the delay line value is d. If SUT is jitter free, the time interval between A and B equals the average period, T_{avg}, and the phase comparison result only depends on d. In Fig. 2(b), since d is chosen to be less than T_{avg}, B will always lead A'. Nevertheless, affected by jitter, B drifts away from its ideal position which may reverse the phase relationship. The one probability of the phase comparator, i.e., the probability that B leads A', is also described by Eq. (2).

For both jitter CDF sampling methods, to obtain $CDF_{edge}(-\Delta d)$, a number of phase comparisons are made and $CDF_{edge}(-\Delta d)$ equals the resulting one probability. To collect more CDF samples, one simply varies Δd (using the adjustable delay line) and repeats the same procedure. Note that the number of phase comparisons must be sufficiently large to limit the CDF sample errors to an acceptable extent.

Note that, in [7], an ideal reference clock is needed as the time reference. Thus, the sampled jitter CDF is the time interval error (TIE) CDF. The method in [3], on the other hand, does not need a reference clock as it compares the period of each cycle to the delay line value. The sampled jitter CDF is the period jitter CDF.

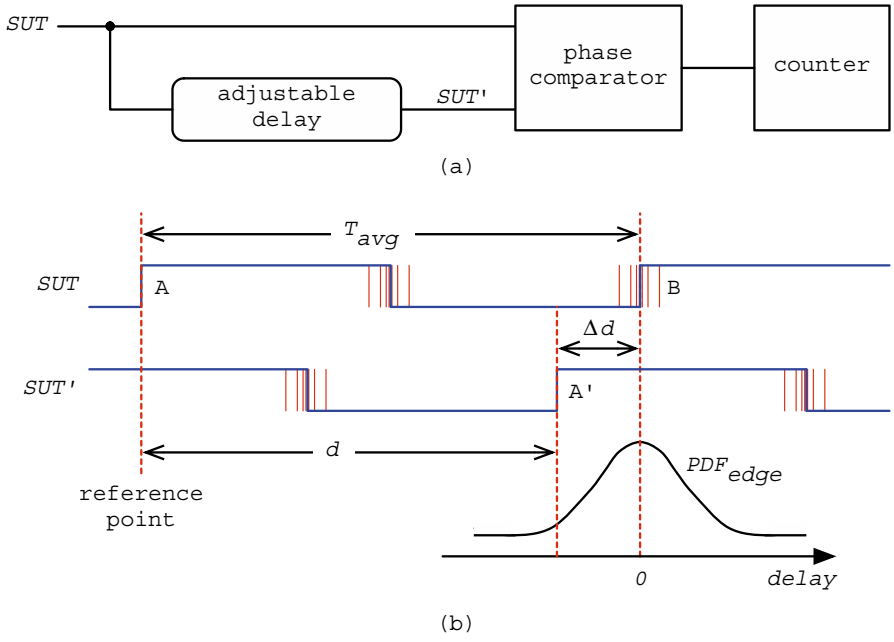

Fig. 2. The jitter CDF sampling method in [3].

2.2. *Delay Line Calibration*

One approach to measure the delay line value is the oscillation method. Figure 3 depicts one possible implementation. During calibration, the delay line together with the inverter forms an oscillator of which the oscillation period is $2\cdot(d + d_{inv})$, where d_{inv} is the inverter delay. During the observation window, the frequency counter counts the number of oscillation cycles from which the oscillation period can be derived.

One seemingly severe problem is that the oscillation loop consists of not only the delay line but also the inverter. As will be seen latter, the inverter delay will be cancelled out in the jitter computation algorithm and have no effect on the test results.

The main error sources of the oscillation approach include numerical rounding errors, systematic and random errors of the observation time window, and time and temperature variations. Except for the last one which has to be solved through circuit design, the negative effects of the other sources can be reduced to an acceptable level by lengthening the observation window [5].

2.3. *RMS Jitter Computation*

Ideally, one CDF sample is sufficient to derive the RMS value of a Gaussian distribution jitter. Let F_X be the normalized Gaussian CDF. The one probability p and the delay line value d_i are related by

$$p = F_X\left(\frac{d_i - d_0}{J_{RMS}}\right) \tag{3}$$

where J_{RMS} is the RMS jitter and d_0 is the delay line value for which the jitter CDF is 0.5. (In [7], d_0 equals the fixed delay value if the ideal *SUT* edges and the reference clock edges coincide. In [3], d_0 is the average *SUT* period.) From Eq. (3), RMS jitter can be derived by:

$$J_{RMS} = \frac{d_i - d_0}{F_X^{-1}(p)} \tag{4}$$

where F_X^{-1} denotes the inverse normalized Gaussian CDF. In reality, the use of Eq. (4) is difficult because measuring

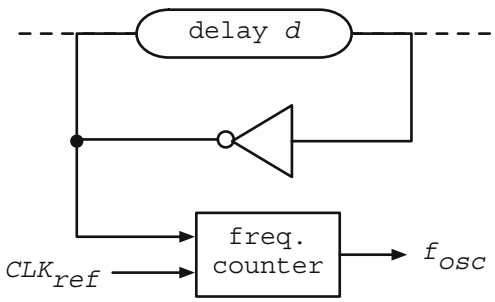

Fig. 3. Delay line calibration.

