On-the-fly Computation Method in Field-Programmable Gate Array for Analog-to-Digital Converter Linearity Testing

Darwin C. Mangca[1], **Olga Joy Gerasta**[1], **Anne Lorraine Luna**[2],
Xi Zhu[3] **& Jefferson A. Hora**[1,3*]

[1]Microelectronics Laboratory, EECE Department, MSU–Iligan Institute of Technology, Andres Bonifacio Avenue, Tibanga, Iligan City 9200, Philippines
[2]Analog Devices General Trias, Inc., Gateway Business Park, General Trias, Cavite 4107, Philippines
[3]University of Technology Sydney, 15 Broadway Ultimo NSW 2007, Australia
[*]E-mail: jefferson.hora@g.msuiit.edu.ph

Abstract. This paper presents a new approach to linearity testing of analog-to-digital converters (ADCs) through on-the-fly computation in field-programmable gate array (FPGA) hardware. The proposed method computes the linearity while it is processing without compromising the accuracy of the measurement, so very little overhead time is required to compute the final linearity. The results will be displayed immediately after a single ramp is supplied to the device under test. This is a cost-effective chip testing solution for semiconductor companies, achieved by reducing computing time and utilization of low-cost and low-specification automatic test equipment (ATE). The experimental results showed that the on-the-fly computation method significantly reduced the computation time (up to 44.4%) compared to the conventional process. Thus, for every 100M 12-bit ADC tested with 32 hits per code, the company can save up to 139,972 Php on electricity consumption.

1 Introduction

Because of the growing demand for high-performance applications with mixed signal integrated circuits, such as music recording, wireless telecommunications, data exchange systems and digital signal processing, researchers have been motivated to design high-performance, high-resolution, high-fidelity and low-noise analog-to-digital converters (ADC) [1]. An ADC is a device that converts analog signals to digital signals. As the production volume of high-resolution ADC increases and the fabrication process is scaled up, the cost of testing increases proportionally. Evaluating high-resolution ADCs requires expensive mixed-signal automated test equipment (ATE) [2]. As the resolution of the ADC increases, the testing time increases, which means additional costs [3]. Companies are spending millions of dollars on building

trust with clients by ensuring that the ADCs they are selling work properly and that their design specifications are properly met.

In measuring the performance of an ADC, the histogram method is used. With this method, offset, gain and non-linearity errors of the ADC can be measured [2-17]. When testing the performance of the ADC, the non-linearity test is the most time-consuming. The only way to decrease the testing time is to improve the method of testing for non-linearity errors.

The process of testing ADC ICs by semiconductor companies is shown in Figure 1. In testing of ADCs, an ATE and a device interface board are required [3]. The role of the ATE is to produce a precise stimulus signal and to capture and measure the performance of the device under test (DUT) [1,4]. It also requires a device interface board (DIB) that interfaces the ATE to the DUT. Inside the ATE, a digital signal processing (DSP) is responsible for computing the linearity performance of the DUT [6]. Before the DSP can be used, all data output from the DUT must be gathered to calculate the linearity, for which computing time is needed. This study investigated the On-the-fly Computation Method (OFCM) in FPGA hardware to decrease the testing time by removing this computing time.
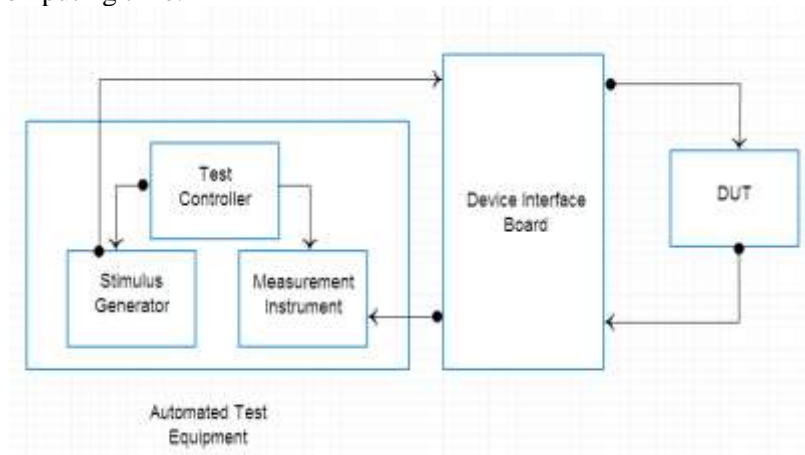
Figure 1 Current ADC test set-up for linearity testing by semiconductor companies.

