An Event-Driven System-Level Noise Analysis Methodology for RF Systems

Christoph Beyerstedt, Jonas Meier, Fabian Speicher, Ralf Wunderlich, Stefan Heinen
Chair of Integrated Analog Circuits and RF Systems
RWTH Aachen University
Aachen, Germany
mailbox@ias.rwth-aachen.de

***Abstract*—This paper presents an approach for a system-level noise simulation in the frequency domain for RF systems. The noise analysis is able to depict frequency conversion due to nonlinear or time-varying circuits and can also consider the signal processing in the digital part. Therefore, it is possible to analyze the noise from all parts of the system at a point of choice, e.g. directly at the demodulator input. The approach is based on analog models of the genuine circuit level implementation used for system level exploration or verification purpose. It can be integrated in a conventional system level simulation with nearly no overhead. The noise analysis is implemented on top of an already existing event-driven RF simulation approach which uses a combination of SystemVerilog and C++ for modeling. MATLAB is used for post-processing and visualization of the results.**

***Index Terms*—Noise, mixed-signal, RF, event-driven, modeling, verification**

I. Introduction

Modern integrated RF systems have to fulfil a variety of different specs including tight noise and nonlinearity requirements. Usually, the RF systems are designed using a spreadsheet or, for more accuracy, parts of the system are simulated. For a complete system simulation, the simulator needs to handle the high frequencies in the gigahertz range on the one hand and on the other hand a large digital circuitry. Conventional RF simulation methods like the harmonic balance or the periodic steady state analysis usually fail when it comes to large systems with multiple excitation frequencies and a transient analysis is taking an unacceptable long time on a SPICE-like simulator. Therefore, a simulation method which suits both the RF and the digital parts is required. It has been shown, that an event-driven simulation of the system-on-chip (SoC) with real number models (RNM) of the analog and RF blocks is a good trade-off between the simulation speed and the accuracy [1]. While approaches to cover nonlinear effects by using so called spectral signals are already developed [2] [3], up to the authors knowledge a proper noise analysis method for event-driven simulators similar to a SPICE-like simulator one's is missing. The goal of this work was to implement a noise analysis on top of the event-driven RF simulation method presented in [2]. The analysis calculates the impact from the different noise sources on specific nodes. Frequency conversion due to nonlinear effects or time-varying circuits is considered. It is possible to split the resulting noise signal into the different noise sources to see which part of the circuit has the greatest influence. Furthermore, not only the analog but also the signal processing of the digital circuitry is taken into account. In this paper, the following notations are used: Large scale signals which influences the operation point of a circuit or system will be printed in capital letters whereas small scale signals which have no influence on an operation point will be denoted with small letters. Vectors are marked with an underscore and matrices are printed bold. The Dirac distribution is represented by a delta symbol δ and the $\star$ denotes the convolution operator.

II. Analysis concept

In the here presented analysis the common assumption is made that the noise level is small enough to neglect its influence on the large signal operating point of the system. The influence of noise on one or more nodes in the system is analyzed in two phases: The simulation phase and the presentation and post-processing phase. In the simulation phase, the algorithm starts at the nodes of interest and recursively looks for the noise sources which determine the noise at these nodes. The algorithm is exemplary explained using a generic receiver shown in Fig. 1. The analog-to-digital converter's (ADC's) input noise in a certain bandwidth is analyzed stepping backwards through the receiver structure calling recursively a *noise()* function, which collects the noise contributing in the analyzed frequency range from the current block and looks for additional noise sources further down the chain. The linear transfer functions as well as noise folding through nonlinearities (shown in Fig. 2) or time-varying circuits like mixers is taken into account. Figure 3 exemplarily describes the gathering of the different noise contributions. A linear, frequency dependent and a static, nonlinear block are cascaded. The nonlinear block performs a frequency conversion with the conversion function $T(f) = a_0 + a_{-1}\delta(f + f_1) + a_1\delta(f - f_1)$ and the linear block has the transfer function $H(s)$. The analysis steps backwards through the blocks and recursively calls the noise functions for the preceding blocks at the frequencies the noise contributions are coming from. The returned values are weighted with the transfer function or the factors of the conversion functions. Due to the frequency conversion in the nonlinear block, noise contributions from f_0 and $f_0 \pm f_1$ of preceding blocks must be taken into account. The output referred internal noise of the block is added afterwards. The

