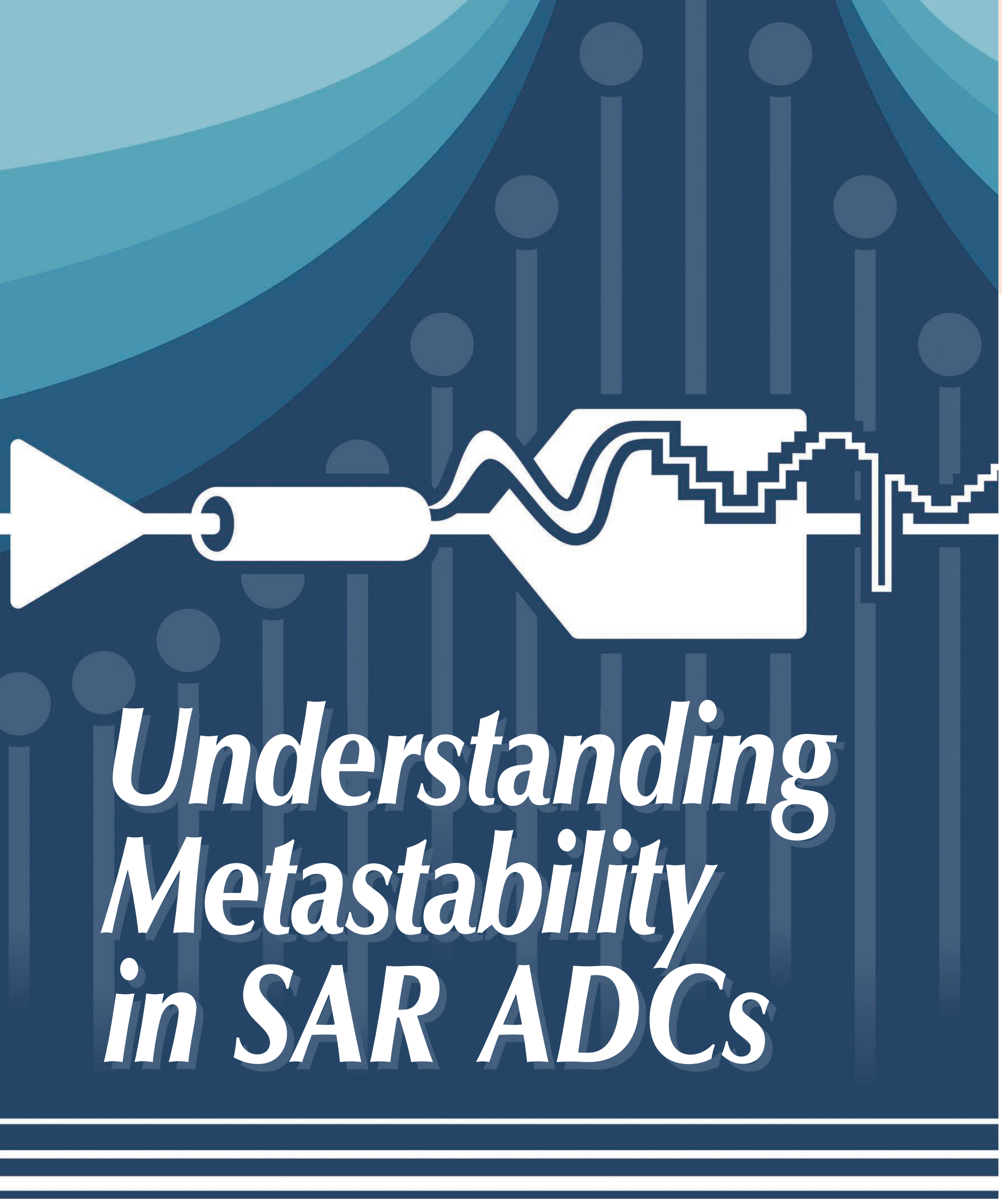
Andrew Yu, Daniel Bankman, Kevin Zheng, and Boris Murmann



Understanding Metastability in SAR ADCs

Part II: Asynchronous

This article is the second part of our tutorial on metastability in successive approximation register (SAR) analog-to-digital converters (ADCs). Having covered the synchronous topology in Part I in the Spring 2019 issue of *IEEE Solid-State Circuits Magazine*, we now investigate its asynchronous (or self-timed) counterpart. Even though the idea of successive approximation is rather old, it was not until 2006 that the benefits of asynchronous timing were unleashed through a seminal contribution by Chen et al. [1]. Since then, the asynchronous variant has emerged as a very popular option to achieve higher speeds and tame metastability issues more gracefully. The latter aspect is the focus of this article.

We begin by examining the differences that arise from asynchronous timing and then enumerate their impact on the metastability rate and metastability error distribution. As in Part I, we first consider the noiseless operation and then look at the combined effects of thermal noise and metastability. Next, we apply the derived models to an ADC-based serial link application and finish with remarks about metastability detectors.

The material in this article is most closely related to the works of Cai et al. [2] and Waters et al. [3]. However, just as in Part I, our distinguishing features are our combined treatment of noise and metastability and our framework for computing the combined error probability mass function (PMF).

Digital Object Identifier 10.1109/MSSC.2019.2922890
Date of publication: 27 August 2019

From Synchronous to Asynchronous

In the synchronous architecture, one clock period T_{CLK} is allocated to each bit cycle of the successive approximation algorithm. This clock period must be long enough to accommodate both the regeneration-independent delay in the SAR loop T_{FIX} [SAR logic, digital-to-analog converter (DAC) settling, buffers, and so on] and the comparator regeneration time. If one clock period cannot accommodate both of these delays in any given bit cycle, a metastability error occurs. The number of comparator time constants (τ) that fit into the time available for regeneration per bit cycle is enough information to fully characterize the metastability behavior of the synchronous SAR ADC. We refer to this timing parameter as $N=(T_{\mathrm{CLK}}-T_{\mathrm{FIX}})/\tau$. As explained in Part I, assuming a uniform input signal distribution, the probability of a metastability error is $P_{\mathrm{meta}}=2(2^{B}-1)e^{-N}$, where B is the number of bits.

As the comparator input becomes arbitrarily small, its regeneration time becomes arbitrarily large. However, during how many bit cycles can the comparator input actually become arbitrarily small? The answer is only one of them. During successive approximation, the input signal is tested against B DAC voltages, but it can reside within ±0.5 least significant bits (LSBs) of at most one DAC voltage. This is often called the *hard* decision, because it can cause an arbitrarily long regeneration time. All other decisions are *easy*, in the sense that regeneration happens quickly for comparator inputs with magnitude greater than 0.5 LSBs. In designing the synchronous SAR ADC, we must set the value of N assuming that any decision could be the hard decision. This is, in fact, quite wasteful, because, in all bit cycles but one, the time available for regeneration is overprovisioned. Rather than waiting through a full T_{CLK}, it would be more efficient to proceed from one bit cycle to the next immediately after the regeneration and fixed delays have elapsed. This is the basis for the asynchronous architecture.

Figure 1(a) shows a block diagram of the asynchronous architecture, and Figure 1(b) compares its timing to that of its synchronous cousin. The main implementation difference is in the SAR logic, which must now detect when the comparator resolves, capture its output, switch the DAC, reset the comparator, and kick off the next decision, all without relying on an externally applied clock. In a real implementation, DAC settling and comparator reset happen concurrently. The longer of these two delays determines the timing, which we simply refer to as T_{FIX}, the regeneration-independent delay in the SAR loop.

In the synchronous architecture, we allocate the same timing margin for metastability in each bit cycle, color-coded as red in Figure 1(b). If

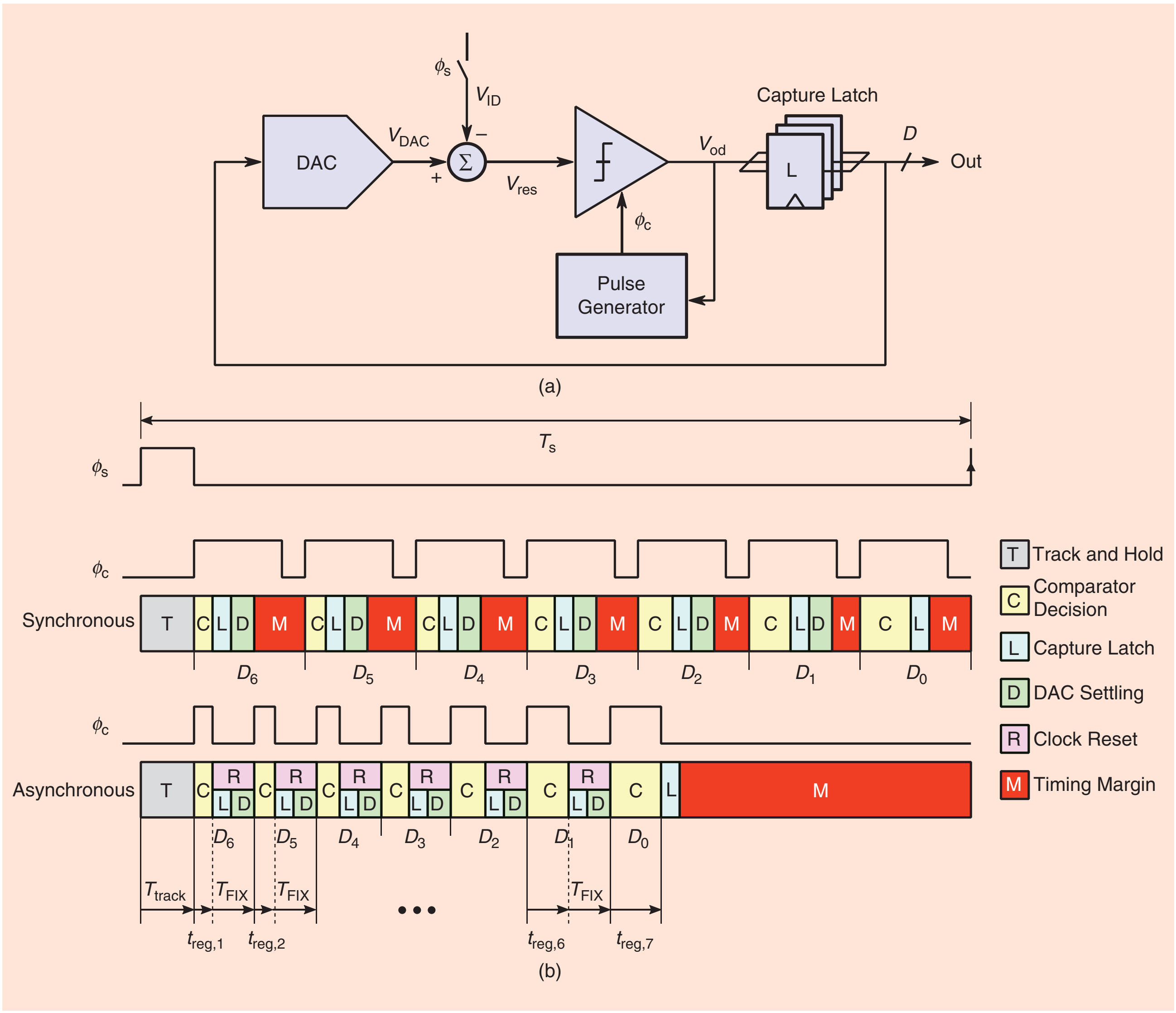


FIGURE 1: (a) An asynchronous SAR ADC block diagram. (b) A timing diagram comparison between the synchronous and asynchronous topologies. While a synchronous design allocates timing margin to each bit cycle individually, an asynchronous scheme allocates a single block of timing margin for all bit cycles to share. v_{od}: differential output voltage; D: decision.