In a typical process, all the data are captured first, after which the captured data are processed and the linearity performance of the ADC chip is measured as soon as the data capturing process is done. With this method, there is idle time for both the capturing module and DSP module. With background processing as shown in Figure 2, more time can be saved by allowing no idle time for both

modules. Background processing can measure the linearity performance of the first ADC while it captures the data of a 2nd ADC.
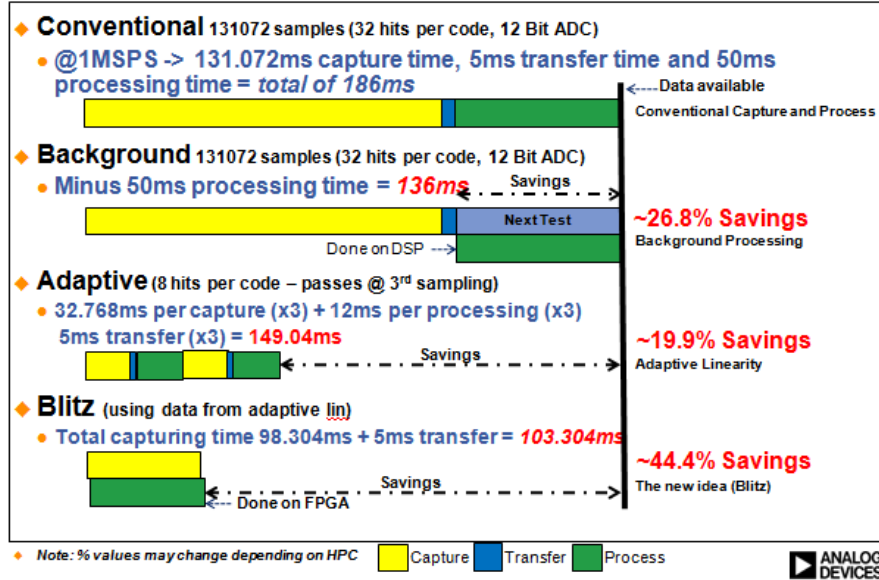
Figure 2 Test time composition for existing methods: (a) conventional capture and processing, (b) background processing, (c) adaptive linearity, (d) On the Fly Computation Method [18].

Another way of increasing time-saving is by applying adaptive linearity. This is a method that carries out linearity testing with a small number of hits per code but with multiple iterations until the maximum number of iterations is reached. With this method, if the linearity passes at the 3rd iteration, then there will be substantial time-saving because further iterations do not need to be carried out.

Lastly, applying OFCM using FPGA can further increase time-saving by computing the linearity performance of the ADC chip while data capturing is in progress. As can be seen in Figure 2(d), there is a small overhead in the computing time when the linearity results are almost ready in order to show them after capturing the ADC output [18].

2   ADC Linearity Test

2.1   BIST Advantages and Disadvantages

Having a built-in self-test (BIST) is beneficial for the following reasons: (a) low cost of testing, (b) better fault coverage since special structures can be

incorporated into the chips, (c) shorter test times if the BIST can be designed to test more structures in parallel, (d) easier customer support, and (e) capable of performing tests outside the production electrical testing environment [19]. The last advantage allows the consumer to test the chips prior to mounting or even when they are on the application board.

There are also disadvantages of using a BIST. They are: (a) additional silicon area and fabrication processing requirements for the BIST circuits, (b) reduced access times, (c) additional pin requirements since the BIST circuit needs a way to interface with the outside world to be effective, and (d) possible issues with the correctness of the BIST results since the on-chip testing hardware itself can fail.

Aside from the advantages and disadvantages, there are other issues that need to be addressed when implementing a BIST. They are: (a) faults to be covered by the BIST and how these will be tested for, (b) how much chip area will be occupied by the BIST circuits, (c) external supply and excitation requirements of the BIST, (d) test time and effectiveness of the BIST, (e) how the BIST will impact the production electrical test processes that are already in place [19].

Furthermore, when using a BIST, ATE is no longer needed when measuring the linearity performance of an ADC. Thus, any ATE that was bought by a company will become obsolete and thus an investment loss. Because of these disadvantages, a new method is proposed that measures the linearity performance of ADCs using a field-programmable gate array (FPGA).

2.2 FPGA Background

2.2.1 Advantages of Using FPGA

An FPGA is a type of integrated circuit (IC) that can be programmed for different algorithms after fabrication [20]. Modern FPGAs consist of up to 2 million logic cells that can be configured to implement a variety of software algorithms. A traditional FPGA is similar to a regular IC, providing low production costs and the same level of performance. There are 5 benefits to using FPGA technology [21]:

1. Performance – When taking advantage of hardware parallelism, using an FPGA can execute more processes in each clock cycle than DSP processors by breaking up several sequential executions.
2. Time to Market – Using an FPGA hardware prototyping is easy and fast. You can test your concept on an FPGA without going through the long fabrication process of ASIC design.
3. Cost – FPGAs are mass-produced, thus the cost per chip is low.

