Exploiting Pipeline ADC Properties for a Reduced-Code Linearity Test Technique

Asma Laraba, Haralampos-G. Stratigopoulos, Salvador Mir, Hervé Naudet

Exploiting Pipeline ADC Properties for a Reduced-Code Linearity Test Technique

Asma Laraba, Haralampos-G. Stratigopoulos, *Member, IEEE*, Salvador Mir, *Member, IEEE*, and Hervé Naudet

***Abstract*—Testing the static performances of high-resolution Analog-to-Digital Converters (ADCs) consumes long test times that are disproportionally high with respect to the test time devoted to other types of circuits embedded in a modern System-on-Chip (SoC). In this paper, we review the state-of-the-art of reduced-code linearity test methods for pipeline ADCs and we propose a new approach that increases the efficiency and accuracy of the method. We show that by exploiting some inherent properties in the architecture of pipeline ADCs we can achieve significant static test time reduction while maintaining the accuracy of the standard histogram test. The proposed method is demonstrated on a 55nm 11-bit 2.5-bits/stage pipeline ADC.**

***Index Terms*—Design-for-test, analog-to-digital converter testing, static testing, linearity testing, pipeline analog-to-digital converters, reduced code linearity testing, histogram testing.**

I. Introduction

Differential Non Linearity (DNL) and Integral Non Linearity (INL) are the two main static performances that are measured during production testing of Analog-to-Digital Converters (ADCs). In the standard testing scheme, a saturated sine-wave or ramp is applied to the input of the ADC and the number of occurrences of each code at the output is obtained to construct the histogram, from which DNL and INL can be readily calculated.

This standard static test approach requires the collection of a large volume of data since each code needs to be traversed many times to average noise. The volume of data increases exponentially with the resolution of the ADC to a degree where the static test time becomes prohibitively large for high-resolution ADCs. Nowadays, the static test of high resolution ADCs is addressed with the same standard approach used for low-resolution ADCs. As a result, the static test time is disproportionally high as compared to the silicon area that the ADCs occupy on the die of a System-on-Chip (SoC) and as compared with the test time of other types of mixed-signal circuits in the SoC. According to published data from industry [1], although mixed-signal circuits occupy an area less than 5% in a modern SoC, testing the mixed-signal functions takes up to 30% of the total test time. Given that ADCs are among the most commonly met mixed-signal circuits in SoCs and since high static test times translate to high test costs, reducing the static test time for ADCs is an area of industry focus and innovation.

Many alternative test techniques aiming at reducing the static test time for ADCs have been reported in the literature. Techniques which permit measuring a pre-defined code width with on-chip circuitry are proposed in [2]–[4]. Integrating a ramp generator for on-chip histogram test has been investigated in [5]–[11]. This is a promising approach since it can eliminate the need to transfer a large volume of data off-chip to the Automatic Test Equipment (ATE) and it can also alleviate the problems related to noise and unstable electrical contacts. However, ADC linearity test standards dictate that the test stimulus must be at least two bits more linear than the ADC itself [12], [13], thus a high-resolution ramp generator needs to be designed which is extremely challenging. There are also issues regarding the robustness of the built-in test circuitry and memory storage. Instead, in [14], the entire ADC input range is exercised by small-amplitude triangular waves that are superimposed to a progressively increased DC level. In [15], [16], methods are introduced based upon first identifying and computationally removing the source non-linearity and then accurately estimating the ADC static performances. In [17], [18], an exponential waveform is employed in the analysis. These last three approaches relax the requirements on the input stimulus linearity, however, they do not reduce the test time. In [19], a technique is proposed for deriving the INL of the ADC from the outcome of the Fast Fourier Transform (FFT) used in standard spectral testing. However, deducing the static performances from a spectral test can only be applied in some cases and cannot be generalized. Model-based ADC linearity test techniques are proposed in [20], [21].

In this paper, we propose an efficient reduced-code linearity test technique for pipeline ADCs. In general, reduced-code testing can be applied to ADCs that, by virtue of their operation, have groups of output codes which have the same width. Examples of such ADCs include pipeline, Successive Approximation Register (SAR), logarithmic, sub-ranging, cyclic, etc. Thus, instead of considering all the codes in the testing procedure, we can consider measuring only one code out of each group to extract the complete transfer characteristic of the ADC, thus reducing significantly the static test time. In other words, we exploit the inherent properties of the ADC architecture to reduce the static test time. Performing a successful reduced-code testing scheme requires fully un-

A. Laraba was with Université Grenoble Alpes, CNRS, TIMA, Grenoble, France. She is now with Xilinx, Dublin, Ireland (e-mail: asma.laraba@xilinx.com).

H.-G. Stratigopoulos was with Université Grenoble Alpes, CNRS, TIMA, Grenoble, France. He is now with Sorbonne Universités, UPMC Univ. Paris 06, CNRS, LIP6, France (e-mail: haralampos.stratigopoulos@lip6.fr).

S. Mir is with Université Grenoble Alpes, CNRS, TIMA, Grenoble, France (e-mail: salvador.mir@imag.fr).

H. Naudet is with STMicroelectronics, Grenoble, France (e-mail: herve.naudet@st.com).

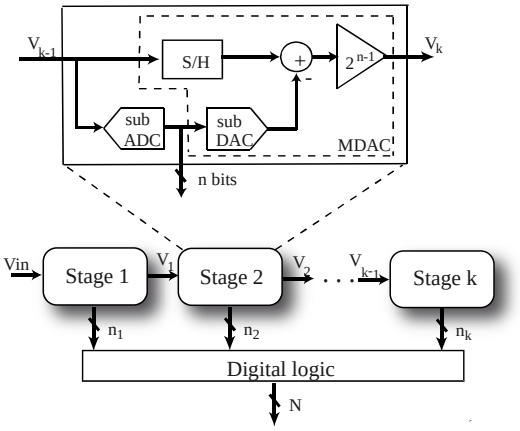


Fig. 1. Architecture of a pipeline ADC.

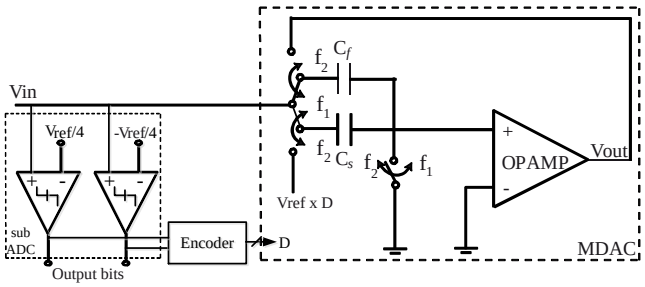


Fig. 2. MDAC implementation of an 1.5-bit stage.

derstanding the aspects of the ADC architecture. Published work has so far consider SAR ADCs [22] and pipeline ADCs [23]–[26]. SAR ADCs typically have lower conversion rate and higher resolution as compared to pipeline ADCs. Reduced code testing is useful for both architectures since static test time is inversely proportional to the conversion rate and proportional to the resolution.

The rest of the paper is organized as follows. In Section II, we present the reduced-code testing principle and the state-of-the-art focusing on pipeline ADCs. In Section III, we discuss how the accuracy of the estimated static performances by reduced-code testing is affected due to the presence of noise. In Section IV, we explain the concepts and definitions that set the grounds for developing a technique to cancel out the noise. In Section V, we explain in detail the proposed method to cancel out the noise and apply reduced-code testing with confidence. In Section VI, we show experimental results obtained on a 55nm 11 bit 2.5-bits/stage pipeline ADC designed by STMicroelectronics. Finally, in Section VII, we conclude the paper.

