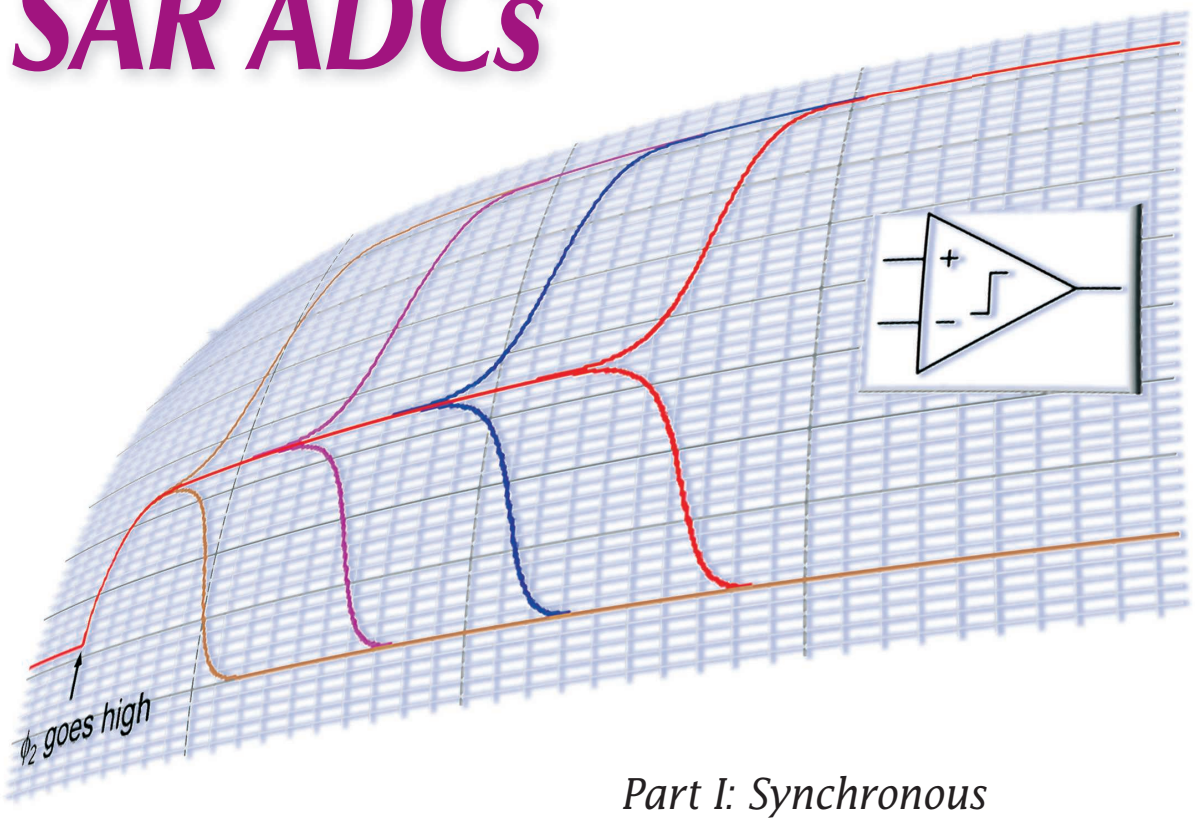
Daniel Bankman, Andrew Yu, Kevin Zheng, and Boris Murmann

Understanding Metastability in SAR ADCs



Part I: Synchronous

By strongly leveraging technology scaling in its building blocks, the successive approximation architecture resides comfortably at both the high-speed and high-resolution frontiers of the analog-to-digital conversion landscape [1]–[3]. This particularly digital-friendly set of building blocks includes MOS switches, metal fringe capacitors, and regenerative latches, each of which has analog nonidealities that can limit the attainable speed and resolution of a successive approximation register (SAR) analog-to-digital converter (ADC). Whereas circuit techniques such as bottom-plate sampling and mismatch calibration have a proven track record for mitigating sources of nonlinearity, techniques that address the problem of metastability in the regenerative latch are still under development [4]–[6]. This two-part tutorial aims to help readers follow the most recent advancements in this area, by providing an all-in-one introduction to the subject of metastability in SAR ADCs not yet available in the literature or textbooks. A clear picture of how metastability, noise, and the SAR algorithm interact is a useful prerequisite for deciding where and how potential metastability errors should be dealt with in the design of SAR ADCs.

Digital Object Identifier 10.1109/MSSC.2019.2910647
Date of publication: 24 June 2019

1943-0582/19©2019IEEE

Metastability and thermal noise in the regenerative latch have both been analyzed individually in the context of the SAR algorithm [7]–[9]. It is well known that metastability can cause as large as quarter-scale errors in the ADC output code with very small probability that increases steeply as a function of the regenerative latch time constant [10]. Although conventional wisdom suggests that the comparator's thermal noise can be directly referred to the input of the SAR ADC, [9] showed that this is not exactly the case, because the comparator generates an independent noise sample on every successive decision. Whereas metastability causes large code errors with small probability, comparator noise causes small code errors with relatively large probability. How should we think about metastability and thermal noise in a combined framework? Can we just overlay the code error distributions? Or do these two phenomena interact in some interesting way?

This article is the first in a two-part series that examines how metastability and thermal noise interact to shape the code error distribution of the SAR ADC. We consider the synchronous architecture in this part and the asynchronous architecture in Part II. To start, we need a clear baseline for how metastability and thermal noise affect the synchronous SAR loop in isolation. One gap in existing models for metastability in SAR ADCs is the assumption that the comparator decision immediately following a metastable event is random (i.e., a 50/50 coin toss) [7], [8]. Unfortunately, assuming that metastability and noise interact in this simple way does not allow us to predict the effect of increasing or decreasing the amount of comparator noise in the SAR loop. Although the aftermath of a metastable event can be heavily influenced by noise, metastability itself is a deterministic phenomenon.

In the following section, we build up a model for the metastability code error distribution where the ADC input signal is random but the response of the SAR loop is completely deterministic. This will show that the digital-to-analog converter (DAC) switching scheme plays a significant role in determining the shape of the distribution. Then, we review the effects of thermal noise in the SAR loop. By extending this thermal noise model to include the possibility of causing metastability, we reveal the code error distribution that faithfully captures the interaction of the two effects.

A Deterministic Model for Metastability

Rather than calling it a coin toss, let's try to track down how the SAR loop actually responds to a metastable event. To do this, we need to look inside the SAR logic. Which circuit elements are required, and how do they function together?

The SAR ADC effectively runs a binary search [Figure 1(a)]. At each successive approximation step, the polarity of the residuum determines which direction the DAC should switch to create a next guess that's closer to the input. A voltage comparator converts the polarity of the residuum to a digital bit, which then controls the switching of one binary-weighted DAC input in order from most to least significant. As the conversion proceeds, a memory element must hold each comparator decision at the appropriate DAC input to build a voltage waveform like that in Figure 1(a). This is typically achieved by pairing each DAC input with a capture latch, driving all latch inputs with the comparator, and selecting the appropriate latch at each step [Figure 1(b)]. What's important for metastability is that the receiver of the comparator output is itself a latch.

In the SAR ADC, the comparator's job is to amplify its analog input to a valid digital output as fast as possible. Regenerative comparators have become extremely popular because the latch is the fastest amplifier available, as long as gain is required only at discrete instants in time. The regenerative latch [Figure 1(c)] amplifies its sampled voltage exponentially with time:

$$v_{OD}(t) = v_{OD,0}\, e^{t/\tau}.$$

In the case of the comparator, this sampled voltage is the residuum. In the case of the capture latch, this sampled voltage is the comparator output. Although the exponential transient is the fastest option available, the