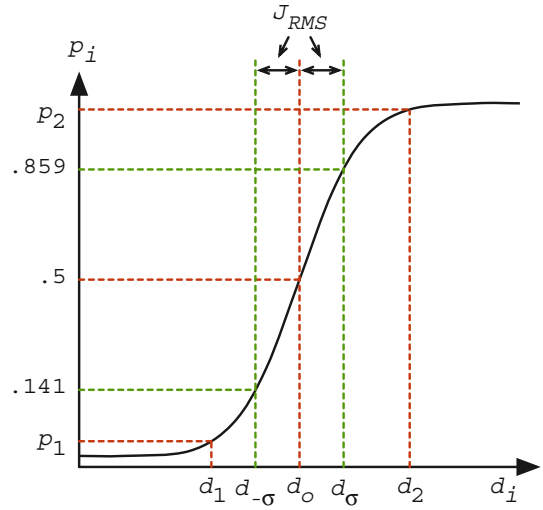

Fig. 4. Computing RMS value for Gaussian distribution jitters.

the actual value of d_1 (and d_0) is very challenging, if possible at all, for DfT applications.

To avoid measuring the exact delay values, the jitter computation method in [3] uses two CDF samples instead. Illustrated in Fig. 4, two CDF samples, (d_1, p_1) and (d_2, p_2), are obtained using a two-tap delay line with delay values d_1 and d_2. Under the Gaussian distribution assumption, the jitter CDF is a scaled (by J_{RMS}) and shifted (by d_0) version of F_X; therefore, RMS jitter can be derived by:

$$J_{RMS} = \frac{d_1 - d_2}{F_X^{-1}(p_1) - F_X^{-1}(p_2)} \tag{5}$$

$$= \frac{d_1 - d_2}{x_1 - x_2} \tag{6}$$

Note that the $x_i = F_X^{-1}(p_i)$ notation will be used throughout the rest of this paper. Compared to Eq. (4), knowledge of the delay difference, rather than the exact values, is needed to solve for J_{RMS}, which makes the previously mentioned oscillation method a viable delay line calibration solution. Because there is no closed form expression for the Gaussian CDF, solving the inverse Gaussian CDF is computation intensive. In practice, one can use a look-up table to store the inverse Gaussian CDF function.

The technique in [7] takes a different approach to RMS jitter computation. By taking more CDF samples (about 30 in the reported results), the approximate jitter CDF is obtained. From the jitter CDF, the two delay values that correspond to one probabilities of 14.1% and 85.9%, i.e., the two delays that are one RMS away from d_0, are identified or approximated. Let's denote the two delay values by $d_{-\sigma}$ and d_σ, respectively. The RMS jitter value is then derived by

$$J_{RMS} = \frac{d_\sigma - d_{-\sigma}}{2} \tag{7}$$

Compared to [3], this approach does not need inverse Gaussian CDF computation because more CDF samples are available.

Delay line deviations may jeopardize the success of both methods. In [3], d_1 and d_2 are selected to be within a few sigma's from d_0 to ensure high measurement accuracy. For the method in [7], the delay values must be close enough and should cover $d_{-\sigma}$ and d_{σ}. These design requirements may be invalidated by the inevitable process variations. Note that the actual RMS jitter value has a similar effect—larger/smaller RMS jitters effectively scale down/up the delay values. To resolve this problem, the technique in [3] utilizes the most process variation tolerant delay values, i.e., $T_{avg} \pm J_{spec}$ where J_{spec} is the RMS jitter pass/fail threshold. As for [7], one may increase the delay line tap count and/or resolution, implying more hardware overhead and longer test time.

3. The Proposed Algorithm

The proposed RMS jitter testing algorithm is based on jitter CDF sampling due to its lower DfT hardware complexity. In practice, depending on whether TIE or period jitter is of interest, one may choose the DfT circuits in Fig. 1 or Fig. 2, respectively. The distinctive features of the algorithm include the utilized low-tap count coarse delay line configuration, and its ability to tolerate up to 30% process variations and handle a wide RMS jitter range.

3.1. The Jitter Test Flow

Illustrated in Fig. 5 is the proposed jitter testing flow consisting of four phases.

3.1.1. Phase 1: Delay calibration and CDF sampling. For each delay tap t_i, its delay value (d_i) is measured using the oscillation method shown in Fig. 3. Then, CDF sampling is performed by making a sufficiently large number of comparisons for each tap t_i and recording the corresponding one probability p_i.

3.1.2. Phase 2: Pass test. The pass test is intended to determine if the RMS jitter is *less than* the jitter specification without explicit RMS jitter computation. In addition to reducing post-processing efforts, the pass test is important as it enables one to make a pass decision when the RMS jitter is well below the jitter specification—for such cases, jitter computation errors are usually unacceptable as the CDF samples themselves are not accurate enough. If a pass decision can be made, the test process terminates; otherwise, it proceeds to next phase.

3.1.3. Phase 3: Fail Test. Similar to the pass test, the fail test is intended to determine if the jitter RMS is *greater than* the jitter specification without jitter computation. Its purpose is to reduce post-processing efforts.

3.1.4. Phase 4: Jitter Computation. If no pass/fail decision can be made in phases two and three, jitter computation is performed to determine if RMS jitter is within specification. Recall that the jitter computation equation [Eq. (6)] only needs two CDF samples. A simple heuristic is developed to select a pair of CDF samples for jitter computation. In the following, we will discuss the last three phases. For ease of illustration, we will start from the jitter computation phase.

3.2. Jitter Computation

According to Eq. (6), jitter computation inaccuracies originate from errors associated with the delay value measurement and the one probability estimation. Since the delay value measurement errors (regardless of the constant offset) can either be reduced by lengthening the observation window or has to be resolved through circuit design techniques, our analyses focus on the impact of CDF sampling errors on the resulting jitter testing accuracy.

