Hindawi Publishing Corporation
International Journal of Microwave Science and Technology
Volume 2015, Article ID 757591, 7 pages
http://dx.doi.org/10.1155/2015/757591

Research Article

Implementation of a Cross-Spectrum FFT Analyzer for a Phase-Noise Test System in a Low-Cost FPGA

Patrick Fleischmann,[1] Heinz Mathis,[1] Jakub Kucera,[2] and Stefan Dahinden[2]

[1] *Institute for Communication Systems ICOM, University of Applied Sciences of Eastern Switzerland, 8640 Rapperswil, Switzerland*
[2] *Anapico Ltd., 8152 Glattbrugg, Switzerland*

Correspondence should be addressed to Heinz Mathis; heinz.mathis@hsr.ch

Received 1 June 2015; Revised 1 September 2015; Accepted 2 September 2015

Academic Editor: Giovanni Ghione

The cross-correlation method allows phase-noise measurements of high-quality devices with very low noise levels, using reference sources with higher noise levels than the device under test. To implement this method, a phase-noise analyzer needs to compute the cross-spectral density, that is, the Fourier transform of the cross-correlation, of two time series over a wide frequency range, from fractions of Hz to tens of MHz. Furthermore, the analyzer requires a high dynamic range to accommodate the phase noise of high-quality oscillators that may fall off by more than 100 dB from close-in noise to the noise floor at large frequency offsets. This paper describes the efficient implementation of a cross-spectrum analyzer in a low-cost FPGA, as part of a modern phase-noise analyzer with very fast measurement time.

1. Introduction

Phase noise, the random phase fluctuations of a periodic signal, is an important parameter to characterize high-frequency devices, in particular reference oscillators and microwave synthesizers. Phase noise is important because it has a large impact on the performance of many applications [1], such as high-speed communications [2], radar, and precision navigation [3].

There exist various methods for phase-noise measurement, of which the cross-correlation method achieves the best sensitivity and the widest frequency range, at the expense of a relatively complex setup [4–6]. Various fully automated integrated phase-noise analyzers that implement this method are available on the market, for example, the Anapico APPH6040/20G (7 or 26 GHz), the Agilent E5052B, or the Rohde & Schwarz FSUP.

In this paper, we will first review the basics of modelling phase noise and give an outline of the associated terminology. After that, the basics of phase-noise measurement methods will be discussed. Finally, the design and implementation of a novel cross-spectrum FFT analyzer in a low-cost FPGA are described in detail.

2. Phase-Noise Modelling and Terminology

A perfect fixed-frequency oscillator without noise would produce a perfect sine wave. In reality, any oscillator is affected by internal random noise processes, such as thermal and flicker noise, as well as aging and external influences, such as temperature and vibrations. To be able to characterize the phase noise of a real oscillator, its output signal can be modelled by

$$V(t) = \left(V_0 + \epsilon(t)\right) \sin\left(2\pi\nu_0 t + \phi(t)\right), \qquad (1)$$

where $\phi(t)$ denotes the random phase fluctuations, V_0 the nominal amplitude, and ν_0 the nominal frequency. The random amplitude fluctuations $\epsilon(t)$ can generally be neglected for high-quality oscillators [7].

Phase fluctuations are characterized in the frequency-domain by their one-sided power spectral density $S_\phi(f)$, defined as

$$S_\phi(f) = \phi^2(f)\frac{1}{\mathrm{BW}}\left[\mathrm{rad}^2/\mathrm{Hz}\right], \qquad (2)$$

where $\phi(f)$ is the root mean squared (rms) phase fluctuation and BW is the measurement bandwidth. The Fourier frequency f ranges from 0 to ∞ in this one-sided spectrum which contains the power of both sidebands around the nominal frequency [8].

The standard measure of phase noise in the frequency-domain is the single-sideband phase-noise $\mathscr{L}(f)$ defined by the IEEE standard 1139 [8] as

$$\mathscr{L}(f) \equiv \frac{1}{2} S_\phi(f). \quad (3)$$

$\mathscr{L}(f)$ is usually specified in dBc/Hz, that is, dB below the carrier in a 1 Hz bandwidth.