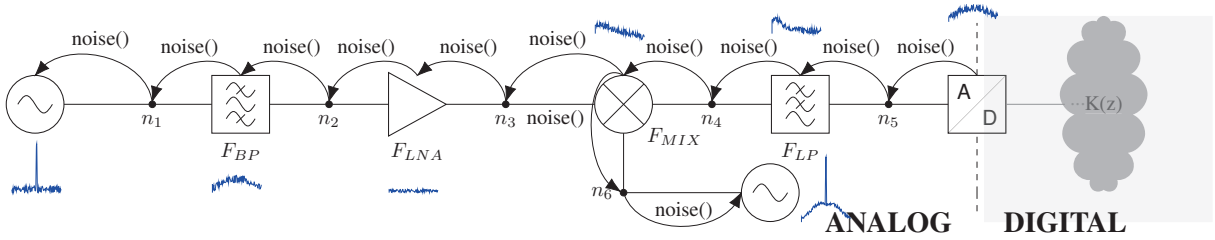

Fig. 1: Generic RF receiver.

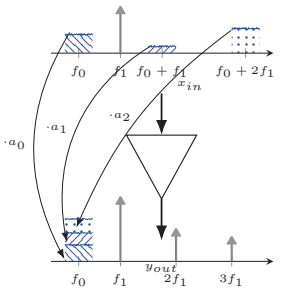

Fig. 2: Noise folding due to nonlinearities. The output noise in the observed output band is composed of the weighted input noise components from different frequency bands downconverted by the large scale input signal.

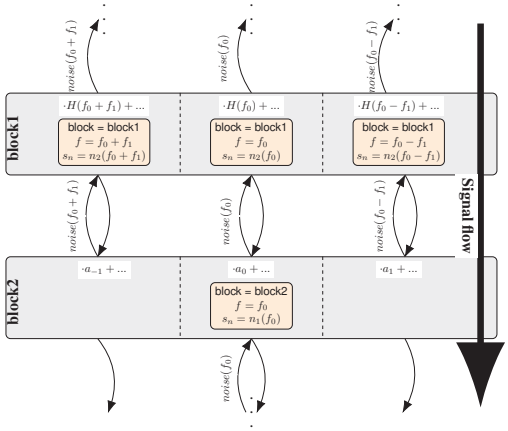

Fig. 3: Cascade of a linear, frequency dependent block (block1) and a static nonlinear block (block2).

recursion stops if a block with no inputs, e.g. a source or a node with no connected output ports is found. In the simulation phase, each noise contribution is regarded as independent and uncorrelated complex-valued small signal component to take multipath propagation and cancellation effects correctly into account. In the presentation phase, the spectral density n_k of the collected noise contributions are calculated by taking the squared absolute value of the complex-valued small signals $s_{noise,k}$ of each noise contributors (eq. 1).

$$n_k(f) = |s_{noise,k}(f)|^2. \tag{1}$$

A. Analog/RF blocks

The output signal s_{out} for linear circuits can be calculated with the knowledge of the transfer function $H(s)$ of the linear block and the input signal s_{in} (eq. 2).

$$s_{out}(f) = H(f) \cdot s_{in}(f). \tag{2}$$

In case of a linear circuit, there is no frequency translation from input to output noise thus the small signal of each noise component passing such a block is weighted by the transfer function evaluated at the frequency of the noise contribution.

For nonlinear circuits, the small signal conversion function for an arbitrary nonlinear block is examined. It is considered, that the transfer function of the block can be described by a nonlinear function $F(\xi)$. It is further assumed that the block is excited by a signal $X(t) = S(t) + s(t)$ which is the sum of a large signal $S(t)$ and a small signal $s(t)$. The output signal $Y(t)$ is calculated according to eq. 3.

$$Y(t) = F(X(t)) = F(S(t) + s(t)). \tag{3}$$

and can be approximated by the Taylor series by eq. 4.

$$Y(t) \approx F(S(t)) + \frac{dF(\xi)}{d\xi}\Big|_{\xi=S(t)} \cdot s(t). \tag{4}$$

