NOISE FIGURE CALCULATIONS IN DSP SYSTEMS

Torbjorn Larsson
Paradiddle Communications Inc.
torbjorn@paradiddle.us

INTRODUCTION

As digital signal processing (DSP) continues to make inroads into the RF domain, the design of radio solutions in DSP is evolving from being merely an application of DSP to a recognized engineering discipline. *Radio DSP design*, as we will call it here, is the marriage of radio systems engineering and traditional (baseband) DSP systems engineering, but with its own unique set of tools and design patterns. The basic objectives of radio system design remain the same, of course. The digital signal path in a DSP-based receiver, just like the analog signal path in a traditional radio receiver, must be carefully crafted to strike the right balance between sensitivity, linearity, selectivity and hardware cost (or power consumption). However, the ways in which basic parameters like noise level, signal gain and linear range are controlled in a DSP radio differ significantly from a traditional radio. This white paper is the first in a series of papers in which we intend to lay down the basic rules of radio DSP design and discuss a number of important design patterns. In the first paper, we take a look at one of the most fundamental tools of radio systems engineering: *noise figure*. We begin by demonstrating that the noise figure concept applies equally well in the digital domain, and that noise figure calculations in DSP systems can be surprisingly simple. The noise figures of two common DSP structures, a FIR decimator and a CORDIC rotator, are derived as an illustration. Next, we consider calculating the noise figure of a system comprised of cascaded DSP blocks. The result is used to calculate the noise figure of a digital down-converter (DDC).

Although the focus here is on custom DSP solutions (meaning algorithm-specific datapath solutions for ASIC or FPGA), it is worth mentioning that the results presented below are generic and can be applied to programmable DSP architectures as well. As will soon become clear, much of the design work in a custom DSP solution is concerned with controlling the *physical bitwidth* along the datapath. When dealing with a programmable DSP architecture, on the other hand, one must distinguish between *word length* and *effective bitwidth*. The word length is a feature of the processor architecture, whereas the

effective bitwidth is the number of bits occupied by the signal at a particular point in the signal path. The effective bitwidth, which is algorithm-dependent and varies along the signal path, plays the same role in programmable DSP as the physical bitwidth does in custom DSP. In the following, we will let the term “bitwidth” refer to the physical bitwidth of the datapath, since the assumption is that we are dealing with a custom DSP solution. However, the same results can be applied to a programmable DSP architecture by simply interpreting “bitwidth” as “effective bitwidth”.

NOISE IN DSP SYSTEMS

Consider the datapath in Figure 1. This is a custom DSP implementation of a FIR (finite impulse response) decimation filter with decimation factor 2, a structure commonly found in wireless receivers. The signal samples as well as the filter coefficients in the decimator are signed integers in 2’s complement representation. The de-multiplexer at the decimator input alternates between the upper and lower branch, so that each branch receives one out of every two incoming samples. Both branches are FIR filters, although the FIR filter in the upper branch only has a single tap, due the half-band nature of the filter design [1]. The output samples from the two branches are summed to produce the decimator output, resulting in a sample rate reduction by a factor of 2. Note that the entire circuit except the de-multiplexer runs at the output (i.e. lower) sample rate, a significant implementation advantage in high-speed DSP applications. Also note that only those registers that are essential for the correct operation of the decimator have been included in the figure. It is often necessary to insert additional pipeline registers in order to satisfy the timing constraints of a specific implementation, but such registers have no impact on the noise figure.

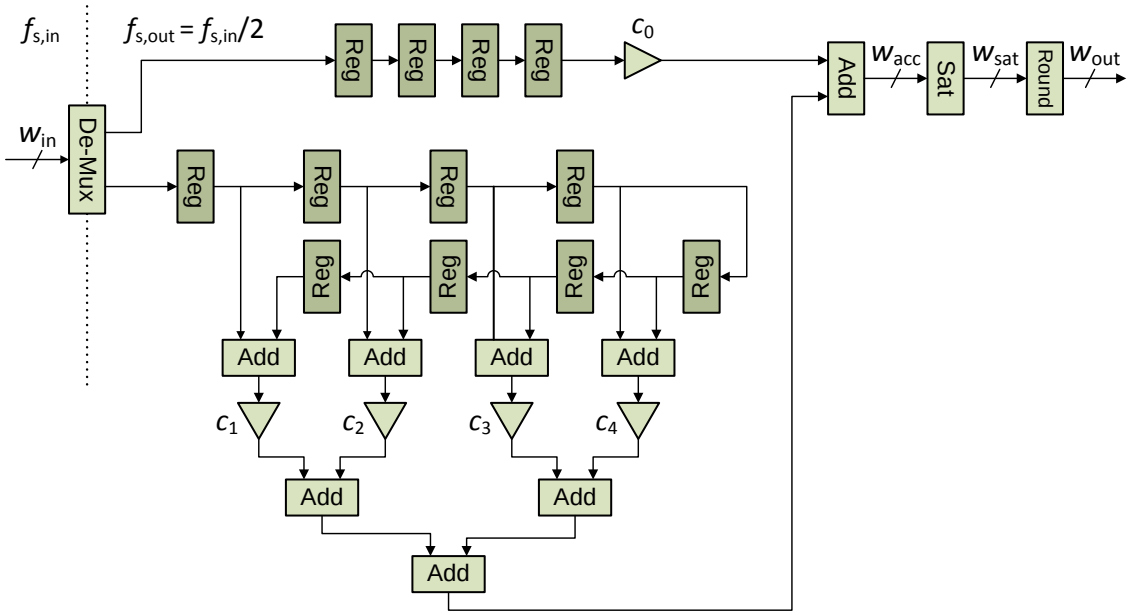

Figure 1 2x FIR decimation filter.

The datapath architecture in Figure 1, which is essentially a block diagram mapped directly onto a datapath, is often referred to as a *constant multiplier architecture*. The name derives from the fact that the multiplications in the filter taps are carried out by dedicated shift-and-add logic rather than a generic multiplier. Other architectures can be used to implement the same block diagram, the most common being the multiply-accumulate loop [2] [3]. However, the constant multiplier architecture is an ideal starting point for a discussion of noise figure in DSP systems, because there is no need in this case to distinguish between *datapath* (digital circuit) and *block diagram* (algorithm). Later on, we will see that for noise figure calculations it is the algorithm that matters, not the datapath architecture.

The design of the datapath in Figure 1 is based on a few simple principles. Starting from the input bitwidth w_{in} and proceeding left-to-right, the signal bitwidth in the datapath is gradually expanded inside all adders and constant multipliers to ensure that overflow cannot occur. For instance, the bitwidth after the first line of adders is $w_{\text{in}} + 1$, whereas the bitwidth after multiplication with a coefficient c_k is $w_{\text{in}} + 1 + \text{ceil}(\log 2(|c_k|))$, and so on. This continues all the way up to the last adder on the right hand side, at which point the bitwidth is reduced by a *saturate* operation followed by a *round* operation. Due to the bitwidth expansion, the whole circuit up to the last adder is perfectly linear and noise free. The signal bitwidth cannot be allowed to keep on growing indefinitely, however, which is why the saturate and round operations are needed. The task of the saturate operation is to remove a fixed number (typically one or two) of the most significant bits in each signal sample, while ensuring that no wrap-around occurs. This is done by limiting (i.e. clipping) those sample values that cannot be represented in the saturation bitwidth w_{sat}. It should be said right away that clipping, although preferable to wrap-around, is a non-linear behavior that is very much undesirable in a wireless receiver. However, by setting the saturation bitwidth correctly, the probability of clipping can be made small enough that the effect of the saturate operation is negligible. This brings us to the round operation located at the output of the decimator. The purpose of this operation is to remove a fixed number of the least significant bits from each signal sample. Rounding is used instead of simple truncation in order to avoid a bias, or DC offset, in the output signal.

