Implementation of BIST Technique for A to D Converters in FPGAs

M.V. Sushumna
Department of ECE, CVR College of Engineering, Hyderabad, India
Email: sushumna451@gmail.com

Abstract—**When designing an Analog to digital Converter (ADC) it is desirable to test its performance at two different points in the development process. The first is characterization and verification testing when a chip containing the ADC has been taped-out for the first time, and the second is production testing when the chip is manufactured in large scale. It is important to have a good correlation between the results of characterization and the results of production testing. This paper investigates the feasibility of using a built-in self-test to evaluate the performance of embedded ADCs in Field Programmable Gate arrays (FPGA), by using the FPGA fabric to run necessary test algorithms. The idea is to have a common base of C code for both characterization and production testing. The code can be compiled and run on a computer for a characterization test setup, but it can also be synthesized using a high-level synthesis (HLS) tool, and written to FPGA fabric as part of a built-in self-test for production testing. By using the same code base, it is easier to get a good correlation between the results, since any difference due to algorithm implementation can be ruled out. The algorithms include a static test where differential nonlinearity (DNL), integral nonlinearity (INL), offset and gain error are calculated using a sine-wave based histogram approach. A dynamic test with an FFT algorithm, that for example calculates signal-to-noise ratio (SNR) and total harmonic distortion (THD), is also included. All algorithms are based on the *IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters* (IEEE Std 1241). To generate a sine-wave test signal it is attempted to use a delta-sigma DAC implemented in the FPGA fabric. For the static test there was a perfect match of the results to 10 decimal places, between the algorithms running on a computer and on the FPGA, and for the dynamic test there was a match to two decimal places.**

Index Terms—**FPGA, built in Self Test (BIST), ADC, dynamic test, static test, linearity, DNL, INL, offset, gain error, FFT, SNR, THD, delta-sigma, sigma-delta, Digital to analog converter (DAC), high-level synthesis, HLS, IEEE Standard 1241**

I. INTRODUCTION

A field-programmable gate array is a type of programmable logic circuit, which can be used to realize integrated circuit designs. In contrast to an application specific integrated circuit (ASIC) an FPGA is not manufactured to fit a certain application, but can instead be used in a large variety of applications. One of the advantages is that the development time for a design can be much shorter for an FPGA than for an ASIC, and thus the time-to-market is shorter. Another advantage is flexibility. If there is an error in the design, the bug can be fixed and the device reprogrammed with the correct version of the design. If a product is produced in large volumes, ASICs have the advantage that they are cheaper to manufacture per unit, but since there is a high initial cost for manufacturing integrated circuits, FPGAs are more cost effective for lower volumes.

This paper explains the testing of analog-to-digital converters (ADCs) that are embedded in field-programmable gate array (FPGA) chips. By exploiting the fact that an FPGA can be reprogrammed, its fabric can be utilized as part of a built-in self-test (BIST) for the embedded ADC, and the test logic can then be removed when it is not needed anymore. If the test algorithms are written in C code, and mapped to the FPGA using high-level synthesis, then the exact same algorithms can also run on a computer as part of larger test setup. The BIST is suitable for production testing and demonstrations, and the computerized test setup is suitable for more versatile characterization testing. By using the same code base it is easier to get comparable measurement results from the two. In order to test the ADC a test signal also has to be generated, and this can possibly be done with the help of a delta-sigma modulator realized in the FPGA fabric. These two topics are using C code for an FPGA ADC test, and generating a test signal with a delta-sigma modulator in the FPGA fabric and are investigated in this paper. The C code test algorithms are mainly based on an IEEE standard for ADC testing [5], and related research papers, and the delta-sigma modulator is inspired by an application note from Xilinx[4]. C code test algorithms and Verilog code for delta-sigma modulators were developed in Xilinx's Vivado design suite.