In the frequency domain, the output signal $s_{out}(f)$ is computed by eq. 5:

$$\begin{aligned} Y(f) &= \mathcal{F}\{F(S(t))\} + \underbrace{\mathcal{F}\Big\{ \frac{dF(\xi)}{d\xi}\Big|_{\xi=S(t)} \Big\}}_{T(f)} \star \mathcal{F}\{s(t)\} \\ \Rightarrow s_{out}(f) &= T(f) \star s(f). \end{aligned} \tag{5}$$

The simulation method in [2] describes the signals as quasi-periodic signals in their steady-state for every point in time of the simulation which is a good assumption for high frequency signals with slowly varying carrier and also used in other simulation techniques like envelope transient harmonic balance [4]. Thus, the input signal S and the transfer function T can be expressed as sum of weighted Dirac distributions with the weights a_n. The small signal output $s_{out}(f)$ can be calculated according to eq. 6:

$$s_{out}(f) = \Big(\sum_n a_n \delta(f - f_n) \Big) \star s(f) = \sum_n a_n s(f - f_n) \tag{6}$$

In Fig. 2 the noise folding is exemplarily shown for an amplifier which is excited with a sinusoidal signal with the frequency f_1. The output noise signal in a certain frequency band around f_0 is composed of the input noise signal around f_0 weighted with a factor a_0, which equals the amplification of the block, the input noise signal around $f_0 + f_1$ weighted with a_1 and the input noise signal around $f_0 + 2f_1$ weighted with a_2. The above shown calculations can be generalized to circuits

with multiple inputs as well, e.g. mixers. In eq. 3, the signals X, S and s are replaced by a vectorial representation where each vector entry represents an input signal. The function F is dependent on the input vector and instead of the a simple derivative the gradient of F is used.

B. Mixed-signal feedback systems

One has to be careful when describing feedback loops in event-driven simulators because it can lead to infinite loops in the simulation. For the here presented noise analysis method, purely analog loops without delay will lead to this problem. But often, a RF system consist not only of analog components but has a large digital part as well which interacts with parts of the analog circuitry. A $\Delta\Sigma$-ADC or a phase-locked loop (PLL) are examples where the loop is partially in the analog and partially in the digital domain. How the proposed noise analysis method can deal with this type of mixed-signal feedback systems is shown by the example of a $\Delta\Sigma$-ADC. Figure 4 shows a $\Delta\Sigma$-ADC with an analog loop filter, an A/D converter, a feedback path in the digital domain and a digital-to-analog converter (DAC), which translates the digital feedback signal to an analog one. The noise analysis for

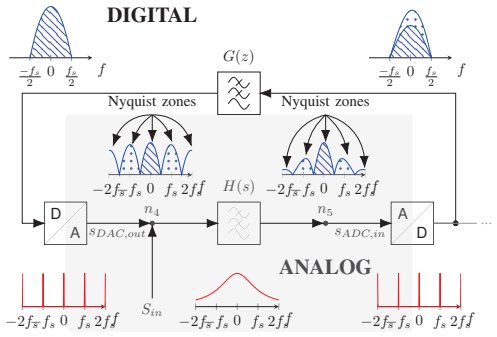

Fig. 4: Noise folding due to sampling in a $\Delta\Sigma$-ADC.

mixed-signal systems with feedback is split into two steps. In the first phase, a noise analysis for the analog part of the system is performed. The digital-to-analog converters are treated as independent noise sources with an amplitude of 1. The second phase of the noise analysis takes place in the post-processing step. The small signal originating from the DAC at the ADC input is extracted and can be identified as small signal transfer functions from the DAC to the ADC. Now, the noise propagation in the digital has to be considered. The transfer function from the ADC output to the DAC in the feedback path is usually known to the user and is often only a scaling factor or a delay but also every other linear transfer function can be considered by this method. Due to the sampling with a certain rate f_s, signal parts outside of the first Nyquist zone are folded in the $-\frac{f_s}{2}, \frac{f_s}{2}$ band. Regarding the input signal as vector with the signal parts split in the different Nyquist zones as the vector entries, the folding due to the sampling in the A/D process can be described as follows:

$$s_{ADC,out}(f) = \underbrace{\begin{pmatrix} k & .. & k & .. & k \end{pmatrix}}_{\boldsymbol{H_{ADC}}} \cdot \begin{pmatrix} s_{ADC,in}(f - nf_s) \\ .. \\ s_{ADC,in}(f) \\ .. \\ s_{ADC,in}(f + nf_s) \end{pmatrix}, \tag{7}$$

with the gain k of the quantizer. The digital-to-analog conversion process can be similarly described by eq. 8:

$$\begin{aligned} \underline{s}_{DAC,out} &= \boldsymbol{H_{DAC}} \cdot s_{DAC,in}(f) \\ \boldsymbol{H_{DAC}} &= (c..c..c)^T, \end{aligned} \tag{8}$$

with the gain c of the DAC. The DAC input can be calculated according to eq. 9.

$$s_{DAC,in}(f) = s_{ADC,out}(f) \cdot G(z)\big|_{z=exp(j2\pi f)} \tag{9}$$

The output of the analog loop filter can be determined according to eq. 10. The transfer function is equal to the simulated DAC's noise signal which can be split in the Nyquist zones to get the matrix entries $H(f + kf_s)$.

$$\underline{s}_{LF,out} = \underbrace{\boldsymbol{diag}(H(f - nf_s), .., H(f), .., H(f + nf_s))}_{\boldsymbol{H_{LF}}} \cdot \underline{s}_{LF,in} \tag{10}$$

Eq. 7 - eq. 10 leads to the eq. 11. The closed loop signal $\underline{s}_{cl}$ corresponds to the signal $\underline{s}_{ol}$ which describes an ADC's input signal without considering the loop. The conversion matrix $\boldsymbol{M}$ gives the relationship between $\underline{s}_{cl}$ and $\underline{s}_{ol}$.

$$\begin{aligned} \underline{s}_{ol} &= \underbrace{[\mathbf{1} - \boldsymbol{H_{LF}} \cdot \boldsymbol{H_{DAC}} \cdot G(e^{j2\pi f T_s}) \cdot \boldsymbol{H_{ADC}}]}_{\boldsymbol{M}^{-1}} \cdot \underline{s}_{cl} \\ \Rightarrow \underline{s}_{cl} &= \boldsymbol{M} \cdot \underline{s}_{ol}. \end{aligned} \tag{11}$$

This matrix can be used to compute the input referred noise at the ADC's input by using the cyclostationary noise conversion theory derived in [5] and [6]. It is stated that the noise spectrum due to a conversion with a transfer matrix results in eq. 12

$$\boldsymbol{N_{cl}} = \boldsymbol{M} \cdot \boldsymbol{N_{ol}} \cdot \boldsymbol{M}^{\dagger}, \tag{12}$$

where $\boldsymbol{N_{cl}}$ and $\boldsymbol{N_{ol}}$ are Fourier-transformed autocorrelation matrices of the real ADC's input noise, respectively the simulated ADC's input noise without feedback loop. If the noise sources are uncorrelated, the matrix $\boldsymbol{N_{ol}}$ is a matrix with entries only on the main diagonal. $\boldsymbol{M}^{\dagger}$ denotes the conjugate transpose of $\boldsymbol{M}$. The resulting closed loop spectral noise densities for the different Nyquist zones are located on the main diagonal of the matrix $\boldsymbol{N_{cl}}$. The other matrix entries describe the cross correlation between the noise signals.

1) Quantization noise: It is considered that the quantization noise is equally distributed over the first Nyquist zone and behaves like white noise. The quantization noise is added as extra noise contribution and its spectral density n_Q is calculated as shown in eq. 13 [7], with the sampling frequency f_s, and the quantization step size Δ.

$$n_Q(f) = \begin{cases} \frac{\Delta^2}{12 f_s} & -\frac{f_s}{2} \leq f < \frac{f_s}{2} \\ 0 & \text{else} \end{cases} \tag{13}$$

C. Digital signal processing

In the post-processing phase of the analysis it is also possible to consider the influence of digital signal processing on the noise contributions. This is for example useful if the noise at the demodulator's input needs to be known after digital filtering of the ADC's output signal. For this purpose, the ADC's input noise spectral densities of the different noise sources are calculated and then folded in the first Nyquist area (analogous to eq. 7). The resulting noise spectral densities can be computed by multiplying the ADC's output noise spectral densities with the absolute squared value of the transfer function $K(z)$ of the digital signal processing which must be known to the user. At the moment, in the digital domain only linear, time-invariant transfer characteristics are considered but the analysis can be extended to other systems as well.