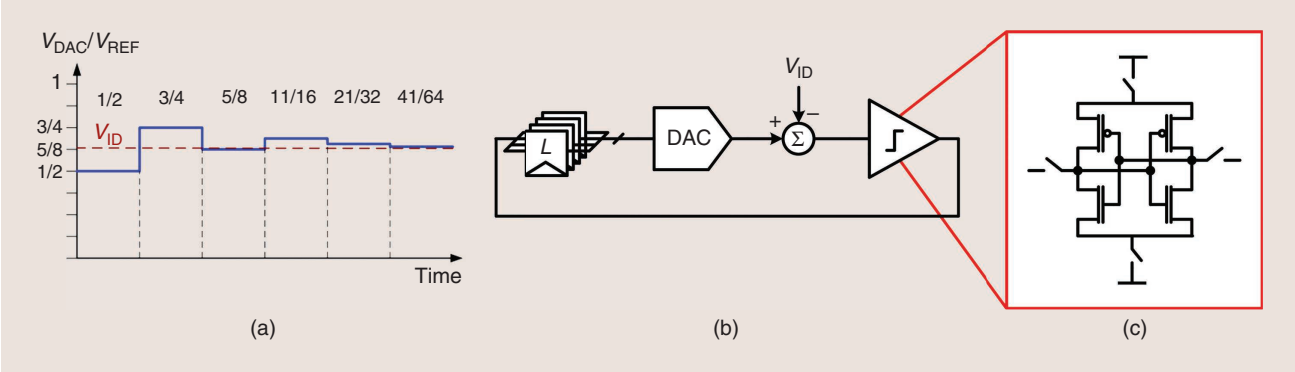


FIGURE 1: The basic synchronous SAR ADC (a) conversion waveform and (b) block diagram. The SAR loop effectively runs a binary search. Both the comparator and capture latch are built around a regenerative latch formed by (c) cross-coupled inverters.

fundamental issue remains that, as the sampled voltage becomes arbitrarily small, the time needed to hit the digital rails and switch the DAC becomes arbitrarily large. If this delay extends into the next clock cycle, the SAR algorithm malfunctions, and a metastability error occurs.

Figure 2 shows the three most likely scenarios for timing at the comparator to the capture latch interface. The comparator and the capture latch are enabled simultaneously while the clock is high. On the falling edge, the comparator is reset, and the capture latch becomes opaque. The most likely scenario [Figure 2(b)] is that the comparator regenerates completely before reset and that the capture latch samples a valid digital bit on the falling edge. For a small range of residual voltages, the comparator will not finish regenerating before reset, and the capture latch will instead sample an input somewhere between the rails [Figure 2(c)]. Here, we assume that the regenerative latches inside the comparator and the capture latch are identical, such that the capture latch continues regeneration seamlessly after it samples. In Figure 2(c), the capture latch finishes regenerating before the next comparator decision; hence, the SAR algorithm still runs as expected. However, for an even smaller range of residual voltages, the capture latch will not finish regenerating until sometime during the next clock cycle [Figure 2(d)]. In this case, the DAC does not switch to the appropriate next guess in time for the next comparator decision. Clearly, this causes a problem for binary search, the nature of which depends on the DAC switching scheme.

At this point, let's pause to think deeply about two assumptions of the model. First, we assumed that the capture latch continues regeneration seamlessly after comparator reset. In a practical implementation, the comparator and the capture latch may not have the same metastable operating point, making seamless operation impossible. However, in this case, the range of residual voltages leading to a metastability error will stay the same; it will just be shifted away from zero differential by a diminishingly small amount, which drops exponentially with the time available for regeneration. When we consider the probability that the input signal excites metastable behavior, it will be clear that such a shift is negligible relative to the spread of the input signal distribution. Second, it is worth noting that the range of residual voltages making the regeneration time last more than two clock cycles is so small that we call it *zero* for the purposes of this article. If we considered such tiny voltages, then this analysis would show that the probability of sampling them with the comparator falls below 10^{-20}.

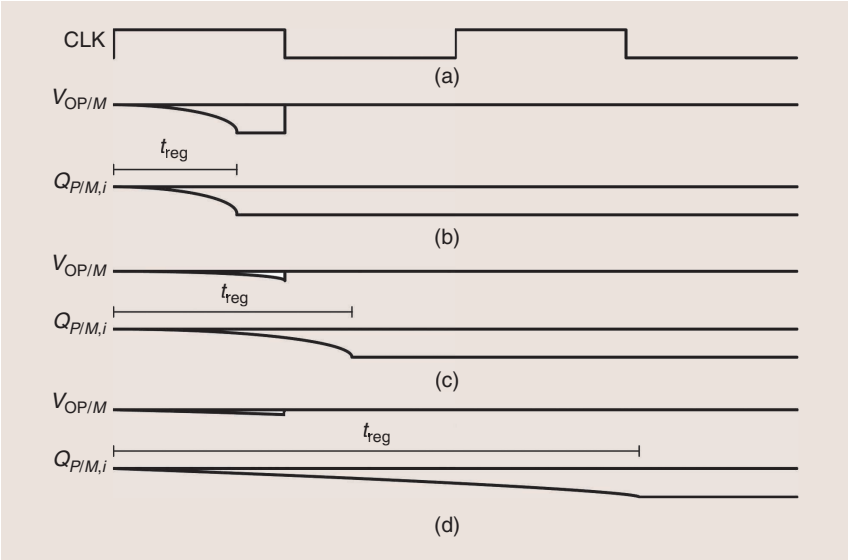


FIGURE 2: Waveforms illustrating comparator outputs $v_{OP/M}$ and capture latch outputs $Q_{P/M,i}$ for the three most likely regeneration scenarios. It is assumed that the comparator and capture latch contain identical regenerative latches and that their outputs are reset (precharged) high. (a) The comparator clock. (b) The comparator fully regenerates its sampled input to the rails before reset. (c) The comparator does not fully regenerate before reset, but the capture latch finishes regeneration before the next comparator decision. (d) The capture latch finishes regeneration sometime during the next clock cycle, causing the SAR algorithm to malfunction.

Bottom-Plate Versus Top-Plate Sampling

In recent years, several DAC switching schemes have emerged for the purpose of reducing the CV^2 energy it takes to run the SAR algorithm. A key point for metastability is the distinction between the classic set–reset procedure and the more recent energy-efficient idea of switching the DAC strictly in response to a comparator decision [11], [12]. Which of these two categories the switching scheme falls under depends on the use of bottom-plate versus top-plate sampling. Figure 3 shows the schematics and timing diagrams for the bottom-plate sampling scheme of [11] [Figure 3(a)–(c)] and the top-plate sampling scheme of [12] [Figure 3(d)–(f)].

With bottom-plate sampling, some form of DAC switching is inherently required in between the hold instant and the most significant bit (MSB) decision because, during the track interval, the comparator sees a differential short. This scheme must use a set–reset procedure for switching the DAC, whereby each bit is set to a logic 1, tested, and then either held or reset to 0 according to the comparator decision. If the capture latch is regenerating toward a logic 1, then it doesn't matter how long it takes to get there, because the corresponding DAC input is already set to logic 1 to begin with. However, if the capture latch is regenerating toward a logic 0, then this needs to happen within one clock cycle; otherwise, the DAC will not switch to the appropriate next

guess before the next comparator decision. This introduces a bias into the metastability code error distribution for bottom-plate sampling.

With top-plate sampling, the ADC input voltage appears at the comparator input during the track interval and after the hold instant. Because of the differential architecture, no initial DAC switching is needed to resolve the MSB, which is simply the polarity of the sampled voltage. From the MSB decision down to the second least significant bit (LSB) decision, the DAC switches strictly in response to each comparator decision. If the capture latch does not regenerate the comparator decision within one clock cycle, then the DAC doesn't switch at all. This leads to the somewhat startling conclusion that the decision immediately following a metastable decision is also metastable. Of course, in the presence of thermal noise, such double metastable events occur with practically zero probability. But, to establish a baseline for the effect of metastability on a noiseless SAR loop, we must consider this behavior.

A Delay Model for the SAR Logic

The SAR logic schematics in Figure 3(c) and (f) contain two more signal path elements in addition to the comparator and capture latch. These are the fixed delay and an ideal threshold. The fixed delay models regeneration-independent delay sources, such as buffers, DAC settling, and set–reset logic for bottom-plate sampling. The ideal threshold allows us to model DAC switching as an abrupt but delayed jump, similar to the analysis of metastability in pipeline ADCs presented in [13]. Here, we assume that the effect of partial DAC settling can be referred into the model's T_{FIX} parameter.