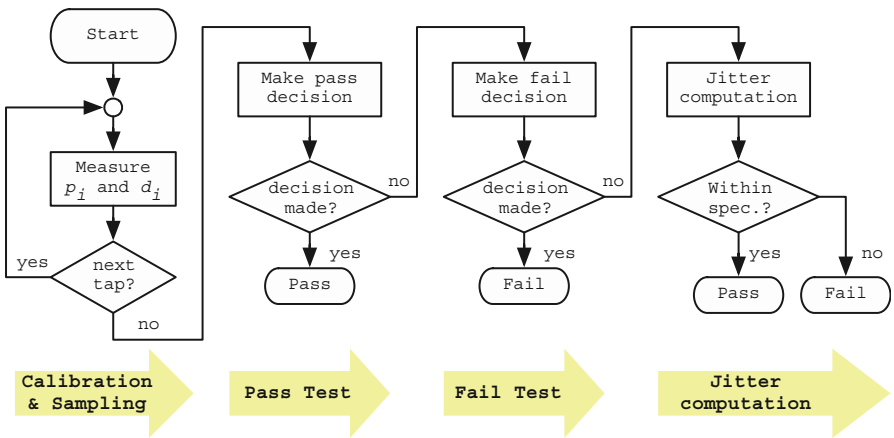

Fig. 5. The proposed RMS jitter testing algorithm.

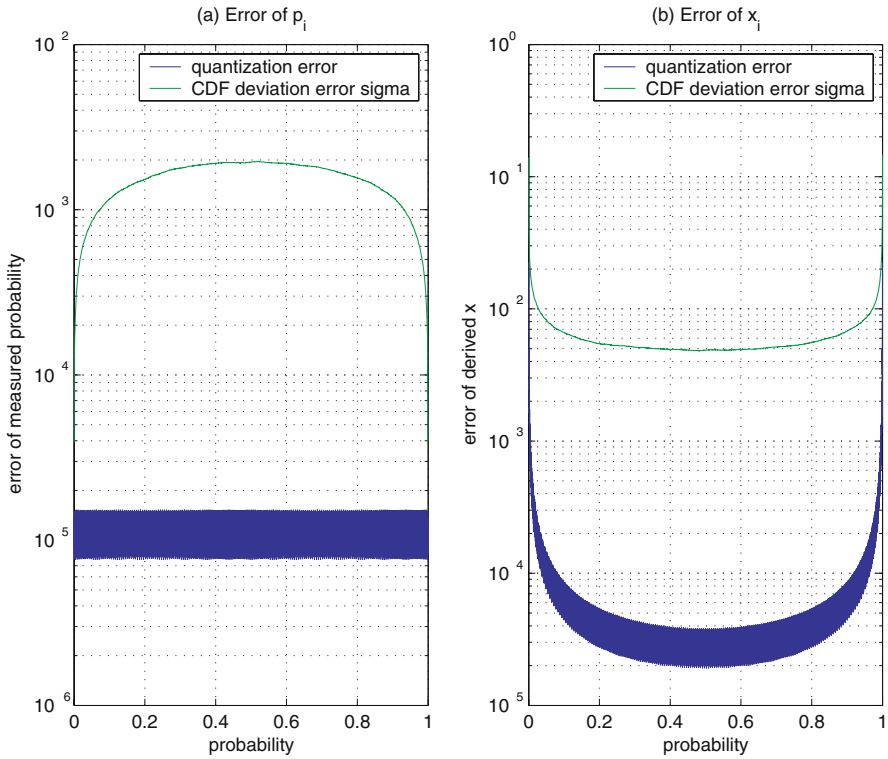

Fig. 6. Errors of measured probabilities (**a**) and computed x (**b**) ($N = 2^{14}$).

3.2.1. Quantization Error. If N phase comparisons are made to estimate the one probability p_i, the obtained p_i can assume only $N+1$ discrete values, i.e., $0, \frac{1}{N}, \frac{2}{N}, \cdots, \frac{N-1}{N}, 1$, which leads to quantization error. The bottom curve of Fig. 6(a) is the quantization error for $N = 2^{14}$. (The x and y axes of Fig. 6(a) are the ideal probability and the corresponding quantization error, respectively.) Apparently, the probability quantization error is bounded by $\frac{1}{N}$. Doubling the number of phase comparisons will effectively reduce the quantization error by a factor of two.

3.2.2. Finite Sample Size Error. Due to the limited number of phase comparisons that can be made, the observed CDF curve inevitably deviates from the ideal Gaussian distribution, which also induces CDF sampling errors. In Fig. 6(a), the upper curve is the *standard deviation* of the probability error caused by CDF deviation. Note that for each probability value, the corresponding error is a Gaussian distribution. Therefore, quadrupling the sample size will reduce the error by 50%. The CDF deviation error is in general greater than the quantization error, and the largest error occurs when the probability is 0.5.

3.2.3. CDF Sample Selection for Jitter Computation. In Fig. 6(b), the errors of computed x, $x = F_X^{-1}(p)$, are shown. (The upper and lower curves correspond to the quantization and CDF deviation errors, respectively.) Due to the steep tails at both ends of the inverse Gaussian CDF curve, the errors of x with respect to both the quantization and the CDF deviation errors become bathtub like curves.