III. Implementation

A. Event-driven simulation phase

The here proposed noise analysis methodology is built on top of an event-driven RF simulation method presented in [2]. Therefore, a short introduction in this method is given for a better understanding. In [2], SystemVerilog (SV) models which replace the design's building blocks just above the transistor level are used for simulation. To extend the modeling possibilities of SV regarding analog circuits, the direct programming interface (DPI) is used to connect C++ classes to the SV models. In C++, the richness of libraries and already implemented algorithm allows an accurate modeling of analog behavior. When starting the simulation, all blocks, ports and nets on SV side are linked to special C++ classes representing the specific object. In the simulation flow, SV serves for event-handling and initial creation of the C++ objects. Every object has a pointer on the preceding and subsequent objects. Therefore, it is possible to start from a net, e.g. at the end of a receiver chain, and recursively go back through the design using these pointers which is used for the noise analysis. The C++ objects are implemented in a special C++ class hierarchy (see. [2]) with a root class and all other class are derived from that class in a tree structure. This simplifies the implementation of the noise analysis for already existing designs because the needed extensions only need to be implemented once on the highest possible parent class and the behavior will be inherited by the child classes. The linear and nonlinear transfer characteristics are stored in special C++ objects. Among other properties the nonlinear objects have a method for analytical calculation of their derivative. The nonlinearity could for example stored as polynomial and the derivative could be computed straightforwardly from its coefficients. The signals in the simulation environment are represented as a special signal type which can carry a multi-tone RF signal with multiple fundamentals and different numbers of harmonics. Each frequency component is represented as baseband equivalent signal [1]. A signal $S(t)$ can be described as shown in eq. 14. The baseband equivalent signal components $I_k(t_{ev}) + jQ_k(t_{ev})$ and the center frequencies $\omega_k(t_{ev})$ only needs to be updated at the sampling points t_{ev} of the baseband signal components, which highly reduces the sampling rate for the simulation because there is no need to sample at radio frequency.

$$S(t) = \sum_k (I_k(t_{ev}) + jQ_k(t_{ev})) \cdot \exp(j\omega_k(t_{ev})t). \quad (14)$$

With the help of a multidimensional Fourier transform, one can easily switch between time and frequency domain in the signal representation depending on which description fits best for the particular operation on the signal. In [2] a nonlinear LNA and a mixer is given as an example. The nonlinear operations are performed in the time domain whereas the final signal representation is given in the frequency domain to show the generated harmonics and intermodulation products. The noise analysis is implemented on top of the already available modeling and simulation flow. Therefore, nearly no overhead is generated for setup and model creation. The analysis itself is implemented in the C++ environment while the result presentation and the post-processing is done in MATLAB. A special signal type called *AMS_SmallSignal* was developed which carries the noise components and on which operations like addition, a multiplication with a constant, a frequency shift or the processing with a linear transfer function can be performed. The *AMS_SmallSignal* signal type is composed of a vector of the different signal component objects *AMS_SmallSignalComponent*. The *AMS_SmallSignalComponent* class possess a variable *origin*. It stores the block where the signal was created and the original frequency of the signal component in this block. When evaluating the noise signal at the measurement point, the user can decompose the signal by source block and frequency so that it can be found out from which block and which frequency band the noise contributions come from. Because it is assumed that the noise sources are uncorrelated, only the amplitude value of noise components with identical origin block and frequency can be added. Other components are added by attaching the *AMS_SmallSignalComponent* objects to the vector of the resulting *AMS_SmallSignal* object. In eq. 15, the addition of two noise signals is illustrated. The different noise components consist of the origin variable with the source block S^k, the frequency f_i of the noise contribution at its source and the signal value a_i^k. The noise analysis algorithm is started at a by the user determined moment. On this event, a noise evaluation function is called, which starts with the lowest frequency to analyze. The function calls the *noise* function of the input ports, which recursively calls the *noise* function of the attached nets, attached output ports and their related blocks. At the time of the analysis, a snapshot of the large scale signals is taken and these specific signals are used to calculate the transfer characteristics. If the block is nonlinear or time-varying, the conversion function T is calculated via the Fourier transform of the transfer function's derivative with the input signal as function's arguments. For this purpose, the spectral input signal is transformed via IFFT in the time-domain, the derivative of the transfer function is calculated with the