The time it takes to regenerate some residual voltage to the rails is

$$t_{\mathrm{reg}} = \tau \ln\left(\frac{V_{\mathrm{DD}}}{v_{\mathrm{res}}}\right).$$

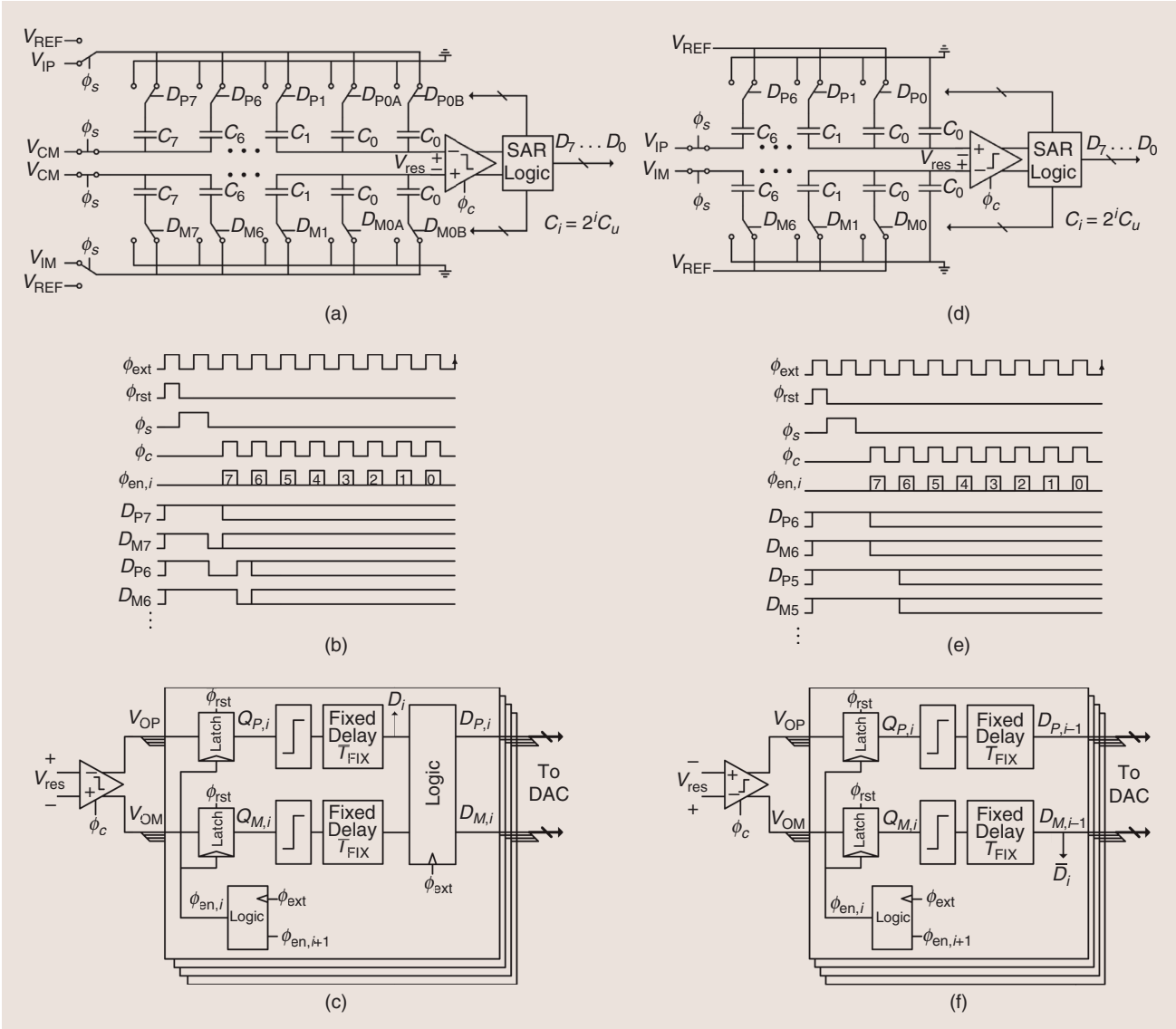


FIGURE 3: The schematic, timing diagram, and logic delay model for the synchronous SAR with (a)–(c) bottom-plate sampling and (d)–(f) top-plate sampling.

Because we assume that the capture latch continues regeneration seamlessly with the same τ as the comparator, we can use this expression without further consideration of the comparator reset. If the DAC does not switch before the next comparator decision due to late regeneration, a metastability error occurs. This happens when $t_{reg} + T_{FIX} > T_{CLK}$ or, equivalently, when

$$|v_{res}| < V_{DD}\, e^{-(T_{CLK}-T_{FIX})/\tau} = V_{DD}\, e^{-N},$$

where $N = (T_{CLK} - T_{FIX})/\tau$ is the number of latch time constants that fit into the time available for regeneration per clock cycle. This parameter is all we need to fully characterize the metastability code error distribution, independent of the specific frequency of operation.

Impact of Late Regeneration on Code Error

To complete our deterministic model of the SAR loop response to metastability and find the resulting code error, we'll consider two illustrative examples, shown in Figure 4. Without loss of generality, we assume that the ADC full-scale range V_{FS} is equal to the comparator supply voltage V_{DD}. In these examples, $T_{FIX} = T_{CLK}/2$, and all decisions are instantaneous except for the second MSB. The ADC input voltage resides just under $V_{FS}/4$, close enough that the magnitude of the second MSB residual voltage is smaller than $V_{DD}\, e^{-N}$. The second MSB decision creates a logic 0, with regeneration time equal to T_{CLK}. Accounting for the fixed delay, the corresponding DAC input switches half a cycle after the next comparator decision, causing an error. The resulting DAC waveforms are different, but the code errors turn out to be the same for a positive residuum.

Figure 4(b) shows the behavior for the bottom-plate sampling DAC switching scheme. Because the second MSB decision regenerates one cycle late, the second MSB of the DAC is still set to a logic 1 by the time of the third MSB decision. With both the second and third MSB set to logic 1, the DAC creates a guess that is erroneously higher than the input voltage. This leads to a logic 0 for the third MSB decision, rather than the correct logic 1 value. However, by the time of the fourth MSB decision, the second MSB capture latch has finally regenerated its logic 0. From this point forward, the DAC waveform is consistently lower than it should have been, but this still leads to all remaining bits being resolved correctly as logic 1s. For this 8-b example, the correct ADC output code is 10111111. Because of metastability on the second MSB decision, this output code becomes 10011111. The code error is –32.

Figure 4(c) illustrates what happens with the top-plate sampling switching scheme for the same input voltage. Late regeneration of the second MSB means that the DAC doesn't switch at all before the third MSB decision. In the noiseless SAR loop, this implies that the residual voltage is unchanged and that the third MSB decision will be an erroneous logic 0 with the same late regeneration time. By the fourth MSB decision, the second MSB capture latch has finally regenerated its logic 0, and all subsequent bits are resolved correctly as logic 1s. Interestingly, the DAC voltage takes another negative step at time $4T_{CLK}$. This is because the third MSB