3. Measuring Phase Noise

The simplest phase-noise measurement method is the direct spectrum measurement using a spectrum analyzer. However, this method is only suitable for sources with relatively high noise because the phase noise of the spectrum analyzer must be significantly lower than the noise of the device under test. Furthermore, the dynamic range of this method is very limited because the carrier signal is not suppressed.

Another class of measurement methods are the frequency-discriminator methods. The advantage of these methods is that they do not require a reference oscillator. However, these methods cannot achieve the sensitivity of the phase detector methods described below [6].

The method whose implementation is described in Section 4 is the cross-correlation method. Therefore, the remainder of this section will first describe the single-channel quadrature method, which is the basis of the cross-correlation method, before the cross-correlation method is explained.

3.1. Quadrature Method Phase-Noise Measurement. The basic principle of the quadrature method is depicted in Figure 1. The device under test (DUT) signal is mixed with a reference (REF) signal at the same frequency using a phase detector mixer. A phase locked loop (PLL) ensures that the DUT and REF signals stay in phase quadrature during the measurement, so that the output of the mixer (after low-pass filtering) will be approximately proportional to the phase fluctuations of the input signals. Thus, the mixer operates as a phase detector. The output voltage can then be measured using a baseband FFT spectrum analyzer [6].

The main disadvantage and the limiting factor for the measurement accuracy of this method is that the reference source must exhibit significantly lower phase noise than the DUT because any noise on the reference is added to the DUT noise. One possible solution of this problem is to use two identical sources as DUT and reference, so that the two sources contribute the same amount of noise to the output. The measured noise power is then twice the noise power of a single source, assuming the phase noise of the two sources is uncorrelated.

Another disadvantage of the quadrature method is that the PLL forms a high-pass filter for the phase noise, as it inherently tries to compensate for phase fluctuations. Therefore, the PLL loop bandwidth must be made substantially lower than the lowest required noise frequency. Depending on the frequency stability of the DUT and reference, the loop bandwidth cannot be made arbitrarily small because the PLL might lose lock [6]. Therefore, the high-pass effect of the PLL is often canceled after the measurement, using signal processing.

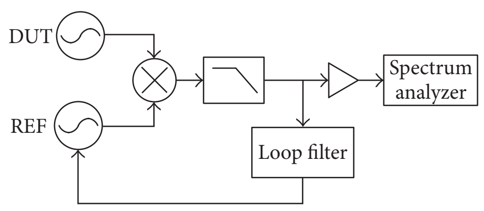


Figure 1: Block diagram of the quadrature method. The reference oscillator (REF) is phase-locked to the device under test (DUT). The output of the phase detector (mixer) is first amplified with a low noise amplifier and then measured with a baseband spectrum analyzer.

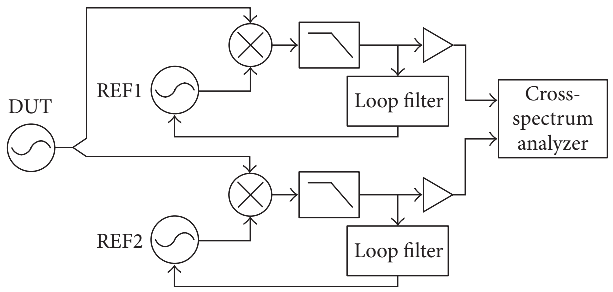


Figure 2: Block diagram of the cross-correlation method. This method uses two independent reference sources (REF1, REF2) and phase detectors (mixers). The two baseband outputs are measured with a cross-spectrum analyzer.

3.2. The Cross-Correlation Method. The cross-correlation method solves the problem of the reference source noise by using two independent reference sources and phase-detector circuits (see Figure 2). The basic reasoning behind this method is that the noise of the reference sources can be averaged away by cross-correlating and averaging the outputs of the two mixers [4, 5]. In practice, the noise floor can be improved by about 20 dB over the single-channel quadrature method [9].

In a cross-spectrum FFT analyzer, the discrete Fourier transforms (DFTs) of the two input signals are computed and the DFTs are multiplied pointwise, taking the complex conjugate of one signal, to obtain an estimate of the cross-spectrum. Several of these cross-spectra can then be averaged.