$$\begin{aligned}&\big[(<S^0,f_0>,a_0^0),(<S^0,f_1>,a_1^0),..,(<S^k,f_j>,a_j^k),..\big]\oplus\big[(<S^1,f_0>,b_0^1),(<S^1,f_1>,b_1^1),..,(<S^k,f_j>,b_j^k),..\big]\\&\big[(<S^0,f_0>,a_0^0),(<S^0,f_1>,a_1^0),(<S^1,f_0>,b_0^1),(<S^1,f_1>,b_1^1),..,(<S^k,f_j>,a_j^k+b_j^k),..\big]\end{aligned}\tag{15}$$

derivative method of the C++ nonlinear class and evaluated at the input signal values. The resulting signal is transformed back with the FFT to the spectral domain and corresponds to the conversion function T (see. Fig. 5). The FFTW3 library [8] was used for FFT and IFFT operations. If the block is

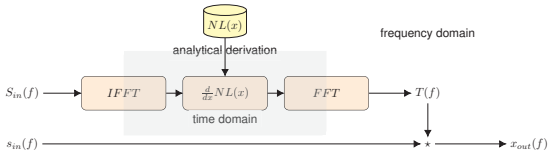

Fig. 5: Calculation of the conversion function $T(f)$.

linear and frequency-dependent, the transfer function has to be evaluated for every new frequency of each noise-component in the noise signal. After the conversion functions respectively the transfer functions are calculated, the noise contributions coming from the inputs and the block itself are added up taking the calculated conversion factors and frequency shifts, respectively the weighting through the transfer function into account. The returned *AMS_SmallSignal*-noise signal contains the amplitudes from all small signal noise components of each block and each frequency band. The results are stored in a *.csv*-file which can later be read-in by MATLAB. After the analysis for this frequency point is finished, the frequency is increased by the defined step size and the next analysis is started until the stop frequency is reached. The noise sources in the individual blocks can be either set by a numerical value or the block can load a noise file from simulation or measurement to cover noise spectra which have an arbitrary shape.

B. Presentation/post-processing implementation

The result file is read-in by a MATLAB graphical user interface (GUI) in which the noise components can be selected and plotted. Furthermore, MATLAB does the post-processing of the noise data. As described in subsection II-B, it is possible to calculate the closed loop noise response of a mixed-signal circuit. For this purpose, the read-in noise contributions are split in the different Nyquist zones (eq. 16). The superscript illustrates in which Nyquist area the component is and the subscript corresponds to the frequency in the first Nyquist zone the component is folded to.

$$\begin{aligned}&[\underbrace{{}^{-n}_{f_0}s\quad {}^{-n}_{f_1}s\quad ..}_{\text{-nth Nyquist zone}}\quad \underbrace{{}^{-n+1}_{f_0}s\quad {}^{-n+1}_{f_1}s\quad ..}_{\text{-nth+1 Nyquist zone}}\quad \underbrace{{}^{1}_{f_0}s\quad {}^{1}_{f_1}s\quad ..}_{\text{first Nyquist zone}}\quad ..]\\&\rightarrow[\underbrace{{}^{-n}_{f_0}s\quad {}^{-n+1}_{f_0}s\quad ..\quad {}^{0}_{f_0}s\quad ..}_{f_0\underline{s}}],[\underbrace{{}^{-n}_{f_1}s\quad {}^{-n+1}_{f_1}s\quad ..\quad {}^{0}_{f_1}s\quad ..}_{f_1\underline{s}}],..\end{aligned}\tag{16}$$

$\boldsymbol{N_{ol}}$ can now be determined for each frequency $f_0, f_1, ...$ as the matrix with the absolute value squared of the entries of the corresponding vectors ${}_{f_0}\underline{s}, {}_{f_1}\underline{s}, ...$ on the main diagonal. Furthermore, the matrices $\boldsymbol{H_{LF}}$ for the frequency $f_0, f_1, ...$ can be filled with the components of the split signal vector coming from the DAC according to eq. 10.