From Fig. 6(b), it seems that one should pick the CDF samples of which the one probabilities are closer to 0.5 for jitter computation because the corresponding errors of x_i's are smaller. However, if the two CDF samples are too close, the denominator of Eq. (6) will be too small and as a result magnify the x_i errors. To determine the suitable CDF sample pair for jitter computation without too much computation overhead, we define a probability value p_{cmpt}, $\left(0 < p_{cmpt} < 0.5\right)$, to screen out less accurate CDF samples. As a compromise between the errors of x_i's and the magnitude of the denominator, the two CDF samples of which the one probabilities are the greatest and smallest within the range $\left[p_{cmpt}, 1 - p_{cmpt}\right]$ are selected for jitter computation.

```
1  C ← φ;
2  foreach tap t_i do
3  |   if p_cmpt ≤ p_i ≤ (1 − p_cmpt) then
4  |   |   C ← C ∪ {t_i};
5  |   end
6  end
7  if size(C) ≥ 2 then
8  |   t_up ← max(C);
9  |   t_down ← min(C);
10 |   x_up ← invCDFLookup(p_up);
11 |   x_down ← invCDFLookup(p_down);
12 |   J_RMS ← (d_up − d_down) / (x_up − x_down);
13 end
```

Algorithm 1: The jitter computation algorithm.

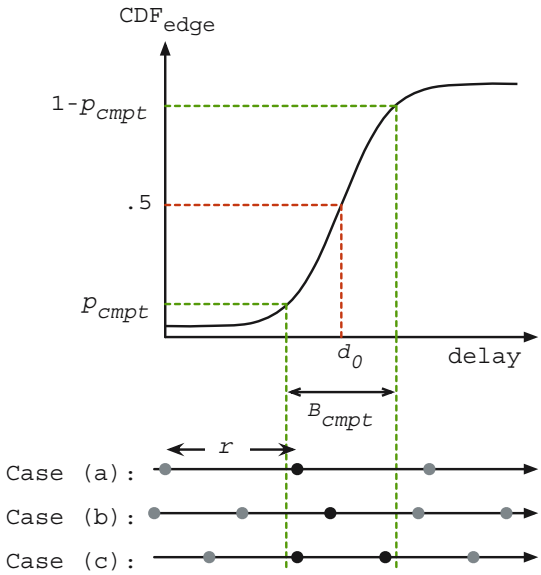

Fig. 7. The dependence of $Size(C)$ on r and J_{RMS}.

Shown in Algorithm 1 is the complete jitter computation algorithm. First, the delay taps of which the one probabilities are within $[p_{cmpt}, 1-p_{cmpt}]$ are identified and stored in the set C (lines 1–6). Jitter computation will not be performed if C contains less than two taps (line 7); otherwise, the minimum and maximum delay taps in C, denoted by t_{down} and t_{up} respectively, are selected for jitter computation (lines 8–9). In lines 10–11, p_{up} and p_{down} are the one probabilities of t_{up} and t_{down} respectively, and inverse CDF table lookup is performed to obtain x_{up} and x_{down}. RMS jitter is then derived using Eq. (6) (line 12). The computed RMS jitter is compared to the jitter specification J_{spec} to make the pass/fail decision.

Note that there are occasions, i.e., $Size(C) < 2$, when jitter computation is not applicable. Let's define the jitter computation band, B_{cmpt}, as the difference of the two delay values of which the one probabilities are p_{cmpt} and $1-p_{cmpt}$ (Fig. 7), respectively. One then has

$$B_{cmpt} = 2 \cdot J_{RMS} \cdot F_X^{-1}(1-p_{cmpt}) \tag{8}$$

Figure 7 illustrates how the size of C is affected by J_{RMS}, the delay line resolution r, and the delay value offset. (Each dot in the three cases represents a delay tap.)

Case (a) In this case, $r = 1.2 \cdot B_{cmpt}$. As a result, only one CDF sample falls inside the jitter computation band.

Case (b) r is reduced to $0.8 \cdot B_{cmpt}$. However, because of the delay value offset, $Size(C)$ is still less than two.

Case (c) Here r is the same as that in Case (b) but the delay offset is different. In this case, $Size(C) = 2$ and jitter computation is applicable.

The relationship between $Size(C)$ and B_{cmpt} (assuming arbitrary offset) is as follows:

$$Size(C) = \begin{cases} 0-1 & if \quad 0 < B_{cmpt} < r \\ 1-2 & if \quad r \le B_{cmpt} < 2r \\ 2-3 & if \quad 2r \le B_{cmpt} < 3r \\ \cdots & \\ k-k+1 & if \quad kr \le B_{cmpt} < (k+1)r \end{cases} \tag{9}$$

It is interesting to note that, since B_{cmpt} is proportional to J_{RMS}, for small RMS jitter values, the probability that jitter computation can be applied, i.e., $Size(C) \ge 2$, is low. One apparent solution is to reduce the delay line resolution r, which will however increase the number of delay taps and lengthen the delay calibration and CDF sampling time. To resolve this problem, the *pass test* is introduced.

3.3. The Pass Test

The purpose of adding the pass test to the test flow is to handle the cases when RMS jitter value is so small that jitter computation is either inaccurate or not applicable. In the pass test procedure, another parameter p_{pass}, $0 < p_{pass} < 0.5$, is introduced to decide if the pass test is applicable. The idea of pass test is depicted in Fig. 8 where

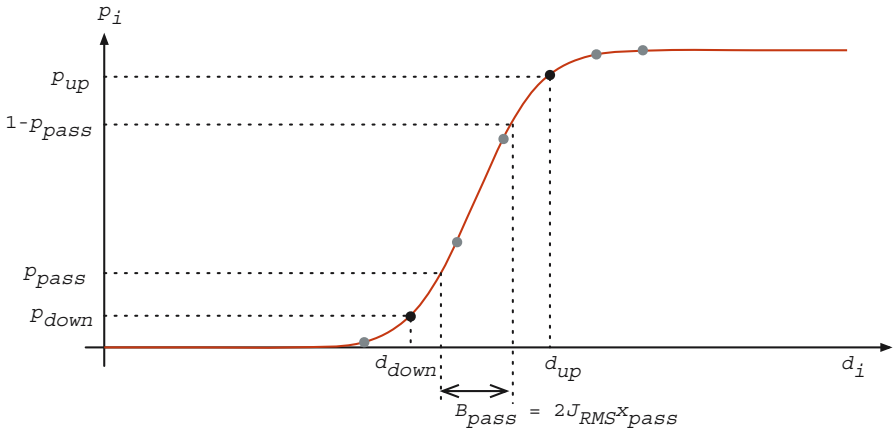

Fig. 8. Derivation of the RMS jitter upper bound.