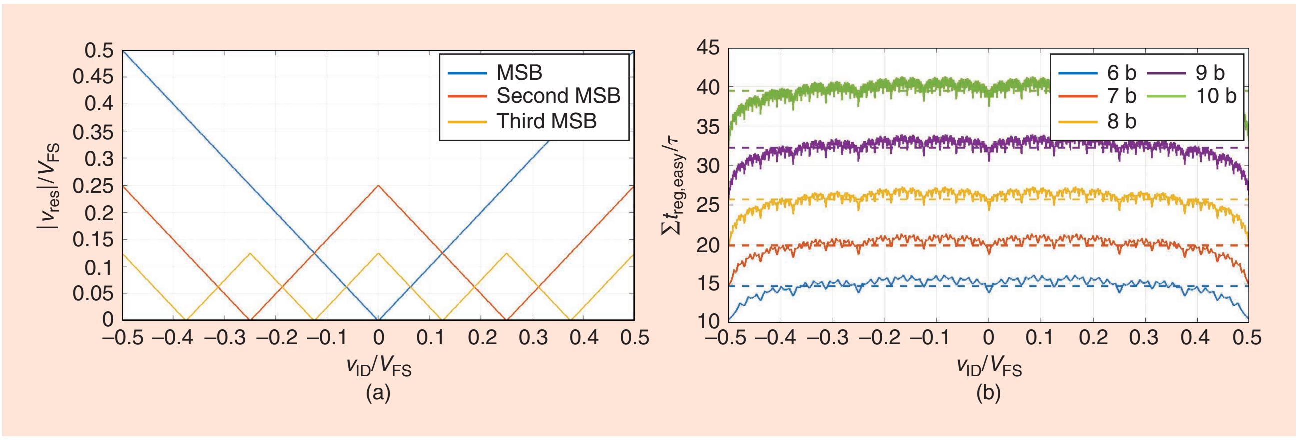


FIGURE 2: (a) A plot of residual voltage during the MSB, second MSB, and third MSB decisions as a function of ADC input voltage. Given the input voltage, we know the residual voltage for every decision, and therefore we know the regeneration time. (b) The total regeneration time of easy decisions as a function of ADC input voltage (solid) and average value (dashed).

a bit cycle doesn't use up its timing margin (because the comparator input happens to be large), this time is wasted. Similarly, if a bit cycle exhausts its timing margin (because the comparator input happens to be small), then a metastability error occurs. Timing margin cannot be borrowed from the subsequent bit cycle, even though all other decisions are guaranteed to be easy. In contrast, the asynchronous architecture allocates a single block of timing margin for metastability, which any individual bit cycle can use as needed. We would expect that for a fixed metastability probability (P_{meta}), we can run the ADC faster, while for a fixed speed, we can achieve a much lower P_{meta}. How can we calculate P_{meta} for the asynchronous architecture?

We can answer this question by finding the available regeneration time for the hard decision, converting this to a range of residual voltages (i.e., the comparator input) that cause the hard decision to run out of time, and then finding the probability that the voltage lies in this range. To find the available time, we consider the total regeneration time of the easy decisions. An easy decision must have a residual voltage greater than 0.5 LSBs in magnitude, so its regeneration time must satisfy

$$t_{reg,easy} < \tau \ln\left(\frac{V_{DD}}{\frac{1}{2}\frac{V_{DD}}{2^B}}\right) = \tau \ln(2^{B+1}).$$

While the total regeneration time of all easy decisions cannot exceed $B-1$ times the bound on the right-hand side, this corresponds to the impossible scenario in which the comparator input is 0.5 LSBs for every decision. Figure 2(a) shows the residual voltage for the three most significant bit (MSB) decisions as a function of the ADC input v_{ID}, which is all we need to calculate a better estimate. We're interested only in the total regeneration time of easy decisions, so, for each decision, we ignore the range of inputs v_{ID} that would bring its residual voltage within ±0.5 LSBs of zero (corresponding to that particular decision being hard). Summing up the regeneration times, we obtain the plot in Figure 2(b), which shows the total regeneration time of easy decisions as a function of v_{ID} for several ADC resolutions.

At this point, we invoke the assumption that the ADC input v_{ID} is uniformly distributed across the full-scale range. This key assumption of the article is applied to the derivations and results that follow. However, for any reasonably well-behaved input signal distribution, we expect similar results. The dashed lines in Figure 2(b) represent the average value of the total regeneration time of all the easy decisions, which we denote as T_{easy}/τ in our subsequent analysis. Interestingly, T_{easy}/τ can be hand-calculated (see "Calculating T_{easy}"), giving the values summarized in Table 1.

The conversion period of the ADC, T_s, must be long enough to accommodate the track-and-hold interval T_{track}, $B-1$ fixed delays, T_{easy}, and the regeneration time of the hard decision $t_{reg,hard}$. If the regeneration time of the hard decision exceeds the time allocated to it,

$$t_{reg,hard} > T_s - T_{track} - (B-1)\,T_{FIX} - T_{easy},$$

then a metastability error occurs. Let us now write the timing parameter N in a way that works for both synchronous and asynchronous approaches, by replacing $T_{CLK} - T_{FIX}$ from the synchronous case

TABLE 1. THE TOTAL REGENERATION TIME OF EASY DECISIONS AVERAGED OVER FULL-SCALE RANGE FOR 6–14-B RESOLUTION.

B	T_{easy}/τ
6	15
7	20
8	26
9	32
10	39
11	47
12	56
13	65
14	75

with $(T_s - T_{\text{track}} - T_{\text{FIX,TOT}})/B$, a more general way of spelling out the time available for regeneration per bit cycle [$T_{\text{FIX,TOT}}$ is the total fixed delay per conversion: BT_{FIX} for synchronous and $(B-1)T_{\text{FIX}}$ for asynchronous]:

$$N = \frac{(T_s - T_{\text{track}} - T_{\text{FIX,TOT}})/B}{\tau}$$

The range of residual voltages for the hard decision leading to a metastability error is then

$$|v_{\text{res,hard}}| < V_{\text{DD}}\, e^{-(BN - T_{\text{easy}}/\tau)}.$$

The residual voltage on the hard decision lies within ±0.5 LSBs, and this is, in fact, the quantization error of the ADC. For well-behaved inputs, the quantization error can be treated as uniformly distributed. The probability of a metastability error is then the amount of quantization error probability mass that falls within the metastability window given previously. For the asynchronous architecture, we have $P_{\text{meta}} = 2^{B+1} e^{-(BN - T_{\text{easy}}/\tau)}$.

In the synchronous architecture, the time available for regeneration per bit cycle N determines the metastability rate $P_{\text{meta}} = 2(2^B - 1)e^{-N}$. In the asynchronous architecture, by contrast, the metastability rate is instead set by the total time available for regeneration from all bit cycles (BN) minus the time needed to regenerate the easy decisions T_{easy}/τ (also a function of B). This is the mathematical manifestation of the design principle "allocate a single block of timing margin for metastability that individual bit cycles can eat into as needed." Figure 3 plots P_{meta} for both synchronous and asynchronous architectures as a function of N for 6–8-b resolution. The key difference on the log scale is that P_{meta} for the asynchronous architecture drops B times faster than P_{meta} for the synchronous case. Let's now get a sense for the raw numbers. At 8-b resolution and $N = 25$ comparator time constants available for regeneration per bit cycle, P_{meta} with synchronous timing is 7.1×10^{-9}, whereas P_{meta} with asynchronous timing is 1.4×10^{-73}. This demonstrates that, at the same speed ($N = 25$), P_{meta} is dramatically lower for the asynchronous architecture. Similarly, at 8-b resolution and $P_{\text{meta}} = 10^{-12}$, the asynchronous architecture runs at $N = 7.5$, more than four times faster than the synchronous architecture, which runs at $N = 33.9$.

In Part I, we termed P_{meta} the *conservative* metastability rate, because P_{meta} is the probability of any metastable event. However, not all metastable events are equally catastrophic in terms of the resulting ADC output code errors. In practice, an answer to the question "If I want the probability of code errors greater than X to be smaller than 10^{-Y}, what value of N do I need?" can be more valuable than just P_{meta}. To answer this question, we must derive the PMF over ADC output code errors for the asynchronous architecture. The end goal is to derive this distribution considering both metastability and thermal noise. Similar to our process in Part I, we begin by building a model in which the ADC input signal is random, but the response of the SAR loop is completely

Calculating T_{easy}

In Figure 2(a), the absolute value of the residual for each bit cycle i (where $i = 1, 2, \ldots B$) forms a triangle wave with period $V_{\text{FS}}/2^{i-1}$ and peak-to-peak value $V_{\text{FS}}/2^i$. Noting that each period is identical and assuming a uniform input signal distribution, the average regeneration time of bit cycle i is the average of one of these periods centered at zero.

To find average $t_{\text{reg,easy},i}$ per bit cycle, we calculate the average of the region where the comparator input $> \pm 0.5$ LSB, giving

$$t_{\text{reg,easy},i}/\tau = \frac{1}{\left(\frac{V_{\text{FS}}}{2^{i-1}}\right)}\left[\int_{-\frac{V_{\text{FS}}}{2^i}}^{-\frac{V_{\text{FS}}}{2^{B+1}}} \ln\left(-\frac{V_{\text{DD}}}{x}\right)dx + \int_{\frac{V_{\text{FS}}}{2^{B+1}}}^{\frac{V_{\text{FS}}}{2^i}} \ln\left(\frac{V_{\text{DD}}}{x}\right)dx\right]$$
$$= 1 - 2^{i-B-1} + [i - (B+1)2^{i-B-1}]\ln(2) + (1 - 2^{i-B-1})\ln\left(\frac{V_{\text{DD}}}{V_{\text{FS}}}\right).$$

The average T_{easy} for B bits is thus

$$\frac{T_{\text{easy}}}{\tau} = \frac{1}{\tau}\sum_{i=1}^{B} t_{\text{reg,easy},i} \approx [1 + \ln(2)]\,(2^{-B} - 1) + B\left[1 + \left(2^{-B} - \frac{1}{2}\right)\ln(2)\right] + \frac{B^2\ln(2)}{2}$$

where we assume $V_{\text{FS}} \approx V_{\text{DD}}$.

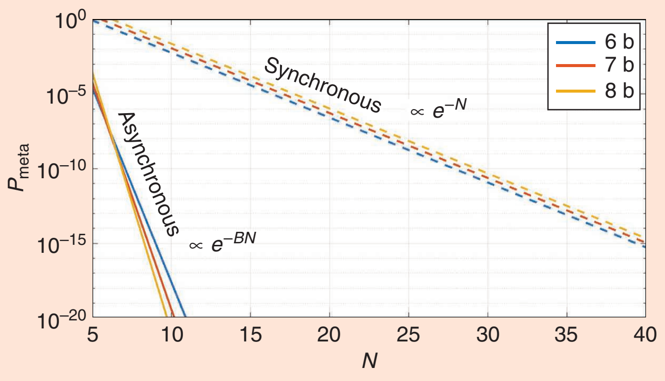


FIGURE 3: A graph of the probability of a metastable event (P_{meta}) versus timing parameter N for asynchronous and synchronous architectures and 6–8-b resolution. On the log scale, P_{meta} drops B times faster with asynchronous timing.