II. Reduced-Code Testing Principle and State-Of-The-Art for Pipeline ADCs

A. Overview of pipeline ADCs

A pipeline ADC consists of a cascade of stages, as shown in Fig. 1 [27]. Each stage consists of a sample-and-hold (S/H)

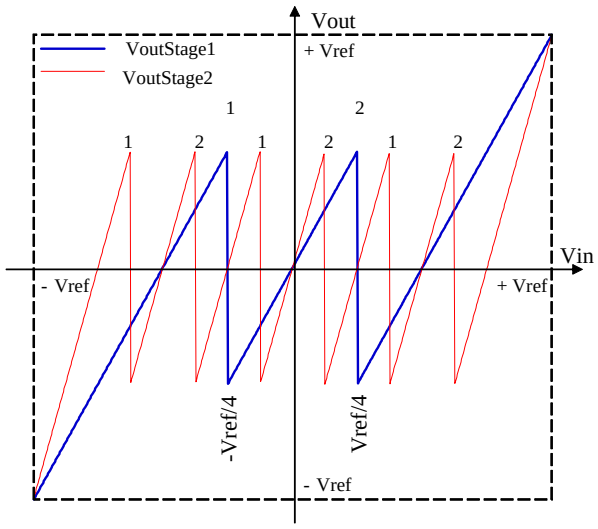


Fig. 3. Residue of the first and second stages of a 1.5-bit/stage pipeline ADC.

circuit, a sub-ADC typically implemented with a flash ADC, a sub-DAC, a subtractor, and an op-amp. The input signal to each stage is first converted by the sub-ADC to a digital code which is the output of the stage. The result of the conversion is reconverted by the sub-DAC to an analog signal and subsequently subtracted from the input signal. The result of the subtraction (e.g. the residue) is amplified so as to use the same reference voltage in all stages. The residue of the first stage is sampled by the second stage and so forth. The S/H, sub-DAC, substractor, and op-amp are typically implemented as a switched-capacitor circuit, which is commonly referred to as Multiplying Digital-to-Analog Converter (MDAC). As an example, Fig. 2 shows the implementation of an 1.5-bit stage. The digital logic assembles the digital codes of the cascaded stages, performs the digital correction to reduce the accuracy requirement of the flash ADCs (in particular, to moderate the effect of comparators' offset), and provides the digital output of the ADC.

B. Underlying principle

Let us consider Fig. 3 which plots together the residues of the first and the second stages of a 1.5-bit/stage pipeline ADC. The number placed above the peak of a transition indicates which of the two comparators of the sub-ADC is being exercised (e.g. its threshold is crossed) at this transition. Notice that each time a comparator is exercised, there is a transition in the digital output of the stage. As it can be seen, if we traverse the input dynamic range of the ADC, the two comparators of the first stage are exercised once. The first, second, and third segment of the first stage residue traverse the output ranges $[-V_{ref}, V_{ref}/2]$, $[-V_{ref}/2, V_{ref}/2]$, and $[-V_{ref}/2, V_{ref}]$, respectively. Therefore, for each segment of the first stage residue, each of the two comparators of the second stage is exercised once, that is, if we traverse the input dynamic range of the ADC, then the two comparators of the

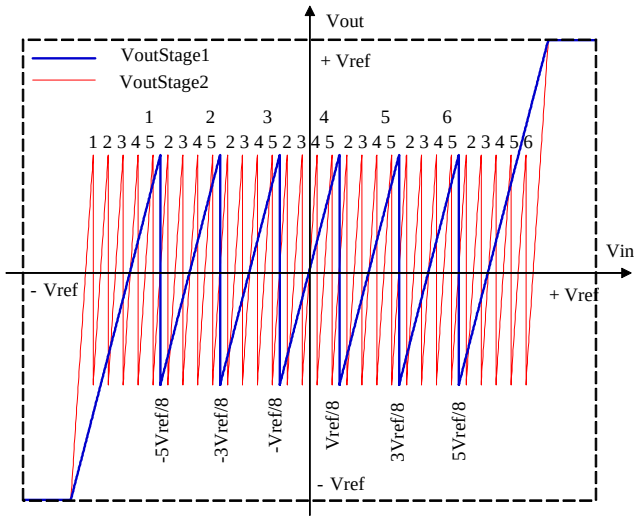


Fig. 4. Residue of the first and second stages of a 2.5-bit/stage pipeline ADC.

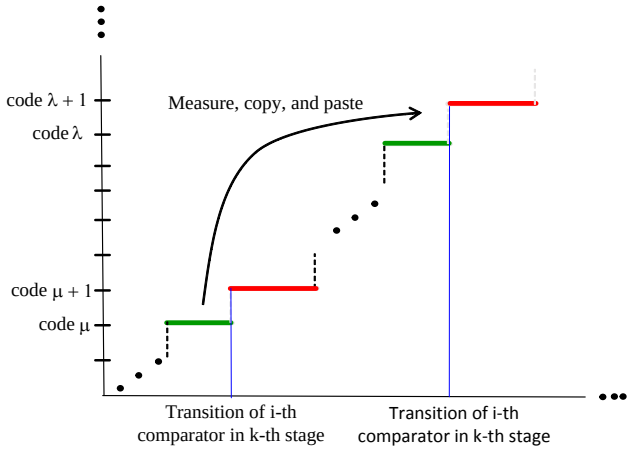


Fig. 5. Principle of reduced code testing of pipeline ADCs.

second stage are exercised three times each in total. Following a similar argument, if we traverse the input dynamic range of the ADC, then the two comparators of the third stage are exercised seven times each.

In the case of a 2.5-bit stage, the sub-ADC consists of six comparators and provides a 3-bit output. Fig. 4 plots together the residues of the first two stages of a 2.5-bit/stage pipeline ADC. As it can be seen, the six comparators in the first stage are exercised once if we traverse the input dynamic range of the ADC. In contrast, in the second stage, the first and sixth comparators are exercised just once while the rest of the comparators are exercised seven times each.

The bottom line of the above discussion is that a comparator in a second or later pipeline stage is exercised several times. This implies that in the ADC output there will be transitions that are due to the same comparator. As an illustration, in Fig. 5 we show two transitions in the ADC output that are due to the i-th comparator in the k-th stage. An ADC output code shares two adjacent transitions that involve two different comparators. The DNL error, that is, the variation of the width of the code from the ideal one Least Significant Bit (LSB) width, is mainly due to the presence of different error sources (i.e. finite op-amp gain, capacitor mismatch, op-amp offset, sub-ADC comparators offset, etc.) in the first stages of the pipeline since a stage in the pipeline dominates all subsequent stages in terms of the error produced in the transfer characteristic. Notice also that any error source in a stage that produces a DNL error of an ADC output code will be seen at the ADC output code transition that has been caused by the comparators of that stage. In the example of Fig. 5, let us assume that the i-th comparator in the k-th stage dominates the comparators with which it shares codes μ, $\mu + 1$, λ, and $\lambda + 1$. This means that the width of these codes are principally affected by the errors in the k-th stage where the i-th comparator belongs to. Furthermore, it means that the widths of the codes λ and $\lambda + 1$ are practically equal to the widths of the codes μ and $\mu + 1$, respectively. Thus, we need to measure the width of either λ or μ and the width of either $\lambda + 1$ or $\mu + 1$. Extending this argument, let us assume that we know the mapping between the transitions in the ADC output and the comparators in the pipeline stage that are being exercised to produce these transitions. If we measure only the codes around a representative set of ADC output transitions such that this set covers all comparators in all stages and each comparator is represented once in this set, then, by relying on the mapping, we can readily assign values to the widths of unmeasured codes around the unselected ADC output transitions. In other words, we measure a reduced number of codes in the histogram and we fill in the rest of the histogram automatically by relying on the information in the extracted mapping.

C. Static test time reduction

Relying on a reduced number of code width measurements to extract the complete transfer characteristic of the ADC translates in static test time reduction. Specifically, the test time is mainly governed by the transfer time of data from the ADC under test to the memory of the ATE and from the ATE to the workstation where the data will be processed for constructing the histogram. The pure electrical test time is rather negligible compared to the data transfer time. Thus, if we measure X% of the codes, which compared to the standard histogram technique translates into transferring only X% of the data, then we drastically reduce the test time. Overall, we expect to have a test time reduction slightly below (100-X)% given that compared with the standard histogram technique we have the extra step of deriving the mapping.