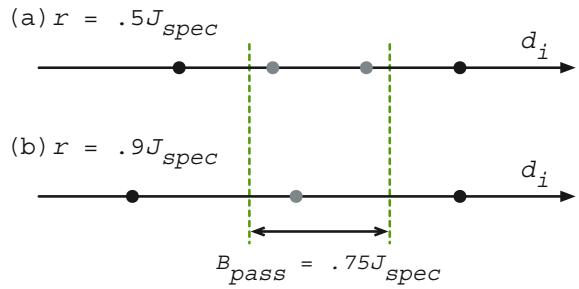

Fig. 9. Effect of tap resolution on pass test.

the jitter CDF is plotted, each dot on the CDF curve represents a CDF sample, and the pass band B_{pass} is defined as

$$B_{pass} = 2 \cdot J_{RMS} \cdot x_{pass} \tag{10}$$

where $x_{pass} = F_X^{-1}(1 - p_{pass})$. To determine if J_{RMS} is greater than J_{spec}, the two CDF samples (d_{up}, p_{up}) and (d_{down}, p_{down}) that enclose the one probability range $[p_{pass}, 1 - p_{pass}]$ are utilized to compute the *upper bound* of the RMS jitter. Since p_{up} and p_{down} are outside $[p_{pass}, 1 - p_{pass}]$, one has

$$d_{up} - d_{down} > 2 \cdot J_{RMS} \cdot x_{pass}$$

An upper bound of J_{RMS} can then be computed.

$$J_{RMS} < \frac{d_{up} - d_{down}}{2 \cdot x_{pass}} \tag{11}$$

Clearly, a pass decision can be made if the derived upper bound is less than the jitter specification J_{spec}, which corresponds to the following inequality:

$$d_{up} - d_{down} \leq 2 \cdot x_{pass} \cdot J_{spec} \tag{12}$$

Since J_{spec} and x_{pass} are pre-defined parameters, the right hand side of Eq. (12) can be computed in advance.

The pass test algorithm is shown in Algorithm 2. First, the candidate CDF samples are identified and stored in C_{down} (if less than p_{pass}) or C_{up}(if greater than $1 - p_{pass}$) (lines 1–10). If both C_{down} and C_{up} are non-empty, the

```
1  C_up ← φ;
2  C_down ← φ;
3  foreach tap t_i do
4  |   if p_i < p_pass then
5  |   |   C_down ← C_down ∪ {t_i};
6  |   end
7  |   else if p_i > (1 − p_pass) then
8  |   |   C_up ← C_up ∪ {t_i};
9  |   end
10 end
11 if size(C_up) > 0 and size(C_down) > 0 then
12 |   t_up ← min(C_up);
13 |   t_down ← max(C_down);
14 |   if d_up − d_down ≤ 2 · x_pass · J_spec then
15 |   |   return pass;
16 |   end
17 end
```

Algorithm 2: The pass test algorithm.

pass testis performed (line 11). In lines 12–13, the minimum delay tap t_{up} in C_{up} and the maximum delay tap t_{down} in C_{down} are selected. Then, Eq. (12) is used to determine if a pass decision can be made (lines 14–16).

Like jitter computation, there are occasions when the pass test cannot be applied or the pass decision cannot be made even though $J_{RMS} < J_{spec}$. In Fig. 9, it is assumed that $J_{RMS} = 0.5 \cdot J_{spec}$ and $x_{pass} = 0.75$. Thus, $B_{pass} = 0.75 \cdot J_{spec}$.

Case (a) In this case, $r = 0.5 \cdot J_{spec}$. The left and right-hand sides of Eq. (12) are $3r = 1.5 J_{spec}$ and $2 \cdot 0.75 \cdot J_{spec} = 1.5 \cdot J_{spec}$, respectively. Thus, Eq. 12 is true and a pass decision is made.

Case (b) Here we have $r = 0.9 \cdot J_{spec}$. The left and right-hand sides of Eq. (12) are $2r = 1.8 \cdot J_{spec}$and $1.5 \cdot J_{spec}$, respectively. Consequently, the inequality is not true and the pass decision is not made even though $J_{RMS} < J_{spec}$. Jitter computation is necessary.

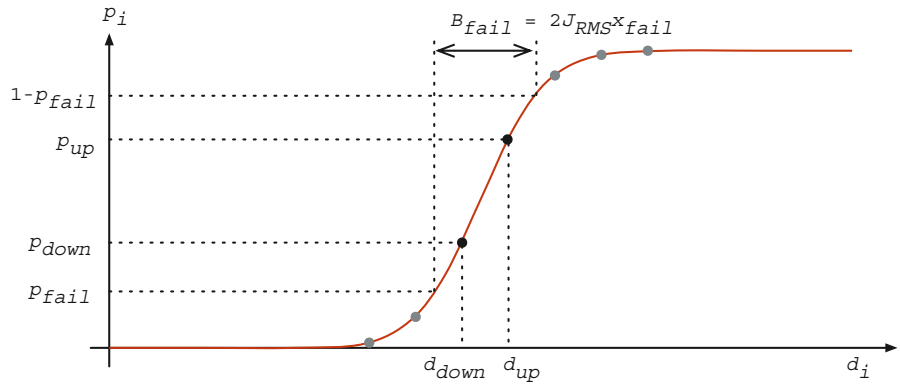

Fig. 10. Derivation of the RMS jitter lower bound.