The asynchronous architecture allocates a single block of timing margin for metastability, which any individual bit cycle can use as needed.

deterministic, i.e., it excludes comparator input-referred thermal noise. We then augment this model to include the effects of comparator noise.

Metastability Code Error Distribution With Random Input Signal and Noiseless Comparator

In the absence of noise, the ADC input v_{ID} determines the entire timing of the asynchronous SAR ADC. We invoked similar reasoning in the previous section, when we used Figure 2 to add up the regeneration times of easy decisions for all inputs across the full-scale range. Given v_{ID}, we can calculate the DAC voltage in each bit cycle. It's just a matter of how long each bit cycle takes and whether there's time to finish all cycles before the end of the conversion period T_s. How long the ith bit cycle takes is determined by the residual voltage $v_{res,i} = v_{DAC,i} - v_{ID}$, the difference between the DAC voltage (determined by v_{ID}) and v_{ID} itself. Given v_{ID}, we can calculate how long the comparator spends regenerating each decision. Between the end of one decision and the beginning of the next, there is the fixed delay T_{FIX}. We can draw a timing diagram showing the self-timed comparator enable signal as well as the comparator output voltage, all based strictly on knowledge of the ADC input v_{ID}. An example is constructed in Figure 4(a), where v_{ID} resides very close to the bottom of the input range that maps to the code 101.

Let's walk through the timing diagram in Figure 4(a) step by step. During the track interval (ϕ_S), the sampling signal is asserted, and the ADC output code is reset (to 000 in this example). Immediately following the track interval, the comparator enable signal (ϕ_C) is asserted, beginning the regeneration time for the MSB decision. During this time, the comparator output is in a metastable state, denoted as X. The ADC output code is X00. Then, the comparator finishes regenerating a logic 1. Immediately following regeneration, the comparator is reset, and the fixed delay begins. In the course of this time, the ADC output code is 100. At the end of the fixed delay, the comparator enable signal is asserted again, beginning the regeneration time for the second MSB decision. The ADC output code is now 1X0. The comparator soon resolves a logic 0, the ADC output code switches to 100, and another fixed delay begins. Finally, the comparator enable signal is asserted for the LSB decision, and the ADC output code switches to 10X. Notice, though, that the comparator does not resolve the LSB decision by the end of the conversion period. A metastability error is born.

Now imagine that v_{ID} is not close enough to the 100–101 transition level to cause a metastability error but instead resides right in the middle of the code 101. We can use the dark red, dark green, and blue timing bands in Figure 4(a) to find out what happens to the regeneration times of the MSB, second MSB, and LSB decisions, respectively. Recall that, given v_{ID}, all regeneration times are determined. The widths of the dark timing bands represent the regeneration times, and those of the light timing bands represent the fixed delays (notice how the fixed delay widths are constant). If v_{ID} moves from the bottom to the middle of code 101, the MSB regeneration time decreases, the second MSB regeneration time increases, and the LSB regeneration time decreases significantly. Now there's enough time for the comparator to resolve the LSB decision before the end of the conversion period.

The timing bands can, of course, be extended from code 101 to the entire full-scale range, depicted in Figure 4(b). It's just important to remember that the ADC output code inside each band is different. For example, in code 010, the timing bands would instead be labeled (left to right) 000, X00, 000, 0X0, 010, 01X, and finally 010, the ideal ADC output code. We now have a graphical way to find the ADC output code given v_{ID} at any instant in time. There is one instant in time of particular importance, and that is when the conversion ends. In the first example (v_{ID} very close to the bottom of code 101), the ADC output code at the end of conversion is 10X. Now, all we need to do is decide how to treat the X, and we will know the ADC output code error caused by this metastable event.

Following [4] and Part I of this tutorial, we assume that the capture latch output switches in an abrupt but delayed jump. In other words, we assume that the capture latch output stays in its reset state until regeneration is over, at which time it switches instantaneously to the correct value. Under this assumption, X can be replaced by 0 in the ADC output codes of Figure 4(a): 000, 000, 100, 100, 100, 100, 101. With v_{ID} very close to the bottom of code 101 and insufficient time to resolve the LSB decision, the ADC output code at the end of conversion is 100, and the code error is –1.

The method of timing bands gives us a function from v_{ID} to the ADC output code error. How can we use this to find the code error PMF? Imagine drawing a vertical line on Figure 4(b), exactly at time T_s, the end of conversion. Figure 4(c) plots the 1D cross section of the bands at this time. Here, we can easily see the ranges of v_{ID} that cause the conversion to terminate during the MSB regeneration (dark red), first fixed delay (light red), second MSB regeneration (dark green), second fixed delay (light green), and so on. The regions that are not distinctly colored correspond to the conversion terminating after all decisions have regenerated, resulting in zero code error. By integrating the probability mass of v_{ID} in each range and incrementing the probability of the associated code error by this amount, we can construct the ADC output code error PMF (see "Computing the Noiseless Probability Mass Function").

Previously, we found that the probability of *any* metastable event is $P_{\mathrm{meta}} = 2^{B+1} e^{-(BN - T_{\mathrm{easy}}/\tau)}$. T_{easy}/τ depends strictly on B, so P_{meta} is determined entirely by a single design variable: N. For the synchronous architecture, we found that N determined both P_{meta} and the entire shape of the code error distribution. For asynchronous architecture, this is not the case. The shape of the ADC output code error PMF depends on two design variables: N and T_{FIX}/τ, the number of comparator time constants available for regeneration and the number of comparator time constants consumed by fixed delay. Note that, together, these two parameters specify the number of comparator time constants in the total normalized conversion time T_s/τ. To show why the asynchronous code error PMF has a 2D design space instead of a 1D design space, we examine two examples shown in Figure 5.

Let's begin with Figure 5(a), which shows two design points sharing the same value of N. Design 1 has zero fixed delay, and design 2 has nonzero fixed delay. In design 2, the fixed

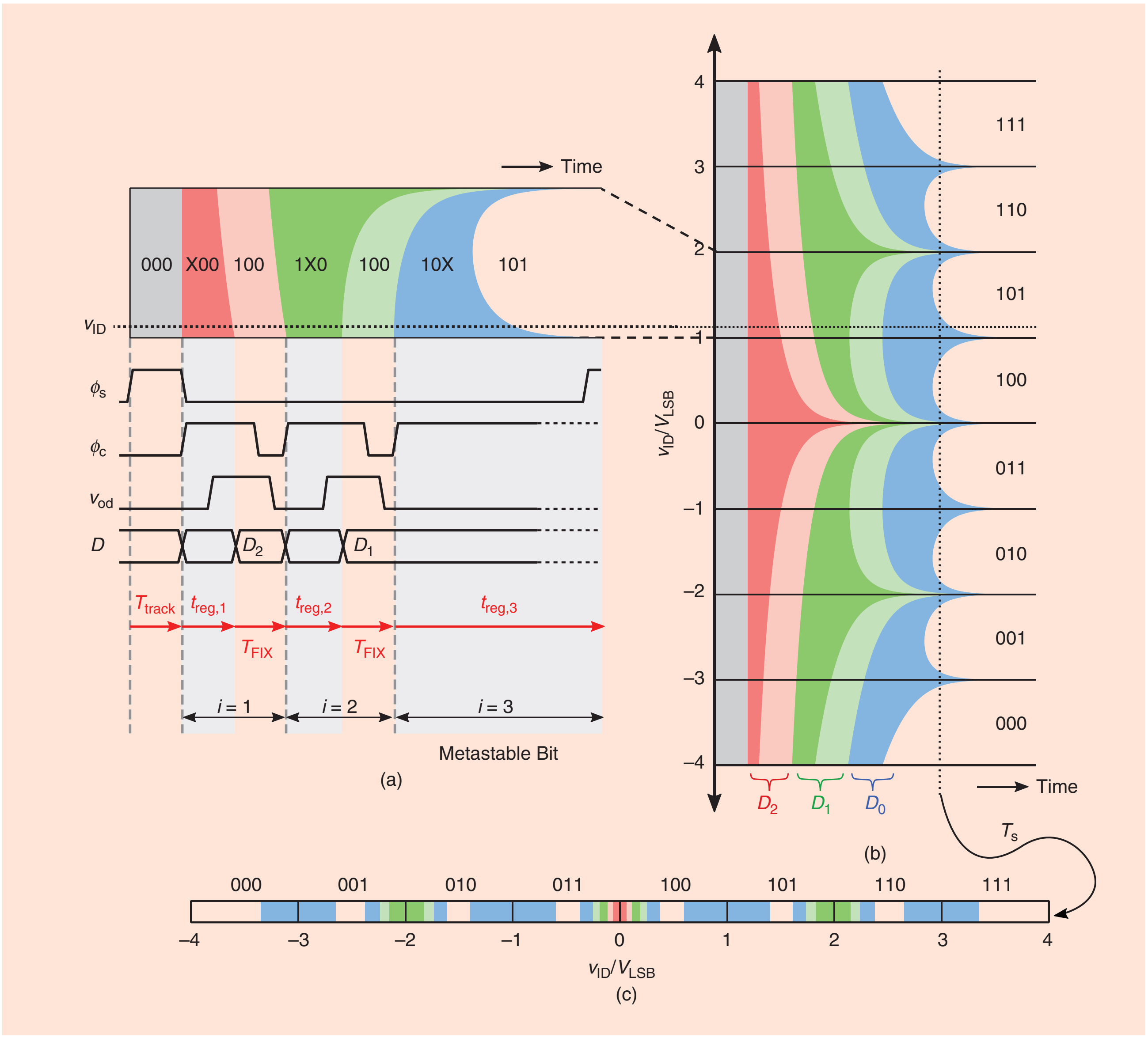


FIGURE 4: (a) A timing diagram and timing bands for a particular input voltage near the bottom of code 101. Given the input voltage, each regeneration time is determined. The timing bands illustrate how these regeneration times change and add together as a function of the input voltage. The 3-b codes indicate ADC output during each particular timing band. (b) The timing bands are extended to cover the full-scale range. Note that the ADC output code during each band changes across the full-scale range. (c) A 1D cross section of timing bands at the end of the conversion time. This breaks the full-scale range down into segments, each corresponding to running out of time during a particular bit cycle and incurring a particular code error. The probability mass of v_{ID} inside each segment is used to increment the probability of the corresponding code error in the PMF.

In comparison with the synchronous architecture, the asynchronous counterpart achieves dramatically better metastability performance for the same conversion time T_s/τ and total fixed delay.

delays are simply added to the total conversion time. From a design perspective, we've allocated the same amount of time for regeneration, which we intend all bit cycles to share. However, the hard decision doesn't see it that way. From the perspective of the hard decision, the added fixed delays of design 2 appear as additional timing margin that can be borrowed (really, stolen) from subsequent bit cycles. MSB errors are less likely to happen simply by virtue of increasing T_s/τ, whether or not the added time is intended to be used for MSB regeneration. This leads to a lower probability for large code errors in design 2, but, because N is kept constant, the total probability of all code errors P_{meta} remains the same. The example shown in Figure 5(b) is easier to interpret. In design 2, fixed delay is introduced by taking time away from the metastability margin in a way that keeps T_s/τ constant. With smaller N, the metastability code error PMF shifts upwards, and P_{meta} increases.

In comparison with the synchronous architecture, the asynchronous counterpart achieves dramatically better metastability performance for the same conversion time T_s/τ and total fixed delay $T_{\text{FIX,TOT}}/\tau$. Not only is the probability of *any* metastable event lower, but also the probability of a particular code error decays much faster as a function of the code error magnitude. The end result is that it is almost impossible to see a large code error in the asynchronous architecture due to the MSB decision running out of time. This is because, to the MSB decision, the entire conversion time is essentially timing margin for metastability, and the probability of running out of time drops exponentially with this margin.