The ADC test covered in this paper can be divided into two parts as static test and dynamic test and each of these two tests consist of a set of algorithms. The algorithms were tested with artificial data generated in Matlab. For the static test an ADC model was written using an if-statement to represent the transition levels of an ideal 4-bit ADC, and then errors were introduced by changing some of the transition level from their ideal values. The model was then used to quantize a sine-wave and the resulting ADC codes were used to test the static test algorithms. For the dynamic test a sine-wave with a specific signal-to-noise ratio was generated using the built-in additive white Gaussian noise function in Matlab. Harmonics of the signal were also added to the signal and the resulting array of data was then used to test the dynamic test algorithms. After these tests with artificial data, the algorithms were ported into C code and the C code was tested using the same artificial data to see that it matched the models in Matlab. Then the codewas synthesized into a RTL representation using HLS and tested in a co-simulation. In the co-simulation both the C code and its synthesized RTL representation are simulated using the same C code test bench. This way the results can be compared to make sure the synthesized design is working as expected. The top-levels of the BIST, one for a dynamic ADC test and one for a static ADC test, were written in Verilog. They are basically finite state machines that collect ADC data and start the test when a button is pressed. In Vivado there is a catalog of IP cores, which are ready-made pieces that can be used in designs by generating code and then instantiating it in Verilog. The HLS blocks (the synthesized C code) can be imported into this library and then instantiated into the design in the same manner as other IP cores. This way the synthesized ADC test algorithms were integrated into the Verilog top-levels. The top-level designs with instantiated HLS blocks were simulated, synthesized, implemented and generated into bitstreams using Vivado, and then tested on the FPGA. When testing them, output codes from the ADC were probed using ILAs and then used as test data in Matlab and C simulations. In this way the results from the BIST could be compared to the results of algorithms running on a computer processing the exact same data, and thus the correlation between them could be investigated.

II. ADC TESTING

This paper is based on the IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters (IEEE Std 1241). The performance measurements of interest here can be divided into two categories; static test and dynamic test. The static test measures an ADC's excursion from its ideal behavior when sampling a slowly varying (relatively low frequency) signal. The dynamic test measures the spectral purity of the of the ADC's digital output signal, when the ADC sample a relatively high frequency signal. The IEEE standard 1241 attempts to provide a common ground for ADC tests and testing methods. If test methods and definitions from this standard are used, and if they are disclosed properly when presenting the results, it is easier to compare ADCs and to reproduce measurement results. This makes the information more usable, and thus more valuable.

A. *Static test*

The idea behind what is usually referred to as a static ADC test, is to excite the ADC with an input signal of low enough frequency so that the signal can be considered a DC, or static signal. The output codes of the ADC are then compared with what is expected from an ideal ADC model. When designing a static ADC test there are a number of choices to be made. There are several, equally valid, ways to define the ideal behavior and the ADC's excursion from this ideal behavior. The static test consists of finding the transition levels of the ADC and compares these voltage levels to the ideal ones. The difference between the tested device and the ideal case is then quantified by four metrics: gain, offset, differential nonlinearity (DNL) and integral nonlinearity (INL).

i. Performance metrics

There are four metrics which are used to quantify the difference between the transfer function of the ADC under test and the ideal transfer function. These are: gain, offset, DNL and INL. Which are defined [1] as Gain is the value by which the input values are multiplied to minimize the mean square deviation from the output values. Gain error is the deviation from the ideal gain in percent. Offset is the value by which the input values are added to minimize the mean square deviation from the output values. Differential nonlinearity (DNL) is the difference between a specified code bin width and the average code bin width, divided by the average code bin width. Integral nonlinearity (INL) is the maximum difference between the ideal and actual code transition levels after correcting for gain and offset.
The gain and offset is calculated using the following equation