4. Reliability – With the advancement of test equipment, FPGAs are well-tested so that all product releases are guaranteed to work.
5. Long-term Maintenance – In digital communications, specifications keep changing over time and modifications may be implemented in the future. Since FPGAs are reconfigurable, they can keep up with future modifications.

3   ADC Linearity Testing With an FPGA

3.1   Design of the OFCM in FPGA

Figure 3 shows a top-level block of an OFCM containing 4 types of modules, including the input-output (I/O) pins that interface with the DIB and DUT.
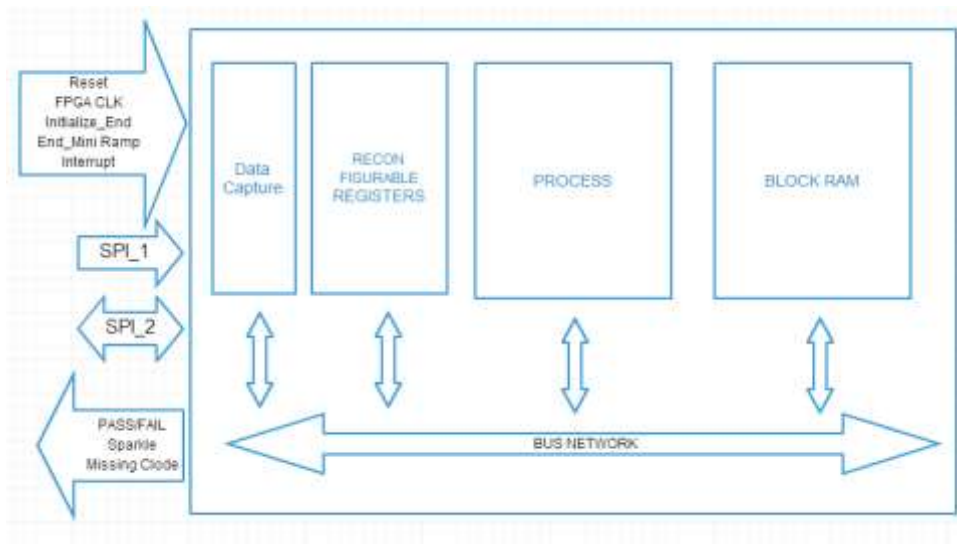
Figure 3   Top-level block diagram of the OFCM system.

3.1.1   Data Capture

Data capture is responsible for capturing the SPI info and converting it to parallel data that can be read by other modules. Figure 4 shows a simple schematic diagram of how to convert serial data to parallel data. Data are converted from serial to parallel format by using a series of D flip-flops that function as shift registers.
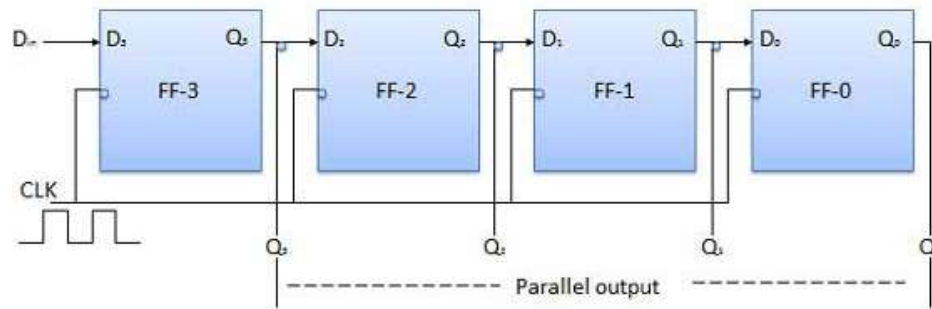
**Figure 4**   Schematic diagram of serial to parallel conversion.

3.1.2   Reconfigurable Registers

By taking advantage of the reconfigurable registers, the ATE can fully control the operation of the ATE. This means that for any operation there is no need to load a specific program, which makes the system reconfigurable without making changes in the program. The list of registers that can be reconfigured are explained below:

1. Window Length – tells the processing module that a sparkle has been detected if it is greater than or smaller than the window length.
2. Number of Mini Ramps – tells the processing module the total number of mini ramps that is supplied by the ATE
3. Number of Hits per Code – tells the processing module the number of hits per code per mini ramp supplied by the ATE
4. DNL min max settings – tells the processing module the allowed DNL pass value for the first mini ramp.
5. INL min max settings – tells the processing module the allowed INL pass value for the first mini ramp.