D. Large linearity errors in the first stages

In practice, to obtain good accuracy when large linearity errors are present, it may be necessary to consider measuring more than two codes around the transitions that involve comparators which belong to the stages that are closer to the front of the pipeline. To explain how we assign values to the

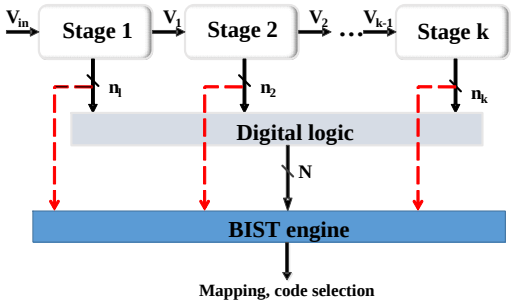


Fig. 6. Monitoring of the digital outputs of the internal stages of the pipeline aiming at a correct mapping between ADC output transitions and comparators.

widths of the unmeasured codes in this case, let us consider that we measure the widths of m codes on the right and m codes on the left of an ADC output transition that is due to the i-th comparator in the k-th stage being exercised. Then, if an ADC output transition is due to a comparator in the j-th stage being exercised, where $j > k$, and this ADC output transition has a distance of less than m steps from an ADC output transition that is due to the i-th comparator in the k-th stage being exercised, then the widths of the codes around this ADC output transition will be overwritten by the corresponding widths in the set of m code widths measured around the ADC output transition that is due to the i-th comparator in the k-th stage being exercised. The number of the codes to be considered around each ADC output transition depends also on the amount of errors that are present in the ADC which is reflected on the minimum and maximum DNL and INL. The larger the DNL and INL errors are and the more towards the front of the pipeline the stage is, the larger this number is recommended to be.

E. Importance of deriving an accurate mapping

In order to make the reduced-code testing technique successful, we need to meet two objectives. First, we need to ensure that an ADC output transition is mapped to the correct comparator. This holds for all ADC output transitions, i.e. those that are selected and those that are not selected and their surrounding codes will be inferred later. Second, for a comparator in a given stage we should avoid selecting an output transition that involves in addition to this comparator a comparator in one of the previous stages. The reason is that the error of the previous stage will overshadow the error of the target stage. If the above two objectives are not met, then the accuracy of the technique degrades, resulting in an erroneous characterization of the static performances of the ADC.

The possibility to apply reduced-code testing on pipeline ADCs was first introduced in [23] where the mapping between the ADC output transitions and the comparators that produce these transitions is performed by considering the nominal succession of ADC output transitions. In [24], it is explained with several detailed examples that this mapping principle can lead to erroneous estimation of static performances when the error sources are more than just the comparator offset.

To meet the aforementioned two objectives, and to avoid pitfalls such as those demonstrated in [24], it is proposed to draw the mapping by monitoring the digital outputs of the internal stages of the pipeline before they undergo digital correction [24], [25], as shown in Fig. 6. The rationale is that when a comparator threshold is crossed it necessarily produces a transition in the digital output of the stage to which it belongs to. A transition in the digital output of a stage provides complete information about which comparator has been exercised. Furthermore, a transition in the digital output of a stage can be mapped to the resulting ADC output transition by simply processing the outputs of the different stages as is done by the digital logic block of the ADC.

F. Summary of reduced-code testing

In summary, the enhanced reduced-code testing technique consists of the following steps:

1) Find the mapping between the ADC output transitions and the comparators that are being exercised.
2) Select a set of ADC output transitions such that all comparators in all stages are represented once in this set.
3) Measure the widths of the codes around the selected ADC output transitions.
4) Infer the widths of the codes around the ADC output transitions that were not selected in step 2.
5) Calculate DNL and INL from the obtained code widths.

For an integrated solution, a digital Built-in Self-Test (BIST) engine could be designed to perform steps 1-2 on-chip. The information about the mapping and the selected codes to be measured is transferred from the ADC under test to the ATE where step 3 is performed. Data from steps 1-3 are subsequently transferred to the workstation where steps 4-5 are performed. If the on-chip resources permit it, steps 4-5 can also be performed on-chip. Integrating a test stimulus generator to perform step 3 will result in a full BIST implementation.

III. Noise Considerations when Applying Reduced-Code Testing

As explained in Section II, the most important step of a reduced code testing scheme is guaranteeing an accurate mapping between the transitions occurring at the output of the internal stages and the corresponding ADC transitions. However, due to the presence of noise in the transitions of internal stages it is unlikely that an accurate mapping will be obtained [26]. An example is shown with the measurement in Fig. 7 which superimposes the digital outputs in decimal of the internal stages of an 55nm 11-bit 2.5-bit/stage pipeline ADC that is also used as our case study in Section VI. This figure zooms in a small part of the whole dynamic range of the pipeline ADC. As it can be seen, some of the transitions are very noisy, which, in turn, will lead unavoidably to inaccuracies in the mapping. This is shown with an example in Fig. 7 where the intention is to calculate the ADC output code that corresponds to the transition from 2 to 3 of the fourth stage that is due to its third comparator being exercised. As it can be seen, the outputs of the first, second, third, and fourth stages to the right of this transition are 4, 2, 2, and 3, respectively. However, due to the presence of noise,

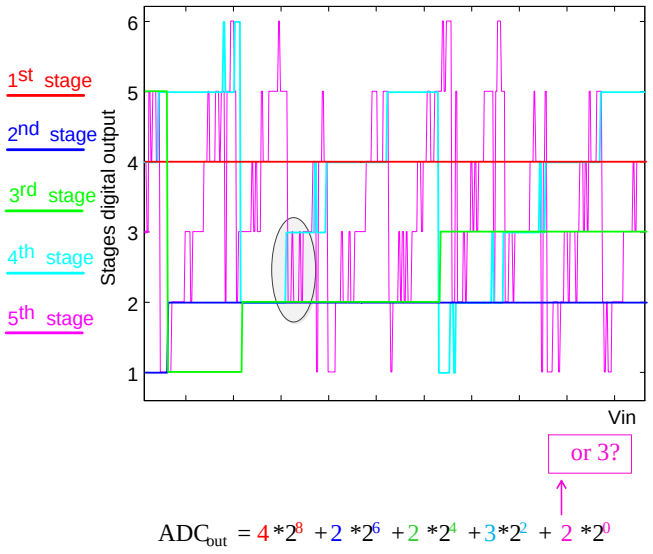


Fig. 7. The effect of noise on the transitions in the stage outputs.

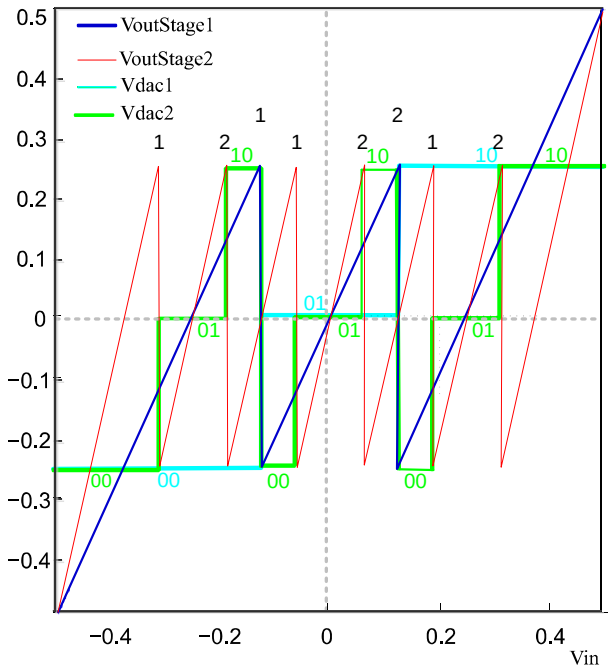


Fig. 8. Residues of the first two stages of a 1.5-bit/stage pipeline ADC plotted together with the output of their sub-DAC.

the output of the fifth stage to the right of this transition is toggling between 2 and 3. In other words, the mapping between the ADC output and internal stage transitions cannot be deterministically established due to noise. This will inadvertently result in an incorrect grouping of the ADC output codes that have equal widths and, thereby, the estimation of DNLs will be inaccurate. Furthermore, successively summing up erroneous values of DNLs may result in a significant error in INL estimation.

In Section IV, we will define the concepts of natural and forced transitions and root codes. These concepts will be used next to develop an enhanced reduced-code linearity testing technique that is immune to noise.