$$GT[k] + V_{OS} + \varepsilon[k] = LSB \cdot (k-1) + T_1 \qquad (1)$$

where G is the gain, T[k] is the measured transition levels, Vos is the offset, ε is the error for each index k and T_1 is the ideal position of the first transition level. For the mid-riser convention the ideal position of the first transition level is at one LSB above the minimum input voltage, so T1 is equal to V_{min} + 1LSB. The right-hand side of Eq. 1 is the ideal transition levels. For the first one, k is equal to 1, so it is located at T_1, which is V_{min} + 1LSB. For the second one k is equal to 2, so it is located at V_{min} + 2LSBs, etc. On the left-hand side are the measured transition levels *T* [*k*]. The task is to find the gain *G* and the offset *Vos* which minimizes the mean squared deviation from the best-fit (linear regression) line. This is done by minimizing the sum of square of the errors (SSE). In this case Eq. 1 can be rearranged as

$$\varepsilon[k] = LSB \cdot (k-1) + T_1 - GT[k] - V_{OS} \qquad (2)$$

The index k is the index of the transition levels and runs from 1 to 2^N- 1 and by assuming T_1 is equal to V_{min} + 1LSB the SSE can be expressed as

$$SSE = \sum_{k=1}^{2^N-1} \varepsilon^2[k] = \sum_{k=1}^{2^N-1} \left(LSB \cdot k + V_{\min} - GT[k] - V_{OS}\right)^2$$

This is a function of the two variables G and Vos and to find the minimum, the partial derivatives are calculated and set to zero.

$$\frac{\partial(SSE)}{\partial V_{OS}} = -2\sum_{k=1}^{2^N-1} \left(LSB \cdot k + V_{\min} - GT[k] - V_{OS}\right) = 0$$

------ (3)

$$\frac{\partial(SSE)}{\partial G} = -2\sum_{k=1}^{2^N-1}\left(LSB \cdot k + V_{\min} - GT[k] - V_{OS}\right)T[K] = 0 \quad \text{------} \quad (4)$$

Eqs. 3 and 4 form an equation system which can be solved for *G* and *Vos*. Doing so results in the following expressions for gain and offset:

$$G = \frac{LSB \cdot \left(2^N - 1\right)\left(\sum_{K=1}^{2^N-1} kT[k] - 2^{(N-1)}\sum_{K=1}^{2^N-1} T[k]\right)}{\left(2^N - 1\right)\sum_{K=1}^{2^N-1} T^2[k] - \left(\sum_{K=1}^{2^N-1} T[k]\right)^2} \quad (5)$$

$$V_{OS} = LSB \cdot 2^{(N-1)} + V_{\min} - \frac{G}{2^N - 1}\sum_{K=1}^{2^N-1} T[k] \quad (6)$$

DNL and INL are calculated with Eq. 7 and Eq. 8 respectively [2].

$$DNL[k] = \frac{G \cdot \left(T[k+1] - T[k]\right) - LSB}{LSB} \quad (7)$$

$$INL[k] = \frac{G \cdot T[k] + V_{OS} - LSB \times (k-1) - T_1}{LSB} \quad (8)$$

The DNL is the difference in code bin width *W*[*k*], after compensating for gain, to the ideal code bin width, which is 1 LSB, expressed in LSBs. The INL is the error of each transition level, after compensating for gain and offset, expressed in LSBs.

B. Dynamic test

In the dynamic test the ADC is excited with a sine-wave signal of high enough frequency so that dynamic errors occur. A number of samples from the sine-wave are collected and the discrete Fourier transform (DFT) is used to analyze the data. The DFT can be calculated using an FFT algorithm[7].

i. Performance metrics

The DFT gives the frequency spectrum of the ADC codes and a number of performance metrics can be calculated from this spectrum. An example of a spectrum obtained from the DFT is illustrated in Figure 1. The dynamic performance metrics defined in the IEEE Std 1241 are given as Total harmonic distortion (THD) is, using a pure sine wave input of specified amplitude and frequency, the root-sum-of-squares of all the harmonic distortion components including their aliases in the spectral output of the ADC. IEEE Std 1241-2010 suggest the ten first harmonics are used to estimate THD. THD is often expressed as a decibel ratio with respect to the rms amplitude of the output component at the input frequency. The total harmonic distortion (THD) is calculated by summing up the harmonics of the fundamental signal, i.e. summing up the power in frequency bins that are at integer multiples of the frequency bin of the input signal. The ratio of the distortion power to the signal power (the difference in dB) is then the THD.