IV. Example

To show the accuracy of the proposed method, the noise at the end of a cascade of a nonlinear amplifier, a filter, a second nonlinear amplifier and a second filter was simulated, once with a harmonic balance noise analysis in SpectreRF on base of VerilogA models and once with SystemVerilog/C++ models by means of the here presented simulation methodology. The first amplifiers has the transfer function $X_{out} = 5X_{in} - 4000X_{in}^3$, the second one the transfer function $X_{out} = 5X_{in} - 750X_{in}^3$ and the linear blocks are first order lowpass filters with $f_{3dB} = 100\,\mathrm{MHz}$ and a gain of 1 at $0\,\mathrm{Hz}$. A single noise source was placed in front of the chain and the structure was exited with a sinusoidal signal with an amplitude of $4\,\mathrm{mV}$ and a frequency of $400\,\mathrm{MHz}$. In Fig. 6, the noise contributions from the different frequency bands the noise is converted from are compared. For the sake of clarity, only components from one side of the spectrum are shown. It can be seen, that the results from the proposed method matches the SpectreRF results accurately. Due to numerical inaccuracies the maximum deviation is $0.335\,\mathrm{dB}$. Furthermore,

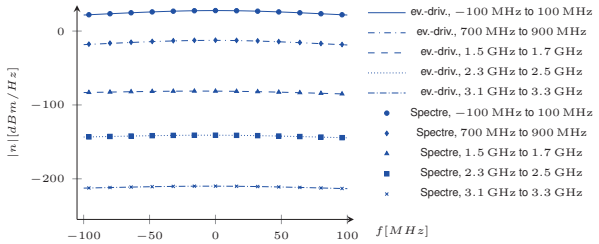

Fig. 6: Comparison of simulation results.

the implemented noise analysis methodology was applied to a generic RF receiver as shown in Fig. 1. It consists of a preselection filter, a low noise amplifier, a mixer driven by a local oscillator and a continuous time $\Delta\Sigma$-ADC as shown in Fig. 4 with a comparator as quantizer. The preselection filter is a bandpass with a center frequency of $2.4\,\mathrm{GHz}$ and a bandwidth of $100\,\mathrm{MHz}$. The LNA is modelled as nonlinear amplifier with the transfer function $X_{out} = 5X_{in} - 4000X_{in}^3$. The mixer model simply multiplies the input and the LO signal and the transfer function of the feedback filter in the $\Delta\Sigma$-ADC is $H(s) = \frac{100}{s}$. The comparator is modelled as single bit quantizer with a high gain ($k = 1e4$). The input signal of the receiver is sinusoidal with the amplitude $A = 10\,\mathrm{mV}$ and the frequency $f_1 = 2.4\,\mathrm{GHz}$. The LO's fundamental is

$f_{LO} = 2.4\,\mathrm{GHz}$ and the amplitude is $1\,\mathrm{V}$. The first harmonic of the LO signal has an amplitude of $0.1\,\mathrm{V}$. It is assumed that the LNA's input referred noise has a peak at $4\,\mathrm{MHz}$, for example due to substrate noise. Figure 7 shows the most relevant contributions to the ADC's input noise assuming an open loop in the $\Delta\Sigma$-ADC. It can be seen, that the noise peaks from the LNA's internal noise has been converted to the ADC's input due to the LNA's nonlinearity and the first harmonic of the LO signal. After closing the loop virtually, aliasing components at multiples of the clock frequency ($32\,\mathrm{MHz}$) arise due to the sampling process in the ADC and the feedback DAC (Fig. 8). The corresponding ADC's output noise spectrum is shown in Fig. 9. For better visibility, only the frequency range from $-6\,\mathrm{MHz}$ to $6\,\mathrm{MHz}$ is plotted. Also, the quantization noise n_Q of the ADC is now shown. To dampen the peaks at $\pm 4\,\mathrm{MHz}$, a digital lowpass filter with the transfer function $G(z) = \frac{1}{8}(1 + z^{-1} + z^{-2} + ... + z^{-7})$ is applied. The resulting noise spectrum is shown in Fig. 10. The noise analysis with 5000 frequency points was computed on a Intel i7-6700 CPU with $3.4\,\mathrm{GHz}$ and 16GB memory. It took less than 5 seconds for the simulation and the post-processing phase. Due to the mixed-signal nature of the system, a comparison to a SpectreRF harmonic balance noise simulation was not possible.