IV. Natural Transitions, Forced Transitions, and Root Codes

A. Natural and forced transitions

Since we will be monitoring the digital outputs of each of the pipeline stages we need to list and label all possible transitions. We classify them into two types: *natural* transitions and *forced* transitions. Looking at the residue of the second stage $VoutStage2$ of a 1.5-bit/stage pipeline ADC shown in Fig. 8, we can observe the six transitions corresponding to the two comparators in the sub-ADC of this stage. As in Fig. 3, we have indicated which of the two comparators is being exercised by placing the comparator's number at the peak of the corresponding transition. By looking at the sub-DAC output of the second stage $Vdac2$, we can also identify which of the two comparators is being exercised each time. We have indicated on the $Vdac2$ curve the corresponding digital output around each $Vdac2$ transition. The first comparator is exercised three times (e.g. transitions $00 \rightarrow 01$) and the second comparator is exercised also three times (e.g. transitions $01 \rightarrow 10$). We observe also that in addition to the transitions $00 \rightarrow 01$ and $01 \rightarrow 10$, there is another transition $10 \rightarrow 00$ occurring twice. This transition happens under the influence of transitions in the residue of the first stage $VoutStage1$. The residue of the first stage becomes suddenly lower than the threshold of the second comparator in the second stage and, thus, the digital output of the second stage transitions from 10 to 00. We refer to these transitions as forced transitions because the digital output of the stage transitions due to a sudden change at its input which is caused due to one of the comparators of the previous stages being exercised. Conversely, when the digital output of the stage transitions due to a smooth change at its input causing one of the comparators in this stage to be exercised, we refer to these transitions as natural transitions.

B. Root codes

For the discussion in this Section we use a behavioral model of a 10-bit ADC that comprises four 2.5-bit stages and a last 2-bit stage. Let us consider the third comparator in the second stage of this ADC.

Fig. 9 superimposes the output of the ADC (left y-axis in decimal) on the digital output of the first and second stages (right y-axis in decimal) as we traverse the whole input dynamic range. From this plot we can identify the output codes of the ADC that are associated with the transitions from 2 to 3 of the second stage that are due to the third comparator in this stage being exercised. On the left y-axis of Fig. 9 we show the ADC output codes on the right of these transitions. As it can be seen from Fig. 9, the first ADC output code 112 corresponds to the case where the output of the first stage is 0, the second ADC output code 240 corresponds to the case where the output of the first stage is 1, the third ADC output code 368 corresponds to the case where the output of the first

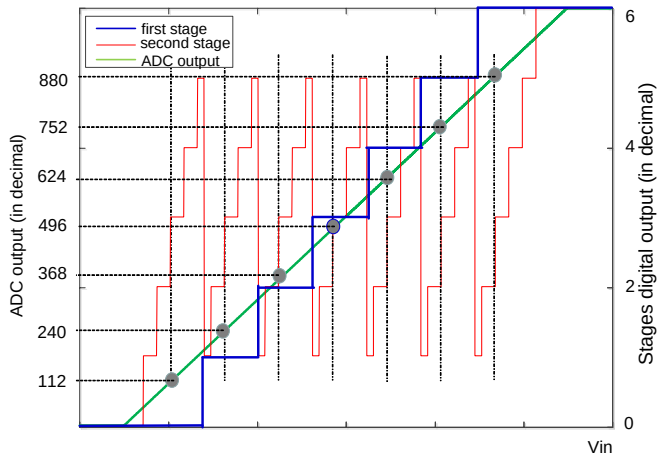


Fig. 9. Transitions in the first and second stages and corresponding ADC output codes.

stage is 2, and so forth. All these ADC output codes can be derived from code 112 by adding a term that is obtained by multiplying the value in decimal of the output of the first stage with the weight of the first stage. In particular, considering that for this specific ADC the weight of the first stage is equal to 2^7, we can write: $112 = 112 + 0 \cdot 2^7$; $240 = 112 + 1 \cdot 2^7$; $368 = 112 + 2 \cdot 2^7$; $496 = 112 + 3 \cdot 2^7$; $624 = 112 + 4 \cdot 2^7$; $752 = 112 + 5 \cdot 2^7$; $880 = 112 + 6 \cdot 2^7$. We refer to the output code 112 as the *right root code* of the third comparator in the second stage. Similarly, by looking at the ADC output codes on the left of the transitions from 2 to 3 of the second stage that are due to the third comparator in this stage being exercised, we can define the *left root code* of the third comparator in the second stage.

To generalize, let us divide the ADC stages into two groups. The first group contains stages 1 to $k-1$ and the second group contains stages k to N, where N is the total number of stages. Let us also define a function $f(x, comp_k^i, w)$ where:

- x refers to the rank of the stage in the pipeline,
- $comp_k^i$ refers to i-th comparator of the k-th stage,
- w refers to the right side (e.g. $w = R$) or to the left side (e.g. $w = L$) of the transition of the digital output of the stage that is due to the comparator $comp_k^i$ being exercised.

We define $f(x, comp_k^i, w)$ as follows: given that the i-th comparator of the k-th stage is exercised producing a transition in the digital output of the x-th stage, $f(x, comp_k^i, w)$ is the digital output of the x-th stage on the w side of this transition. For example, $f(1, comp_2^3, R)$ refers to the digital output of the first stage on the right of a transition that is due to the third comparator of the second stage being exercised.

By definition, every time the i-th comparator in the k-th stage is exercised, the digital output of the k-th stage transitions from a value equal to $f(k, comp_k^i, L) = i - 1$ to a value equal to $f(k, comp_k^i, R) = i$. Furthermore, every time the same comparator is exercised in a stage, the residue of this stage, which is the analog input to the following stages, always transitions between the same two values. This implies that every time the i-th comparator in the k-th stage is exercised, the digital output of the x-th stage, $x = k+1, \cdots, N$, is always equal to the value $f(x, comp_k^i, L)$ before the transition and equal to the value $f(x, comp_k^i, R)$ after the transition.

We define:

$$L_k^i = [f(k, comp_k^i, L), f(k+1, comp_k^i, L), \cdots, f(N, comp_k^i, L)] \quad (1)$$

$$R_k^i = [f(k, comp_k^i, R), f(k+1, comp_k^i, R), \cdots, f(N, comp_k^i, R)]. \quad (2)$$

If we sum up the elements of L_k^i respecting the weight of each stage, then we obtain the left root code of the i-th comparator in the k-th stage. If we sum up the elements of R_k^i respecting the weight of each stage, then we obtain the right root code of the i-th comparator in the k-th stage.

V. Obtaining the Mapping for Accurate Reduced-Code Testing

A. Cancelling out the effect of noise

In Section IV-B, we have shown that the ADC output codes corresponding to the same comparator can be expressed as a function of the root code of this comparator and the contributions of the stages preceding the stage where this comparator belongs to. This property can be used to cancel out the effect of noise when performing the mapping between the ADC output transitions and the comparators.

We first apply a ramp and we observe the type of the transitions at the digital output of each stage. For each natural transition that is due to the i-th comparator in the k-th stage being exercised, we obtain L_k^i and R_k^i from Eq. (1) and (2). If n is the number of natural transitions, then we have at hand n values of each element $f(x, comp_k^i, L)$ of L_k^i and n values of each element $f(x, comp_k^i, R)$ of R_k^i, $x = k, \cdots, N$. Due to the presence of noise, these n extracted values are not necessarily the same for $x = k+1, \cdots, N$. In other words, the left and right root codes calculated starting from different natural transitions may not be the same.

For $x = k+1, \cdots, N$, let $\mu^{x,L}_{comp_k^i}$ and $\mu^{x,R}_{comp_k^i}$ be the values of $f(x, comp_k^i, L)$ and $f(x, comp_k^i, R)$, respectively, that are most frequently met in the n values that we have at hand. In this way, we obtain the noise-free L_k^i and R_k^i as follows:

$$L_k^i = [i-1, \mu^{k+1,L}_{comp_k^i}, \cdots, \mu^{N,L}_{comp_k^i}] \quad (3)$$

$$R_k^i = [i, \mu^{k+1,R}_{comp_k^i}, \cdots, \mu^{N,R}_{comp_k^i}]. \quad (4)$$

From the noise free L_k^i and R_k^i we can calculate the noise-free left and right root codes of the i-th comparator in the k-th stage.