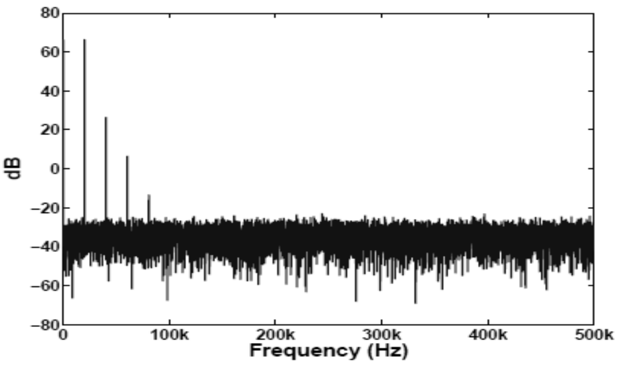

Figure 1: Spectrum, of a 20 kHz sine-wave, obtained using DFT

The signal-to-noise ratio (SNR) is calculated by summing up the noise, i.e. the power in all the frequency bins up to Nyquist frequency, except for the bins containing the DC, the fundamental signal and its harmonics. The ratio of the signal to the noise (the difference in dB) is then the SNR. Signal-to-noise ratio is the ratio of the rms amplitude of the ADC output signal to the rms amplitude of the output noise, using a pure sine wave input of specified amplitude and frequency. This does not include the harmonic distortion components that are used for the estimate of THD. Signal-to-noise-and-distortion ratio (SINAD) is the same as SNR with the only difference that the harmonics are included in the noise summation[9]. The spurious free dynamic range (SFDR) is the ratio of signal to the highest noise spur (the difference in dB) in the spectrum. Lastly the effective number of bits (ENOB) is calculated using the SINAD in the formula for SNR for an ideal ADC (Eq.9). Replacing SNR with SINAD in this formula and solving for N gives the ENOB[8].

$$SNR = 6.02N - 1.76dB \quad (9)$$

Effective number of bits is a measure of the signal-to-noise-and distortion ratio used to compare the actual ADC performance to an ideal ADC.

III. BUILT-IN SELF-TEST

Built-in self-test(BIST), means that extra circuitry is added to a design in order for the design to be able “perform operations on itself to prove correct operation” [2]. This is then combined with a so called scan technique which allows the registers in the design to be read, so that the result of the test can be examined. Because of the added circuitry, there is a circuit overhead related to the test, but because the test time can be reduced the overall system cost may be lower. A BIST in a FPGA has the advantage that it does not have the circuit overhead normally associated with a BIST. When the BIST circuitry is not needed in the design anymore it can be removed and the design re-synthesized, which is a huge advantage compared to a BIST in an ASIC. In Figure 2 a block diagram of the BIST is shown. A delta-sigma modulator in the FPGA fabric is used to generate a test signal. This signal is fed out through the I/O of the FPGA and is then filtered by an external analog filter before going to the dedicated analog inputs of the ADC. The filter is necessary to remove high-frequency quantization noise. The test signal is generated continuously and the ADC is sampling at a fixed sample rate. The top-level Verilog block consists of a finite state machine (FSM). When a start button on the KC705 board is pressed the top-level collects samples from the ADC, and when enough samples have been collected it starts algorithms that calculates the performance of the ADC based on the collected samples. These are the algorithms that were written in C code and synthesized using HLS. Once the algorithms are finished, a done signal is returned and the top-level FSM goes to an idle state where it resides until the start button is pressed again.