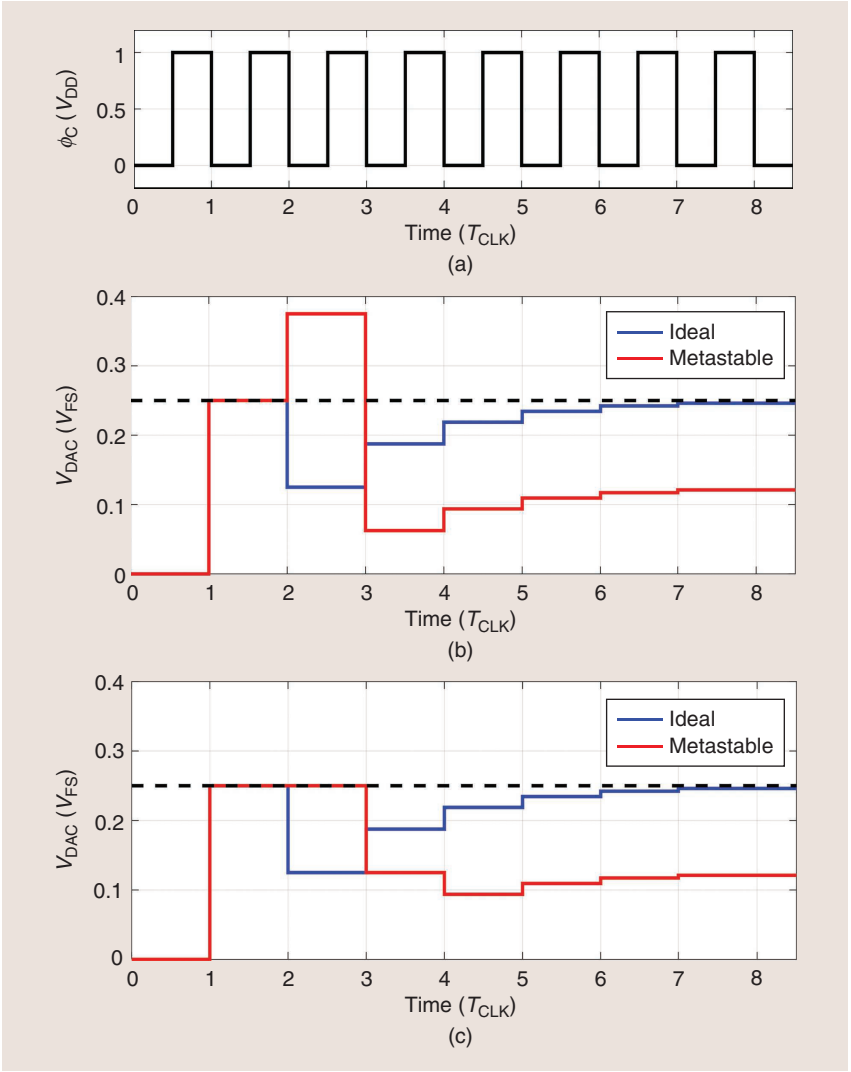


FIGURE 4: (a) The comparator clock waveform. Example waveforms of the SAR loop response to second MSB metastability for (b) bottom-plate sampling and (c) top-plate sampling.

decision, a logic 0, reaches the DAC input at this time. It has double the weight and negative the polarity of the fourth MSB, a logic 1. Just as with bottom-plate sampling, the end result is that the bit immediately following the first metastable bit becomes flipped. The code error is again –32.

If the residuum on the second MSB decision had the same small magnitude but negative polarity, it would have no effect on the bottom-plate sampling switching scheme. This is because, with the set–reset procedure, each DAC input is set to a logic 1, tested, and only reset to logic 0 if the capture latch regenerates a logic 0 at some point. With a negative residuum, the capture latch regenerates a logic 1, but when this happens is irrelevant because the DAC input is set to a logic 1 to begin with. However, with top-plate sampling, there is no such set–reset procedure to bias the DAC input in one direction. The resulting behavior for top-plate sampling is that the code error is negated if the residual voltage is negated at the metastable decision. This symmetry is broken by special cases on the LSB and second LSB due to the time available before output capture, but the resulting code errors are bounded to ±1 LSB and hence not worth discussing further.

To summarize, for bottom-plate sampling, metastability on bit i with positive residuum results in a code error of -2^{i-1} and with negative residuum results in no code error (see "Metastability Code Errors"). For top-plate sampling, metastability on bit i, with positive residuum, results in a code error of -2^{i-1} and, with negative residuum, results in a code error of 2^{i-1}. These results are similar to [7] and [8], but here we have carried out a completely deterministic analysis, without one logic gate tossing a coin. Now that we have a circuit model for this deterministic SAR loop behavior, we can augment it with thermal noise to find out precisely how metastability and thermal noise interact. In developing this model, we saw that the bottom-plate sampling DAC switching scheme can actually bias metastability code errors to a single polarity. We will see later that thermal noise does not undo this bias. As our last step in pure metastability analysis, let's now consider how likely a random input signal is to excite these late regeneration events.

Metastability Code Errors

The following expressions provide an explicit description of the code errors due to metastability for the successive approximation register analog-to-digital converter in Figure 5 (with arbitrary resolution).

For bottom-plate sampling, metastability on decision i will result in code error

$$A_i = \begin{cases} -2^{i-1} & \text{if } v_{\text{res},i} > 0,\ i > 0 \\ 1 & \text{if } v_{\text{res},i} > 0,\ i = 0 \\ 0 & \text{if } v_{\text{res},i} < 0 \end{cases},$$

and, for or top-plate sampling,

$$A_i = \begin{cases} -2^{i-1} & \text{if } v_{\text{res},i} > 0,\ i > 0 \\ 0 & \text{if } v_{\text{res},i} > 0,\ i = 0 \\ 2^{i-1} & \text{if } v_{\text{res},i} < 0,\ i > 1 \\ 0 & \text{if } v_{\text{res},i} < 0,\ i = 1 \\ -1 & \text{if } v_{\text{res},i} < 0,\ i = 0 \end{cases}.$$

Metastability Code Error Distribution With Random Input Signal

Previously, we established a deterministic mapping from the position of the metastable bit and the polarity of its residuum to the corresponding ADC output code error. Using this mapping and the distribution of a random input signal, we can now find the probability mass function (PMF) over ADC output code errors due to metastability. For the armchair probabilist, the easiest way to do this is to partition the sample space into the following groups of $2^{B+1}-1$ events: for each of the 2^B-1 ADC transition levels, the events that the input voltage resides within $V_{\text{DD}}e^{-N}$ of the transition above and below, and the event that the input voltage does not reside within $V_{\text{DD}}e^{-N}$ of any transition level (i.e., the trivial case of no metastability). The nontrivial metastable events are depicted graphically in Figure 5 as red stripes surrounding the ADC code transitions. If the input voltage falls within one of these stripes, then the bit at the corresponding position will become metastable (e.g., the input approaching $\pm V_{\text{FS}}/4$ implies second MSB metastability). Given the position of the metastable bit and whether the input voltage resides above or below the transition (the polarity of the residuum), the resulting ADC output code error is determined. The probability that the input resides within a stripe can be calculated by integrating the input signal probability density function (PDF) in that voltage range. By iterating through each of these $2^{B+1}-2$ possible metastable events, calculating its probability, and incrementing the probability of the resulting ADC output code error

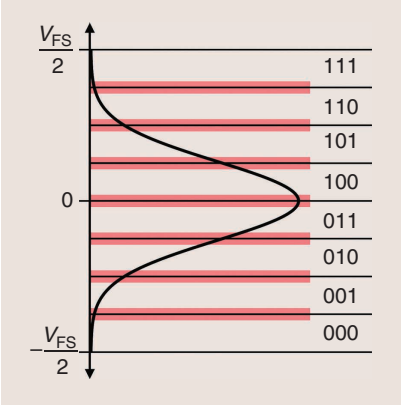


FIGURE 5: A 3-b example showing a Gaussian input signal with overlaid voltage intervals corresponding to metastable events.

by this amount, we can compute the code error PMF.

Figure 6 shows the metastability code error PMF for an 8-b synchronous SAR ADC with $N = 25$ latch time constants allocated for regeneration and a Gaussian input signal distribution. For both top-plate and bottom-plate sampling, the total probability of metastable events (of any error magnitude) is 7.11×10^{-9}. Had we assumed a uniform input signal distribution rather than Gaussian, we could have used the conventional expression to predict $P_{\text{meta}} = 2(2^B-1)e^{-N} = 7.08\times10^{-9}$. The small difference arises simply from the shapes of the two distributions. However, for both bottom-plate and top-plate sampling, all metastability errors are not equally catastrophic. The maximum code error magnitude is quarter scale, and this occurs with probability 4.26×10^{-11}, two orders of magnitude lower than the conventional P_{meta}. The most common nonzero code errors are +1, with probability 1.8×10^{-9} for bottom-plate sampling, and –1, with probability 2.7×10^{-9} for top-plate sampling. But wouldn't we expect code errors of ±1 LSB to be swamped by thermal noise anyway? If designing for the conventional P_{meta} is too conservative, then what feature of this code error distribution should we try to control with our design knobs? To answer these questions, we now find the code error distribution due to thermal noise and then consider its interaction with metastability.

Thermal Noise

Let's now shift gears to thermal noise to see how it affects the SAR loop in isolation. In [9], it was shown that comparator noise cannot be directly referred to the input of a SAR ADC. This is because the comparator generates an independent noise sample on every decision. How do noise samples on every decision combine to produce an error in the ADC output code?

This problem is made tractable by first assuming a deterministic input voltage to the ADC. In this manner, the comparator noise samples are the only source of randomness, and it's straightforward to track down their combined effect on the ADC output code. Any findings for a deterministic input can be extended to an arbitrary input PDF by approximating it with a discrete PMF, calculating the code error distribution for each possible input value, and summing these results weighted by the probability of each input value (i.e., invoking total probability). This works well as long as we quantize the input signal PDF using a step size much smaller than the thermal noise standard deviation.