Because rounding and truncation are so essential to a discussion of noise figure in digital systems, it is necessary that we first describe these operations in more detail. Truncating the n least significant bits of a signal sample can be thought of as a two-step procedure, where 1) the sample value is divided by 2^n, thus yielding a number with an n-bit fraction, and 2) the fraction is erased to obtain an integer result. With 2's complement number representation, erasing the fraction always makes the output number *smaller*. Therefore, when truncation is performed on a sequence of unbiased samples, the resulting output sequence will be negatively biased. The round operation, on the other hand, modifies the behavior in step 2 by rounding to the nearest integer, i.e. by adjusting the output number upwards when the fraction is greater than 0.5. For example, suppose we wish to remove the 3 least significant bits from the sample value $0100101_2 = 37_{10}$. With truncation, the result is $0100_2 = 4_{10}$, which can be written as 0100.101 – 0.101, or in decimal notation as 4.625 – 0.625. The truncation error is therefore –0.625. If instead we use rounding, the result becomes $0101_2 = 5_{10}$, since the number 4.625 is rounded

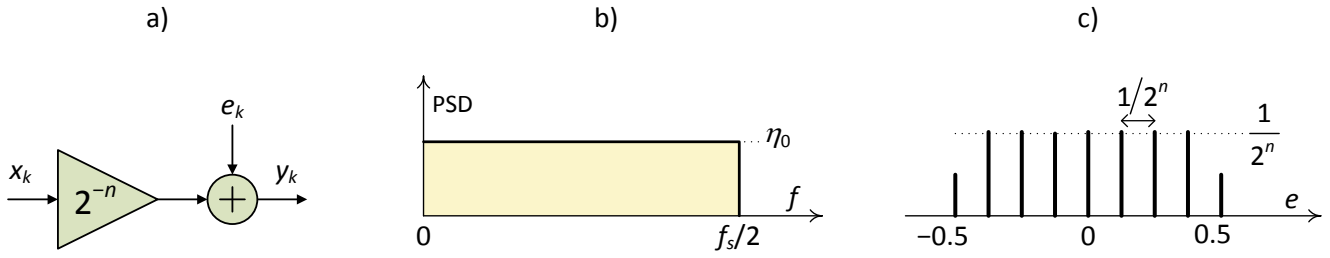

Figure 2 Mathematical representation of rounding. a) Equivalent model. b) Power spectral density of round-off noise. c) Probability mass function of round-off noise (n = 3).

upwards to 5. The rounded output can be written as 4.625 + 0.375, so the rounding error in this case is +0.375. With rounding, special attention must be paid to the case when the fraction is exactly 0.5, i.e. when the number to be rounded is right in between two integer values. In this case, some rule must be used to ensure that rounding in the positive direction occurs with the same frequency as rounding in the negative direction. This will completely eliminate the bias and is known as *convergent rounding* [4]. A common method to implement convergent rounding is to select the nearest *even* integer value when the fraction is 0.5.

Figure 2 depicts the mathematical model of a convergent round operation that removes n bits. We see that the round operation acts as an attenuator followed by an additive noise source. Under normal conditions, the noise samples generated by the round operation will form a stationary white process with power spectral density (PSD) as shown in Figure 2b. Further, all noise (error) values except the two boundary values ±0.5 will occur with equal probability, as illustrated in Figure 2c. For example, a round operation that removes 3 bits will generate 2^3+1 = 9 different noise values given by 0, ±0.125, ±0.25, ±0.375 and ±0.5, where the two boundary values ±0.5 occur with probability 1/16 and all other values occur with probability 1/8. From the probability mass function (PMF) in Figure 2c, we can calculate the power (mean square value) of the round-off noise. The result is

$$\overline{e^2} = \frac{1}{12}\left\{\frac{3}{2^n} + \frac{\left(2^n - 2\right)\left(2^n - 1\right)}{2^{2n}}\right\} \tag{1}$$

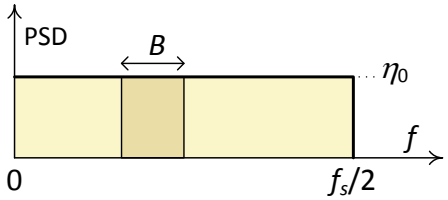

Figure 3 Only the portion of the round-off noise that falls within the user's bandwidth is relevant.

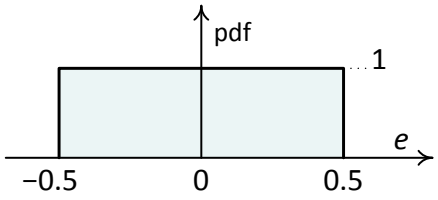

Figure 4 Probability density function of ideal quantization noise.

Notice that with increasing n, the expression inside the brackets quickly approaches 1 and the noise power approaches 1/12 ≈ 0.083. Already for n = 4, the noise power is 0.084. Good system design dictates that round operations should be used sparingly (to limit the amount of round-off noise), which means that every round operation in the system will normally remove a fair number of bits, typically 10 or more. Under these circumstances, $\overline{e^2} = 1/12$ is a very a good approximation. In the following, we will therefore base our noise analysis on the assumption that *every round operation in the system generates the same noise power* 1/12.

Naturally, this does not mean that every round operation will have the same impact on the system SNR. Recall that the desired (user) signal typically only occupies a portion of the frequency axis, as depicted in Figure 3. Only the noise power that falls within the user bandwidth (B) is relevant, since the remaining noise will be rejected by filters located further downstream in the receiver. Consequently, what matters is not the noise power, but the noise *power spectral density*. As can be seen in Figure 3, the one-sided noise PSD, denoted by η_0, is obtained by dividing the total noise power 1/12 with half the sample rate (f_s/2), or equivalently, by multiplying the noise power with twice the sample period ($2T_s = 2/f_s$). The result is a parameter so fundamental to the analysis of noise in DSP systems that it deserves to be framed:

$$\eta_0 = \frac{1}{6f_s} = \frac{T_s}{6} \tag{2}$$

This PSD level applies not only to round-off noise, but also to the quantization noise in an A/D converter. Ideal quantization noise has the probability density function shown in Figure 4, which can be thought of as the limiting case of the probability mass function in Figure 2c for $n \to \infty$. As expected, the mean square value of the probability density function in Figure 4 evaluates to 1/12 and the power spectral density of quantization noise is therefore the same as for round-off noise. In fact, *any* digital (i.e. quantized) waveform with uncorrelated quantization noise will exhibit a noise floor with PSD given by Eq. (2). This noise floor is always present, even in the absence of other noise sources.

As an illustration of the importance of Eq. (2), consider the digital down-converter (DDC) in Figure 5.

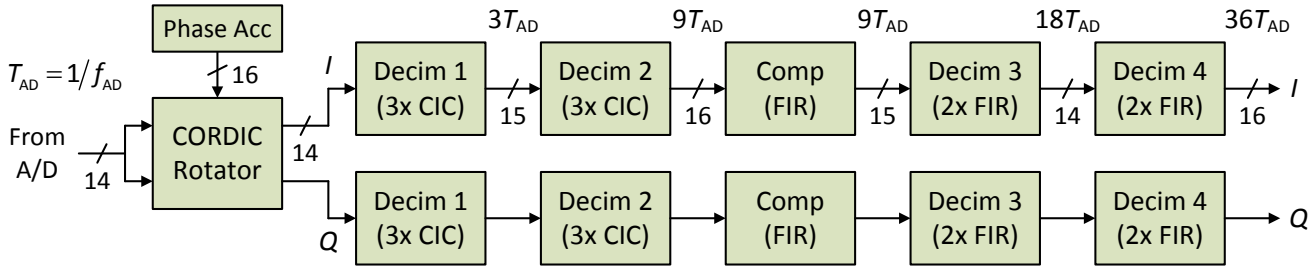

Figure 5 Digital Down-Converter.