The uncorrelated noise products will have random amplitude and phase in the DFT and will therefore be eliminated by the averaging; they will decrease proportionally to $1/\sqrt{m}$, where m denotes the number of averages, if they are completely uncorrelated. This means that the measurement sensitivity will be increased by at most 5 dB for a tenfold increase in the number of averages [9].

However, for the correlated part of the noise, the product equals the squared magnitude. Therefore, more averages

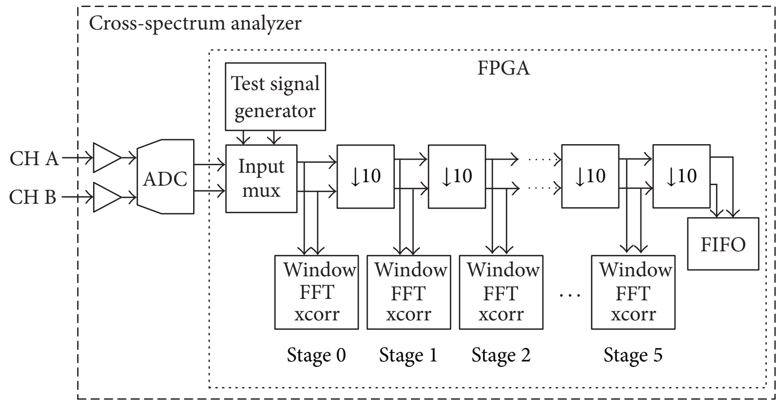


FIGURE 3: Architecture of the cross-spectrum analyzer and the signal processing inside the FPGA.

will improve the estimation of the correlated noise, that is, the phase noise of the DUT. Once the uncorrelated noise is averaged away, the variance of the power estimate will decrease proportionally to $1/m$ [5, 10].

In effect, we can accurately measure a noise source that has a lower noise level than the noise floor of a single measurement channel by using the cross-correlation method. An extensive tutorial of the cross-correlation method can be found in [5].

4. FPGA Cross-Spectrum Analyzer Implementation

This section describes the implementation of a wideband cross-spectrum analyzer for phase-noise measurement in a low-cost Spartan-6 FPGA (Field Programmable Gate Array) from Xilinx.

Figure 3 shows an overview of the complete cross-spectrum analyzer. The two analog input channels A and B are connected to two independent quadrature measurement systems, as depicted in Figure 2. The inputs are digitized using a high dynamic range ADC (Analog-to-Digital Converter) operating at a sampling rate of 100 to 150 MHz. Sufficient ADC resolution is required to cope with largely changing phase-noise profiles.

Inside the FPGA, the two channels are processed by a cascade of decimators with downsampling factors of 10 and then fed to the signal processing stages to compute the cross-spectral density of the two channels. Decimating by a factor of 10 has the advantage that the resulting plot has a constant number of samples per decade. Multiple correlations are summed up in an accumulator memory for averaging in every stage. The accumulated correlations are read out via a softcore microprocessor that provides the output interface. Figure 4 shows the architecture of one signal processing stage. The first signal processing stage operates at the full ADC sampling rate of 125 MHz and each following stage operates at a sampling rate ten times smaller than the preceding stage. This architecture simultaneously produces an estimation of the cross-spectral density in multiple, logarithmically spaced frequency ranges. The samples with the lowest sampling rate,

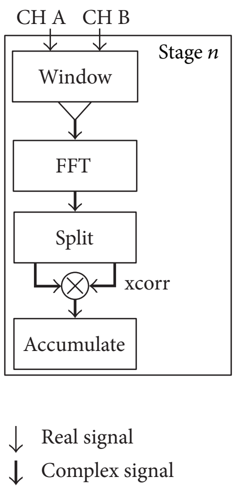


FIGURE 4: Architecture of one signal processing stage. One complex FFT block, in combination with a split block, is used to compute the DFTs of the two real input channels.

which come out of the last downsampling stage, are not processed in a hardware signal processing block but are stored in a FIFO memory (first in, first out). The FIFO memory is periodically read out by the microprocessor and the signal processing for these low-frequency samples is performed in software.