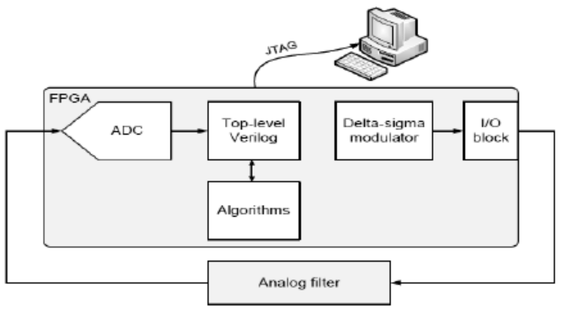

Figure 2: Block diagram of the BIST

Integrated into the design are ILAs which can be used to read the test results. Each ILA has a trigger and a capture signal that can be chosen by the designer. If the ILA is armed, this is done in Vivado's hardware manager; it starts to capture data from a designated register in the design when the trigger signal goes high. The register is then sampled by the ILA each clock cycle a chosen condition, based on the capture signal, is met. If for example the condition is that a capture signal C0 is equal to one, then the register will only be sampled at the clock cycles where "C0 = 1". If there is no capture condition set, however, the ILA will sample the register every clock cycle after the trigger goes high. Each sample that an ILA collects is stored in its memory, which consists of block RAM, and the data capture continues until the memory is full. The memory content can then be read through a JTAG interface using Vivado's hardware manager[3]. There is a data ready signal that goes high one clock cycle for each ADC conversion, and this signal can be used both as trigger and capture signal for an ILA that captures ADC codes from the ADC data register. Thus data from the ADC can be obtained in a convenient way.

One important thing to note about this setup is that there are no external active components used in the test. Typically a signal generator or external DAC would be used to generate the test signal for the ADC. The argument for not using external active components is that those components then would have to be calibrated, or their accuracy somehow guaranteed in order to ensure that the errors found in the test are indeed in the ADC itself and not the test signal. It is desirable to avoid this situation. The Specifications of test case ADC are shown in the following Table 1.The first four parameters after the resolution in Table 1 integral nonlinearity, differential nonlinearity, offset and gain errors are measures of the ADC's static performance. Regarding gain and offset error there is a setting for automatic calibration which gives smaller errors, but the un calibrated values are shown here. The sample rate can be chosen between 0.1 and 1 MS/s, but usually it is the performance at the highest possible sample rate that is of most interest. Signal to noise ratio and total harmonic distortions are measures of the ADC's dynamic performance. The ADC supports both uni-polar and bipolar mode. For the bipolar mode a fully differential input signal can be used. For a 1 MS/s sample rate a 26 MHz ADC clock frequency is needed, but a higher frequency system clock can be divided down and used as ADC clock.

TABLE I
SPECIFICATIONS OF TEST CASE ADC

Parameter	Value
Resolution (bits)	12
Integral nonlinearity (LSB)	±3
Differential nonlinearity (LSB)	±1
Offset error (LSB)	±6
Gain error (%)	±0.5
Sample rate (MS/s)	0.1 – 1
Signal-to-noise ratio (dB)	60
Total harmonic distortion (dB)	70
Unipolar input range (V)	0 – 1
Bipolar input range (V)	-0.5 – 0.5
Unipolar common mode range (V)	0 – 0.5
Bipolar common mode range (V)	0.5 – 0.6
ADC clock frequency (MHz)	1 – 26

IV. SYSTEM IMPLEMENTATION

This paper consists of two main parts; investigating signal generation with a delta-sigma modulator realized in FPGA fabric, and mapping C code algorithms for ADC testing to FPGA fabric using HLS. A second-order delta-sigma modulator was investigated because it gives a sufficient SQNR with a reasonable oversampling ratio. Its implementation in Verilog is discussed below. Two BISTs for ADC testing were developed, one for dynamic test and one for static test.