The input of this DDC is connected to an A/D converter with sample rate f_{AD} and sample period $T_{AD} = 1/f_{AD}$. The signal path inside the DDC consists of a CORDIC phase-rotation block [5] serving as a complex synthesizer/mixer, two Cascaded Integrator Comb (CIC) decimation stages [1] with decimation factor 3, a low-order FIR filter to compensate for the passband droop in the two CIC filters [6], and two FIR decimation stages with decimation factor 2. Taken together, the four decimation stages achieves a decimation factor of $3 \cdot 3 \cdot 2 \cdot 2 = 36$. The CORDIC rotator block, in combination with the phase accumulator, acts as both a synthesizer and a mixer, down-converting the desired user signal to baseband by phase-rotating the input samples. The third decimation stage in the DDC is the FIR decimation filter discussed earlier and shown in Figure 1. Let us consider the impact of the round operation in this filter, compared to the impact of a round operation inside the CORDIC rotator (this block will be described later). The round operation in the CORDIC rotator runs at sample rate $1/T_{AD}$ and therefore generates a noise PSD equal to $T_{AD}/6$ Hz^{-1}. As the signal travels downstream in the DDC, the noise added by the various round operations in the signal path gradually get "hotter" due to the falling sample rate (or equivalently, the growing sample period). When the signal reaches the output of third decimation stage, the sample rate has dropped by a factor $3 \cdot 3 \cdot 2 = 18$. Therefore the noise level contributed by the round operation in this decimation stage is $T_{AD}/108$ Hz^{-1}, which is 12.5 dB higher than the noise level contributed by the round operation inside the CORDIC rotator.

Another parameter that influences the relative impact of two round operations located at different points in the system is the signal gain along the intermediate signal path. For example, by the time the round-off noise from the CORDIC rotator has made its way to the round operation in the third decimation stage, it has been amplified by the accumulated gain in the signal path that runs from the input of the first decimation stage to the output of the third decimation stage (including the attenuator *inside* the round operation of that stage, see Figure 2a). Consequently, if we wish to reduce the relative impact of the round operation in the third decimation stage, we must raise the gain somewhere along this signal path. One option is to increase the output bitwidth in the third decimation stage, since this will reduce the number of bits removed by the round operation, thereby reducing the attenuation just before the round-off noise is added. The other option is to raise the signal gain in one of the upstream (preceding) blocks.

Although we generally recommend the use of convergent rounding, and especially so for wireless applications, there are a few cases where rounding can be replaced by truncation without any deleterious effect. One such case is the CORDIC algorithm discussed below. Truncation can be thought of as a sort of round operation that always "rounds" to the nearest s*maller* integer, causing the truncation error to fall between 0 and −1. The PMF of truncation noise has the same general appearance as the PMF of round-off noise shown in Figure 2c, but is centered at −0.5 rather than zero. Truncation noise can therefore be modeled as round-off noise riding on a DC component with amplitude −0.5. The power in the noise term is the same as for convergent rounding (= 1/12); the only difference is the presence of a DC component.

NOISE FIGURE IN THE DIGITAL DOMAIN

From the discussion above, it may appear that calculating the total noise level generated by the round operations in the DDC is a very complicated task. This is not so. All we need is a parameter that expresses the amount of noise a block contributes to the system, and a way to calculate this parameter as a function of the other parameters in the block, such as the input and output bitwidths. In other words, what we need is a definition of *noise figure for DSP systems*. Our starting point will be the classic definition of noise figure as it applies in the analog domain. Figure 6a shows the basic setup [7] [8]. The device-under-test (DUT) is a two-port connected to a signal source and a load. The SNR is measured both at the input and the output, across the user bandwidth B. Because of the noise generated within the DUT, the output SNR will always be less than the input SNR. This degradation of SNR is measured by the *noise factor*, defined as

$$F = \frac{S_{in}/N_{in}}{S_{out}/N_{out}} \tag{3}$$

When expressed in dB, the noise factor is referred to as *noise figure*, denoted by *NF* (although some authors do not make this distinction and use the two terms interchangeably). Note that the noise factor definition requires that the input (source) noise level N_{in} is specified; without an *input noise level reference*, the definition is ambiguous. When working with the noise factor of an analog system, input impedance matching is usually assumed, i.e. the source resistance is assumed to equal the input resistance of the two-port. The matched condition leads to $N_{\text{in}} = kTB$ where k is Boltzmann's constant and T is the temperature in degrees Kelvin. This is the noise power that is guaranteed to be present across the input terminals of the two-port, even in the absence of other noise sources.

Other than serving as a definition of the noise factor, Eq. (3) is not particularly useful. The dependence on the input signal power S_{in} and the signal bandwidth B makes it overly complicated. To derive a simpler expression, we first note that $S_{\text{out}}/S_{\text{in}} = G^2$, where G^2 is the power gain of the DUT. Observe that G here is defined as the square root of the power gain. Further, if we let η_{in} and η_{out} denote the input and output noise PSD inside the user frequency band, we have $N_{\text{out}}/N_{\text{in}} = (\eta_{\text{out}}B)/(\eta_{\text{in}}B) = \eta_{\text{out}}/\eta_{\text{in}}$ where

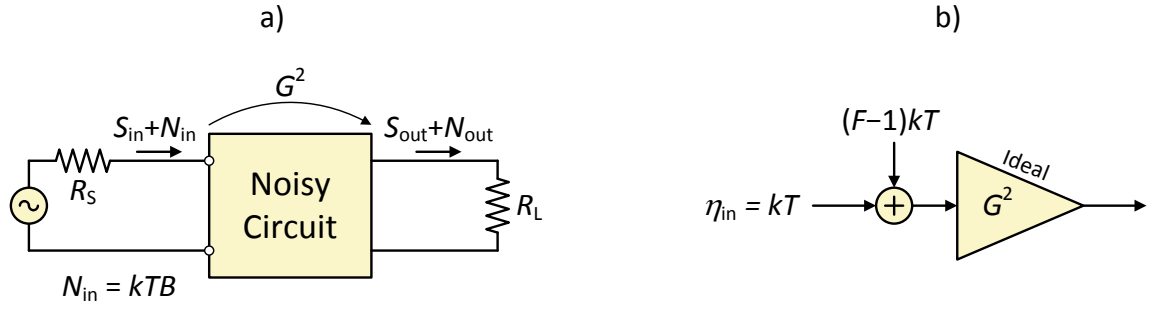

Figure 6 Noise figure in the analog domain. a) Basic setup. b) Equivalent system model with input-referred excess noise PSD.

$\eta_{in} = kT$. Using these results, the noise factor in Eq. (3) can be written as

$$F = \frac{\eta_{out}}{G^2 \eta_{in}} \tag{4}$$

Next, we note that there are two principal noise sources contributing to the noise level at the output of the DUT: the noise originating from the input and the noise generated inside the DUT. The latter is commonly known as the *output-referred excess noise*. Letting η_{ex} denote the PSD of the output excess noise, we can now express the noise factor as

$$F = \frac{G^2 \eta_{in} + \eta_{ex}}{G^2 \eta_{in}} = 1 + \frac{\eta_{ex}}{G^2 \eta_{in}} \tag{5}$$

Conversely, if the noise factor has already been calculated, the equation above can be used to solve for the output excess noise level, given by

$$\eta_{ex} = G^2 (F - 1) \eta_{in} \tag{6}$$

It is often convenient to think of the excess noise as being generated by a fictitious noise source located at the input of an ideal (noiseless) system, as shown in Figure 6b. This theoretical noise source, with PSD given by $\eta_{ex}/G^2 = (F-1)\eta_{in}$, is referred to as the *input-referred excess noise*. Notice that since the input noise level η_{in} is a known quantity, the input-referred excess noise level is completely specified by the noise factor.