Finally, it is worth thinking about the fact that the metastability code error PMF is asymmetric. Is it true that all metastability code errors are negative? This is simply a byproduct of our choice of capture latch reset code: all zeros. If the hard decision's regeneration time eats up the rest of the conversion time, then all subsequent ADC

Computing the Noiseless Probability Mass Function

Let the analog-to-digital converter (ADC) input voltage be $v_{\text{ID}} \in [-V_{\text{FS}}/2, V_{\text{FS}}/2]$. We partition this input domain into each of the 2^B least significant bit-sized regions with a unique ADC output code, such that each v_{ID} is written as

$$v_{\text{ID}} = k\frac{V_{\text{FS}}}{2^B} + \Delta V,$$

where $k = -2^{B-1}+0.5, -2^{B-1}+1.5, \ldots 2^{B-1}-1.5, 2^{B-1}-0.5$ are half integers, and $\Delta v \in [-(V_{\text{FS}}/2^{B+1}), (V_{\text{FS}}/2^{B+1})]$. The digital-to-analog converter voltage during bit cycle i for each k is given by

$$v_{\text{DAC}}(i,k) = \begin{cases} 0 & , \quad i=0 \\ \dfrac{V_{\text{FS}}}{2^i}\left(\left\lfloor \dfrac{k}{2^{B-i}} \right\rfloor + 0.5\right), & i>0' \end{cases}$$

where $y(x) = \lfloor x \rfloor$ is the floor function. Bit cycles begin at $i=1$, but $i=0$ is included as the initial condition of the ADC before any operations begin. The residual for each bit cycle in each interval is

$$v_{\text{res}}(i,k) = v_{\text{DAC}}(i-1,k) - k\frac{V_{\text{FS}}}{2^B} - \Delta v.$$

The edges of each metastable region are found by numerically solving for Δv such that $T_s - T_{\text{track}} = (i-1)T_{\text{FIX}} + \tau\sum_{n=1}^{i}\ln(V_{\text{DD}}/|v_{\text{res}}(n,k)|)$ for each i and k (solutions may not exist for all i, k).

To calculate code errors, we write the ADC output code after cycle i completes as

$$D_{\text{out}}(i,k) = D_{\text{reset}} - D_{\text{reset},i}(i) + D_{\text{out0}}(i,k).$$

D_{reset} is the reset code. We define a reset code of all 0s as $D_{\text{reset}} = -2^{B-1}$. This expression takes the reset code, subtracts out the first i bits of the reset code given by $D_{\text{reset},i}$, and then adds in the new i bits from the comparator decisions given by D_{out0}, which is calculated by assuming a reset code of all 0s to be

$$D_{\text{out0}}(i,k) = 2^{B-i}\lfloor k/2^{B-i} \rfloor.$$

$D_{\text{reset},i}$ can be calculated from D_{out0} by replacing k with D_{reset}. Combining all expressions, the ADC output code in each cycle i for each k is

$$D_{\text{out}}(i,k) = \begin{cases} D_{\text{reset}} & , \quad i=0 \\ D_{\text{reset}} + 2^{B-i}\left(\left\lfloor \dfrac{k}{2^{B-i}} \right\rfloor - \left\lfloor \dfrac{D_{\text{reset}}}{2^{B-i}} \right\rfloor\right), & i>0 \end{cases}.$$

The code error is analytically given by

$$\epsilon(i,k) = D_{\text{out}}(i,k) - \lfloor k \rfloor,$$

where $\lfloor k \rfloor$ gives the integer value of the ideal output code.

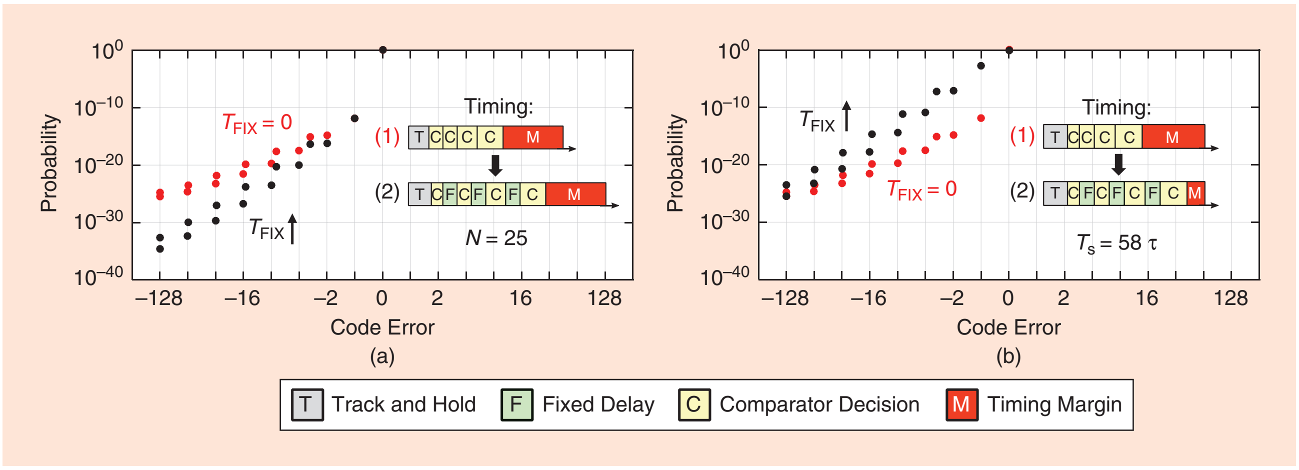


FIGURE 5: The effect of changing T_{FIX}/τ illustrated in metastability code error PMFs. (a) The T_{FIX}/τ parameter is increased while keeping N constant. From the perspective of the MSBs, adding fixed delay is the same as adding timing margin that can be borrowed from the LSBs. Therefore, large code errors caused by running out of time during MSB regeneration become less likely. However, with N kept constant, the total probability of all metastable events (P_{meta}) must remain the same. (b) The T_{FIX}/τ parameter is increased while keeping T_s/τ constant. This reduces N and increases the probability of all metastability errors.

output bits are zeros. The actual ADC output code is always lower than the ideal ADC output code, so all errors are negative. But suppose we prefer metastability code errors not to have a bias; in other words, we prefer a code error PMF with mean close to zero. This can be accomplished by balancing the reset code. Figure 6 shows the code error PMF with reset code equal to 00000000, 11111111, or 01010101. The balanced reset code with alternating zeros and ones leads to a code error PMF with a near-zero mean.

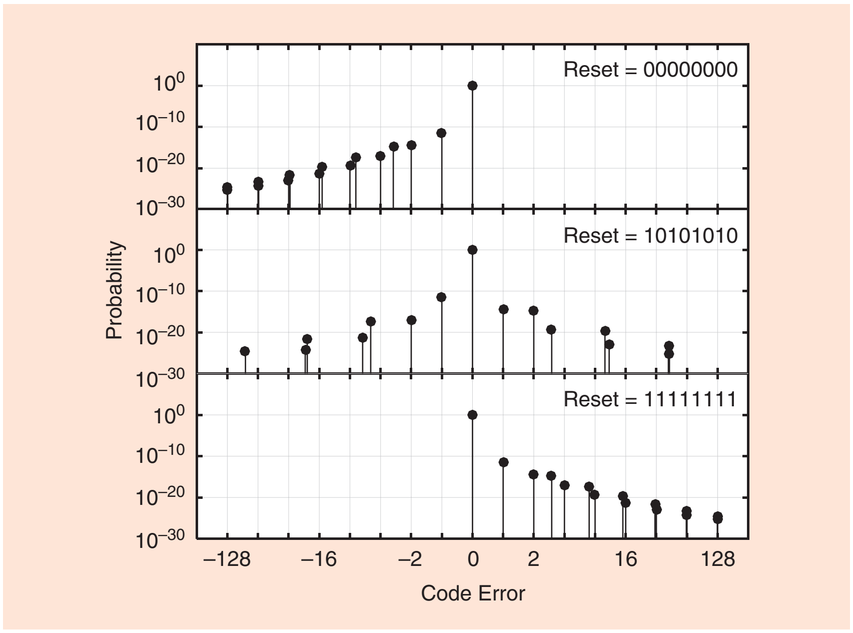


FIGURE 6: A plot of metastability code error PMFs for different reset codes. The balanced reset code (10101010) leads to a distribution with near-zero mean.

Metastability Code Error Distribution With Random Input Signal and Noisy Comparator

Much like that of the synchronous architecture, the model of the asynchronous variant combining comparator noise and metastability can be understood as a straightforward extension of the model for noise in isolation. This noise-only model is applicable to both synchronous and asynchronous architectures because it considers only the correctness of comparator decisions and ignores the timing of the ADC. Let's begin with a review of the model for thermal noise in isolation.

The first step is to approximate the input signal's continuous probability density function (PDF) using a discrete PMF. This works well as long as the step size is much smaller than the thermal noise standard deviation. Then, for each discrete level in the approximate input signal PMF, we enumerate all 2^B possible DAC voltage waveforms. Because of thermal noise, all DAC trajectories are possible, even though some are vanishingly improbable. The probability of a DAC trajectory can be calculated by integrating the distribution of the input plus noise either above or below the DAC voltage at each decision, shown in Figure 7(a) and (b) for a 3-b example. The product of the shaded areas is the probability of that DAC voltage waveform and the associated ADC output code. Finally, we increment the probability of the corresponding code error by this amount (scaled by the probability of the input level), and that's how we build up the code error PMF.

The model combining noise and metastability in the asynchronous architecture starts exactly the same way: iterate over all possible input levels, and, for each level, iterate over all 2^B DAC trajectories. Now we know the probability of the ADC output code we would get if we were willing to wait for all decisions to regenerate. By introducing metastability, we ask, "What happens if the ADC runs out of time?" To answer this question, we need to begin thinking about the regeneration time on each decision as a random variable.