The logarithmically spaced frequency ranges are important because phase-noise power spectral densities are always plotted in a log-log scale. In the low-frequency range, the frequency resolution therefore needs to be much smaller than in the high-frequency range. When computing the DFT (Discrete Fourier Transform) of a signal, the frequency resolution is proportional to f_s/N_{DFT}, where f_s is the sampling rate and N_{DFT} is the length of the DFT in samples. To obtain a frequency resolution of 1 Hz at a sampling rate of 125 MHz, a length of $N_{\mathrm{DFT}} = 125 \cdot 10^6$ samples would be required. This is clearly infeasible on a platform with limited memory resources. However, if the FFT block for the lowest frequency range operates at a sampling rate of only 125 Hz, the frequency resolution with $N_{\mathrm{DFT}} = 1024$ is 0.122 Hz.

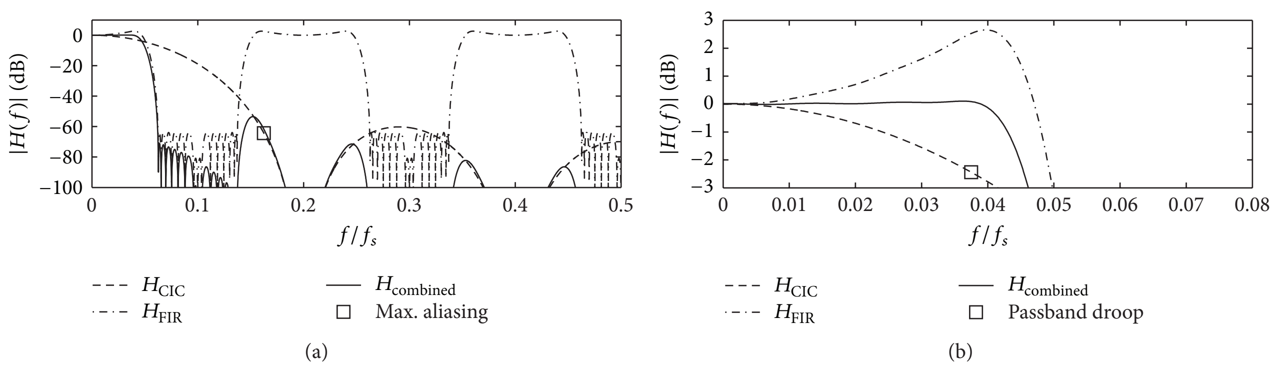


Figure 5: Frequency responses of the CIC filter, FIR filter, and combined filter. The CIC filter and FIR filter have downsampling rates of 5 and 2, respectively. The frequency axis is relative to the input sampling rate.

Table 1: Number of required multipliers for optimal FIR filter implementation, depending on the decimation-rate allocation.

R_{CIC}	R_{FIR}	N_{coeff}	N_{mult}
1	10	119	6
2	5	72	4
5	2	31	2

Another benefit of this architecture is that the measurement time to obtain one estimate of the power spectral density is dramatically reduced in the high-frequency ranges. If we again assume a frequency-resolution requirement of 1 Hz and want to estimate the spectral density using one FFT, we would require a measurement time of 1 s, because the measurement time is inversely proportional to the frequency resolution. Using the implemented architecture, we can perform multiple FFTs and correlations in the high-frequency stages in parallel, while the lowest-frequency stage may only be able to perform one correlation in the given measurement time.

4.1. Decimation. The cascade of decimators provides antialiasing filtering and downsampling for the FFT stages. The first decimator stage operates at an input sampling rate of 125 MHz and is downsampled to 12.5 MHz. It was implemented as a combination of a cascaded integrator comb (CIC) filter [11] and a subsequent finite impulse response (FIR) filter. This configuration was chosen to minimize the number of required multipliers, which are a limited resource in low-cost FPGAs.

The required specification for the decimation filter was alias suppression of at least 60 dB and passband flatness of 0.1 dB. The transition bandwidth should not exceed 50% of the output bandwidth. This means that the usable bandwidth is 75% of the total output bandwidth.