Fig. 10 shows an example with five ADC output codes (e.g. μ, λ, ϵ, γ, and δ) that are on the left of an ADC output transition that is due to a comparator in the second stage being exercised. The ADC output codes are expressed in terms of the weights of the stages multiplied by coefficients that correspond to the values in decimal of the digital outputs of

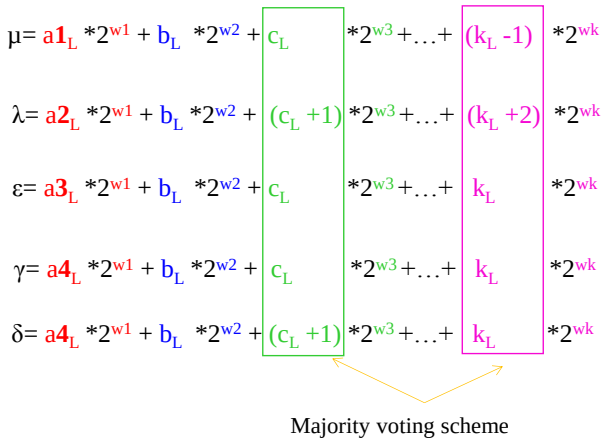


Fig. 10. Majority voting scheme to extract noise-free root codes.

the stages. The coefficients of the first stage (e.g. a_{1L}, a_{2L}, a_{3L}, a_{4L}, a_{5L}) will change depending on the position of the ADC output code in the dynamic range. The coefficient of the second stage is constant (e.g. b_L) for all ADC output codes. As for the coefficients of the stages 2 to k, they are not necessarily constant due to noise. To extract the noise free values of the coefficients, we simply apply a majority voting scheme. Thus, c_L is chosen for the third stage and k_L is chosen for the k-th stage since they are the most frequently met values.

It should be noticed that the number of natural transitions that are due to a specific comparator being exercised varies depending on the number of the stage the comparator belongs to. For example, for an 11-bit, 2.5 bit/stage ADC, for one sweep of the input ramp covering the whole input dynamic range, the number of natural transitions varies from 1 if the comparator is in the first stage to more than 500 if the comparator is in the fifth stage. Given that the noise becomes evident in the transitions of the last stages in the pipeline, as shown in Fig. 7, for these stages we will have a sufficiently large number of samples to apply majority voting with confidence and obtain the noise-free L_k^i and R_k^i and, subsequently, the noise-free left and right root codes.

B. Obtaining the mapping

As it will be explained in this Section, in order to obtain the mapping, we only need (a) the noise-free L_k^i and R_k^i for each of the comparators and (b) the value of the digital output of each stage at the beginning of the dynamic range, denoted by $Out_{k,start}$, and at the end of the dynamic range, denoted by $Out_{k,end}$, where k denotes the number of the stage, $k = 1, \cdots, N$.

$Out_{k,start}$ and $Out_{k,end}$ are straightforward to obtain. In decimal, $Out_{k,start} = 0$ for each stage and $Out_{k,start} = 2^B - 2$, where B is the number of output bits of the k-th stage, unless considerable offsets are present in the ADC. For example, $B = 2$ for an 1.5-bit stage, $B = 3$ for a 2.5-bit stage, etc. In the following, we will assume that $Out_{k,start} = 0$ and $Out_{k,end} = 2^B - 2$.

By looking at the digital output of any stage as we traverse the input dynamic range (for example, see Fig. 9), we observe that it starts at $Out_{k,start}$ and it ends at $Out_{k,end}$ with natural or forced transitions occurring in between. Fig. 11(a) shows the typical digital output of a first 2.5-bit stage. It starts at $Out_{1,start} = 0$, ramping up to $Out_{1,end} = 6$, with the natural transitions of its six comparators occurring in between. Notice that any transition in the digital output of a first stage is a natural transition.

In contrast, a transition in the digital output of a second or later stage could be either natural or forced. As shown in the example of Fig. 9, the digital output of a second 2.5-bit stage starts at $Out_{2,start} = 0$ and ends at $Out_{2,end} = 6$, with both natural and forced transitions occurring in between. The forced transitions are due to the natural transitions of the first stage and the natural transitions occur between every two successive forced transitions. The first forced transition of the second stage is due to the first natural transition of the first stage. The value of the digital output of the second stage to the left and to the right of its first forced transition are given by $f(2, comp_1^1, L)$ and $f(2, comp_1^1, R)$, respectively, as shown in Fig. 11(b). The second forced transition of the second stage due to the second natural transition of the first stage is a transition from $f(2, comp_1^2, L)$ to $f(2, comp_1^2, R)$, the third forced transition of the second stage due to the third natural transition of the first stage is a transition from $f(2, comp_1^3, L)$ to $f(2, comp_1^3, R)$, and so forth. The digital output values of the second stage between $Out_{2,start}$ and $f(2, comp_1^2, L)$, between $f(2, comp_1^i, R)$ and $f(2, comp_1^{i+1}, L)$, for $2 \leq i \leq 5$, and finally between $f(2, comp_1^6, R)$ and $Out_{2,end}$, will be obtained by simply incrementing $Out_{2,start}$ by 1 until $f(2, comp_1^2, L)$, incrementing $f(2, comp_1^i, R)$ by 1 until $f(2, comp_1^{i+1}, L)$, for $2 \leq i \leq 5$, and finally incrementing $f(2, comp_1^6, R)$ by 1 until $Out_{2,end}$. Between these values the comparators of the second stage are exercised one after the other, incrementing the digital output of the stage by 1.

Similarly, the digital output of the third stage can be reconstructed from the digital output of the second stage based solely on $Out_{3,start}$, $Out_{3,end}$, $f(3, comp_2^i, L)$, $f(3, comp_2^i, R)$, $f(3, comp_1^i, L)$, and $f(3, comp_1^i, R)$. $f(3, comp_2^i, L)$ and $f(3, comp_2^i, R)$ are used to account for the forced transitions of the third stage due to the natural transitions of the second stage while $f(3, comp_1^i, L)$ and $f(3, comp_1^i, R)$ are used to account for the forced transitions of the third stage due to the natural transitions of the first stage.

It turns out that we can reconstruct the digital output of any stage from the digital outputs of the preceding stages in the pipeline based on the elements of the noise-free L_k^i and R_k^i and $Out_{k,start}$ and $Out_{k,end}$. In this way, with a single sweep of the input dynamic range, we can identify the transitions in digital outputs of each stage and, then, juxtapose them with the ADC digital output to find the mapping. An algorithm based on nested *for loops* can be used for this purpose, starting from the first stage down to the last stage of the ADC, and recording the required mapping information in the course of the algorithm.

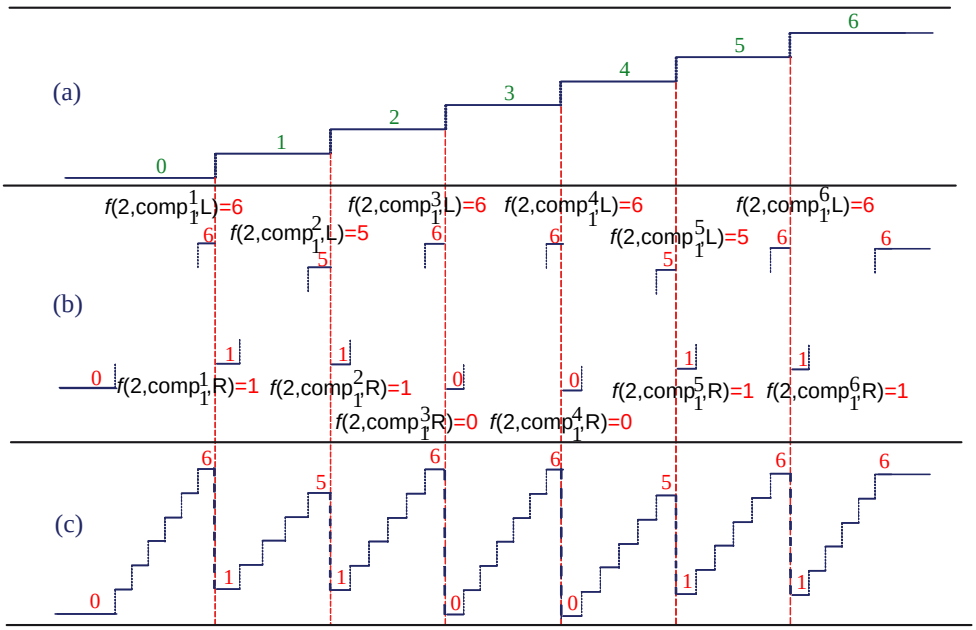


Fig. 11. Reconstructing the digital output of the second stage using $L_1^i(2)$ and $R_1^i(2)$.