The first thing to observe is that, given an input level and a DAC trajectory, all regeneration times are (conditionally) independent. In Figure 7(a) and (b), the conditional distribution of the residual voltage is the shaded area under each noise distribution (normalized to integrate to unity). With regeneration time related to residual voltage by $t_{\text{reg}}/\tau = \ln(V_{\text{DD}}/|v_{\text{res}}|)$ and assuming Gaussian noise, it can be shown (see "Probability Distribution of Regeneration Time With Noise") that the conditional PDF of one decision's regeneration time is

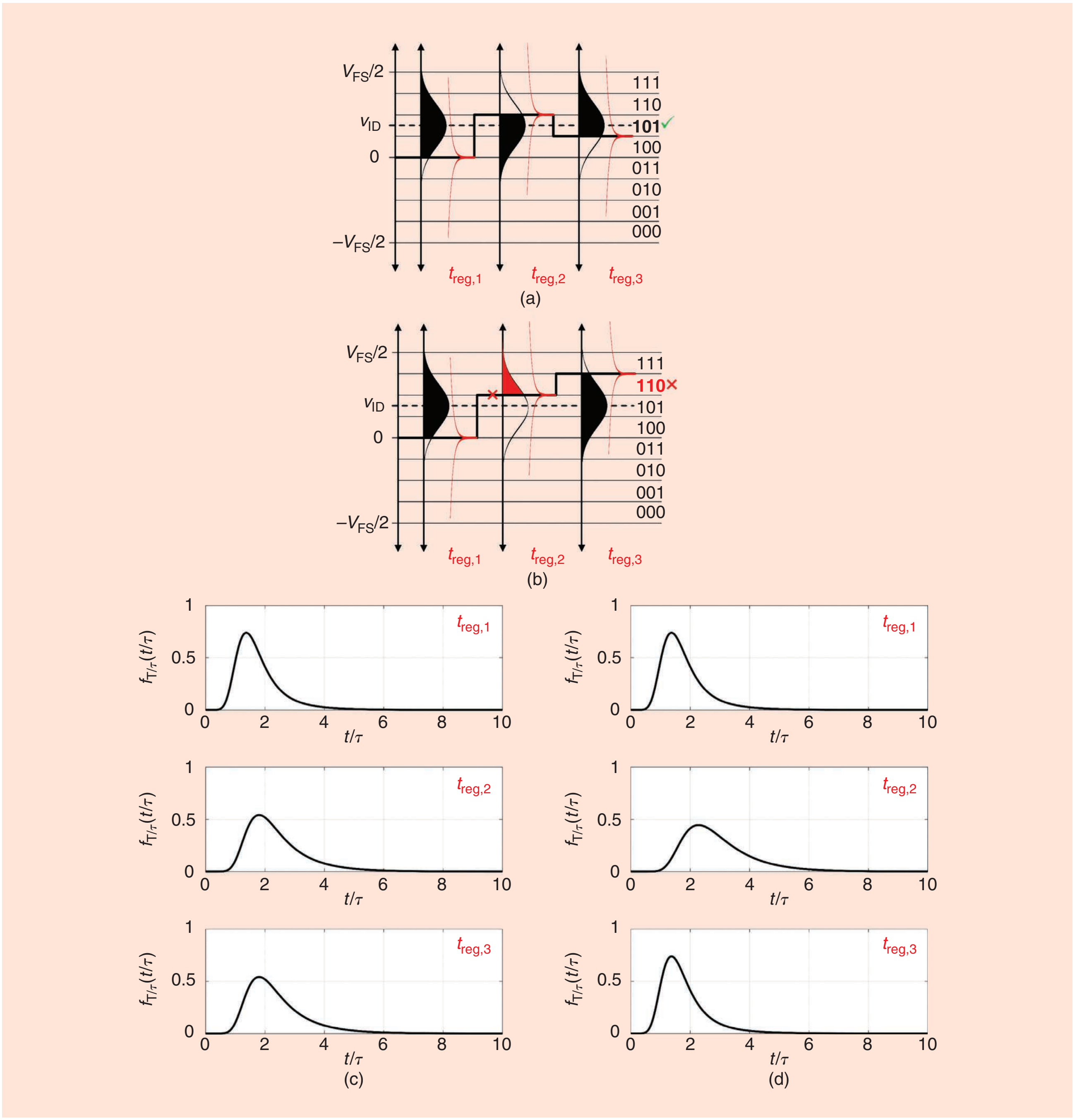


FIGURE 7: A 3-b example showing conditional distribution of (a) and (b) residual voltage and (c) and (d) regeneration time. Given an input voltage and a DAC trajectory, regeneration times are independent random variables.

$$f_{T/\tau}(t/\tau) = \frac{2}{\frac{\sqrt{2\pi}\,\sigma_n}{V_{\mathrm{DD}}}}\left(\frac{e^{-\left(\frac{e^{-\frac{t}{\tau}}\pm(v_{\mathrm{DAC}}-v_{\mathrm{ID}})/V_{\mathrm{DD}}}{\sqrt{2}\,\sigma_n/V_{\mathrm{DD}}}\right)^2-\frac{t}{\tau}}}{1\pm\operatorname{erf}\left(-\frac{\frac{v_{\mathrm{DAC}}-v_{\mathrm{ID}}}{V_{\mathrm{DD}}}}{\frac{\sqrt{2}\,\sigma_n}{V_{\mathrm{DD}}}}\right)}\right).$$

The $\pm$ term is positive if we condition on a negative residual voltage [e.g., the Figure 7(a) MSB decision] and negative if we condition on a positive residual voltage [e.g., the Figure 7(a) second MSB decision]. Distributions $f_{T/\tau}(t/\tau)$ for the MSB, second MSB, and LSB regeneration times are shown in Figure 7(c) and (d).

Now let's consider what it means to run out of time and in how many different ways this can actually happen. For the 3-b example from Figure 7, this can happen in exactly five ways, as shown in Figure 8. The first case is that we run out of time during the MSB decision. This would lead to an ADC output code of X00. The second case is that the MSB regeneration finishes, but we run out of time during the fixed delay that follows, leading to an output code of 100. The third case is that the second MSB regeneration starts but doesn't finish, leading to an output code of 1X0. The fourth case is that we run out of time during the second fixed delay. For this case, the two examples in Figure 7(a) and (b) produce different output codes because of the decision error in Figure 7(b). The fifth case is that we run out of time on the LSB, leading to an X in that position. Finally, if the conversion finishes on time, then the ADC output code takes on the same value it would have for thermal noise in isolation.

How can we calculate the probability of each of these six possible outcomes? First, it's important to see that they can all be expressed using sums of independent random variables (regeneration times). For example, the probability of running out of time during the second MSB regeneration (case 3) is

$$P(t_{\mathrm{reg},1} + T_{\mathrm{FIX}} < T_s \cap t_{\mathrm{reg},1} + T_{\mathrm{FIX}} + t_{\mathrm{reg},2} > T_s).$$

In other words, we're interested in the probability that there's enough time for MSB regeneration and the first fixed delay but not enough for those two plus the second MSB regeneration. We can also visualize this event in the sample space of independent random variables $t_{\mathrm{reg},1}$ and $t_{\mathrm{reg},2}$, shown in Figure 9. The event $t_{\mathrm{reg},1} + T_{\mathrm{FIX}} + t_{\mathrm{reg},2} > T_s$ is the hashed green area. What we really want is the probability of the intersection of this event and the event $t_{\mathrm{reg},1} + T_{\mathrm{FIX}} < T_s$. The event of interest is enclosed in the black trapezoid,

Probability Distribution of Regeneration Time With Noise

To calculate the probability density function (PDF) of regeneration time, we first calculate the cumulative distribution function (CDF) from the condition $t_{\mathrm{reg}} = \tau \ln(V_{\mathrm{DD}}/|v|) < t$, where t is the function input and the noisy residual is $v = v_{\mathrm{DAC}} - v_{\mathrm{ID}} - v_n \sim N(\mu, \sigma_n)$. Here, we describe each residual as a normally distributed voltage with mean $\mu = v_{\mathrm{DAC}} - v_{\mathrm{ID}}$.

To get rid of the absolute value, we condition on $v > 0$ or $v < 0$ (residual either above or below the comparator threshold). For $v > 0$, we find the CDF for t_{reg} using the equivalent voltage condition $v > V_{\mathrm{DD}} e^{-t/\tau}$,

$$F_{\mathrm{reg}}(t \mid v > 0) = 1 - F_v\left(V_{\mathrm{DD}} e^{-\frac{t}{\tau}} \middle| v > 0\right) = \frac{1 - \operatorname{erf}\left(\frac{V_{\mathrm{DD}} e^{-\frac{t}{\tau}} - \mu}{\sqrt{2}\,\sigma_n}\right)}{1 - \operatorname{erf}\left(-\frac{\mu}{\sqrt{2}\,\sigma_n}\right)},$$

where F_v is the CDF for the positive side of a Gaussian truncated at $v = 0$.

We find the PDFs for each side by taking the derivative of the respective CDFs,

$$f_{\mathrm{reg}}(t \mid v > 0) = \frac{2V_{\mathrm{DD}}}{\sqrt{2\pi}\,\sigma_n \tau} \frac{e^{-\left(\frac{V_{\mathrm{DD}} e^{-\frac{t}{\tau}} - \mu}{\sqrt{2}\,\sigma_n}\right)^2 - \frac{t}{\tau}}}{1 - \operatorname{erf}\left(-\frac{\mu}{\sqrt{2}\,\sigma_n}\right)}$$

and

$$f_{\mathrm{reg}}(t \mid v < 0) = \frac{2V_{\mathrm{DD}}}{\sqrt{2\pi}\,\sigma_n \tau} \frac{e^{-\left(\frac{V_{\mathrm{DD}} e^{-\frac{t}{\tau}} + \mu}{\sqrt{2}\,\sigma_n}\right)^2 - \frac{t}{\tau}}}{1 + \operatorname{erf}\left(-\frac{\mu}{\sqrt{2}\,\sigma_n}\right)}.$$

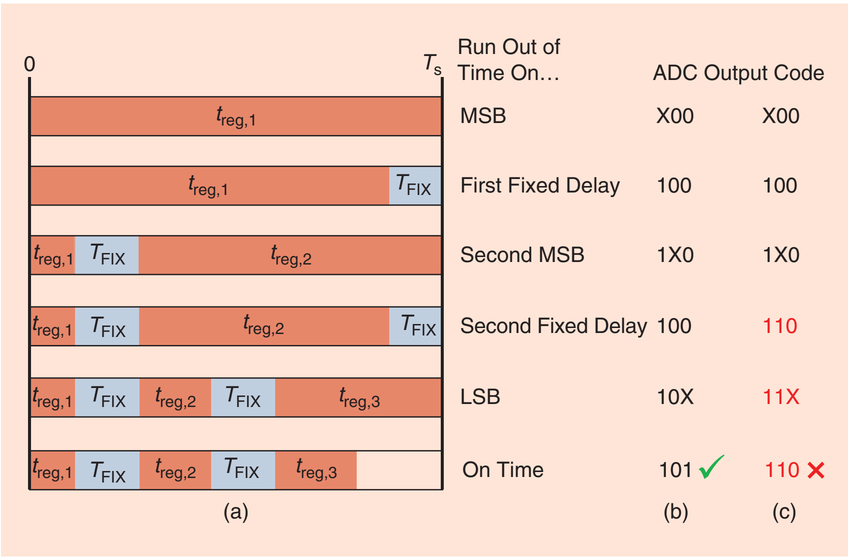


FIGURE 8: For the 3-b example, there are exactly five different ways in which the ADC can run out of time, and each is associated with a particular code error. The two ADC output code columns correspond (b) to Figure 7(a) and (c) to Figure 7(b).

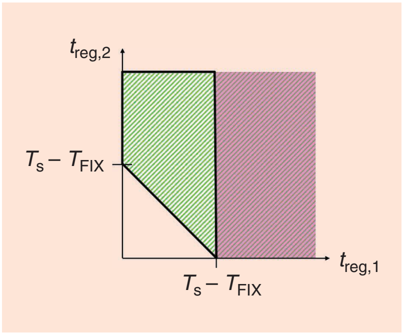


FIGURE 9: A depiction of what it means to run out of time during second MSB regeneration in terms of the sample space of independent random variables $t_{reg,1}$ and $t_{reg,2}$. The hashed green area represents the event in which $t_{reg,1} + T_{FIX} + t_{reg,2}$ exceeds the conversion time. However, we're interested only in the case when this happens and $t_{reg,1} + T_{FIX}$ fits into the conversion time. The purple area represents the event in which $t_{reg,1} + T_{FIX}$ does not fit. By subtracting the probability mass in the purple area from the probability mass in the hashed green area, we obtain the probability of the event of interest.