Although by reducing r a tighter RMS jitter upper bound can be obtained to enhance the likelihood of making a pass decision, this approach will increase the number of delay taps and not desirable.

3.4. The Fail Test

The objective of the fail test is to reduce the post-processing computation complexity—if the RMS jitter is greater than the jitter specification, the fail test could make the fail decision without explicit jitter computation.

To determine if the fail test is applicable, the parameter p_{fail} is introduced. As shown in Fig. 10, the largest and smallest CDF samples, (d_{up}, p_{up}) and (d_{down}, p_{down}), that fall inside the one probability range $[p_{fail}, 1 - p_{fail}]$ are employed to compute the *lower bound* of the RMS jitter. Similar to the pass test analysis, an RMS jitter lower bound is

$$J_{RMS} > \frac{d_{up} - d_{down}}{2 \cdot x_{fail}} \tag{13}$$

where $x_{fail} = F_X^{-1}(1 - p_{fail})$. Therefore, the fail test can make the fail decision simply by examining whether the following inequality is true or not.

$$d_{up} - d_{down} > 2 \cdot x_{fail} \cdot J_{spec} \tag{14}$$

Similar to the pass test, no inverse Gaussian CDF is involved. Thus, the fail test can reduce the computation complexity. Depending on the actual RMS jitter and the delay line resolution, there are cases when the fail decision cannot be made even though J_{RMS} is greater than J_{spec}. For these cases, the fail decision will be made after explicitly computing the RMS jitter.

4. Implementation

The key parameters of the proposed technique include (1) the three CDF sample filtering parameters (p_{cmpt}, p_{pass}, and p_{fail}), (2) the delay line resolution r, and (3) the number of delay line taps. Since these parameters are closely related, the parameters selection process consists of two steps. In step one, we employ SA to determine the filtering parameters and delay line resolution, assuming that there are infinitely many delay taps. Then, in the second step, the number of required delay taps is decided.

4.1. The Delay Line Variation Model

One major concern of the proposed method is the delay line variations. In a regular analog design, the delay line variation could be up to 50% of its nominal value. Although one may apply special analog design techniques to reduce the variations, 30% deviation is still expected.

Since the Gaussian distribution is symmetric about its peak value, it is natural to have the delay values center around d_0 (Section 2.3) where the one probability is 0.5. Thus, the selected variable delay line values are in the form of

$$d_0 \pm (i - 0.5) \cdot r \tag{15}$$

where i is a positive integer.

In this paper, we consider the global process variations and assume that all the delay values are scaled by the same factor. The underlying assumption is that the delay elements are matched. The scaling effect not only affects the delay line resolution, but also introduces an offset—the delay values will no longer center around d_0.

4.2. The SA Search Algorithm

Determination of the test parameters becomes rather complicated after including the delay line variation model. Therefore, an SA-based search algorithm (Algorithm 3) is employed to determine all the parameters except the tap count.

The initial setup of the search algorithm is listed in lines 1–4, where *maxStep* is the maximum number of search steps, *initTemperature* is initial system temperature, *cost* is the lowest cost that is achieved up to now, and *config* denotes the current SA configuration. The initial configuration $(1, 0.1587, 0.1587, 0.1587)$ represents $r = J_{spec}$ and $p_{cmpt} = p_{pass} = p_{fail} = 0.1587$.

During each SA step, a new configuration *newConfig* is generated from *config* (line 9). The update functions for the parameters are in the following form:

$$v_{new} = v_{old} + \delta \cdot (rand() - 0.5) \tag{16}$$

where v is any of the four configuration parameters, δ is the ratio of *temperature* to *initTemperature*, and *rand*() is

```
1  maxStep ← 1000;
2  initTemperature ← 100;
3  cost ← ∞;
4  config ← (1, 0.1587, 0.1587, 0.1587);
5  temperature ← initTemperature;
6  step ← 0;
7  while step < maxStep do
8      step ← step + 1;
9      newConfig ← update (config);
10     newCost ← costFunction (newConfig);
11     if newCost < cost or
       exp(−β · (newCost−cost)/temperature) > rand () then
12         config ← newConfig;
13         cost ← newCost;
14     end
15     temperature ← α · temperature;
16 end
```

Algorithm 3: The SA algorithm.

Table 1. MC setup.

Parameter	Value
Delay line resolution	Uniform distribution within $[0.7r, 1.3r]$
RMS jitter, J_{RMS}	Uniform distribution within $[0.1 \times J_{spec}, 3 \times J_{spec}]$
# Phase comparisons per tap, N	16,384
Allowed test error	5% of J_{spec}

a random number generator. For each new configuration, a total of 1,024 Monte Carlo (MC) simulations are performed to estimate the success rate. The Monte Carlo simulation setup is listed in Table 1. The *actual* delay line resolution is a uniform distribution random variable within $[0.7r, 1.3r]$ to simulate up to 30% delay value deviation. The RMS jitter value is also uniformly distributed within $[0.1J_{spec}, 3J_{spec}]$ to fully exercise the three pass/fail decision phases. For each delay value d_i, $N = 16,384$ phase comparisons are made to obtain the corresponding one probability p_i. Finally, the allowed test error is 5% of J_{spec}. Once p_i's are obtained, the test flow in Fig. 5 is applied and the success rate of making correct pass/fail decisions is obtained after the 1,024 MC simulation runs. The cost function in the SA algorithm is

$$cost = -100 \cdot successRate - 0.5 \cdot r \tag{17}$$

which guides the SA search towards higher success rate and larger delay resolution at the same time. There are two cases when a new configuration is accepted (line 11): (1) the cost of the new configuration is less than the current minimal cost, and (2) the cost is not reduced but the following inequality is true.