We now have everything we need to define the noise factor of a digital system. In fact, all the hard work has already been done. Going back to the noise factor definition in Eq. (3), it should be clear that this definition can also be applied to a digital system, provided that 1) signal power is interpreted as mean square value, and 2) the input noise level is redefined appropriately. The situation is illustrated in Figure 7a. How should the input noise level be defined? The obvious choice is to use the PSD in Eq. (2), since this is the noise level that is guaranteed to be present at the input of a digital system in the absence of other noise sources. Thus, for a digital system we have $\eta_{in} = T_s/6$. With these modifications, Eqs. (4) – (6) apply directly to a digital system and the result is the equivalent model in Figure 7b. We

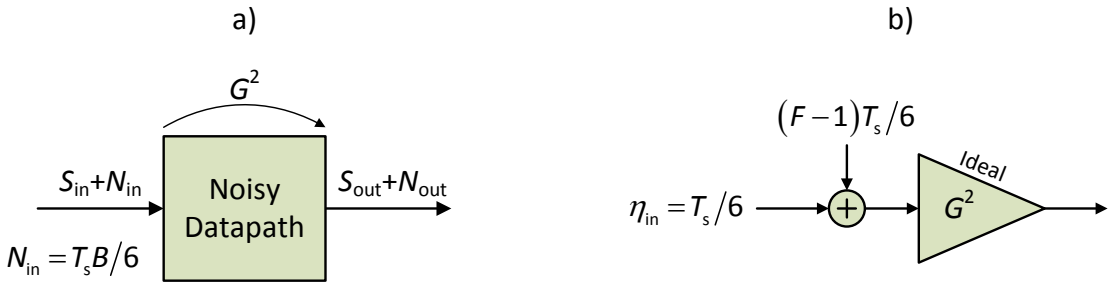

Figure 7 Noise figure in the digital domain. a) Basic setup. b) Equivalent system model with input-referred excess noise PSD.

conclude that the only real difference between "analog noise figure" and "digital noise figure" is that the input noise level kT in the analog domain is replaced by $T_s/6$ in the digital domain.

NOISE FIGURE CALCULATIONS

Now that we have a definition of noise figure for digital systems, it is time to put this new tool to work! Consider first the system in Figure 8. This is a very common DSP structure: a noiseless datapath followed by a single round operation. Because sample rate conversion is so common in DSP systems, we allow the input sample rate $f_{s,in}$ to be different from the output sample rate $f_{s,out}$ and define the *rate conversion factor* of the system as $R = f_{s,out}/f_{s,in} = T_{s,in}/T_{s,out}$. To calculate the noise factor of the system, we first note that the only contributor to the output excess noise is the round operation at the output. It follows directly that $\eta_{ex} = T_{s,out}/6$. Further, from the noise factor definition above, we have $\eta_{in} = T_{s,in}/6$. Substituting these results into Eq. (5), the noise factor is immediately obtained as

$$F = 1 + \frac{T_{s,out}/6}{G^2 T_{s,in}/6} = 1 + \frac{1}{G^2 R} \tag{7}$$

where G is the amplitude gain of the system, inside the frequency band occupied by the user signal. In those cases where G cannot be considered constant within the user band, the power gain G^2 must be calculated first, by averaging the squared magnitude of the frequency response across this band. However, digital filters often have such a modest passband ripple that G can be approximated by the gain at center of the passband (for baseband filters, the DC gain).

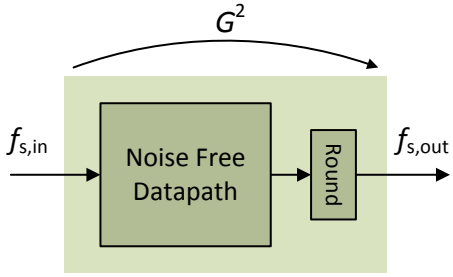

Figure 8 DSP system with one round operation.

Now consider our old friend the decimation filter in Figure 1. Notice that this filter has precisely the structure shown in Figure 8, which means that its noise factor can be calculated using Eq. (7). In fact, *all* decimation filters of FIR and CIC type can (and generally should) be designed this way, and we can therefore conclude that the simple noise factor formula above applies to a surprisingly large class of common DSP structures. To illustrate the usefulness of this formula, let us calculate the noise figure of the decimation filter in Figure 1 with the parameter values listed in Table 1. The frequency response is plotted in Figure 9. This filter was designed to achieve a stopband attenuation of 80 dB and a normalized passband edge of 0.219. The passband ripple is negligible, which means that the DC gain can be used to calculate the noise factor. The DC gain from the input of the filter to the output of the last adder (bitwidth w_{acc}) is given by

$$G_0 = c_0 + 2\sum_{k=1}^{4} c_k \tag{8}$$

To obtain the total DC gain in the decimation filter, the attenuation in the round operation must be included:

$$G = G_0 2^{-(w_{sat} - w_{out})} \tag{9}$$

With the parameter values in Table 1, the total DC gain evaluates to $G = 7968 \cdot 2^{-13} = 0.973$. Also, since this is a 2x decimation filter, we have $R = 1/2$. We can now use Eq. (7) to obtain the noise factor: $F = 1 + 2/0.973^2 = 3.11$. The corresponding noise figure is $NF = 10\log_{10}(3.11) = 4.93$ dB.

There is more to be said about this example, however. The coefficients of a FIR filter are usually scaled to make the internal gain G_0 equal to or slightly less than a power of two, which means that we can write $G_0 \approx 2^{n_0}$ for some integer n_0. In the present case we have $G_0 = 7968 \approx 2^{13}$ and, consequently, $n_0 = 13$. To ensure a small probability of clipping in the saturate operation, the saturation bitwidth must be set to accommodate the increase in signal level caused by the internal gain G_0. This is done by letting $w_{sat} = w_{in} + n_0$, although special circumstances will sometimes warrant a small adjustment of this nominal setting. In the general case, we therefore have

$$w_{sat} = w_{in} + n_0 + \Delta \tag{10}$$

where Δ is a small integer used to modify the signal headroom (defined as the maximum value supported by the bitwidth divided by the signal peak value). We refer to Δ as a *headroom adjustment*. Suppose for instance that $\Delta = -1$. The bitwidth growth from the input of the filter to the output of the

Table 1 Parameter Settings For the 2x FIR Decimation Filter in Figure 1.

w_{in}	15
w_{acc}	29
w_{sat}	27
w_{out}	14
c_0	3984
c_1	−22
c_2	143
c_3	−550
c_4	2421

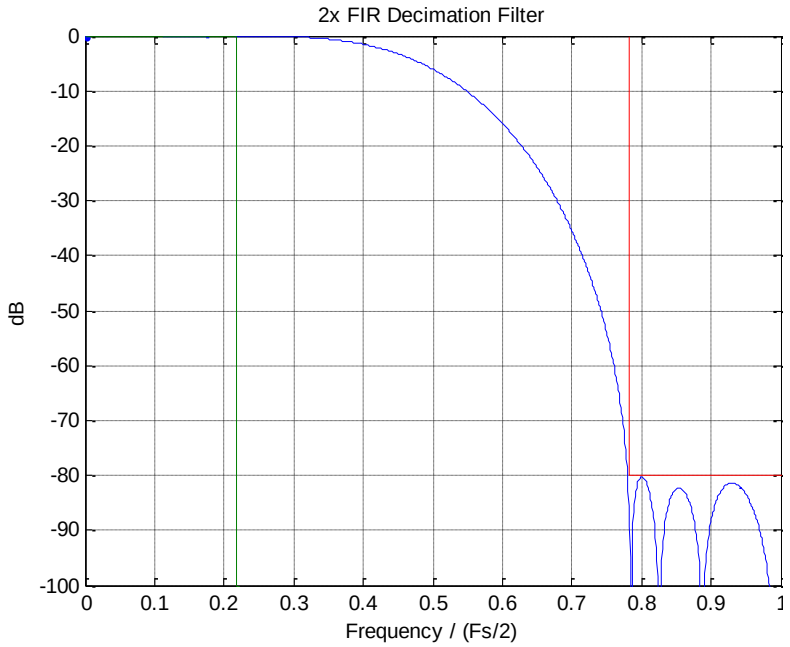

Figure 9 Frequency response of the 2x FIR decimation filter in Figure 1 with parameters as in Table 1.

saturate operation is then $n_0 - 1$ bits. However, the signal level has grown by a factor $G_0 \approx 2^{n_0}$, resulting in a headroom reduction by 6 dB. In the general case, the headroom change in dB is 6Δ.