A. The Delta-Sigma DAC

The delta-sigma modulator was implemented in Verilog and a block diagram of the design can be seen in Figure 3 It consists of four adders, two flip-flops and a quantizer. The signals in the design are represented as 24-bit fixed point two's complement numbers so the buses in the design are 24 bits wide. The adders therefore add 24-bit words and the flip-flops in the diagram represent 24 parallel flip-flops each. Four of the bits are integer bits and 20 are fractional bits. It was found that four integer bits was enough to avoid overflow and 20 fractional bits gave sufficient accuracy. The quantizer is basically a comparator which feeds back +1 if the output is positive, and -1 if the output is negative. Through the output *Xout* one bit is fed out, and that is the inverted sign bit from the 24-bit representation. This way the DAC outputs 1, which corresponds to +VDD volts, if the output of the modulator is positive and 0, which corresponds to 0 volts, if the output of the modulator is negative.

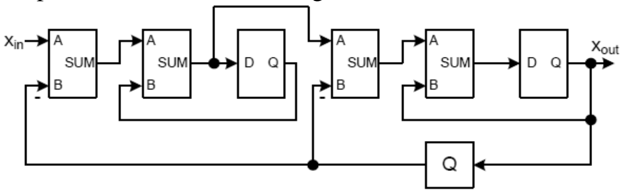

Figure 3: Block diagram of the second-order delta-sigma modulator.

The test setup used for the delta-sigma DAC can be seen in Figure 4. A direct digital synthesizer (DDS) was used to generate a half-scale digital sine-wave of 20 kHz. The DDS is a pre-made design block which is available in Vivado's IP library and can be instantiated in the design. The sine-wave codes are fed to the delta-sigma modulator with the clock rate 8 MHz . The output of the modulator goes through an I/O, traces on the KC705 board and is then fed out to a spectrum analyzer using a GPIO SMA connector. The I/O blocks of the FPGA can be set in many modes (many I/O standards) but for this experiment the choice of standard was limited to the standard LVCMOS18. Before implementing the design two settings to the I/O could be adjusted: drive strength (4-24 mA) and slew rate (fast/slow). For this experiment 8 mA drive strength and fast slew rate were chosen.

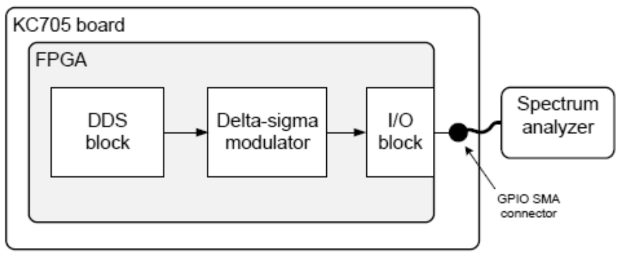

Figure 4: Setup for testing delta-sigma DAC

V. RESULTS

In order to evaluate if synthesizing C code algorithms for ADC testing with HLS is feasible, the test results between code running on a PC and the synthesized design were compared. Except for matching results between PC and BIST, measurement time and FPGA utilization was also considered.

TABLE II
COMPARISON BETWEEN STATIC BIST TEST, C CODE RUNNING ON PC AND MATLAB TEST MODEL USING PROBED HISTOGRAM.

	Matlab model using measured histogram	C code (on PC) using measured histogram	C code algorithms running on FPGA
gain error (%)	-0.1814590398	-0.1814590398	-0.1814590398
offset error (LSB)	-11.4587839949	-11.458783994	-11.458783994
DNL min (LSB)	-0.8265005955	-0.8265005955	-0.8265005955
DNL max (LSB)	0.6550577276	0.6550577276	0.6550577276
INL min (LSB)	-2.1075215934	-2.1075215934	-2.1075215934
INL max (LSB)	2.5614670437	2.5614670437	2.5614670437