For example, if such a filter was implemented as a single FIR filter, the number of required coefficients would be over a hundred; see Table 1. The number of multipliers required for the implementation of an FIR filter in an FPGA can be estimated by dividing the number of coefficients N_{coeff} by the hardware-oversampling rate. The hardware-oversampling rate is just the hardware clock rate f_{clk}, divided by the output sampling rate of the filter $f_{s,\mathrm{out}}$. In this example, the number of multipliers is

$$N_{\mathrm{mult}} \approx \left\lceil \frac{N_{\mathrm{coeff}} f_{s,\mathrm{out}}}{f_{\mathrm{clk}}} \right\rceil = \left\lceil \frac{119 \cdot 12.5\,\mathrm{MHz}}{125\,\mathrm{MHz}} \right\rceil = 12. \quad (4)$$

If the filter coefficients are symmetric, the number of required multipliers can be approximately halved. Consequently, this filter could theoretically be implemented using six multipliers.

To reduce the number of required multipliers, it is often beneficial to choose a filter configuration consisting of a CIC filter followed by an FIR filter and distribute the overall downsampling between the two filters [12]. The advantage of CIC filters is that they do not require multipliers but only adders. Their main drawback is that the passband is not flat (passband droop). However, an FIR filter following the CIC can be used to compensate for the passband droop and also sharpen the transition from the passband to the stopband.

Theoretically there are four different possibilities to distribute the downsampling rate of 10 between two filters: $(1, 10)$, $(2, 5)$, $(5, 2)$, and $(10, 1)$. The last theoretical option would only use a CIC decimation filter with a downsampling factor of 10. However, such a filter cannot meet the requirements. The remaining three options were tested for the number of multipliers they require; see Table 1. The filter configuration with 31 symmetric FIR coefficients, which theoretically only requires two multipliers, was chosen for the implementation. Figure 5 shows the frequency response of the CIC filter, the FIR filter, and the combined filter with the points of maximum aliasing and passband droop.

The number of actually used multipliers depends on how well the algorithm that synthesizes the netlist for the FPGA can exploit the hardware oversampling and the coefficient symmetry. Furthermore, it also depends on the bit-width of the data and coefficients. For this example, the Xilinx tools generated a filter using three multipliers.

4.2. Windowing and Overlapping. Windowing is performed before the FFT to prevent spectral leaking. The implemented window is a 4-term minimum-sidelobe Blackman-Harris window that has large sidelobe suppression of 92 dB but a relatively large equivalent noise bandwidth (ENBW) of 2 bins [13]. The ENBW has to be taken into account to correct the estimated power spectral density.

To increase the number of available FFT samples for averaging, stage 3 to stage 5 employ an overlapping block in front of the windowing. This technique accelerates the convergence of the averaged power spectral densities, which is important to reduce the measurement time at the lower sampling rates.

When the squared magnitude of K independent and identically distributed samples are averaged, the variance of the average σ_{avg}^2 will decrease by

$$\frac{\sigma_{\text{avg}}^2}{\sigma_{\text{samp}}^2} = \frac{1}{K}. \tag{5}$$

If overlapping of 50% is used, the individual FFT samples are correlated. This means that the variance will decrease more slowly. A good approximation of the variance reduction (for more than ten averages) is

$$\frac{\sigma_{\text{avg}}^2}{\sigma_{\text{samp}}^2} \approx \frac{1}{K}\left[1 + 2c^2(0.5)\right], \tag{6}$$

where $c(0.5)$ is the correlation coefficient of the window for 50% overlap [10]. The overlap correlation coefficient of the window used is only $c(0.5) = 3.8\%$. This means that the variance of the average will approximately be reduced as if the individual FFT samples were uncorrelated.

4.3. Two-Channel FFT. The FFT blocks compute the Discrete Fourier Transforms (DFT) of the two windowed input sequences using intellectual property (IP) cores from Xilinx. The length of the DFT at all stages is 1024 samples; the bit-width of the input and output sequences increases substantially from the first stage to the last stages to accommodate the larger dynamic range in the low-frequency stages.

The FFT IP cores compute the standard DFT, which takes one complex-valued input sequence to produce one complex-valued output sequence. In this application, however, we need to compute the DFTs of two real-valued input sequences at the same time. The simplest solution to this problem would be to use two independent FFT cores but this would be a waste of FPGA resources.