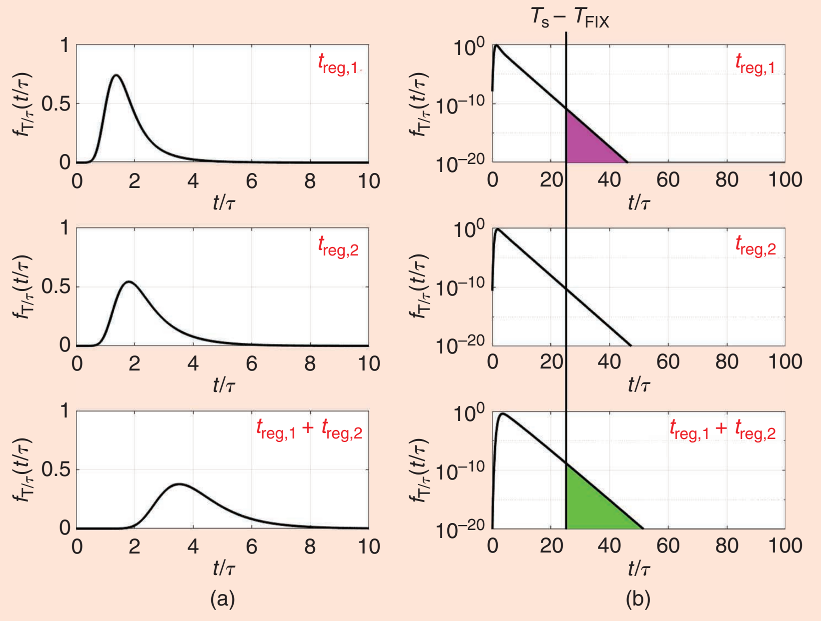


FIGURE 10: The PDFs of MSB regeneration time, second MSB regeneration time, and their sum on (a) linear and (b) log scales for the 3-b example. The PDF of the sum is the convolution of the individual PDFs. The probability mass inside the purple and hashed green areas of Figure 9 is calculated by integrating PDFs in the shaded purple and green areas (visible only on the log scale).

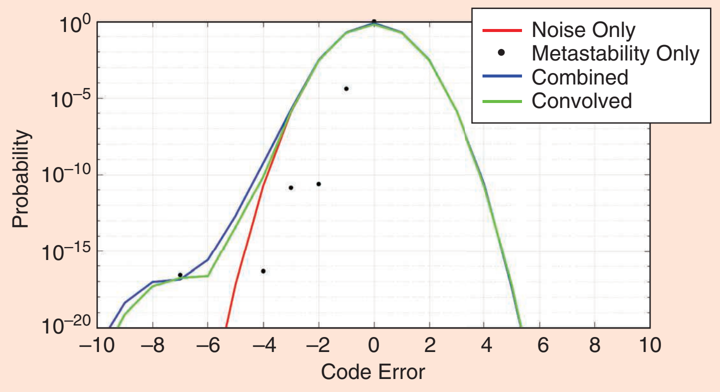


FIGURE 11: A code error PMF overlay of metastability-only (black dots), noise-only (red), combined model (blue), and the convolution of noise-only and metastsability-only PMFs (green). From a 7-b ADC with ENOB = 6, $T_{FIX}/\tau = 8$, and $N = 5$.

and we can obtain its probability by finding the probability mass in the hashed green area and subtracting the probability mass in the purple area. The purple area represents the event $t_{reg,1} + T_{FIX} > T_s$. Therefore, the probability of running out of time during the second MSB regeneration (case 3) is

$$P(t_{reg,1} + t_{reg,2} > T_s - T_{FIX}) \\ - P(t_{reg,1} > T_s - T_{FIX}).$$

We already know the distributions of $t_{reg,1}$ and $t_{reg,2}$, so now we just need the distribution of their sum. Because these random variables are independent, the distribution of their sum is the convolution of their distributions. Figure 10 shows the PDFs of all three ($t_{reg,1}$, $t_{reg,2}$, and their sum) and shows shaded areas in purple and green corresponding to the purple and hashed green areas in Figure 9. Once we take the difference of these shaded areas, we have calculated the probability that we run out of time during second MSB regeneration, given some input level and DAC trajectory. Based on Figure 8, running out of time during the second MSB regeneration corresponds to a code error of –1, so we increment the probability of this code error accordingly. We repeat this procedure for all five ways we can run out of time (as well as for the on-time case), and then we repeat for every DAC trajectory at every input level. Needless to say, this can take a while.

Figure 11 shows code error PMFs for a 7-b ADC with an effective number of bits (ENOB) of 6, $T_{FIX}/\tau = 8$, and $N = 5$. 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 the signal-to-noise ratio, taking *signal* to be a full-scale sinusoid and *noise* to be the sum of quantization noise and comparator input-referred noise (other nonidealities are not considered). What's interesting is that the model combining noise and metastability generates a PMF very close to the convolution of the noise-only and metastability-only PMFs. This tells us that random errors due to noise and metastability effectively combine in an additive way, just as in synchronous architecture. For the practitioner, it's safe to approximate the combined PMF by convolving the metastability-only PMF with the noise-only PMF. We described how to calculate the metastability-only PMF previously, and the noise-only PMF can be approximated by referring comparator noise to the input of the SAR ADC and finding the probability mass that falls into each code (i.e., quantizing a Gaussian). With these approximations, we can compute the code error PMF of an asynchronous SAR ADC with practical resolution in a few minutes.

Lastly, let's return to the question "If I want the probability of code errors greater than X to be smaller than 10^{-Y}, what value of N do I need?" Figure 12 answers this question for a 7-b ADC with ENOB = 6. As with the synchronous architecture, the probability of errors greater than some value has a thermal noise asymptote at large N. Starting slow and then speeding up the ADC (i.e., decreasing N), we eventually hit a corner where metastability begins to cause errors more significant than the thermal

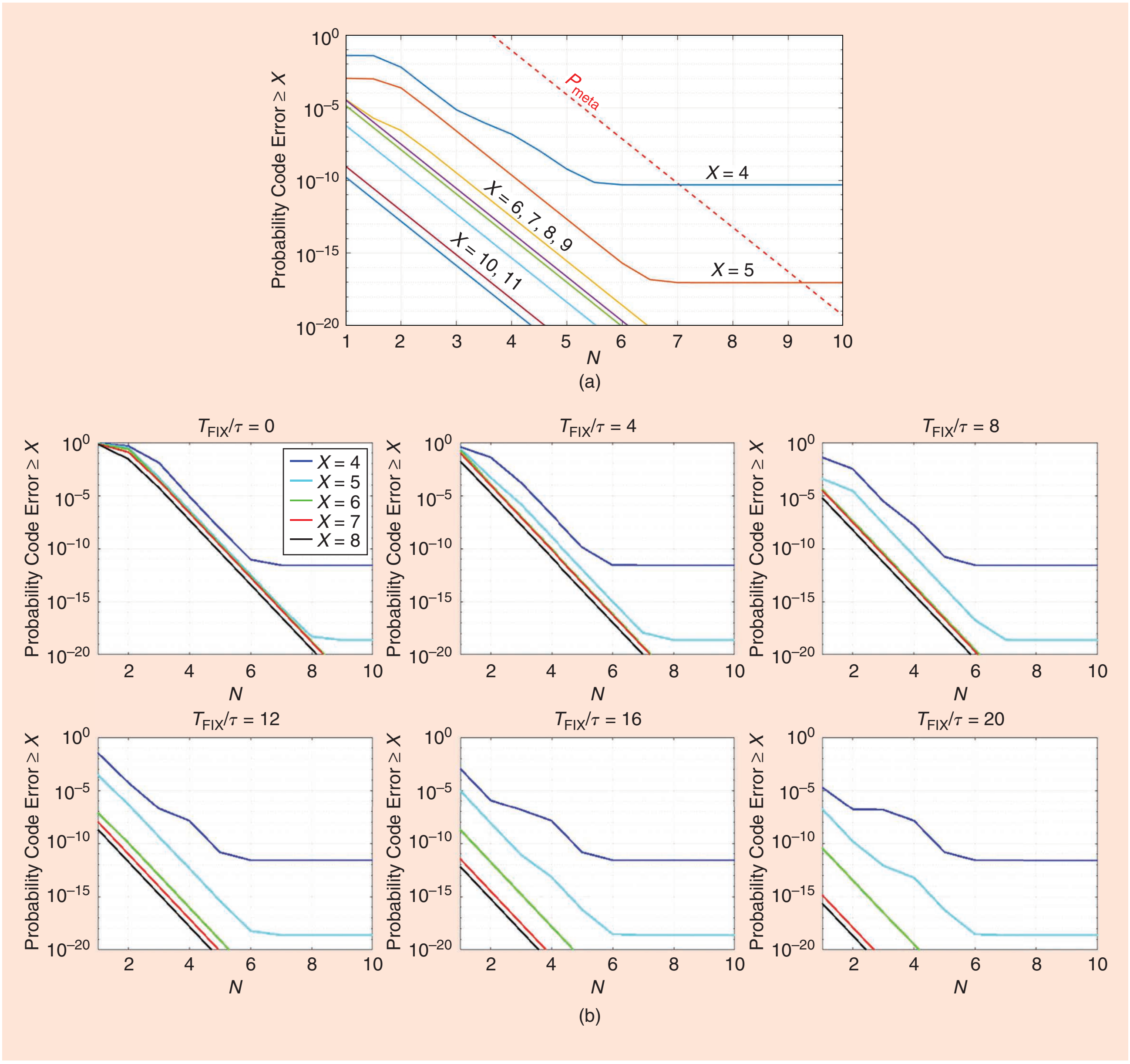


FIGURE 12: A set of plots showing the probability of code errors exceeding a certain amount (X) as a function of N for a 7-b ADC with ENOB $=6$. Given a desired code error ceiling and the tolerable probability of crossing it, these plots allow the designer to read off the required regeneration time in units of comparator time constants. (a) $T_{FIX}/\tau = 8$, computed using the combined model for metastability and noise. The overall rate of metastable events (P_{meta}) versus N is superimposed in dashed red. (b) $T_{FIX}/\tau = 0$, 4, 8, 12, 16, and 20, approximated by convolving the metastability-only PMF with the noise-only PMF.

Starting slow and then speeding up the ADC (i.e., decreasing N), we eventually hit a corner where metastability begins to cause errors more significant than the thermal noise.

noise. Figure 12(a) shows probabilities computed with the combined model for metastability and noise at $T_{\mathrm{FIX}}/\tau = 8$, and Figure 12(b) shows probabilities approximated by convolving the metastability-only and noise-only PMFs at several values of T_{FIX}/τ. Figure 12(a) superimposes a plot of P_{meta} versus N to illustrate that designing for P_{meta} may be overly conservative. For example, if we want the probability of code errors with magnitude greater than 8 LSB to be 10^{-15}, we should design for $N = 4.3$. If we instead design for $P_{\mathrm{meta}} = 10^{-15}$, then we would use $N = 8.6$. By considering the range of code errors that matter, as well as the code error PMF, we learned that we can run the ADC about 40% faster (after including fixed delays in the total conversion time).

Application Example: ADC-Based Serial Link

Ultimately, the error PMF computed in the previous section must be interpreted at the system level and translated into a metric that is useful for the application. To illustrate this process using an example, we consider an ADC-based serial link, which commonly employs a time-interleaved asynchronous SAR ADC. Our analysis is similar to [2], except that we work with our PMF model to find the bit error rate (BER).