Aided by this new insight, we can now approximate the total DC gain in Eq. (9) as

$$G \approx 2^{n_0} \cdot 2^{-(w_{sat} - w_{out})} = 2^{n_0} \cdot 2^{-(w_{in} + n_0 + \Delta - w_{out})} = 2^{-\Delta} \cdot 2^{w_{out} - w_{in}} \tag{11}$$

which, after substituting into Eq. (7), gives the following approximation of the noise factor:

$$F \approx 1 + 2^{2\Delta+1} \cdot 4^{w_{in} - w_{out}} \tag{12}$$

A closer look at the parameter settings in Table 1 reveals that Δ in fact equals −1 in this case (since w_{sat} = 27 = 15 + 13 − 1). Note that a negative headroom adjustment reduces (improves) the noise factor. In a digital down-converter, it is often possible to make a negative headroom adjustment in at least one decimator block because of the drop in signal level that occurs when out-of-band interferers are eliminated by the decimation filters. (We will take a closer look at this and other issues relating to datapath design in an accompanying white paper.) The validity of the approximation in Eq. (12) is easily verified by substituting w_{in} = 15, w_{out} = 14 and Δ = −1, which yields $F \approx 3$ and $NF \approx 4.77$ dB.

Some important observations can be made from the last example. First, notice in Eqs. (11) and (12) that both the gain and the noise factor depend on the *difference* between the input bitwidth and the output bitwidth, an observation that applies to virtually all DSP blocks used in a wireless receiver (that is, virtually all DSP blocks can be *designed* to have this property). This means that for a given input bitwidth, the noise factor can be improved (reduced) by increasing the output bitwidth. In fact, by making the output bitwidth large enough, we can make the noise factor approach 1 (*NF* = 0 dB), effectively obtaining a noiseless system! Of course, when something sounds too good to be true it usually is, and this is no exception. The problem here is hardware cost. Increasing the output bitwidth in one block means that the bitwidths in all the downstream blocks must be increased by the same

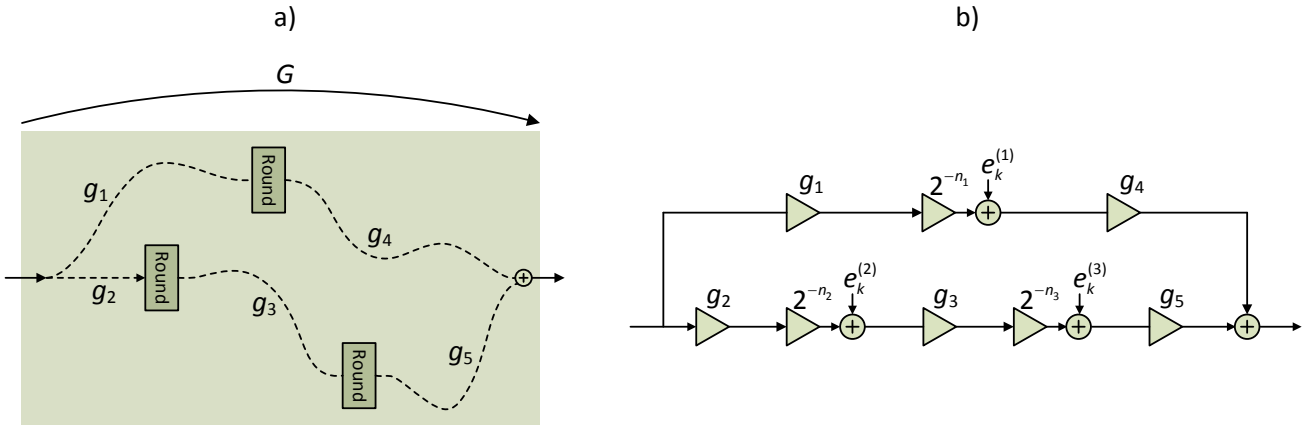

Figure 10 Noise figure calculation in the general case. a) Identify all round/truncate operations and path gains. b) Draw an equivalent signal flow graph.

number of bits, otherwise the noise factors of these blocks would be affected. For a block located in the beginning of the signal path, the added hardware cost in the downstream blocks could be substantial. For this reason, DSP blocks are often (but not always) designed with small gain factors and noise figures > 3 dB. Another observation from the example above is that the noise figure can be calculated without knowledge of the sample rates in the system. Although the input-referred excess noise clearly depends on the input sample rate, as shown in Figure 7b, the noise figure does not. To calculate the noise figure of a system, it is sufficient to know the sample rate conversion factor R.

The general procedure for calculating the noise factor of a digital system can be summarized as follows:

1) Identify all round/truncate operations and signal path gains, as illustrated in Figure 10a.
2) Add up the noise contributions from the round/truncate operations to obtain the output-referred excess noise η_{ex}.
3) Calculate the total gain G from system input to output.
4) Substitute η_{ex} and G into the noise factor formula in Eq. (5).

It is often useful to draw an equivalent signal flow graph (SFG) of the system that identifies all the noise sources and gain factors, as shown in Figure 10b. Depending on the algorithm, the path gains may be signed numbers and may also be random variables, which must be taken into account when calculating the system gain G. Note that it is not necessary to have a definition of the datapath architecture in order to draw the SFG. All that is required is a precise *algorithm* definition, including all round/truncate operations and signal bitwidths. To illustrate this important point, let us calculate the noise figure for the CORDIC rotator block in the DDC in Figure 5. The algorithm is shown in block diagram format in Figure 11 and the parameter settings are given in Table 2. Recall that the CORDIC rotator is used as a combined synthesizer and mixer in the DDC. The input phase bitwidth (W_{phase}) and the number of