In Table 2 a comparison between the BIST, C code running on a PC and the Matlab model for the static test can be seen. The ADC was set in uni-polar mode and an external signal generator was used to generate the input signal. The signal was a single-ended sine-wave with an offset of 500 mV and an amplitude of 550 mV, and the frequency was set to around 20 kHz but non-harmonically related to the sampling frequency. The frequency was somewhat arbitrarily chosen. A lower frequency might be desirable, but the purpose with this experiment was to evaluate the functionality of the BIST rather than evaluate the ADC itself. In the test 200,000 samples were collected and used to generate the histogram. In order to compare the BIST with the C code and Matlab code running on a PC, the histogram was probed with an ILA. Using Vivado's hardware manager the data from the ILA can be obtained in a CSV-file. This way the histogram from the BIST test could be used as input data in Matlab and a C code test bench. Ideally the ADC codes would be probed directly and the histogram would be regenerated with code on the PC, but the ILAs cannot save enough samples. In this comparison, using 10 decimal places, the results match exactly and it can be concluded that synthesizing the algorithms into RTL and running them on the FPGA was successful. Comparing the result to the specifications of the ADC (see Table 1) it can be seen that the test results are reasonable. All results meet specifications except for the offset error, which is probably due to that the accuracy of the offset setting of the signal generator was not good enough.

Figure 5 shows DNL and INL plots from the measurement. The INL has jumps at certain codes where the DNL is large. Note that the INL curve does not start off from zero because the best-fit approach was used and not the terminal based approach.

DOI:10.32377/cvrjst0711

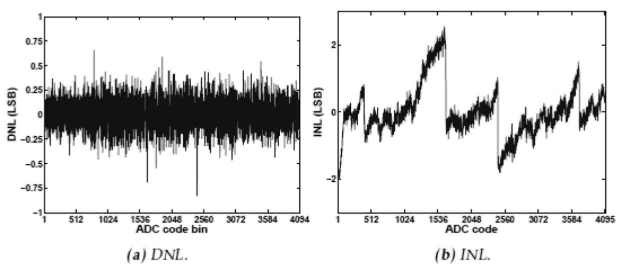

Figure 5: Static test results illustrated with data probed with ILAs.

In Table 3 measurement results from the dynamic BIST are compared with the C code algorithms running on a PC and a MATLAB model. In this case the ADC codes were probed directly with an ILA, a CSV-file with the data was obtained and this data was used in a test bench on the PC. Unfortunately the last ADC sample was not probed correctly by the ILA so to be able to make the comparison, including the FFT, the code of the last sample was estimated by interpolating the curve[10]. Despite this, the results match to two decimal places, except for the THD which matches to one decimal place. This is close enough to deem the experiment successful, and based on the experience from the static test, the match is probably even better in reality. Some difference is expected though, because the design uses floating point operations and the floating point operators used in the FPGA design are not 100% accurate [6].

TABLE III
COMPARISON BETWEEN DYNAMIC BIST TEST, C CODE RUNNING ON PC AND MATLAB TEST MODEL USING PROBED ADC CODES AND PROBED SPECTRUM.

	Matlab model using measured ADC codes	Matlab model using measured spectrum	C code Algorithm running on PC	C code algorithm running on FPGA
THD (dB)	66.030626 9531	66.021016 7941	66.030626 9531	66.021016 7941
SFDR (dB)	47.521079 2495	47.519112 7127	47.521079 2495	47.519112 7126
SINAD (dB)	42.313995 3465	42.313232 9784	42.313995 3465	42.313232 9784
SNR (dB)	42.332489 9726	42.331765 4022	42.332489 9726	42.331765 4022
ENOB	6.7365440 775	6.7364174 383	6.7365440 775	6.7364174 383

Table 3 does show a good match between the algorithms running on a PC and on the FPGA, but the measurement results are much lower than the specification of the ADC (Table 1). One reason for that is that the input signal from the signal generator was not filtered and is therefore not as spectrally pure as desired. Also, and more importantly, there is significant spectral leakage that affects the result. The reason behind the spectral leakage is probably that the frequency of the input signal could not be set with sufficient precision. This highlights a potential difficulty of dynamic ADC testing in a BIST environment. It can be hard to control the input signal frequency accurately enough. In Figure 6, the spectrum obtained by an ILA probe during the test run can be seen. The DFT is calculated directly from the ADC codes and the amplitudes in the resulting frequency spectrum are not normalized. Since the performance metrics are calculated as ratios of the signal and distortions/noise in the spectrum, no normalization is necessary.