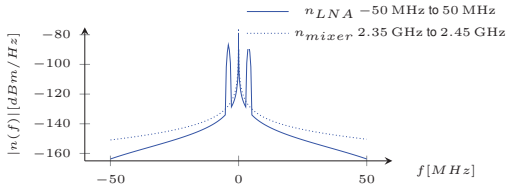

Fig. 7: Noise at ADC input with open loop.

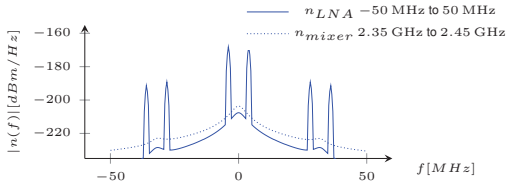

Fig. 8: Noise at ADC input with closed loop.

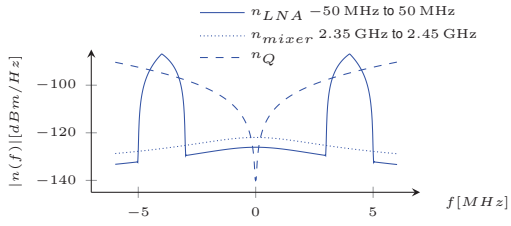

Fig. 9: Noise at ADC output.

V. Conclusion

In this paper, an approach for a system-level noise simulation in the frequency domain for RF receiver is presented which takes frequency conversion due to large scale signals and time varying circuits into account. Furthermore, also digital signal processing and mixed-signal feedback loops can be considered. This allows an analysis of the complete system. The noise contributions are separated by their origin which helps identifying the most critical sources. Due to the integration of the analysis in an event-driven environment, large systems can be quickly analyzed for different test cases. Already available models for the RF system can be reused for this analysis with a little additional source code. The proposed method was compared to a SpectreRF noise simulation and the results showed a strong match. Additionally, a noise scenario in a generic homodyne receiver with a $\Delta\Sigma$-ADC was examined showing the possibility of the analysis to handle mixed-signal circuits. The short simulation duration indicates that also the analysis of more complex systems is possible. The approach is based on the small signal transfer characteristics calculated from the large signal excitations. Therefore, it is also possible to implement a system-level small signal or a sensitivity analysis in a similar fashion.

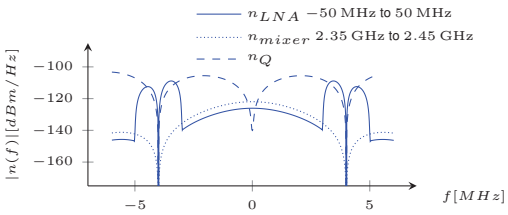

Fig. 10: Digitally filtered noise.

References

[1] J. E. Chen, "A modeling methodology for verifying functionality of a wireless chip," in *2009 IEEE Behavioral Modeling and Simulation Workshop*, pp. 96–101, Sept 2009.

[2] F. Speicher, J. Meier, C. Beyerstedt, R. Wunderlich, and S. Heinen, "Advanced modeling methodology for expedient rf soc verification and performance estimation," in *2018 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, July 2018.

[3] Z. Chen, Z. Liu, L. Liao, R. Wunderlich, and S. Heinen, "A mixed-domain modeling method for rf systems," in *2015 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, Sep. 2015.

[4] J. C. Pedro and N. B. Carvalho, *Intermodulation distortion in microwave and wireless circuits*. Boston, MA : Artech House, 2003.

[5] J. Roychowdhury, D. Long, and P. Feldmann, "Cyclostationary noise analysis of large rf circuits with multi-tone excitations," in *Proceedings of CICC 97 - Custom Integrated Circuits Conference*, IEEE.

[6] D. Held and A. Kerr, "Conversion loss and noise of microwave and millimeterwave mixers: Part 1–theory," *IEEE Transactions on Microwave Theory and Techniques*, vol. 26, pp. 49–55, feb 1978.

[7] R. Lyons, *Understanding Digital Signal Processing (3rd Edition)*. 08 2011.

[8] M. Frigo and S. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, pp. 216–231, feb 2005.