In general, the comparator input at step i is the residual voltage, i.e., the error left in successive approximation

$$v_{\text{res},i} = v_{\text{DAC},i} - v_{\text{ID}}.$$

We can think of the comparator's input-referred noise as adding an independent noise sample to the input voltage on every decision:

$$v_{\text{res},i} = v_{\text{DAC},i} - (v_{\text{ID}} + v_{n,i}).$$

If the noise is large enough to flip the polarity of the residual voltage, a decision error occurs. Figure 7 depicts two possible outcomes for SAR loop behavior in the presence of comparator noise. The Gaussian distribution offset by v_{ID} depicts the distribution of $v_{\text{ID}} + v_{n,i}$ on each decision. Figure 7(a) shows the error-free case that we hope for. The probability mass of $v_{\text{ID}} + v_{n,i}$ on the correct side of $v_{\text{DAC},i}$ is the probability that the decision is correct. Because these decisions are independent, the probability that all decisions are correct is the product of the individual probabilities, i.e., the product of the shaded areas in Figure 7(a). In Figure 7(b), a decision error occurs on the second MSB decision. Having resolved an erroneous logic 1, the DAC steps up accordingly. Then, with high probability, the LSB is resolved correctly as a logic 0. The resulting

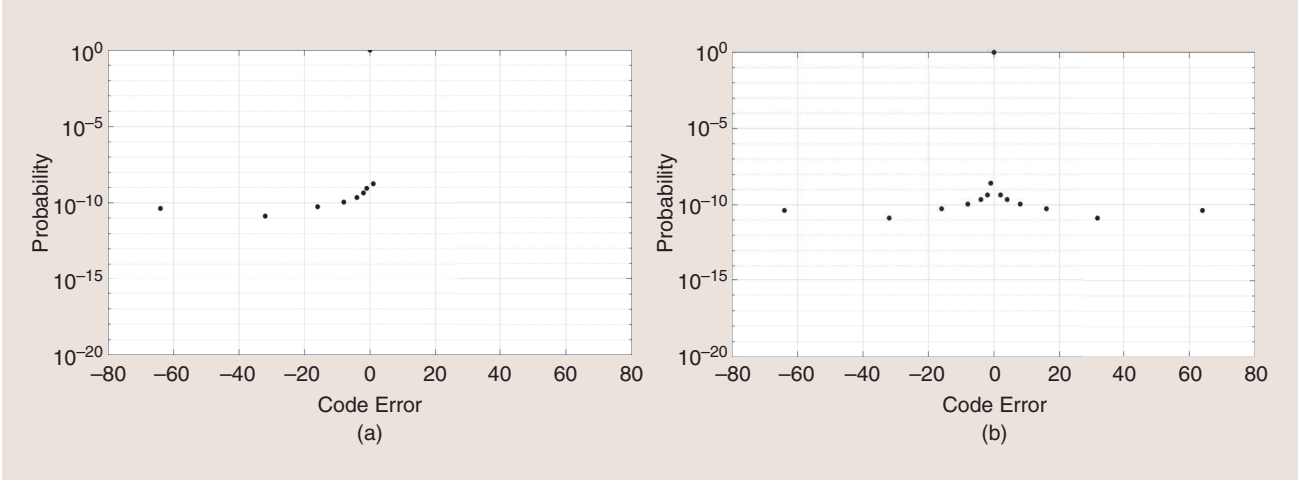


FIGURE 6: Code error PMFs due to metastability. An 8-b ADC with $N = 25$, digitizing a Gaussian input signal with clipping probability 10^{-4}. Distributions for (a) bottom-plate sampling and (b) top-plate sampling.

code error for this outcome is +1, and, due to the improbable second MSB error, this outcome is less likely than the error-free case.

Because the distribution of comparator input-referred noise is well modeled by a Gaussian, every decision has some nonzero probability of being a logic 1 or a logic 0. Even for a deterministic input voltage, all 2^B output codes are possible (although some are diminishingly improbable). To find the code error distribution, we can consider each of the 2^B possible outcomes, find the corresponding code error, and evaluate the probability. Then, we can increment the probability of the corresponding code error by this amount as we iteratively build the code error distribution.

Given a particular ADC output code, the DAC voltage waveform that leads there is determined. Each bit of the output code selects between two directions at each step in the SAR decision tree. We can easily build the DAC voltage waveform in this manner and then integrate the distribution of $v_{ID} + v_{n,i}$ at each step to find the individual probabilities of the decisions that give rise to this DAC voltage waveform. Then, the probability of the ADC output code is the product of these individual probabilities. By extending this method for deterministic inputs to random inputs using total probability as described previously, we can calculate the code error distribution for our Gaussian signal in the presence of just thermal noise. Figure 8 shows the code error distributions for several values of comparator noise, parameterized by effective number of bits (ENOB). Here, ENOB is just a proxy for the ratio of the comparator input-referred noise to the ADC full-scale range. It is calculated from SNR, taking signal to be a full-scale sinusoid and taking noise to be the sum of quantization noise and comparator input-referred noise. It is worth noting that these PMFs do not differ significantly from those predicted by conventional wisdom, i.e. a model that just refers the comparator noise directly to the input of the SAR ADC. However, it is still worthwhile to review this detailed thermal noise model because it is readily extended to capture the interaction between metastability and thermal noise in the SAR loop.

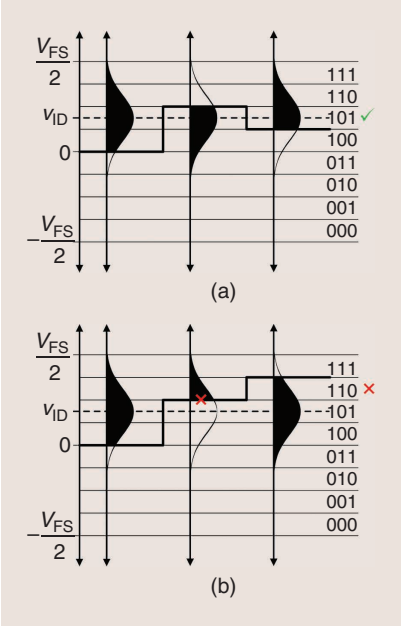


FIGURE 7: A 3-b example showing successive approximation of a deterministic input in the presence of comparator noise. (a) Each bit is resolved correctly, leading to a code error of 0. (b) Thermal noise flips the second MSB decision. The LSB is then resolved correctly with high probability, leading to a code error of +1.

Metastability and Thermal Noise Combined

This framework for calculating the code error distribution in the presence of noise turns out to be extremely versatile. In our model for metastability, each comparator decision has four possible outcomes. These are a logic 1/0 that regenerates on time and a logic 1/0 that regenerates one cycle late. Hence, the SAR loop behavior has 4^B possible outcomes. For each outcome, there is a corresponding DAC voltage waveform, which we can build up step by step by following the sequence of fast or slow, high or low decisions, similar to Figure 7. Then, to calculate the probability of some outcome, we integrate the distribution of $v_{ID} + v_{n,i}$ in the corresponding voltage ranges that give rise to this DAC waveform and find the product of these individual probabilities (see "Computing the Combined Probability Mass Function"). Figure 9 provides an example.

Figure 9(a) shows the error-free case, which results in the same DAC voltage waveform as Figure 7(a). However, a correct decision now entails not only resolving the correct bit, but also resolving it on time. Hence, the red stripes around the transition levels that lead to metastability are excluded from the voltage ranges leading to the error-free

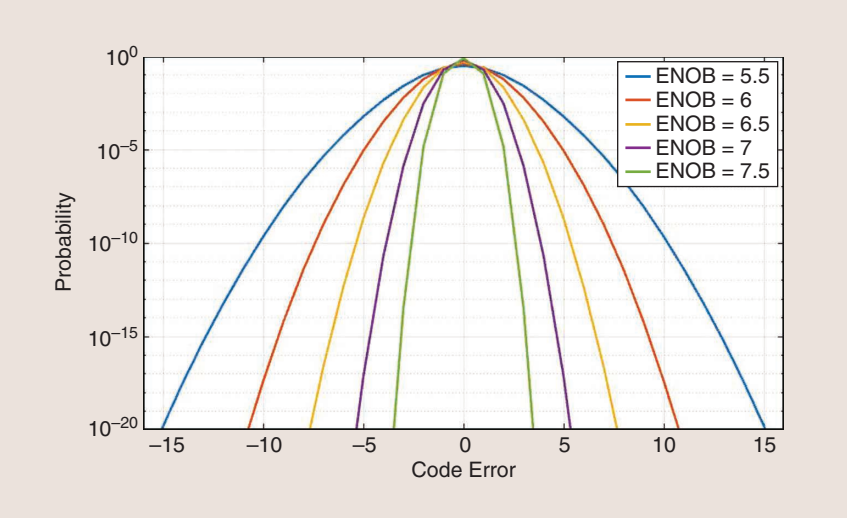


FIGURE 8: Code error PMFs due to thermal noise across a range of ENOB. An 8-b ADC digitizing a Gaussian input signal with clipping probability 10^{-4}.