To save resources, we use the well-known trick for computing the DFTs of two real sequences using only one complex DFT [14]. To explain how this works, we start by defining the DFT of a sequence $x(n)$ of length N as

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1. \tag{7}$$

If $x(n)$ is real-valued, $X(k)$ has a complex conjugate symmetry about $N/2$; that is,

$$X(N-k) = X^*(k), \quad k = 1, 2, \ldots, N-1. \tag{8}$$

In other words, the real component of $X(k)$ exhibits even symmetry and the complex component odd symmetry.

We can exploit this symmetry to compute the DFTs of two real sequences $x(n)$ and $y(n)$ by computing the DFT of the complex input sequence $z(n) = x(n) + jy(n)$. Because the DFT is linear, the transformed sequence is simply

$$\begin{aligned} Z(k) &= \text{DFT}\left[x(n) + jy(n)\right] \\ &= \left\{X_r(k) - Y_i(k)\right\} + j\left\{X_i(k) + Y_r(k)\right\} \\ &= Z_r(k) + jZ_i(k), \end{aligned} \tag{9}$$

where we used the indices r and i to denote the real and imaginary components. Now we can split $Z(k)$ by separating the even and odd parts of its real and complex components to get the DFTs of $x(n)$ and $y(n)$ as follows:

$$\begin{aligned} X(k) &= \frac{1}{2}\left\{Z_r(k) + Z_r(N-k)\right\} \\ &\quad + j\frac{1}{2}\left\{Z_i(k) - Z_i(N-k)\right\}, \\ Y(k) &= \frac{1}{2}\left\{Z_i(N-k) + Z_i(k)\right\} \\ &\quad + j\frac{1}{2}\left\{Z_r(N-k) - Z_r(k)\right\}. \end{aligned} \tag{10}$$

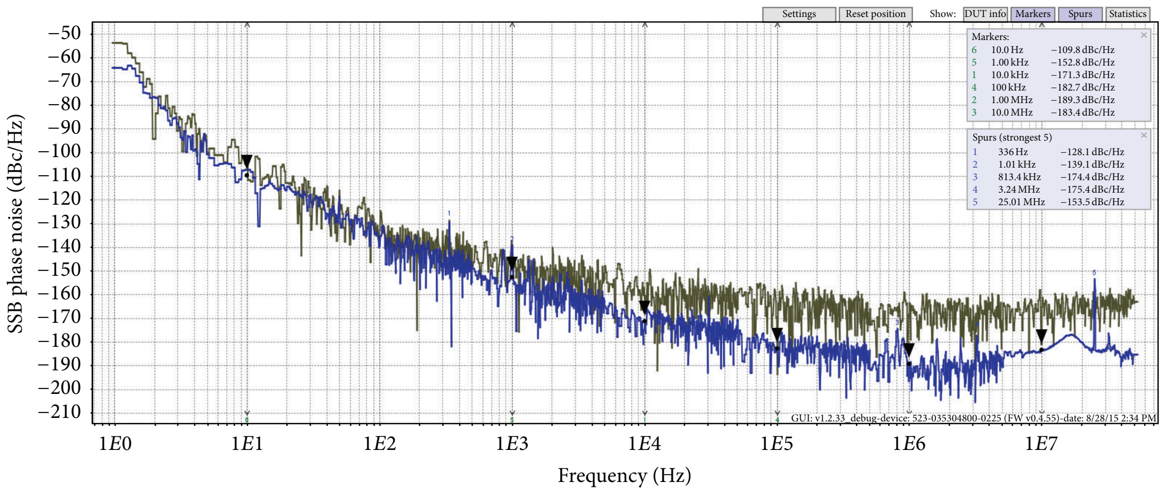
Because of their symmetry, we only need to compute $X(k)$ and $Y(k)$ for $k = 0, 1, \ldots, N/2$.

Using this algorithm, the cost of computing the DFT of two real, length-N sequences is equal to the cost of one complex length-N FFT plus two additions per complex output sample. The memory cost is also increased, compared to one complex FFT, because the complete FFT output sequence has to be stored before splitting can be performed.