$$exp\left(-\beta \cdot \frac{newCost - cost}{temperature}\right) > rand() \tag{18}$$

where $\beta = 8$. The purpose of accepting configurations with higher costs is to prevent the SA search from being trapped in local minimum. The acceptance probability lowers as the system cools down, i.e., *temperature* decreases. After each SA iteration, the system temperature is reduced by the factor $\alpha = 0.98$.

The SA search algorithm identifies a set of parameters that achieves 100 success rate. Listed in Table 2, the delay line resolution is about half the jitter specification, and the other three parameters, p_{cmpt}, p_{pass}, and p_{fail}, are 0.00034, 0.02202, and 0.10276, respectively. In terms of CPU time, the SA search process takes about 2.2 h on a 3 GHz P4 processor. The long simulation time should be acceptable since this process only has to be performed once.

Table 2. The SA search results.

Parameter	Value
r	$0.5065 \cdot J_{spec}$
p_{comp}	0.00034 ($\simeq 3.4\sigma$)
p_{pass}	0.02202 ($\simeq 2\sigma$)
p_{fail}	0.10276 ($\simeq 1.26\sigma$)

Table 3. Success rate vs N and tap count.

	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
4	64.2	77.8	88.6	93.8	95.8	96.9
6	79.8	89.2	94.4	97.1	98.2	99.0
8	89.3	93.5	96.8	98.5	99.2	99.6
10	92.6	94.0	97.9	99.0	99.8	99.9

4.3. *Determining the Delay Line Tap Count*

Recall that the SA algorithm assumes that the delay line has an infinite number of taps. The required tap count is derived through another set of simulation. In the sixth column of Table 3, the achieved success rates with respect to different tap counts (from four to ten) are listed. It shows that an eight-tap delay line will achieve better than 99% success rate.

5. Simulation Results

To validate the parameters obtained in Table 2, another 4,096 MC simulation runs (using the setup in Table 1) are performed and the achieved success rate is 99.2%.

In the following, more detailed analyses on the RMS jitter testing technique are shown.

5.1. *Hardware Overhead/Test Time Tradeoff*

In Table 3, the success rate with respect to different combinations of N's and tap counts are listed. In Table 3, the first column is the delay line tap count, and the first row is the number of phase comparisons, N, per delay tap.

Table 4. Success rate vs N and post-processing order.

	pass → *fail* → *cmpt*	*pass* → *fail*	*cmpt* → *pass* → *fail*	*cmpt*
2^{10}	0.9441	0.6580	0.5613	0.5554
2^{11}	0.9502	0.6602	0.6054	0.6008
2^{12}	0.9748	0.6667	0.8049	0.8015
2^{13}	0.9844	0.6753	0.9048	0.9021
2^{14}	0.9946	0.6626	0.9734	0.9709
2^{15}	0.9941	0.6638	0.9934	0.9895
2^{16}	0.9954	0.6697	0.9959	0.9932

The trend that success rates improve as N's and/or tap counts grow is as expected—larger N reduces the probability estimation error (Sec. 3.2), and more tap counts improves process variation tolerance. To achieve the same success rate, say 99%, one may opt for the $N = 2^{15}$ and six-tap combination to reduce hardware overhead or the $N = 2^{13}$ and ten-tap combination to reduce test time.

5.2. Success Rate Breakdown

To show the effectiveness of the pass and fail tests, the success rates with respect to different combinations and orders of the three post-processing steps are listed in Table 4, where the first column is the number of phase comparisons per delay tap and the first row indicates the order we apply the three steps.

pass → *fail* → *cmpt*: This is the proposed test flow which is intended to minimize the number of inverse CDF lookup operations.
pass → *fail*: The same as the proposed test flow except that the jitter computation step is skipped.
cmpt → *pass* → *fail*: Jitter computation is performed prior to the pass and fail tests.
cmpt: Only jitter computation is performed.
From Table 4, it is observed that

- The pass and fail tests are less sensitive to probability estimation errors—the success rate only changes slightly as N increases from 2^{10} to 2^{16} (column three). On the other hand, jitter computation is much more sensitive (column five).
- From the results of column three, the *pass* → *fail* → *cmpt* flow reduces the number of jitter computation by about two thirds.

Further success rate analyses on the test results for the 4,096 cases $(N = 2^{14})$ are shown in Table 5. In column two, the number of cases when each individual post-processing step is applicable is shown. In columns three and four, the minimum and maximum RMS jitter values of the applicable cases are listed. While jitter computation is applicable for almost all cases and fail test for almost all cases that should be failed, the pass test is applicable for only about 30% of the cases which satisfy the jitter specification.

Table 5. Success rate analyses for 4,096 MC simulations $(N = 2^{14})$.

	# Applicable cases	Min	Max	# Incorrect cases	Min	Max
Jitter computation	4,062	0.100	2.999	18	0.1492	0.8449
Pass test	600	0.100	0.906	0	–	–
Fail test	2,111	1.045	2.999	0	–	–

Table 6. Comparison with [3, 7].