VI. Experimental results

Our case study is a 55nm 11-bit pipeline ADC with digital correction provided by STMicrolectronics. The ADC is composed of four 2.5-bit stages and a last 3-bit stage. For this specific ADC, we had access to the digital output of the internal stages (in total 15 bits) before digital correction is applied. Thus, we extracted the digital outputs of the internal stages off-chip and we did the data processing in Matlab®. The reduced-code testing technique with and without noise cancellation is compared to the standard histogram technique.

The first step in our experiment consists of verifying the three main properties, stemming directly from the architecture of pipeline ADCs, on which the reduced-code testing technique relies upon. The *first* property states that the ADC codes related to the same comparator have equal DNLs. Let us consider Fig. 12 which superimposes the digital output of the complete ADC on the digital output of the second stage as we traverse the whole input dynamic range. As an example, the intersection points between the dashed vertical lines and the digital output of the complete ADC correspond to the codes that are mapped to the transitions that involve the fourth comparator in the second stage (e.g. transition from 3 to 4). The codes on the right of these transitions are shown on the left y-axis of Fig. 12. The first transition is mapped to code 282, the second transition is mapped to code 539, the third transition is mapped to code 795, etc. Next, the DNL of all the codes was calculated using the standard histogram technique, as shown in Fig. 13. By the circles we point to the DNL of the codes shown in Fig. 12. Values are, from left to right, 0.46, 0.44, 0.45, 0.48, 0.43, 0.44, 0.43. As can be observed, they are very close to each other which confirms the first principle of the technique. Thus, instead of measuring the DNL of all these seven codes, we can choose to measure the DNL of just one code and then use this DNL value for the rest of the codes.

The *second* property states that the largest DNL errors occur around the output transitions that involve the comparators in the stages that are towards the front of the pipeline. As an example, Fig. 14 shows the digital output of the first stage superimposed on the digital output of the complete ADC. The six codes shown on the left y-axis of Fig. 14 are the codes on the right of the ADC output transitions that correspond to the transitions of the first stage. In Fig. 13, these codes are indicated with squares. As it can be observed, these codes present the minimum and maximum DNL errors. This second principle tells us that for a given comparator we should avoid selecting a representative ADC output transition that involves in addition to this comparator a comparator in one of the previous stages. It also tells us that we should give priority to the first stages in the step where we extract the unmeasured code widths from the measured ones.

The *third* property states that the ADC output codes corresponding to the same comparator can be expressed as a function of the noise-free root codes of this comparator and the contributions of the stages preceding the stage where this comparator belongs to. Fig. 15 superimposes the digital output of the complete ADC on the digital output of the first and second stages as we traverse the whole input dynamic range. As an example, the intersection points between the dashed vertical lines and the digital output of the complete ADC correspond to the codes that are mapped to the transitions that involve the third comparator in the second stage (e.g. transition from 2 to 3). The codes on the right of these transitions are shown on the left y-axis of Fig. 15. The first transition is mapped to code 232, the second transition is mapped to code 488, the third transition is mapped to code 745, etc. Let us consider these codes and subtract from them the corresponding value in decimal of the first stage multiplied by the weight of the first stage, in order to find the right root code. The

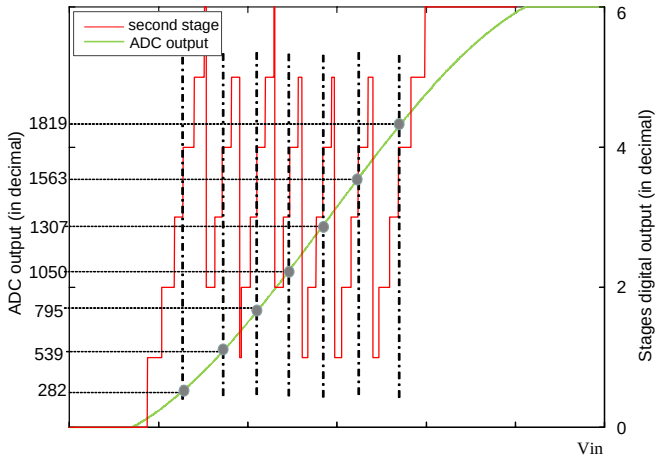


Fig. 12. Searching for the codes that are mapped to the fourth comparator in the second stage.

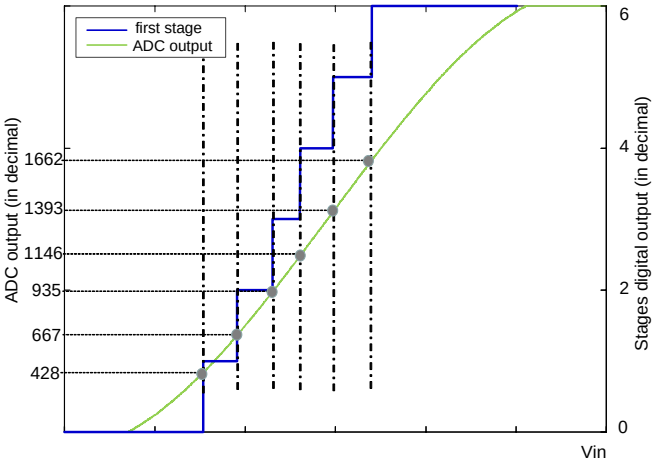


Fig. 14. Searching for the codes that are mapped to the comparators of the first stage.

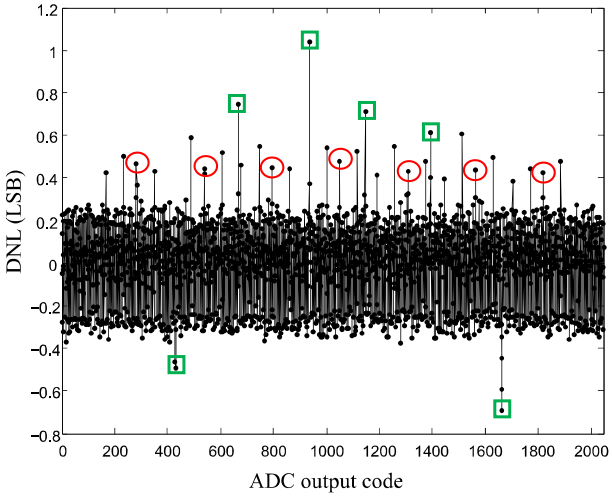


Fig. 13. DNL obtained with the standard histogram technique. The circles point to the codes that are mapped to the fourth comparator in the second stage, while the squares point to the codes that are mapped to the comparators of the first stage.

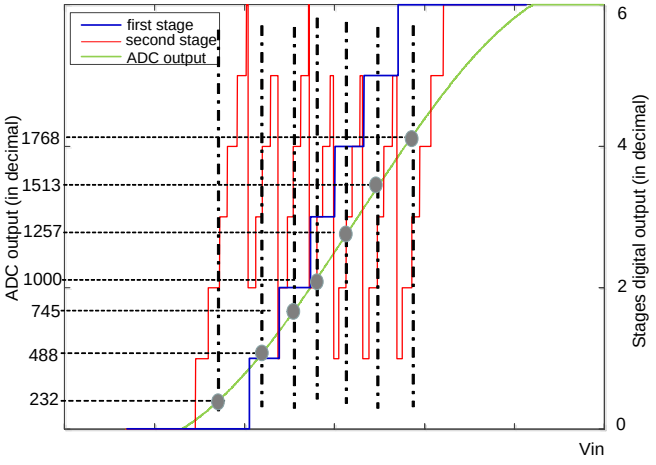


Fig. 15. Root codes example.

subtractions give: $1768-6*2^8=232$; $1513-5*2^8=233$; $1257-4*2^8=233$; $1000-3*2^8=232$; $745-2*2^8=233$; $488-1*2^8=232$; $232-0*2^8=232$. As expected, the right root code is toggling due to the presence of noise. In this example, it is toggling between two values (e.g. 232 and 233). By applying the majority voting scheme, we choose 232 as the right root code of the third comparator of the second stage since it is met more frequently. This means that the codes that should be grouped together as having the same DNLs are not 232, 488, 745, 1000, 1257, 1513 and 1768 as it would have been suggested without using the noise cancelling scheme, but 232, 488, 744, 1000, 1256, 1512 and 1768. These are the codes that are mapped to this comparator recalculated based on the same root code (e.g. 232) by adding the different contributions of the first stage.