Computing the Combined Probability Mass Function

Let the random variable D_i represent the comparator decision at step i (takes values 1 or 0), and let M_i be the indicator random variable that the capture latch regenerates one cycle late (takes value 1 for late, 0 for on-time). For some outcome, the particular values that these random variables take determine the digital-to-analog converter voltage waveform. For bottom-plate sampling, this is

$$v_{\mathrm{DAC},i} = \frac{-V_{\mathrm{FS}}}{2} + \sum_{k=i}^{B-1} \frac{V_{\mathrm{FS}}}{2^{B-k}} - \sum_{k=i+1}^{B-1} (1-d_k) \times \frac{V_{\mathrm{FS}}}{2^{B-k}} u[k-m_k-(i+1)],$$

and for top-plate sampling it is

$$v_{\mathrm{DAC},i} = \sum_{k=i+1}^{B-1} (-1)^{(1-d_k)} \frac{V_{\mathrm{FS}}}{2^{B-k+1}} \times u[k-m_k-(i+1)],$$

where $u[\cdot]$ is the discrete-time step function ($u[0]=1$). The probability of a particular outcome vector (d,m) given the input v_{ID} can be found by integrating the distribution of the residuum at each step and taking the product of these terms:

$$P_{v_{\mathrm{res},i}}(v) = \frac{1}{\sigma_n\sqrt{2\pi}} e^{-(v-v_{\mathrm{DAC},i}+v_{\mathrm{ID}})^2/(2\sigma_n^2)}$$

$$P(D=d \cap M=m \,|\, v_{\mathrm{ID}}) = \prod_{i=0}^{B-1} \int_{v_{\min,i}}^{v_{\max,i}} P_{v_{\mathrm{res},i}}(v)\,dv,$$

where the limits of integration are

$$v_{\min,i} = \begin{cases} V_{\mathrm{DD}}e^{-N} & \text{if } d_i=0 \text{ and } m_i=0 \\ 0 & \text{if } d_i=0 \text{ and } m_i=1 \\ V_{\mathrm{DD}}e^{-N} & \text{if } d_i=1 \text{ and } m_i=1 \\ -\infty & \text{if } d_i=1 \text{ and } m_i=0 \end{cases}$$

$$v_{\max,i} = \begin{cases} \infty & \text{if } d_i=0 \text{ and } m_i=0 \\ V_{\mathrm{DD}}e^{-N} & \text{if } d_i=0 \text{ and } m_i=1 \\ 0 & \text{if } d_i=1 \text{ and } m_i=1 \\ V_{\mathrm{DD}}e^{-N} & \text{if } d_i=1 \text{ and } m_i=0 \end{cases}.$$

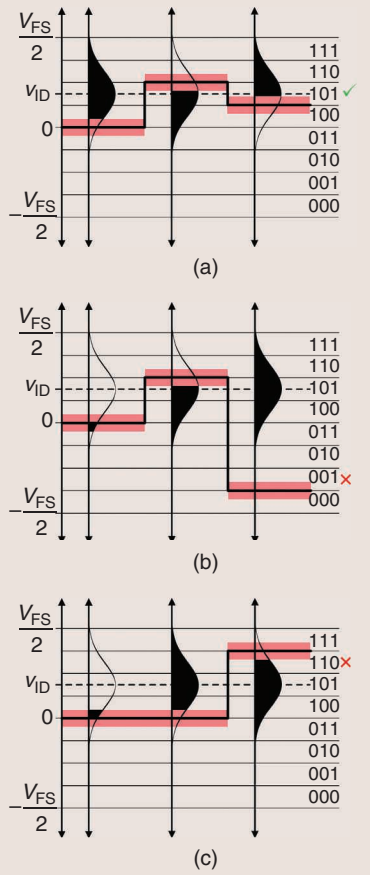


FIGURE 9: A 3-b example showing the successive approximation of a deterministic input in the presence of both metastability and comparator noise. (a) Each bit is resolved correctly and on time, leading to a code error of 0. (b) An example for bottom-plate sampling. The MSB decision is resolved incorrectly and regenerates one cycle late. If the second and third MSBs are regenerated correctly and on time, as shown, the resulting code error is –4. (c) An example for top-plate sampling. The MSB decision is resolved correctly but regenerates one cycle late. With correct and on-time decisions thereafter, the resulting code error is +1.

outcome. Figure 9(b) illustrates an example for the bottom-plate sampling DAC switching scheme in which the MSB not only flips because of noise but also regenerates late. Late regeneration with bottom-plate sampling implies that the DAC steps up, no matter what, for the subsequent decision, which in this case is resolved correctly and on time as a logic 0. However, this unfortunately leads to an even larger code error than if the second MSB were incorrect. By the third MSB decision, the MSB regenerates to a logic 0, and the DAC steps down twice, once for the late MSB reset and once for the second MSB reset. A logic 1 is almost certainly resolved in the LSB decision, but the resulting code error is –4. Had the second MSB decision been incorrectly resolved as a logic 1 due to noise, the resulting code error would be only –2.

Figure 9(c) shows an example for the top-plate sampling DAC switching scheme in which the MSB has the correct polarity and just regenerates late. This implies that the DAC doesn't switch at all before the second MSB decision. However, with this graphical interpretation, we can now see how diminishingly small the product of the shaded areas would become if the second MSB decision were also metastable. By far the most likely scenario is for the slow logic 1 on the MSB decision to be followed by a fast logic 1 on the second MSB decision. By the time of the LSB decision, the DAC steps up twice, due to the combination of late regeneration of the MSB and on-time regeneration of the second MSB. The LSB is resolved correctly as a logic 0, leading to a code error of +1. For both outcomes, the probability is again the product of the shaded areas. Finding the probability of all 4^B outcomes and

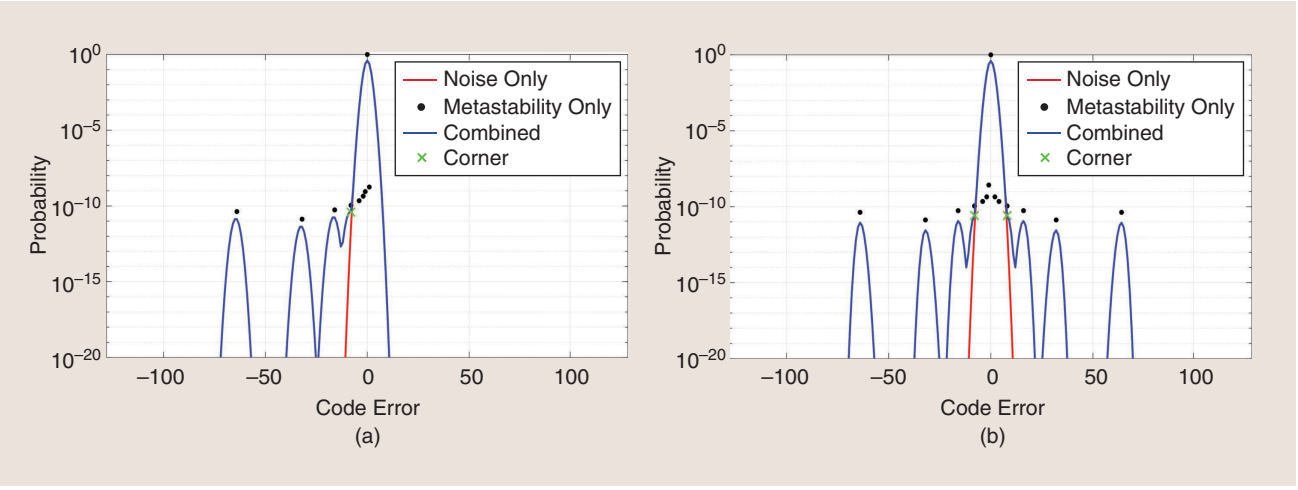


FIGURE 10: Code error PMFs due to metastability and thermal noise combined. Noise-only and metastability-only distributions are overlaid for reference. An 8-b ADC, with $N = 25$ and ENOB = 6, digitizing a Gaussian input signal with clipping probability 10^{-4}. Distributions for (a) bottom-plate sampling and (b) top-plate sampling.

their corresponding code errors completely characterizes the code error distribution in the presence of metastability and noise for a deterministic input, which is readily extended to an arbitrary input distribution using total probability.