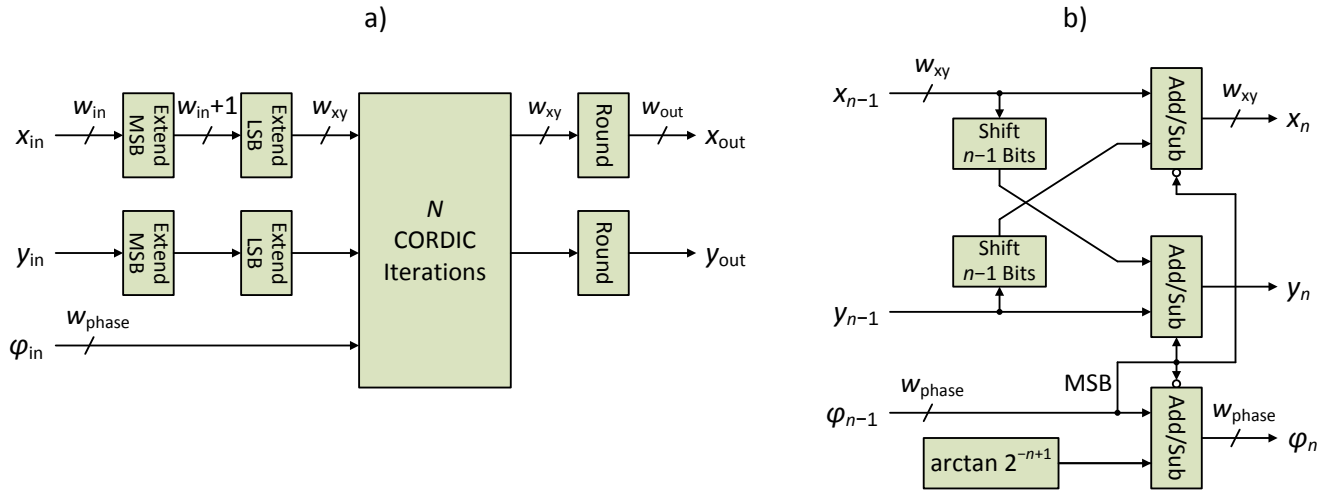

Figure 11 CORDIC rotator block diagram. a) Bitwidth manipulation at input and output. b) Iteration number n.

Table 2 Parameter Settings for the CORDIC Rotator Block.

w_{in}	14
w_{xy}	18
w_{out}	14
w_{phase}	16
N	14

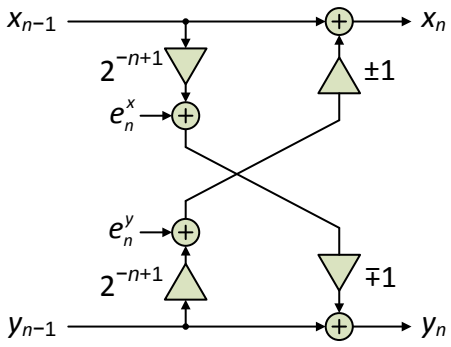

Figure 12 Signal flow graph of a CORDIC iteration.

iterations (N), which together determine the spur level in this type of synthesizer, were chosen to achieve a spurious free dynamic range (SFDR) of at least 80 dB (the actual SFDR is −84.4 dB). The CORDIC algorithm rotates the phase of a complex input sample $x_{in} + jy_{in}$ by performing a sequence of elementary angle rotations (referred to as iterations), the sum of which is made to converge towards the desired rotation angle. For a detailed treatment of the CORDIC algorithm, see references [2], [5] and [9]. Looking first at the block diagram in Figure 11a, we see that the N iterations are carried out after the bitwidth has been extended from w_{in} to w_{xy}. This extension is carried out in two steps. First, a new *most* significant bit is added (by sign extension) to absorb the signal gain inside the CORDIC iterations. Next, another $w_{xy}-w_{in}-1$ *least* significant bits are added (by zero padding) to raise the signal level by a factor $2^{W_{xy}-W_{in}-1}$ prior to the CORDIC iterations. As demonstrated below, this will improve the noise figure of the CORDIC rotator block. After N iterations, the final x and y values are rounded to the output bitwidth w_{out}. Figure 11b defines a CORDIC iteration. Notice that the cross terms are either added or subtracted, depending on the sign bit of the residual phase value obtained from the previous iteration. For our present purpose, we can assume that the sign bit is a random variable that is independent of the sign bits in the other iterations. There is no bitwidth growth in the Add/Sub operations. Also note that the right-shift operations perform truncation, not rounding. The nature of the CORDIC algorithm prevents the formation of a bias at the output, so rounding is not required here. As mentioned earlier, the only difference between truncation and rounding is the bias; the noise power is the same. To get a clearer view of how noise is generated and accumulated inside the CORDIC iterations, we replace the block diagram in Figure 11b with the equivalent SFG shown in Figure 12. The ±1 factors appearing on the crossing signal paths account for the random switching between addition and subtraction in the algorithm and should be treated as random variables. This serves to de-correlate the two signals that converge in the adder, meaning that the gains along these signal paths add in power rather than in amplitude. The signal gain in one quadrature arm (for instance, from input x_{n-1} to output x_n) in the nth iteration is therefore given by

$$g_n = \sqrt{1+2^{-2(n-1)}} \quad , n = 1,2,\ldots,N \tag{13}$$

and the total signal gain after N iterations is obtained as

$$G_{\text{iter}} = \prod_{n=1}^{N} g_n \tag{14}$$

This gain factor is well approximated by 1.647 for all *N* values of practical interest. The total block gain in each quadrature arm is obtained by noting that the 'Extend LSB' operation at the input has a gain of $2^{w_{\text{xy}} - w_{\text{in}} - 1}$, whereas the round operation at the output has a gain of $2^{-(w_{\text{xy}} - w_{\text{out}})}$. The total gain of the CORDIC rotator block is therefore

$$G = 2^{w_{\text{xy}} - w_{\text{in}} - 1} \cdot G_{\text{iter}} \cdot 2^{-(w_{\text{xy}} - w_{\text{out}})} = 2^{w_{\text{out}} - w_{\text{in}} - 1} \prod_{n=1}^{N} g_n \approx 0.82 \cdot 2^{w_{\text{out}} - w_{\text{in}}} \tag{15}$$

in each quadrature arm. We note in passing that the block gain is a function of the difference between the input and output bitwidths, just like in the previous example. The diagram in Figure 13 shows the noise sources and gain factors associated with the *N* iterations. (This is not a signal flow diagram, only an aid in computing the total noise PSD). As before, we have $\eta_0 = T_s/6$. Observe that the first iteration (*n* = 1) does not perform any shift and therefore does not contribute to the excess noise. Let η_{iter} denote the output excess noise PSD in each quadrature arm after the *N*th iteration. Adding up all the noise contributions in the diagram, we get

$$\eta_{\text{iter}} = \eta_0 \left(g_3^2 \cdot g_4^2 \cdot \ldots \cdot g_N^2 + g_4^2 \cdot g_5^2 \cdot \ldots \cdot g_N^2 + \ldots + g_{N-1}^2 \cdot g_N^2 + g_N^2 + 1 \right) = \eta_0 \left(1 + \sum_{n=3}^{N} \prod_{k=n}^{N} g_k^2 \right) \tag{16}$$

The expression inside the parenthesis grows slowly with increasing *N* and evaluates to 13.11 for *N* = 14. To obtain the total excess noise level at the output of the block, we must also include the (squared) signal gain and the noise contribution from the round operation at the output. This gives

$$\eta_{\text{ex}} = \eta_{\text{iter}} 4^{-(w_{\text{xy}} - w_{\text{out}})} + \eta_0 = \eta_0 \left\{ 1 + 4^{-(w_{\text{xy}} - w_{\text{out}})} \left(1 + \sum_{n=3}^{N} \prod_{k=n}^{N} g_k^2 \right) \right\} \tag{17}$$

Finally, we substitute Eqs. (15) and (17) into Eq. (5) to obtain the noise factor of the CORDIC rotator,

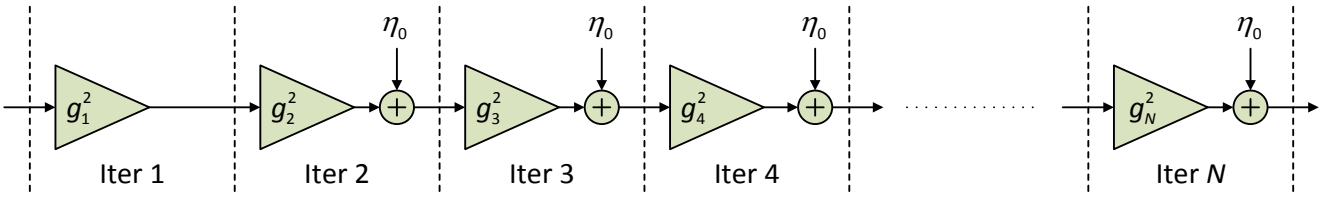

Figure 13 Diagram for calculating the output excess noise level in a CORDIC algorithm (one quadrature arm).