Figure 13(a) shows the considered receiver with a digital feedforward equalizer (FFE) and decision feedback equalizer (DFE). We assume 4-level pulse amplitude modulation (PAM4) signaling with symbols [–1, –1/3, 1/3, 1] and a representative channel with 14-dB loss at Nyquist, which gives the blue pulse response in Figure 13(b). The FFE has five precursor and 15 postcursor taps, which are optimally adapted to yield the values in Figure 13(c). The pulse response at the FFE output is shown in red in Figure 13(b). The first FFE postcursor is assumed to be handled by the one-tap DFE. For simplicity, we neglect quantization effects in the equalizers.

Figure 14(a) looks at the inner workings of the FFE, specifically showing how it processes incoming probability densities. The delay-add operations lead to a weighted convolution of the PDFs/PMFs associated with each sample. As a cartoon example, we show here the PDFs of the ADC's quantization error, but this process works similarly for independent distributions containing all nonidealities,

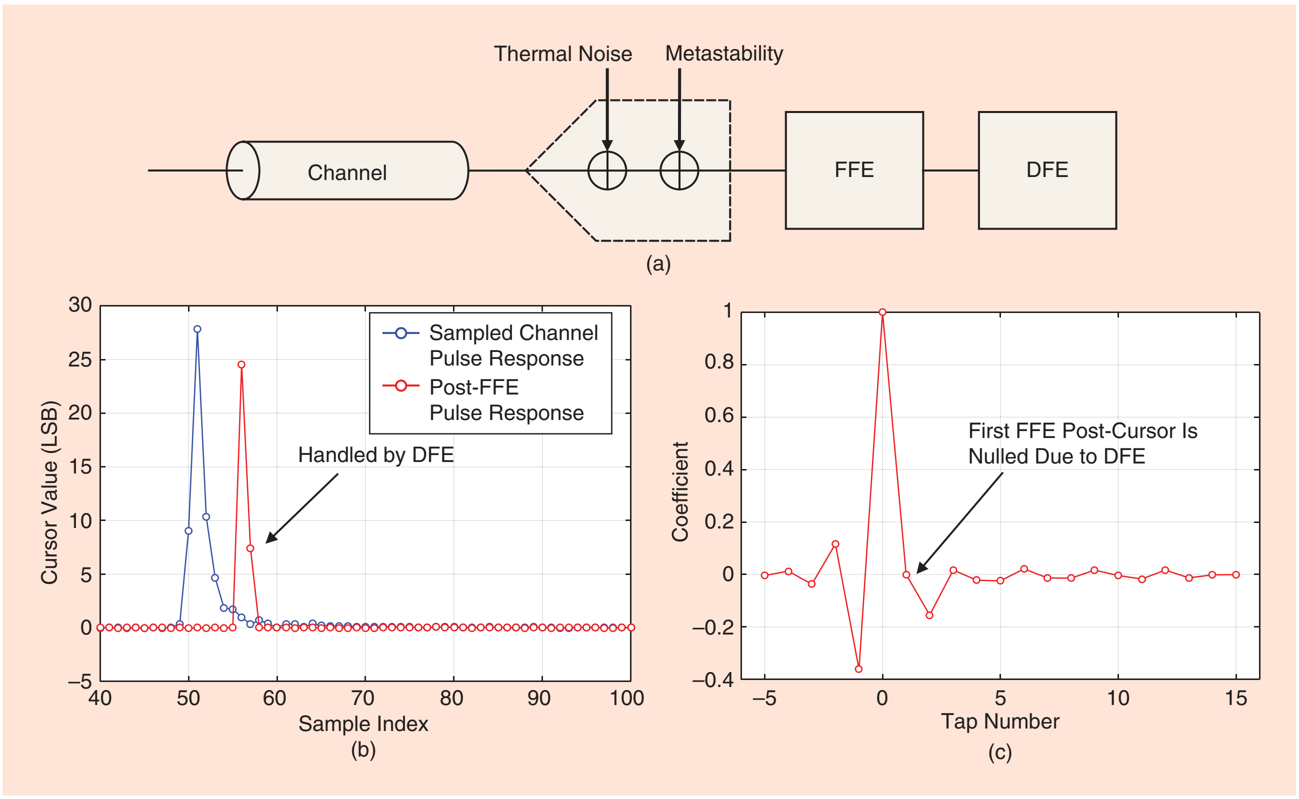


FIGURE 13: (a) A system model for ADC-based serial link analysis. (b) A graph of pre- and post-FFE pulse responses. (c) A graph of adapted FFE coefficients.

including metastability errors. Figure 14(b) shows an example of an incoming PMF in blue and the corresponding FFE-convolved PMF in red. From the convolved PMF (PMF′), we can estimate the BER using

$$\mathrm{BER} \approx \sum_{|\epsilon| > h_0'/3} \mathrm{PMF}'.$$

To a first order (and assuming gray coding), a bit error occurs when the post-FFE error is larger than the eye height, which is one third of the main cursor ($h_0'/3$) for PAM4. This is an approximation, since, for the outer PAM levels (i.e., +/−1), only one error polarity matters. Nonetheless, within the limited scope of this analysis, this will still serve as a useful proxy that gives about the right order of magnitude.

We now study numerical examples with this approach (see Figure 15). We assume 7-b ADCs with small fixed delay ($T_{\mathrm{FIX}}/\tau = 8$, in red) and large fixed delay ($T_{\mathrm{FIX}}/\tau = 16$, in blue). For both cases, we vary the ADC's thermal

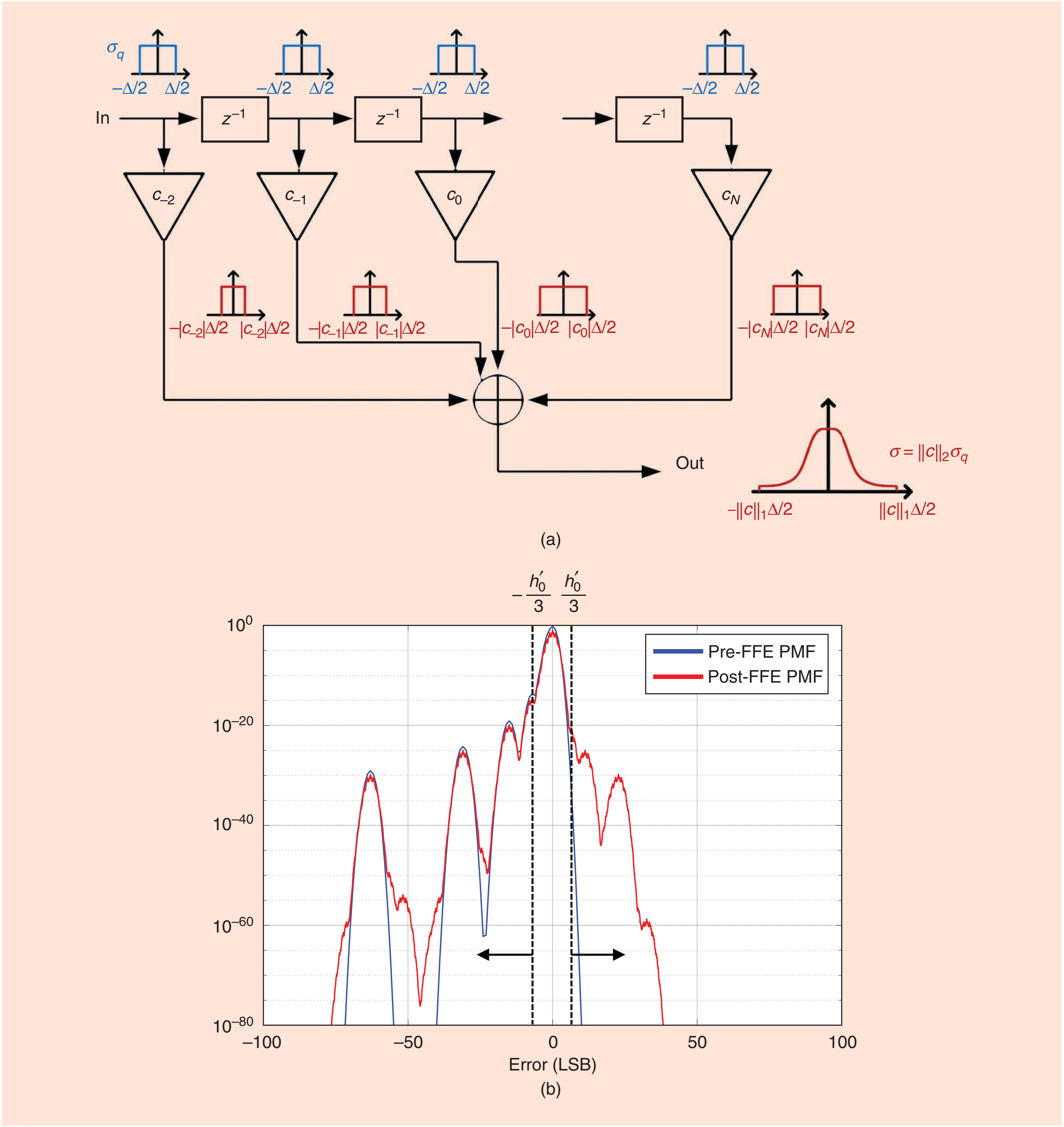


FIGURE 14: (a) An FFE operating on incoming probability distribution. The output distribution follows from a scaled convolution of each input PDF/PMF. (b) A graph of pre- and post-FFE PMF for a 7-b, ENOB = 6 asynchronous SAR ADC with $T_{\mathrm{FIX}}/\tau = 8$ and $N = 4$.

noise level to cover the range between ENOB = 7 (noiseless) and ENOB = 4. Additionally, we sweep the timing parameter N from −4 to 7 and plot the resulting BER estimate against $(T_s - T_{\text{track}})/\tau$, the analog-to-digital conversion time without the input tracking interval. These plots give us a feel for how fast we can run the ADC before we encounter metastability troubles. For large analog-to-digital conversion times, we see that both the fast (red) and slow (blue) designs converge to the same BER at ENOB of 4 and 5. These horizontal asymptotes are set purely by thermal noise and are unrelated to metastability behavior. As we go farther to the left (i.e., as we stress the ADC with a shorter conversion time), we run into steep BER asymptotes set by the combined effect of noise and metastability.

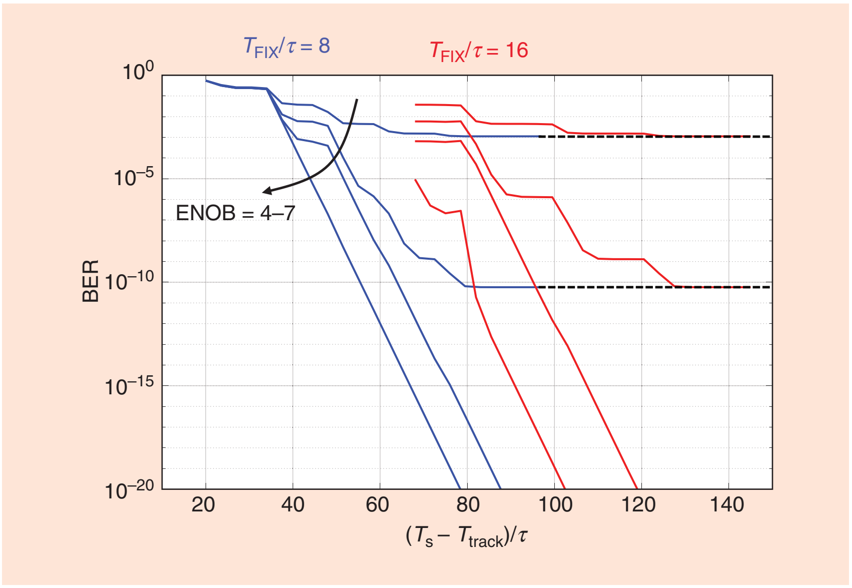


FIGURE 15: A plot of BER estimate versus $(T_s - T_{\text{track}})/\tau$ (analog-to-digital conversion time without input tracking interval). Plots are for 7-b ADCs, with small fixed delay ($T_{\text{FIX}}/\tau = 8$) in blue and large fixed delay ($T_{\text{FIX}}/\tau = 16$) in red. For both cases, the ADC's thermal noise level is varied to cover the range between ENOB = 7 (noiseless) and ENOB = 4. The timing parameter N is swept from −4 to 7.