After having verified the three principles of the reduced-code testing technique, we calculated the DNL and INL based on a reduced set of codes. We considered 8 codes around each of the six ADC output transitions representing the six comparators of the first stage, 6 codes around each of the six ADC output transitions representing the six comparators of the second stage, 4 codes around each of the six ADC output transitions representing the six comparators of the third stage, and 2 codes around each of the six ADC output transitions representing the six comparators of the fourth and fifth stages. In total, we rely only on $8\times6+6\times6+4\times6+2\times6+2\times6=132$ out of 2046 codes of an 11-bit ADC, that is, on only 6% of the codes, which represents a very significant static test time reduction.

The DNL obtained using the standard histogram technique is shown in Fig. 16(a) while the estimated DNL using the reduced-code testing technique without and with noise cancellation is shown, respectively, in Fig. 16(b) and 16(c). The estimated profiles of DNL in Fig. 16(b) and 16(c) are more regular since they are extracted from the DNLs of a reduced

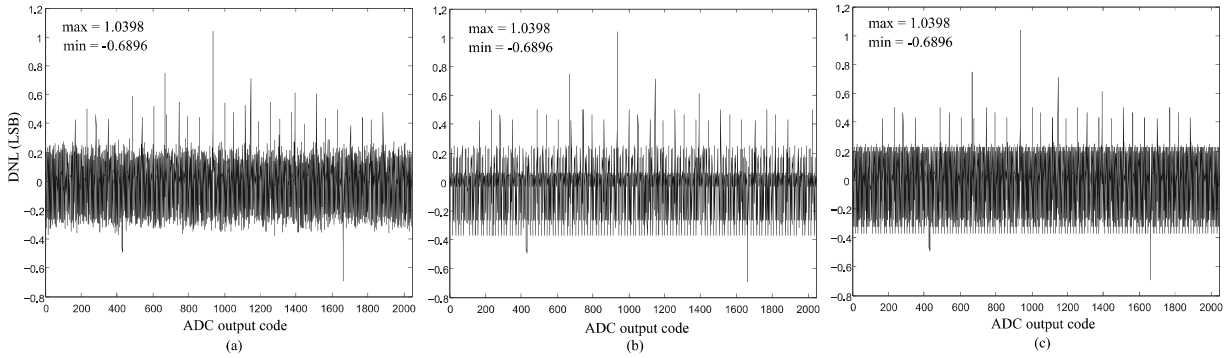


Fig. 16. DNL obtained with: (a) standard histogram technique; (b) reduced-code testing without noise cancelling; and (c) reduced-code testing with noise cancelling.

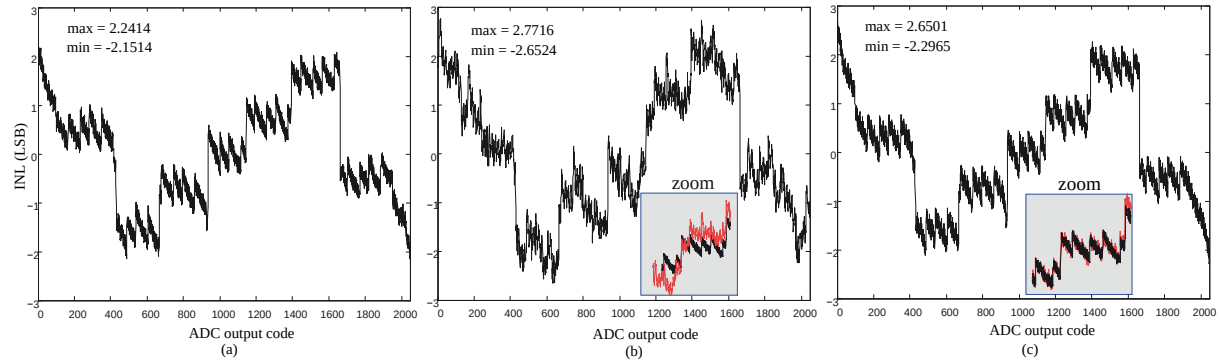


Fig. 17. INL obtained with: (a) standard histogram technique; (b) reduced-code testing without noise cancelling; and (c) reduced-code testing with noise cancelling.

set of codes. The highest DNL errors in the ADC correspond to the first stages in which the transitions are not so noisy. Thus, the minimum and maximum DNLs are well captured regardless whether noise cancellation is used or not. However, when comparing the profile of the smaller absolute DNLs in Fig. 16(b) and Fig. 16(c) with the profile of the smaller absolute DNLs in Fig. 16(a), we observe that the profile in Fig. 16(b) is less ”dense” as opposed to the profile in Fig. 16(c), which implies that there are many codes that have been assigned smaller absolute DNL values. This implies that, unless noise cancelling is used, there are significant errors in the mapping between the ADC output transitions and the comparators that are being exercised in the case where the comparators belong to stages that are towards the end of the pipeline.

The INL obtained using the standard histogram technique is shown in Fig. 17(a) while the estimated INL using the reduced-code testing technique without and with noise cancellation is shown, respectively, in Fig. 17(b) and 17(c). The inset plots in 17(b) and 17(c) show a zoom of the estimated INL profiles superimposed on INL profile obtained using the standard histogram technique. From Fig. 17(c) it is evident that the reduced-code testing technique with noise cancellation offers an excellent INL estimation, implying that we are summing up a succession of well estimated DNLs despite the noise in the ADC output transitions. In fact, the estimated INL is practically indistinguishable from the INL obtained with the standard histogram technique. In contrast, from Fig. 17(b) it is evident that the reduced-code testing technique without noise cancellation, although is capable of capturing the general INL profile, results in evident INL errors due to summing up a succession of poorly estimated DNLs. The general INL profile is captured thanks to the high peaks of DNL. In the case of an ADC with small DNL errors, the reduced-code testing technique without noise cancellation would have failed to capture even the general INL profile. In contrast, if noise cancellation is used, the INL estimation will be excellent independently of the DNL values of the ADC and the level of noise in the measurement environment.

VII. Conclusion

We proposed an enhancement of the reduced-code testing technique for pipeline ADCs that dramatically improves its effectiveness. The reduced-code testing technique is based on the underlying observation that there are output codes that have equal widths thanks to the architectural properties of the pipeline ADC. Therefore, to extract the complete transfer characteristic of the ADC it suffices to measure only a small subset of codes. For this reason, compared to the standard histogram technique that requires measuring irrespectively all codes, the reduced-code testing technique reduces drastically the static test time. The key is to be able to find the mapping between

the ADC codes and the comparators that are being exercised in the internal stages of the pipeline. The proposed enhancement relies, first, on monitoring the outputs of the internal stages of the pipeline and, second, on canceling out the noise, which is an important step in deriving an accurate mapping. For this purpose, we exploit the inherent properties of the architecture of a pipeline ADC. We demonstrate that, compared to generic static test approaches for ADCs, greatest static test cost reduction can be achieved by delving into the ADC architecture to exploit its inherent properties. Experimental results on a 55nm 11-bit 2.5-bit/stage pipeline ADC demonstrate that by measuring only 6% of the codes we can estimate the DNL and INL profiles very accurately. The resulting DNL and INL profiles are practically indistinguishable from those produced by the standard histogram technique.

REFERENCES

[1] F. Poehl, F. Demmerle, J. Alt, and H. Obermeir, "Production test challenges for highly integrated mobile phone SOCs–a case study," in *Proc. IEEE European Test Symposium*, 2010, pp. 17–22.

[2] K. Arabi and B. Kaminska, "Efficient and accurate testing of analog-to-digital converters using oscillation-test method," in *Proc. European Design and Test Conference*, 1997, pp. 348–352.

[3] Y. C. Wen, "A BIST scheme for testing analog-to-digital converters with digital response analyses," in *Proc. IEEE VLSI Test Symposium*, 2005, pp. 221–225.