$$F = 1 + \frac{1 + 4^{-(w_{xy}-w_{out})}\left(1+\sum_{n=3}^{N}\prod_{k=n}^{N} g_k^2\right)}{4^{w_{out}-w_{in}-1}\prod_{n=1}^{N} g_n^2} \tag{18}$$

which evaluates to $F = 2.55$ and $NF = 4.07$ dB for the parameter values in Table 2. Note that the internal bitwidth w_{xy} acts to suppress the second term in the numerator, thereby reducing the noise factor. For increasing w_{xy}, the noise factor quickly approaches the lower limit

$$F_{lim} = 1 + \frac{1}{4^{w_{out}-w_{in}-1}\prod_{n=1}^{N} g_n^2} \approx 1 + 1.48 \cdot 4^{w_{in}-w_{out}} \tag{19}$$

A good design rule is to let $w_{xy} = w_{out} + 4$ (as done in Table 2), which will bring the noise factor to within 0.13 dB of the lower limit. Substituting this design rule into Eq. (18), the noise factor of the CORDIC rotator simplifies to

$$F = 1 + \frac{1 + 4^{-4}\left(1+\sum_{n=3}^{N}\prod_{k=n}^{N} g_k^2\right)}{4^{-1}\prod_{n=1}^{N} g_n^2} \cdot 4^{w_{in}-w_{out}} = 1 + K_F \cdot 4^{w_{in}-w_{out}} \tag{20}$$

where

$$K_F = \frac{4 + \frac{1}{64}\left(1+\sum_{n=3}^{N}\prod_{k=n}^{N} g_k^2\right)}{\prod_{n=1}^{N} g_n^2} \tag{21}$$

For $N = 14$, the factor K_F evaluates to 1.55. Notice that just like in the previous example, the noise factor ends up being a function of the difference between the input and output bitwidths.

It should be evident that the noise factor derivation above is based entirely on the block diagram in Figure 11, which represents an integer algorithm, not an architecture. Several different architectures can be used to implement the CORDIC algorithm. For instance, the *N* iterations may be implemented by a single generic processing element with a datapath designed according to the block diagram Figure 11b that is executed *N* times. Another architectural solution is to use *N* different processing elements arranged in a pipeline, in which each processing element implements a specific iteration (this will increase the throughput by a factor of *N*). Neither of these architectural variations will affect the noise factor, however, since the underlying algorithm remains the same.

CASCADING DIGITAL NOISE FACTORS

So far we have focused on the calculation of noise factors for individual processing blocks. To calculate the noise factor of a more complex digital system such as the DDC in Figure 5, we must also be able to combine the noise factors of cascaded blocks into a single noise factor. Consider the cascade of two DSP blocks in Figure 14a with sample periods T_1, T_2 and T_3. In order to obtain a total noise figure for this cascade, the excess noise contributions from the two blocks must be merged into a single noise source located at the input of block 1. This is done by first referring the excess noise of block 2 to the input of block 1, as shown in Figure 14b. Next, the two excess noise PSDs are summed and the result is expressed in terms of a new noise factor F_{tot}:

$$(F_{\text{tot}}-1)T_1/6 \;=\; (F_1-1)T_1/6+\frac{(F_2-1)T_2/6}{G_1^2} \;=\; \left(F_1-1+\frac{(F_2-1)T_2/T_1}{G_1^2}\right)T_1/6 \tag{22}$$

Comparing the first and last equalities, we find that

$$F_{\text{tot}} \;=\; F_1+\frac{(F_2-1)}{G_1^2R_1} \tag{23}$$

Repeated use of the above equation gives the total noise factor of N blocks:

$$F_{\text{tot}} \;=\; F_1 \;+\; \frac{(F_2-1)}{G_1^2R_1} \;+\; \frac{(F_3-1)}{G_1^2G_2^2R_1R_2} \;+\; \ldots \;+\; \frac{(F_N-1)}{G_1^2G_2^2\cdots G_{N-1}^2R_1R_2\cdots R_{N-1}} \tag{24}$$

Note that in the special case $R_1=R_2=\ldots R_{N-1}=1$, this equation becomes the familiar formula for combining noise factors in the analog domain [7].

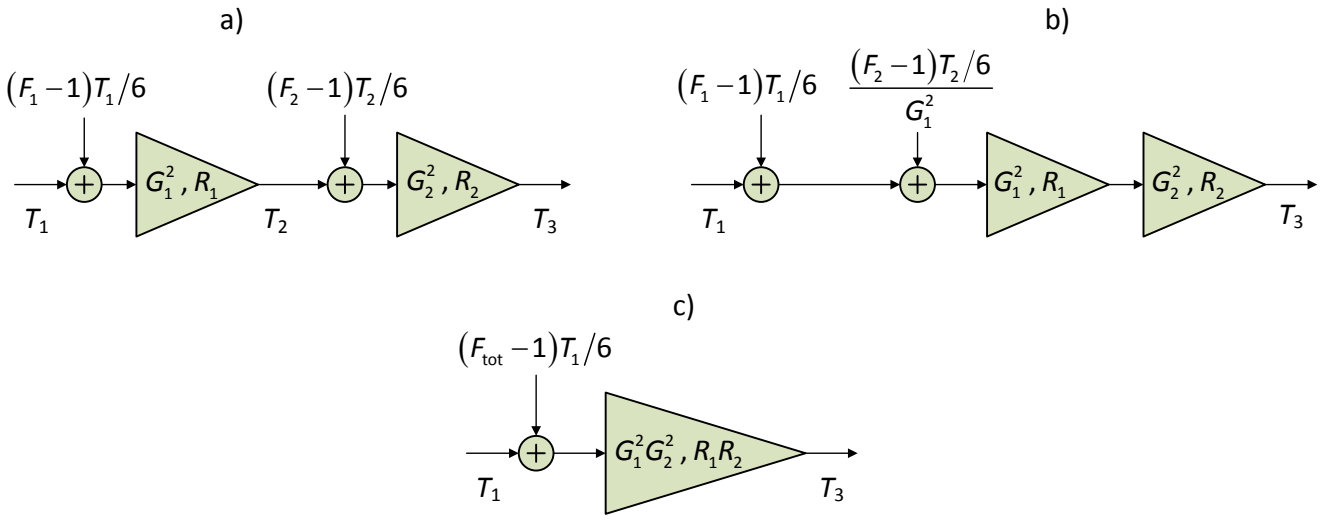

Figure 14 Combining the noise factors of two DSP blocks.

As an application of Eq. (24), consider calculating the noise figure of the DDC in Figure 5. Table 3 lists the gains, rate conversion factors and noise figures of the individual DSP blocks in the DDC. The cascaded (cumulative) noise figures, calculated using Eq. (24), are listed in the last column. For example, the cascaded noise figure of the CORDIC rotator and the first decimation stage is 6.13 dB, and the total cascaded noise figure of the whole DDC is 11.94 dB. This DDC was in fact designed to achieve a total noise figure of less than 12 dB.