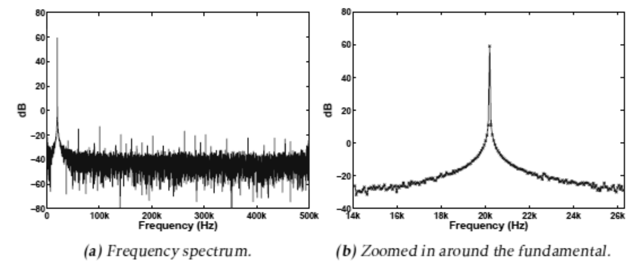

Figure 6: Frequency spectrum calculated by FFT algorithm and probed with ILA.

CONCLUSIONS

The concept of having a common base of C code for running on a PC and for synthesizing with HLS into a BIST running on an FPGA proved feasible. When testing ADCs a number of choices have to be made, especially for the static test. There are multiple ways to define the performance metrics of interest, there are many ways to measure them and the choice of definition and method will affect the result. The IEEE Std 1241 can be used as guidance when making these choices. Definitions and measurement methods are usually not expressed explicitly on data sheets which makes it harder to compare the performance of ADCs.

For the static test, it has to be considered that the accuracy of gain and offset calculations depends on the accuracy of the amplitude of the sine-wave input signal. If very accurate measurements of gain and offset are needed, either the amplitude must be controlled very precisely, or some other measurement method must be used. Perhaps the feedback loop approach can be used to find two of the transition levels and then calculate the amplitude and offset of the input using this information.

For the dynamic test, the precision of the signal frequency setting must be considered carefully. Otherwise windowing or some method to avoid spectral leakage must be used. Lastly, depending on the intended applications, it could be considered to use the BIST to test more aspects of the ADC. For example step response, code noise and two-tone tests.

REFERENCES

[1] 'IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters". IEEE Standard 1241-2010, 2011. Cited on pages 3, 15, 17, 19, 26, 27, and 52.
[2] N. H. E.Weste and D. M. Harris. Integrated Circuit Design. Pearson, Boston, 4 edition, 2011. Cited on page 7.

[3] Vivado Design Suite User Guide – High-Level Synthesis (UG902). Xilinx, July 2012. Cited on pages 5 and 58.

[4] URLwww.xilinx.com/about/company-overview/index.htm. Cited on page 2.

[5] IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters. IEEE Standard 1241-2010, 2011. Cited on pages 3, 15, 17, 19, 26, 27, and 52.

[6] J. Blair. "Histogram Measurement of ADC Nonlinearities Using Sine Waves" IEEE Transactions on Instrumentation and Measurement, 43(3):373–383, June 1994. Cited on pages 4, 15, 27, and 28.

[7] M. Vanden Bossche, J. Schoukens, and J. Renneboog. "Dynamic Testing and Diagnostics of A/D Converters" IEEE Transactions on Circuits and Systems, CAS-33(8):775–785, August 1986. Cited on pages 3 and 26.

[8] W. Kester. Taking the Mystery out of the Infamous Formula, "SNR = 6.02N + 1.76dB," and Why You Should Care. Anlog Devices, www.analog.com/static/imported-files/tutorials/MT- 001.pdf, 2009. Cited on page 28.

[9] W. Kester. Understand SINAD, ENOB, SNR, THD, THD + N, and SFDR so You Don't Get Lost in the Noise Floor. Anlog Devices, www.analog.com/static/imported-files/tutorials/MT-003.pdf, 2009. Cited on page 32.

[10] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. "Discrete-Time Signal Processing" Prentice Hall, New Jersey, 2 edition, 1998. Cited on pages 29, 32, and 53.

DOI:10.32377/cvrjst0711