	Proposed	[7]	[3]
Delay resolution	Coarse $(0.5 \cdot J_{spec})$	Fine[a]	Coarse $(2 \cdot J_{spec})$
Number of delay taps	8	30[b]	2
Process variation tolerance	At least 30% delay line variation	Not available	Not available
Input jitter range[c]	0.1–3 J_{spec}	Not available	0.75–1.75 J_{spec} [d]

[a]: a fraction of the actual jitter value.
[b]: an approximate value from the report.
[c]: 5% test error.
[d]: an upper bound, estimated from reported results.

Column five lists the number of misclassified cases, and columns six and seven are the minimum and maximum RMS jitters of the misclassified cases. It is interesting that the pass and fail tests are more robust than the jitter computation step. The misclassification probability for jitter computation is about 4%. It seems that the SA algorithm makes the pass/fail tests more robust because they are applied prior to jitter computation.

5.3. Discussion

In Table 6, comparisons are made between the proposed technique and [3, 7]. In terms of delay line resolution, both the proposed technique and [3] utilize coarse delay line and thus require less delay taps—the former lowers the delay line design efforts and the latter reduces the delay line calibration time. [7], on the other hand, requires that the delay line resolution be a fraction of the actual jitter value and thus needs more delay taps. For process variations, the proposed technique can tolerate at least 30% delay line variations. Also important is the input jitter range. Our technique is applicable for a rather wide input range, but [3] is rather sensitive.

Not listed in the table are is the incurred post-processing effort. For the proposed technique, when the pass or fail test is able to make a decision, only simple comparisons are needed; otherwise, inverse CDF computation via a lookup table is needed as in [3]. The method in [7], on the other hand, only requires a few mathematical operations. In practice, whether the post-processing should be performed on or off-chip depends on the available on-chip digital resources. For modern SoCs with on-chip processor and memory, the algorithm together with the inverse CDF table may be stored on-chip. For external ATE applications, the delay line calibration results as well as the CDF samples (all in digital formats) may be delivered to the ATE via a customized or the standard boundary scan

test interface. For a 3 GHz signal with $N = 2^{14}$, the required data rate is about 14×183 Kbps.

Finally, the inherent jitter of the DfT circuitry also affects the achievable test accuracy. In addition to more sophisticated design techniques, low-cost and robust jitter calibration techniques should be investigated.

6. Conclusion

A CDF-based RMS jitter testing algorithm for Gaussian or Gaussian-like jitter distribution is proposed in this paper. The features of the proposed algorithm include a low tap-count coarse delay line for CDF sampling and a sophisticated post-processing algorithm that achieves high process variation tolerance. Simulation results show that, in the presence of up to 30% delay line deviations, the success rate of making correct pass/fail decisions is 99%. Currently, we are investigating more efficient algorithms to determine the test algorithm parameters. Also, a prototype chip has been fabricated for silicon proof.

Acknowledgments

This work was partially supported by the National Science Council of Taiwan, R.O.C., under Grant No. NSC94-2220-E-002-006 and NSC94-2220-E-002-011.

References

1. A.H. Chan and G.W. Roberts, "A Synthesizable, Fast and High-Resolution Timing Measurement Device Using a Component-Invariant Vernier Delay Line," in *International Test Conference*, pp. 858–867, 2001.
2. P. Dudek, S. Szczepanski, and J.V. Hatfield, "A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line," *IEEE Transactions on Solid-State Circuits*, vol. 35, no. 2, pp. 240–247, February 2000.
3. J.J. Huang and J.L. Huang, "A Low-Cost Jitter Measurement Technique for BIST Applications," in *Asian Test Symposium*, pp. 336–339, 2003.
4. C.-K. Ong, D. Hong, K.-T. Cheng, and L.-C. Wang, "A Scalable On-Chip Jitter Extraction Technique," in *VLSI Test Symposium*, pp. 267–272, 2004.
5. O. Petre and H.G. Kerkhoff, "On-Chip Tap-Delay Measurements for a Digital Delay-Line Used in High-Speed Inter-Chip Data Communications," in *Asian Test Symposium*, pp. 122–127, 2002.
6. *Serial ATA: High Speed Serialized AT Attachment, Revision 1.0*, Serial ATA International Organization (SATA-IO), August 2001.
7. S. Sunter and A. Roy, "BIST for Phase-Locked Loops in Digital Applications," in *International Test Conference*, pp. 532–540, 1999.
8. S. Sunter, A. Roy, and J.-F. Côté, "An Automated, Complete, Structural Test Solution for SERDES," in *International Test Conference*, pp. 95–104, 2004.
9. S. Tabatabaei and A. Ivanov, "Embedded Timing Analysis: A SoC Infrastructure," *IEEE Design & Test of Computers*, vol. 19, no. 3, pp. 22–34, May–June 2002.
10. C.C. Tsai and C.L. Lee, "An On-Chip Jitter Measurement Circuit for the PLL," in *Asian Test Symposium*, pp. 332–335, 2003.
11. "VCOBIST," Credence Systems Corporation.
12. T.J. Yamaguchi, M. Ishida, *et al*., "A Real-Time Jitter Measurement Board for High-Performance Computer and Communication Systems," in *International Test Conference*, pp. 77–84, 2004.

Jiun-Lang Huang received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, Republic of China, in 1992, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California at Santa Barbara (UCSB) in 1995 and 1999, respectively.

From 2000 to 2001, he was an Assistant Research Engineer in the Department of Electrical and Computer Engineering, UCSB. In 2001, he joined the National Taiwan University, where he is currently an Assistant Professor in the Graduate Institute of Electronics Engineering and Department of Electrical Engineering. His main research interests include design-for-test and built-in-self-test for mixed-signal systems and VLSI system verification.