Let us briefly outline the design flow that lead to the parameter settings in Table 3. Naturally, the first step is to break the DDC into individual DSP blocks and define the structure of each block. Next, all parameters that are *not* signal bitwidths are specified, including filter orders, filter coefficients and certain parameters in the CORDIC rotator. The objective of this design phase is to ensure that the stopband attenuation and SFDR requirements are satisfied. With these requirements out of the way, the final phase of the design is free to focus on the dynamic range in the signal path. As a preliminary step, headroom adjustments are specified for some of the blocks, based on the estimated drop in signal level as adjacent channels and other interferers are rejected by the decimation filters. In the last and final step, the noise figure requirement is addressed by setting the bitwidths along the signal path. How can we set the signal bitwidths to meet a specific noise figure requirement? It is actually simpler than it might appear. When deriving the noise figures of the FIR decimation filter and the CORDIC rotator in the previous section, we observed that in both cases the block gain G as well as the noise factor F were functions of the input and output bitwidths (see Eqs. (11), (12), (15) and (20)). This observation holds true for all six DSP blocks in the DDC. In fact, virtually every linear DSP algorithm defined in integer arithmetic can be designed to have this property. Observe that the CORDIC rotator block is linear, although time-varying. Since there are (in practice) only a finite number of ways to choose the input and output bitwidths, it follows that each DSP block in the DDC has a finite set of possible (G,F) value pairs. Moreover, the output bitwidth of one block is the input bitwidth of the next, which means that the gains and noise figures of individual blocks cannot be selected independently.

Table 3 Parameter Settings for the Digital Down-Converter in Figure 5.

Block	w_{in}	w_{out}	HR Adj	Gain [dB]	Rate Conv	*NF* [dB]	Cascaded *NF* [dB]
CORDIC	14	14	0	−1.69	1	4.07	4.07
Decim 1	14	15	0	4.54	1/3	3.12	6.13
Decim 2	15	16	−1	11.59	1/3	0.82	6.46
Comp Filter	16	15	0	−10.10	1	10.51	8.89
Decim 3	15	14	−1	−0.24	1/2	4.93	11.68
Decim 4	14	16	0	11.96	1/2	0.52	11.94
DDC Total	14	16	−2	16.06	1/36	11.94	

Therefore, when designing a system like the DDC in Figure 5, the system designer does not exercise direct control of the gain and noise figure settings (as would be the case in an analog design). Instead the designer controls the *vector of inter-block bitwidths*. The inter-block bitwidths determine the gains and noise figures of the individual blocks in the DDC, which in turn determine the total (cascaded) noise figure.

The vector of inter-block bitwidths chosen for the particular design in Table 3 is listed in the third column (labeled 'w_{out}') and is also shown in Figure 5. Note that the DDC input bitwidth, which is also the input bitwidth of the CORDIC rotator, will normally be specified before the DDC is designed and should not be considered a design variable here. Starting from a set of specifications that includes the DDC input bitwidth and the maximum tolerated noise figure, an experienced system designer can easily constrain the range of each inter-block bitwidth to a small set of values. For instance, with a DDC input bitwidth of 14, it is reasonable to assume that the bitwidth at the output of the CORDIC rotator should be selected from the set {13,14,15}. Since the DDC signal path is comprised of six DSP blocks, limiting the range of each inter-block bitwidth to three values will define a design space consisting of 3^6 = 729 possible bitwidth vectors, the noise figures of which can be quickly evaluated by a computer algorithm. The design of the DDC signal path now amounts to selecting one particular vector from the design space. Many of these vectors will produce a noise figure that exceeds the required 12 dB and can therefore be discarded immediately. The remaining vectors should be evaluated based on hardware cost. Having satisfied the noise figure requirement, the designer's goal is now to select a bitwidth vector that minimizes, in some sense, the total hardware cost of the DDC. The minimization of hardware cost is outside the scope of the present discussion, but we will return to this issue in another white paper.

What if we have implemented the DDC in Table 3 with a noise figure of 11.94 dB and later find that a smaller noise figure is required? Rather than re-designing the DDC from scratch, a simpler solution is to *scale* the existing datapath to improve the noise figure. Suppose first that we increment all the signal bitwidths in the DDC datapath by one. The input bitwidth is now 15 and the output bitwidth is 17. What happens to the noise figure of the DDC when the datapath is scaled in this way? Unless the reader's answer is "nothing!", he or she may want to go back to the beginning of the previous section and read it again. Remember that both the gain and the noise figure of a DSP block are functions of the difference between its input and output bitwidths. Consequently, incrementing all signal bitwidths will

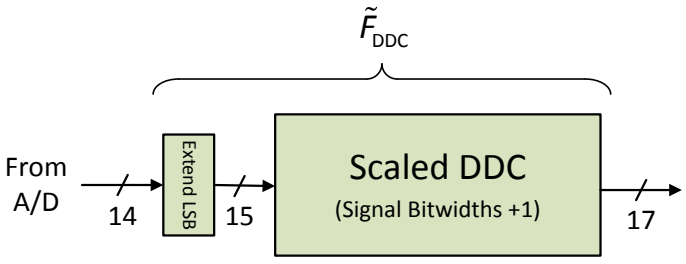

Figure 15 Scaling the DDC datapath.

have no affect on the gains and noise figures of the individual blocks in the DDC and therefore no affect on the total DDC noise figure. Then why bother? Because we can use an 'Extend LSB' operation to expand the bitwidth of the samples from the A/D converter from 14 to 15, as depicted in Figure 15. This operation is equivalent to a left-shift and therefore acts as a 6 dB gain stage in front of the DDC. To calculate the resulting noise factor, we can use the formula for cascaded noise factors in Eq. (23). The noise factor of the 'Extend LSB' operation is 1 (since this operation does not add any noise) and the amplitude gain is 2. The old DDC noise factor is 15.63 (11.94 dB). This gives a new, cascaded noise factor of

$$\tilde{F}_{\text{DDC}} = 1 + \frac{(15.63 - 1)}{2^2} = 4.66 \tag{25}$$

which corresponds to a noise figure of 6.68 dB. We have managed to reduce the DDC noise figure by 11.94 – 6.68 = 5.26 dB. Now the question is how much additional hardware is required to achieve this improvement. The hardware sizes in a DSP block, such as the amount of combinational logic, register storage and RAM storage, are roughly proportional to the input bitwidth. Of the six DSP blocks in the DDC, two blocks consume significantly more hardware resources than the rest, namely the CORDIC rotator and the fourth decimation filter. Both these blocks have an input bitwidth of 14 in the original DDC, growing to 15 in the scaled version. It is therefore reasonable to assume that scaling the DDC datapath by one bit will increase the hardware cost by roughly a factor 15/14 ≈ 1.07. Trading a 7% hardware increase for a 5.26 dB noise figure improvement is not a bad deal! The scaling process can be repeated, although with diminishing returns. For example, incrementing the signal bitwidths in the DDC a second time, while scaling the input samples by another 6 dB, will produce a noise figure of 2.82 dB. This equates to an additional reduction by 3.86 dB, for the price of an additional 7% increase in hardware cost.

REFERENCES

[1] F. J. Harris, *Multirate Signal Processing for Communication Systems*, Prentice Hall, 2004.

[2] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2004.

[3] L. Wanhammar, *DSP Integrated Circuits*, Academic Press, 1999.

[4] P. Lapsley, J. Bier, A. Shoham and E. A. Lee, *DSP Processor Fundamentals*, IEEE Press, 1997.

[5] J. Vankka, *Digital Synthesizers and Transmitters for Software Radio*, Springer, 2005.

[6] R. G. Lyons, "Turbocharging Interpolated FIR Filters", in *Streamlining Digital Signal Processing*, R. G. Lyons, Ed., IEEE Press, 2007.

[7] D. M. Pozar, *Microwave and RF Design of Wireless Systems*, John Wiley and Sons, 2001.

[8] W. F. Egan, *Practical RF System Design, John Wiley and Sons*, 2003.

[9] J. H. Reed, *Software Radio*, Prentice Hall, 2002.