4.4. Cross-Correlation and Vector-Averaging. After the splitting block, the two transformed sequences are multiplied using a complex multiplier block to compute the complex-valued cross-spectrum. Before multiplication, the imaginary part of the sequence from channel b is inverted to obtain the complex conjugate. Finally, multiple correlations are summed up in an accumulator memory which can accommodate more than 10,000 correlations. The accumulator memories of all stages can then be read out and the averaged cross-spectrum can be displayed to the user.

5. Measurements

The cross-spectrum analyzer described above was successfully implemented in the commercially available APPH6040 signal source analyzer from Anapico [15]. Figures 6 and 7 show example phase-noise measurements with the APPH6040, demonstrating the excellent sensitivity achieved by the cross-correlation method. To that end, two traces are plotted in both figures. One trace shows the result of only one correlation, whereas the other trace shows the average of many correlations, which results in a significantly lower noise floor.



Markers:

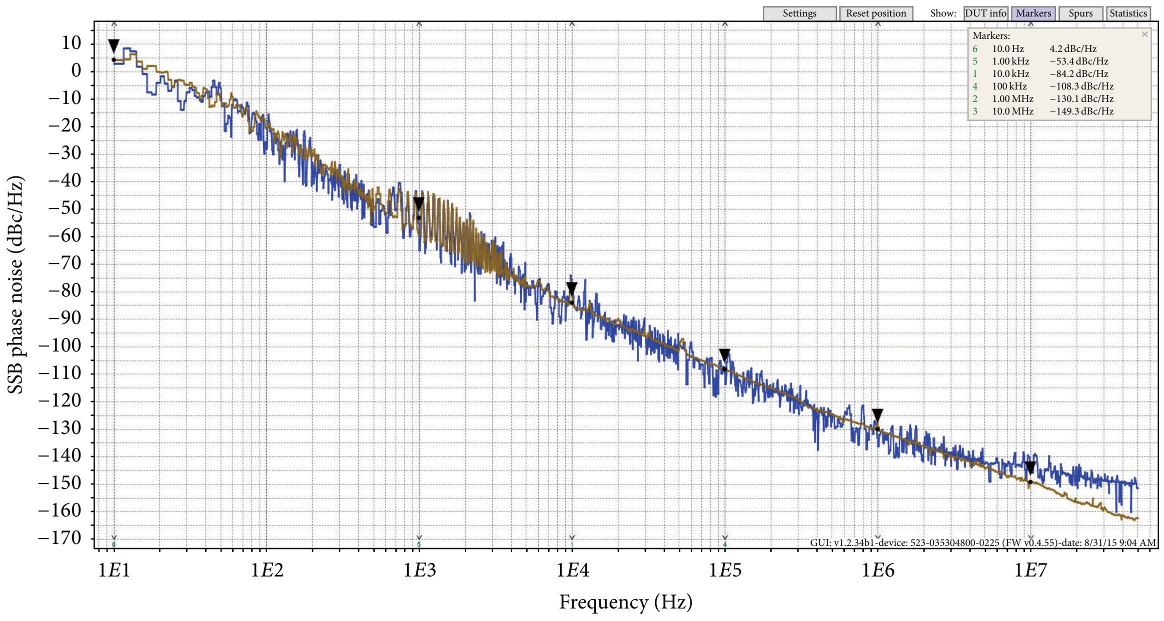
6	10.0 Hz	−109.8 dBc/Hz
5	1.00 kHz	−152.8 dBc/Hz
1	10.0 kHz	−171.3 dBc/Hz
4	100 kHz	−182.7 dBc/Hz
2	1.00 MHz	−189.3 dBc/Hz
3	10.0 MHz	−183.4 dBc/Hz

Spurs (strongest 5)

1	336 Hz	−128.1 dBc/Hz
2	1.01 kHz	−139.1 dBc/Hz
3	813.4 kHz	−174.4 dBc/Hz
4	3.24 MHz	−175.4 dBc/Hz
5	25.01 MHz	−153.5 dBc/Hz

GUI: v1.2.33_debug-device: 523-035304800-0225 (FW v0.4.55)-date: 8/28/15 2:34 PM

Figure 6: Cross-correlation phase-noise measurement with the APPH6040. The DUT was ultra-low-phase-noise 100 MHz OCXO. The grey trace shows a measurement with only one correlation. The blue trace shows a measurement with 10^6 correlations in stage 0 (5–50 MHz), 10^5 correlations in stage 1 (0.5–5 MHz), and so on. Both measurements were performed with the internal reference sources of the APPH6040. Same measurement time of ca. 11 sec for both measurements.