[4] H. Xing, H. Jiang, D. Chen, and R. L. Geiger, "High-resolution ADC linearity testing using a fully digital-compatible BIST strategy," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 8, pp. 2697–2705, 2009.

[5] B. Provost and E. Sanchez-Sinencio, "Auto-calibrating analog timer for on-chip testing," in *Proc. IEEE International Test Conference*, 2010, pp. 541–548.

[6] E.S. Erdogan and S. Ozev, "An ADC-BIST scheme using sequential code analysis," in *Proc. Design Automation Test in Europe Conference Exhibition*, 2007, pp. 1–6.

[7] F. Azais, S. Bernard, Y. Bertrand, X. Michel, and M. Renovell, "A low-cost adaptive ramp generator for analog BIST applications," in *Proc. IEEE VLSI Test Symposium*, 2001, pp. 266–271.

[8] W.-T. Lee, Y.-Z. Liao, J.-C. Hsu, Y.-S. Hwang, and J.-J. Chen, "A high precision ramp generator for low cost ADC test," in *Proc. International Conference on Solid-State and Integrated-Circuit Technology*, 2008, pp. 2103–2106.

[9] B. Provost and E. Sanchez-Sinencio, "On-chip ramp generators for mixed-signal BIST and ADC self-test," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 2, pp. 263–273, 2003.

[10] B. Provost and E. Sanchez-Sinencio, "A practical self-calibration scheme implementation for pipeline ADC," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 2, pp. 448–456, 2004.

[11] Y. Wang, J. Wang, F. Lai, and Y. Ye, "Optimal schemes for ADC BIST based on histogram," in *Proc. IEEE Asian Test Symposium*, 2005, pp. 52–57.

[12] "IEEE standard for terminology and test methods for analog-to-digital converters," *IEEE Std 1241-2010 (Revision of IEEE Std 1241-2000)*, pp. 1 –139, 2011.

[13] "IEEE standard for digitizing waveform recorders," *IEEE Std 1057-2007 (Revision of IEEE 1057-1994)*, pp. 1 –142, 2008.

[14] F. Alegria, P. Arpaia, A.M. Cruz Serra, and P. Daponte, "Performance analysis of an ADC histogram test using small triangular waves," *IEEE Transactions on Instrumentation and Measurement*, vol. 51, no. 4, pp. 723–729, 2002.

[15] L. Jin, K. Parthasarathy, T. Kuyel, D. Chen, and R.L. Geiger, "Accurate testing of analog-to-digital converters using low linearity signals with stimulus error identification and removal," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1188–1199, 2005.

[16] M. A. Jalon, A. Rueda, and E. Peralias, "Enhanced double-histogram test," *Electronics Letters*, vol. 45, no. 7, pp. 349–351, 2009.

[17] H.-K. Chen, C.-H. Wang, and C.-C. Su, "A self calibrated ADC BIST methodology," in *Proc. IEEE VLSI Test Symposium*, 2002, pp. 117–122.

[18] R. Holce, L. Michaeli, and J. Saliga, "DNL ADC testing by the exponential shaped voltage," *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 3, pp. 946–949, 2003.

[19] F. Adamo, F. Attivissimo, N. Giaquinto, and M. Savino, "FFT test of A/D converters to determine the integral nonlinearity," *IEEE Transactions on Instrumentation and Measurement*, vol. 51, no. 5, pp. 1050–1055, 2002.

[20] C. Wegener and M.P. Kennedy, "Linear model-based testing of ADC nonlinearities," *IEEE Transactions on Circuits and Systems*, vol. 51, no. 1, pp. 213–217, 2004.

[21] Z. Yu and D. Chen, "Algorithm for dramatically improved efficiency in ADC linearity test," in *Proc. International Test Conference*, 2012, pp. 1–10.

[22] S. Goyal and A. Chatterjee, "Linearity testing of A/D converters using selective code measurement," *Journal of Electronic Testing: Theory and Applications*, vol. 24, no. 6, pp. 567–576, 2008.

[23] J. Lin, S. Chang, T. Kung, H.Ting, and C. Huang, "Transition-code based linearity test method for pipelined ADCs with digital error correction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 12, pp. 2158–2169, 2010.

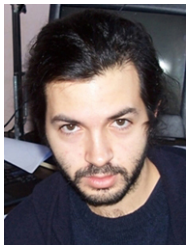
[24] A. Laraba, H. G. Stratigopoulos, S. Mir, H. Naudet, and C. Forel, "Enhanced reduced code linearity test technique for multi-bit/stage pipeline ADCs," in *Proc. IEEE European Test Symposium*, 2012, pp. 50–55.

[25] A. Laraba, H. G. Stratigopoulos, S. Mir, H. Naudet, and G. Bret, "Reduced code linearity testing of pipeline ADCs," *IEEE Design and Test of Computers*, vol. 30, no. 6, pp. 80–88, 2013.

[26] A. Laraba, H. G. Stratigopoulos, S. Mir, H. Naudet, and G. Bret, "Reduced code linearity testing of pipeline ADCs in the presence of noise," in *Proc. IEEE VLSI Test Symposium*, 2013, pp. 1–6.

[27] F. Maloberti, *Data Converters*, IEEE Computer Society Press, 2007.

Asma Laraba (S'12) received a MSc in micro- and nanoelectronics from Joseph Fourier University, Grenoble, France in 2010 and a PhD in electrical engineering from National Institute Polytechnic of Grenoble in 2013. Her PhD thesis research was conducted at TIMA Laboratory, Grenoble, France and focused on DFT and BIST of analog-to-digital converters. She is the recipient of the 2012 European Test Symposium best paper award. She is currently an analog design engineer in Xilinx, Dublin, Ireland.



Haralampos-G. Stratigopoulos (S'02-M'07) received the Diploma in electrical and computer engineering from the National Technical University of Athens, Greece, in 2001 and the Ph.D. in electrical engineering from Yale University, USA, in 2006. From October 2007 to May 2015 he was a Researcher with the French National Center for Scientific Research (CNRS) at TIMA Laboratory, Grenoble, France. Currently he is a researcher with the CNRS at LIP6 Laboratory, Paris, France. His main research interests are in the areas of design-for-test and built-in test for analog, mixed-signal, RF circuits and systems, computer-aided design, and machine learning. He was the General Chair of the 2015 IEEE International Mixed-Signal Testing Workshop (IMSTW), the Program Chair of the 2011 IEEE International Mixed-Signal, Sensors, and Systems Testing Workshop (IMS3TW), and the Program Chair of the 2013 IEEE International Workshop on Test and Validation of High-Speed Analog Circuits (TVHSAC). He has served on the Technical Program Committees of Design, Automation, and Test in Europe Conference (DATE), IEEE International Conference on Computer-Aided Design (ICCAD), IEEE VLSI Test Symposium (VTS), IEEE European Test Symposium (ETS), IEEE International Test Conference (ITC) and several others international conferences. He is an Associate Editor of Springer Journal of Electronic Testing: Theory & Applications, IEEE Design & Test Magazine, and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. He received the Best Paper Award in the 2009, 2012, and 2015 IEEE European Test Symposium (ETS).

Salvador Mir (M'99) has an Industrial Engineering (Electrical, 1987) degree from the Polytechnic University of Catalonia, Barcelona, Spain, and M.Sc. (1989) and Ph.D. (1993) degrees in Computer Science from the University of Manchester, UK. He is a Research Director of CNRS (Centre National de la Recherche Scientifique) at TIMA Laboratory in Grenoble, France. He is director of TIMA Laboratory since January 2015, and he has lead the RMS (Reliable Mixed-signal Systems) Group from 2002 to 2014. His area of expertise is analog/mixed-signal/RF/MEMS test. He has published many papers in this field and he has been General, Program, Topic Chair and member of the Steering Committee of many IEEE/IFIP International Conferences. He is editor of two books on silicon Microsystems.

Hervé Naudet leads the DFT and test engineering teams in the Digital Product Group of STMicroelectronics in Grenoble, France. His research interests include systems-on-chip (SoCs), systems-in-packages (SiPs), mixed-signal testing, analog and digital DFT, and industrial test strategy. He has been a presenter in several international conferences, including CNRS conference in Paris (2002), Design Technology Conference in Grenoble (2010), and VOICE in San Jose (2012). He has an MEA engineer diploma from the Science Engineering Institute of Montpellier.