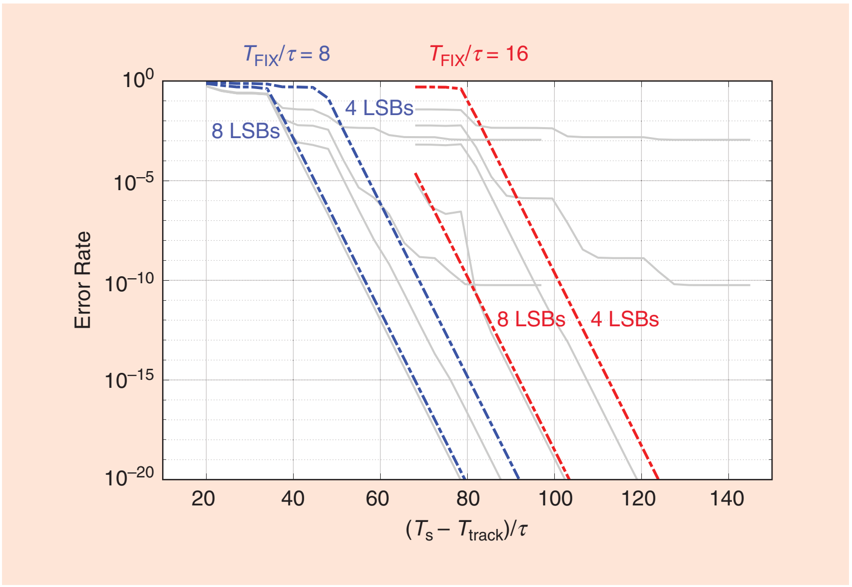


FIGURE 16: A plot of the rates of metastable events errors with magnitudes of ≥ 4 and ≥ 8 LSBs, for large fixed delay (blue) and small fixed delay (red). The data from Figure 15 are overlaid in gray. The x-axis $(T_s - T_{\text{track}})/\tau$ represents the analog-to-digital conversion time without the input tracking interval.

While the previously discussed plot is very useful for design validation, producing it is labor intensive and requires detailed system knowledge, including the exact channel model. Is there a shortcut that lets us obtain reasonable estimates more quickly? Indeed, we can rely on some intuition gained from the PMF characteristics seen in Figure 11. In an asynchronous design, metastability errors of large magnitude are extremely rare and will not significantly impair the BER. Additionally, small errors on the order of 1–2 LSBs are hard to distinguish from thermal noise and will therefore not lead to dramatic shifts in the horizontal BER asymptotes. Ultimately, what should matter most are metastability errors with magnitudes of about 4 or 8 LSBs. In Figure 16, we plot the rates at which pre-FFE errors ≥ 4 LSBs and ≥ 8 LSBs occur for both the slow and fast designs and with the plots from Figure 15 overlaid in gray. We observe that the 8-LSB error rate essentially coincides with the BER of the noiseless design and, hence, constitutes a useful and easy-to-construct asymptote without knowing the specifics of the FFE. As a more conservative option, the 4-LSB curve could be considered.

Figure 17 shows how this approach can be used to get a quick feel for the maximum possible ADC speed. Given a certain target BER (e.g., 10^{-6} or 10^{-12}), we can look for the intersections that define the possible operating ranges (marked in green). Note that the ADC can run only slightly faster for the higher BER target. Lastly, it is interesting to consider the overlaid P_{meta} curves, which are the probability of any metastable event in the ADC. Clearly, designing with this pessimist metric would leave a significant amount of performance on the table and could send us down the wrong path for our design.

Metastability Detectors

Can we detect a metastable state and simply override the ADC output when we're so close to a presumably known decision level? Such schemes have indeed been proposed over the years, but we believe the jury is still out on their effectiveness for asynchronous designs. The situation for synchronous designs is somewhat different, because there is a clear opportunity for eating into the excess timing margins of each bit cycle [6].

The first issue with adding a metastability detector in an asynchronous design is that it will typically load the comparator (directly or via downstream fan-out) and/or increase the delay of subsequent gates, which will exercise the extremely steep tradeoff between the error rate and loop timing (see Figures 3 and 15). Once the detector is added, the PMF may worsen significantly, and the design would have to come from behind to clean things up. This is difficult, especially when the metastability detection logic itself may introduce new metastable states or race conditions that are hard to analyze or even be aware of (see [7] for similar pitfalls with synchronizers). Lastly, because the metastability PMF decays very quickly for large code errors, practical asynchronous designs that don't target extreme performance levels won't show metastability errors that go far beyond the thermal noise level [5], making detection unnecessary.

For anyone interested in demonstrating the effectiveness of a metastability detector for asynchronous SAR ADCs, the following experiment should be considered: 1) implement a best-effort SAR loop without any metastability detection circuitry (simply disabling the detection circuitry at its output is not useful for comparisons); 2) implement the modified circuit with the detection circuitry added; and 3) measure the error PMFs of the two designs in the lab, and provide a quantitative analysis that corroborates the observed results. The methods described in this article should be helpful for the latter.

In an asynchronous design, metastability errors of large magnitude are extremely rare and will not significantly impair the BER.

Summary

In this article, we developed a combined framework for analyzing metastability and thermal noise in the asynchronous SAR ADC. The key takeaways can be summarized as follows.

- In the synchronous architecture, timing margin for metastability is allocated to each bit cycle individually. If the comparator input happens to be large during some bit cycle, its timing margin is wasted. In the asynchronous architecture, a single block of timing margin is allocated for all bit cycles to share. In this manner, timing margin isn't wasted, and the hard decision can take longer, if necessary.
- The method of timing bands depicted in Figure 4 allows us to see how regeneration times change and add together for ADC inputs across the full-scale range.
- The synchronous architecture has a 1D design space in N, whereas the asynchronous architecture has a 2D design space in N and T_{FIX}/τ. This is because, in the asynchronous architecture, the fixed delays of the LSBs appear as additional timing margin for the MSBs to borrow.
- Large code errors caused by running out of time during regeneration of MSBs, such as half-scale or quarter-scale errors, typically have negligible probabilities because the required residual voltage to make this happen is extremely small. On the other hand, small 1–2-LSB code errors are typically drowned out by thermal noise. These conclusions are consistent with the observations made in [5]. We should mainly be concerned with mid-sized code errors (e.g., 4–8 LSBs for a 7-b ADC) due to metastability in an asynchronous design.
- Random errors due to comparator noise and those due to metastability essentially combine in an additive manner. We can do a

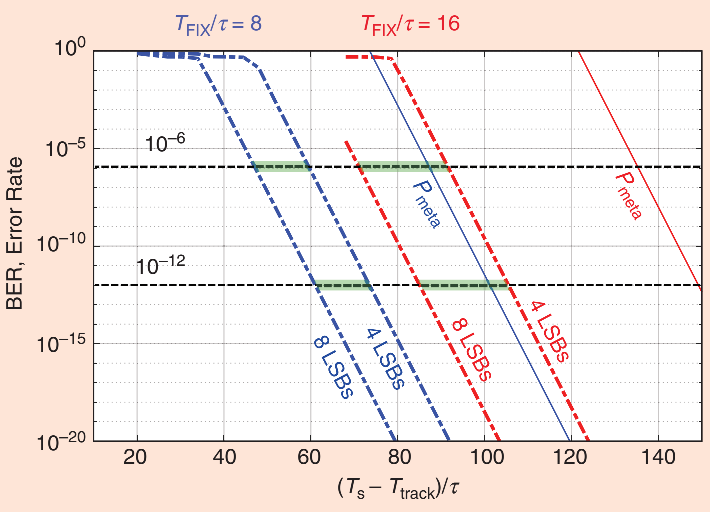


FIGURE 17: The asymptotes from Figure 16 with possible operating regions (for two BER values) marked in green. Also shown for reference are the P_{meta} curves for each case (probability of any metastable event in the ADC). The x-axis $(T_S - T_{track})/\tau$ represents the analog-to-digital conversion time without the input tracking interval.

Practical asynchronous designs that don't target extreme performance levels won't show metastability errors that go far beyond the thermal noise level, making detection unnecessary.

decent job of approximating the combined model by convolving the noise-only and metastability-only PMFs.

- The probability of code errors greater than some value has a minimum set by thermal noise. As we run the ADC faster (i.e., decrease N), metastability eventually causes this probability to increase. This two-part tutorial provides designers with a framework for finding this corner.
- The derived PMF model can be used to estimate the BER of an ADC-based serial link receiver. If the equalizer parameters are known, we can propagate the PMFs via convolutions and obtain a BER estimate by comparing the post-equalizer error magnitude with the eye height. However, a quicker method is to leverage the observation that the mid-size metastability errors matter most and consider their rate of occurrence as a first-order BER bound.
- Given the extremely steep tradeoff curves between the loop timing and metastability error rates in an asynchronous design, it may be counterproductive to introduce even the slightest bit of extra loop delay for the purpose of detecting and resolving metastable states. We leave it as an unproven conjecture that the most practical design choice is to simply focus on minimizing all loop delay components.

References

[1] S.-W. M. Chen and R. W. Brodersen, "A 6b 600MS/s 5.3mW asynchronous ADC in 0.13 μm CMOS," in *IEEE 2006 Int. Solid-State Circuits Conf.–Digest of Technical Papers*, Feb. 2006, pp. 2350–2359.

[2] S. Cai, A. Shafik, S. Kiran, E. Z. Tabasy, S. Hoyos, and S. Palermo, "Statistical modeling of metastability in ADC-based serial I/O receivers," in *Proc. 2014 IEEE 23rd Conf. Electrical Performance Electronic Packaging Systems*, Portland, Oregon, Oct. 2014, pp. 39–42.

[3] 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.

[4] 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.

[5] R. Kapusta, J. Shen, S. Decker, H. Li, E. Ibaragi, and H. Zhu, "A 14b 80 MS/s SAR ADC with 73.6 dB SNDR in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 48, no. 12, pp. 3059–3066, Dec. 2013. doi: 10.1109/JSSC.2013.2274113.

[6] J. P. Keane et al., "16.5 An 8GS/s time-interleaved SAR ADC with unresolved decision detection achieving −58dBFS noise and 4GHz bandwidth in 28nm CMOS," in *Proc. IEEE 2017 Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2017, pp. 284–285.

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

About the Authors

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 ICs 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.

Daniel Bankman (dbankman@gmail.com) 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 at Stanford University, California, 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.

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., San Jose, California.

Boris Murmann (murmann@stanford.edu) received his Dipl.-Ing. (FH) in communications engineering from Fachhochschule Dieburg, Germany, in 1994, his M.S. in electrical engineering from Santa Clara University, California, in 1999, and his Ph.D. 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 IC 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 for *IEEE Journal of Solid-State Circuits*. He is a Fellow of the IEEE.

SSC