Figure 10(a) and (b) shows the code error distributions in the presence of metastability and noise for the bottom-plate and top-plate sampling DAC switching schemes. In both cases, for low-magnitude code errors, the combined distribution behaves as if only noise were present. Then, at a code error magnitude that we can call the *metastability corner*, the combined distribution breaks away from the noise-only distribution and shows an interesting interaction between metastability and noise. The combined distribution resembles the convolution of the noise-only and metastability-only distributions. From afar, the two phenomena look like they interact as independent noise sources that add together, resulting in the convolution of their code error distributions. However, it can be shown analytically that the peaks of the combined distribution are shifted down with respect to the naïve convolution by a factor of about 1.5 for bottom-plate sampling and a factor of about 2 for top-plate sampling. In a sense, thermal noise has a dithering effect on metastability errors. The probability mass spreads out around the power-of-2 code errors exhibited by metastability alone, resulting in less probability mass on these power-of-2 codes.

Note that the combined code error distribution for bottom-plate sampling is still biased, despite the addition of thermal noise to the model. Assuming a coin flip at the capture latch due to a metastable comparator output would not have produced this result, but instead something symmetric. Here, we have viewed the cascade of the comparator and the capture latch as a multistage amplifier and referred all of its noise to its input. This is an equivalent and valid way of viewing the comparator-to-capture latch interface, and also makes the analysis tractable. For high-resolution instrumentation applications where synchronous SAR ADCs are frequently used, this biased code error distribution may have important implications.

Coming back to the metastability corner, it's interesting to think about how it moves as a function of parameters such as N and ENOB, which we illustrate in Figure 11 for top-plate sampling. A smaller N, i.e., less time available for regeneration, should move the "metastability floor" higher, hence moving the corner to lower code

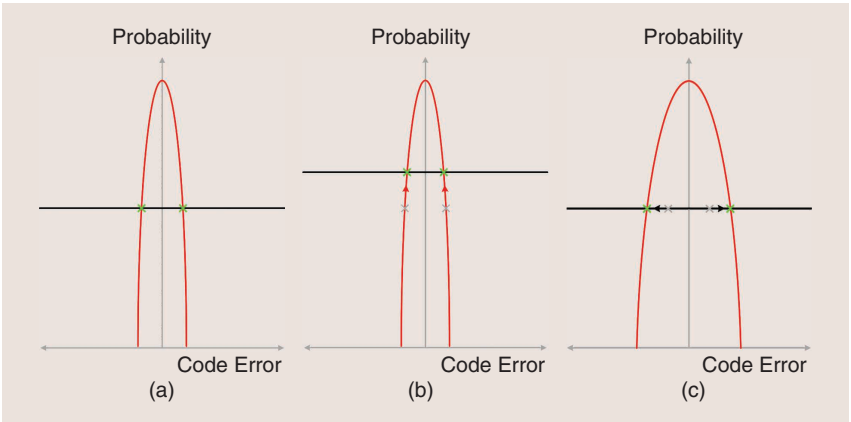


FIGURE 11: The thermal noise code error distribution (red) and "metastability floor" (black). (a) The baseline. (b) A decreasing N, i.e., allocating less time for regeneration, shifts the metastability floor exponentially upward in probability. Following the thermal noise code error distribution, the metastability corner then shows up at a lower code error. (c) A decreasing ENOB, i.e., increasing thermal noise power, broadens the code error distribution due to thermal noise. Following the metastability floor, the metastability corner then shows up at a larger code error.

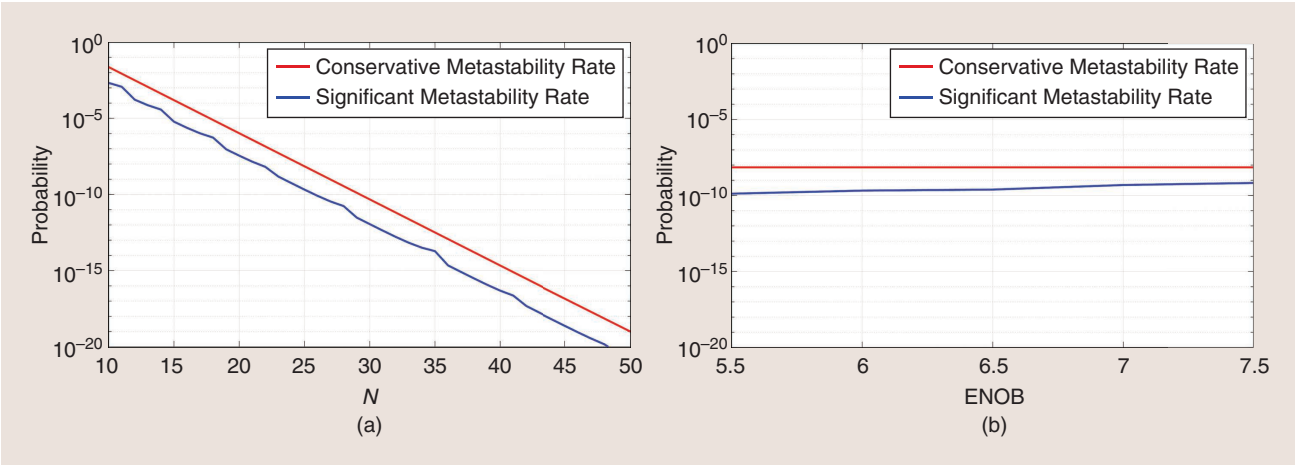


FIGURE 12: An 8-b ADC with top-plate sampling. A conservative metastability rate (the probability of any metastability error) and the significant metastability rate (probability of code error larger than thermal noise) as a function of (a) *N* and (b) ENOB.

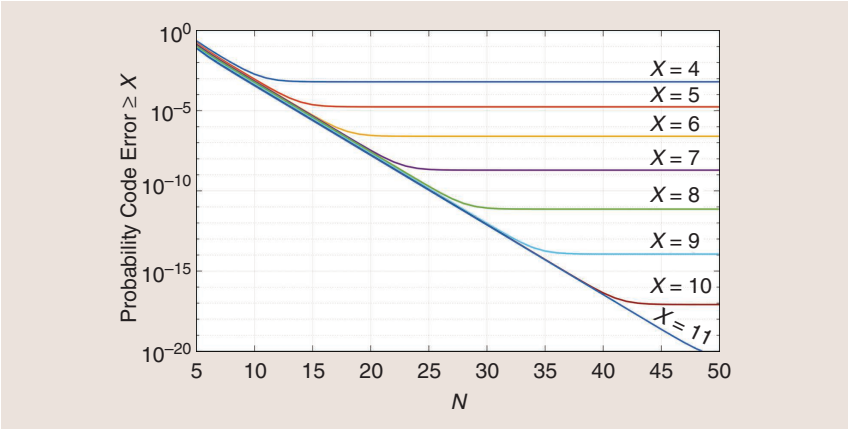


FIGURE 13: The probability of code errors exceeding a certain amount (*X*) as a function of *N*. An 8-b ADC with 6 ENOB and top-plate sampling, digitizing a Gaussian input signal with clipping probability 10^{-4}. Given a desired code error ceiling and the tolerable probability of crossing it, this plot allows the designer to read off the required regeneration time in units of latch time constants.

errors and higher probability [Figure 11(b)]. A smaller ENOB won't affect the metastability floor, but it will move the corner out to larger code errors [Figure 11(c)]. To make this quantitative, one number worth considering is the probability of a code error beyond the metastability corner. This is a more refined estimate of the metastability rate of the ADC. It tells us the probability of a significant metastability error—one that appears above the thermal noise. Figure 12 plots the probability of a significant metastability error with the conservative metastability rate $P_{\text{meta}} = 2(2^B - 1)e^{-N}$ superimposed. The probability of a significant metastability error is about an order of magnitude lower than the conservative metastability rate suggests. On the scale from 10^{-5} to 10^{-20}, one order of magnitude isn't particularly huge; on the other hand, when we think about this in terms of mean time to failure, this more refined estimate is an appealing proposition.