Markers:

6	10.0 Hz	4.2 dBc/Hz
5	1.00 kHz	−53.4 dBc/Hz
1	10.0 kHz	−84.2 dBc/Hz
4	100 kHz	−108.3 dBc/Hz
2	1.00 MHz	−130.1 dBc/Hz
3	10.0 MHz	−149.3 dBc/Hz

GUI: v1.2.34b1-device: 523-035304800-0225 (FW v0.4.55)-date: 8/31/15 9:04 AM

Figure 7: Cross-correlation phase-noise measurement of a free-running wideband VCO at 2.16 GHz with the APPH6040. Single correlation (blue) and maximum correlation (brown) per individual decades. Same measurement time of 1.2 sec for both traces.

6. Conclusion

We have presented the implementation of a novel cross-spectrum FFT analyzer architecture for phase-noise measurements, which was successfully integrated into the commercially available APPH6040 signal source analyzer from Anapico.

Several efficient signal processing techniques had to be employed to enable integration of the FFT analyzer into a low-cost FPGA. For example, the use of CIC filters for signals with high sample rates dramatically reduces the number of required hardware multipliers, a scarce resource in low-cost FPGAs. Furthermore, the successive downsampling architecture uses FFT blocks with small length to cover a large measurement bandwidth, which saves memory inside the FPGA.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

Financial support by the Swiss Commission for Technology and Innovation is gratefully acknowledged (CTI projects 11904.2 PFNM-NM and 13461.1 PFLE-NM).

References

[1] W. P. Robins, *Phase Noise in Signal Sources: Theory and Applications*, vol. 9, IET, 1984.

[2] S. Wu and Y. Bar-Ness, "OFDM systems in the presence of phase noise: consequences and solutions," *IEEE Transactions on Communications*, vol. 52, no. 11, pp. 1988–1996, 2004.

[3] W. Yu, G. Lachapelle, and S. Skone, "PLL performance for signals in the presence of thermal noise, phase noise, and ionospheric scintillation," in *Proceedings of the 19th International Technical Meeting of the Satellite Division (ION GNSS '06)*, pp. 1–17, Fort Worth, Tex, USA, September 2006.

[4] W. F. Walls, "Cross-correlation phase noise measurements," in *Proceedings of the IEEE Frequency Control Symposium*, pp. 257–261, Hershey, Pa, USA, May 1992.

[5] E. Rubiola and F. Vernotte, "The cross-spectrum experimental method," http://arxiv.org/abs/1003.0113.

[6] U. L. Rohde, A. K. Poddar, and A. M. Apte, "Getting its measure," *IEEE Microwave Magazine*, vol. 14, no. 6, pp. 73–86, 2013.

[7] J. Rutman, "Characterization of phase and frequency instabilities in precision frequency sources: fifteen years of progress," *Proceedings of the IEEE*, vol. 66, no. 9, pp. 1048–1075, 1978.

[8] J. Vig, *IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology-Random Instabilities (IEEE Std 1139-1999)*, vol. 1, IEEE, New York, NY, USA, 1999.

[9] J. Breitbarth, "Cross correlation in phase noise analysis," *Microwave Journal*, vol. 54, no. 2, pp. 78–86, 2011.

[10] P. D. Welch, "The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.

[11] E. B. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 155–162, 1981.

[12] R. Lyons, "Understanding cascaded integrator comb filters," embedded.com, 2005, http://www.embedded.com/.

[13] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[14] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, "Realvalued fast Fourier transform algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 6, pp. 849–863, 1987.

[15] *APPH6040/APPH20G Specification V2.11*, AnaPico, 2015, http://www.anapico.com/.