A question that arises frequently in practice is "If I want the probability of code errors greater than X to be smaller than 10^{-Y}, what value of N do I need?" The model described here can be used to answer this question, with an example shown in Figure 13 for an 8-b ADC with 6 ENOB. Each curve in the family has two regimes, corresponding to limits imposed by metastability and thermal noise. Pick a code error X and imagine that the time available for regeneration is so long (i.e. large N) that the metastability floor appears below 10^{-20}. The total probability of code errors greater than X in this case is set by thermal noise and is independent of small changes in N, corresponding to the flat thermal noise asymptote. Now imagine that N is decreased until the metastability corner appears exactly on top of the code error X. This is the value of N at the transition between the flat thermal noise asymptote and the metastability asymptote. Finally, imagine further decreasing N. In this case, the probability of code errors greater than X is set by the metastability floor, which varies as e^{-N}. The bunching up of metastability asymptotes for different code errors X is to be expected because the difference in probability mass under the metastability floor between code errors is very small relative to the global shift in the metastability floor brought on by changing N.

Summary

In this article, we developed a combined framework for thinking about metastability and thermal noise in the synchronous SAR ADC. The key takehomes can be summarized as follows:

- Metastability is a deterministic phenomenon. To make sense of the way metastability and noise actually interact, we must first understand the deterministic behavior of metastability in isolation. Introducing coin-flips provides us with no way to tell how increasing or decreasing the noise amplitude makes a difference.

- As armchair probabilists, we can compute probability distributions by partitioning the sample space into a finite (and tractable) set of events, iterating through each one, calculating its probability, and incrementing the probability of the corresponding random variable value by this amount.
- The method described in the previous item works for computing the code error distribution for the three cases we studied: metastability in isolation, noise in isolation, and the two combined.
- The framework for analyzing noise in SAR ADCs presented in [9] is extremely versatile. It can be extended from deterministic inputs to random inputs using total probability, and it can be extended to model metastability by treating each decision as a four-way outcome (fast/slow and 1/0).
- The code error PMF in the combined framework follows the thermal noise PMF at low magnitude code errors. At the metastability corner, the PMF breaks off from the noise limit and extends to high magnitude code errors characteristic of metastability.
- The combined PMF can be approximated by convolving the noise-only and metastability-only PMFs. In this sense, noise and metastability interact as if they are independent noise sources combining additively.

References

[1] B. Murmann, "The successive approximation register ADC: A versatile building block for ultra-low-power to ultra-high-speed applications," *IEEE Commun. Mag.*, vol. 54, no. 4, pp. 78–83, Apr. 2016. doi: 10.1109/MCOM.2016.7452270.

[2] L. Kull et al., "A 24-to-72GS/s 8b time-interleaved SAR ADC with 2.0-to-3.3pJ/conversion and >30 dB SNDR at nyquist in 14-nm CMOS FinFET," in *Proc. 2018 IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, 2018, pp. 358–360.

[3] Y. Shu, L. Kuo, and T. Lo, "27.2 an oversampling SAR ADC with DAC mismatch error shaping achieving 105dB SFDR and 101dB SNDR over 1kHz BW in 55nm CMOS," in *Proc. 2016 IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, 2016, pp. 458–459.

[4] R. Ginosar, "Fourteen ways to fool your synchronizer," in *Proc. Ninth Int. Symp. Asynchronous Circuits and Systems, 2003*, Vancouver, BC, Canada, 2003, pp. 89–96.

[5] S. Beer and R. Ginosar, "Eleven ways to boost your synchronizer," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1040–1049, June 2015. doi: 10.1109/TVLSI.2014.2331331.

[6] Y. Duan and E. Alon, "A 6b 46GS/s ADC with >23GHz BW and sparkle-code error correction," in *Proc. 2015 Symp. VLSI Circuits (VLSI Circuits)*, Kyoto, Japan, 2015, pp. C162–C163.

[7] J.-E. Eklund and C. Svensson, "Influence of metastability errors on SNR in successive-approximation A/D converters," *Analog Integr. Circuits Signal Process.*, vol. 26, no. 3, pp. 183–190, 2001. doi: 10.1023/A:1008387223956.

[8] A. Waters, J. Muhlestein, and U.-K. Moon, "Analysis of metastability errors in conventional, LSB-first, and asynchronous SAR ADCs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 11, pp. 1898–1909, Nov. 2016. doi: 10.1109/TCSI.2016.2594256.

[9] W. P. Zhang and X. Tong, "Noise modeling and analysis of SAR ADCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 12, pp. 2922–2930, Dec. 2015. doi: 10.1109/TVLSI.2014.2379613.

[10] W. Kester, "Find those elusive ADC sparkle codes and metastable states," *Application Note of Analog Devices Tutorial MT-011*, pp. 1–10, 2009.

[11] J. L. McCreary and P. R. Gray, "All-MOS charge redistribution analog-to-digital conversion techniques. I," *IEEE J. Solid-State Circuits*, vol. 10, no. 6, pp. 371–379, Dec. 1975. doi: 10.1109/JSSC.1975.1050629.

[12] C.-C. Liu, S.-J. Chang, G.-Y. Huang, and Y.-Z. Lin, "A 10-bit 50-MS/s SAR ADC with a monotonic capacitor switching procedure," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 731–740, Apr. 2010. doi: 10.1109/JSSC.2010.2042254.

[13] S. Hashemi and B. Razavi, "Analysis of metastability in pipelined ADCs," *IEEE J. Solid-State Circuits*, vol. 49, no. 5, pp. 1198–1209, May 2014. doi: 10.1109/JSSC.2014.2305075.

About the Authors

Daniel Bankman (dbankman@stanford.edu) received his S.B. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 2012 and his M.S. degree from Stanford University, California, in 2015. He is a Ph.D. candidate in the Department of Electrical Engineering, Stanford University advised by Prof. Boris Murmann. His research focuses on mixed signal processing circuits, hardware architectures, and neural architectures capable of bringing machine learning closer to the energy limits of scaled semiconductor technology. He has held internship positions at Analog Devices, Lyric Labs, and Intel AI Research, and he served as the instructor for EE315 Analog-Digital Interface Circuits at Stanford University in 2018.

Andrew Yu (acyu@mit.edu) received his B.S. and M.S. degrees in electrical engineering from Stanford University, California, in 2017 and 2018, respectively. He is currently a Ph.D. student in electrical engineering and computer science at the Massachusetts Institute of Technology, Cambridge, advised by Prof. Max Shulaker. His research focuses on system-level design and fabrication of monolithic 3D integrated circuits that leverage new nanomaterials, including carbon nanotubes and resistive random-access memory. In 2017, he interned in the SerDes Technology Group at Xilinx, Inc., San Jose, California.

Kevin Zheng (kevinz@xilinx.com) received his B.S. and M.E. degrees in electrical engineering and computer science in 2012 and 2013 from the Massachusetts Institute of Technology, Cambridge, and his Ph.D. degree in electrical engineering in 2018 from Stanford University, California. In 2013, he received the J. Francis Reintjes Excellence in VI-A Industrial Practice Award and the David Adler Memorial EE M.Eng. Thesis Award. He is currently a staff mixed-signal design engineer in the SerDes Technology Group at Xilinx, Inc.

Boris Murmann (murmann@stanford.edu) received his Dipl.-Ing. (FH) in communications engineering from Fachhochschule Dieburg, Germany, in 1994, his M.S. degree in electrical engineering from Santa Clara University, California, in 1999, and his Ph.D. degree in electrical engineering from the University of California, Berkeley, in 2003. He is a full professor with the Department of Electrical Engineering, Stanford University, California. His research interests include mixed-signal integrated circuit design, with special emphasis on data converters, sensor interfaces, and circuits for embedded machine learning. He was a corecipient of the Best Student Paper Award at the Very Large-Scale Integration (VLSI) Circuits Symposium in 2008 and a recipient of the Best Invited Paper Award at the IEEE Custom Integrated Circuits Conference in 2008, the Agilent Early Career Professor Award in 2009, and the Friedrich Wilhelm Bessel Research Award in 2012. He served as an associate editor of *IEEE Journal of Solid-State Circuits*. He is a Fellow of the IEEE.