3.1.3   Process

The process module is responsible for controlling the operation of the FPGA using the reconfigurable register values and the captured values in parallel format. This also controls the operation of the block RAM's read and write processes. Using the OFCM in the process module, the pass/fail results are almost ready as soon as the capturing of data is finished. Aside from the OFCM, new functions were also added to detect sparkle codes. The block diagram for the process module is shown in Figure 5.
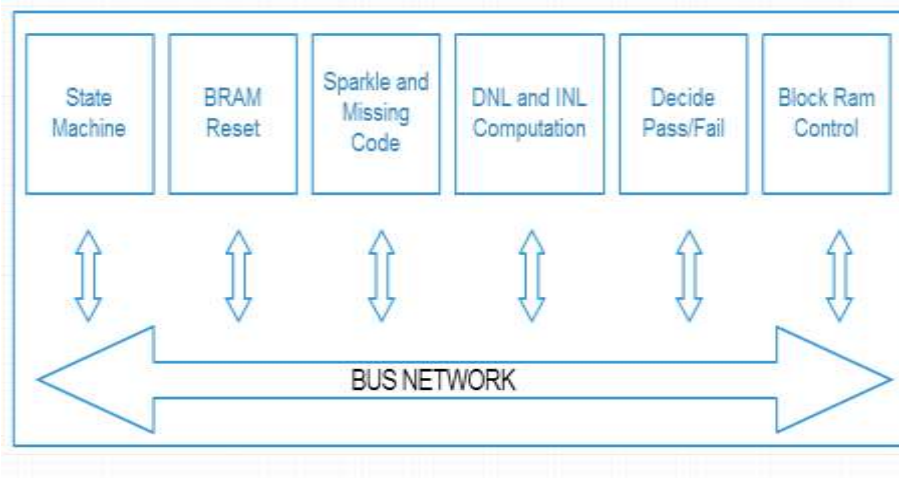
Figure 5 Process module block diagram.

3.1.4   State Machine

All operations are supervised by the state machine. This module can tell the current process state and can trigger the operation of a certain module. State machines are very helpful in working with sequential circuits. The system uses a finite state machine (FSM), which means that there is a finite number of states and the machine can only be in one state at a time, called the current state. A transition happens when one state changes to another if triggered by a condition. When there is a transition, some modules will be triggered and some modules will not be triggered. Using a state machine promotes good design techniques because it makes the code more efficient, easier to debug, and helps to organize the program flow.

3.1.5   BRAM Reset

FPGA block RAMs (BRAMs) do not have a global reset option for resetting all the BRAM values. Implementing a global reset on BRAMs is not a good idea because it can consume many resources when implemented. A problem arises when resetting 2000+ addresses, which can consume 2000+ clock cycles and therefore much time is wasted in this process. To solve this issue, BRAM Reset was created to manually reset the block RAM one address at a time. It resets the first ten address and then resets all the remaining addresses while the data capturing process is in progress. This way, less time is needed for resetting all the registers.

3.1.6   DNL and INL Computation

Differential nonlinearity (DNL) and integral nonlinearity (INL) are the most important electrical characteristics in measuring the linearity performance of an ADC. These two parameters are evaluated by using the histogram method. The formulas for solving the DNL and INL are in Eqs. (1) and (2) as follows:

$$DNL = \frac{H_i - H_{ideal}}{H_{ideal}} \qquad\qquad (1)$$

$$INL = INL_{i-1} - DNL_i \qquad\qquad (2)$$

In solving the DNL, *H* (ideal) is the number of hits per code and is provided by the testing personnel using the reconfigurable registers. DNL is the result of dividing the difference of actual hits per code by the ideal number of hits per code. To solve for the INL, you simply add all the DNL values. The DNL and INL values will be provided by the testing personnel. A 12-bit 2-channel SAR ADC from ADI (AD7866) has typical DNL and INL values of -0.95/+1.25 and ±1.5 respectively.

3.1.7   Pass/Fail Decision

Having the ability to identify if the linearity performance of the ADC passes or fails almost immediately after the data capturing process is a great advantage of using FPGA. Using OFCM, the DSP module found on the ATE will no longer be needed to measure the linearity performance of the DUT. Now the only purpose of the ATE is to provide accurate positive ramp signals and data to control the operation of the test. Using this method, more DSP modules can be added on the DIB, which solves the issue of having 1 DSP module per ATE. With 1 DSP per ATE, linearity can be solved 1 at a time only. Having more DSP modules embedded in the DIB, parallel processing can be applied to test more ADCs per ATE. Passing or failing the DUT is based on the computed DNL and INL values.

3.1.8   BRAM Control

The BRAM control module is responsible for controlling the read and write operations of the BRAM. The block RAM is a special add-on functionality of the FPGA for the purpose of storing large quantities of binary data. To communicate with the BRAM, block RAM control is needed. Three BRAMs are needed to run the system and 3 controllers for the read and write operations. These 3 controllers were described and defined below:

1. Code Count Control – controls the read and write operation for the block RAM code count

2. INL Count Control – controls the read and write operations for the block RAM computed INL
3. DNL Count Control – controls the read and write operations for the block RAM computed DNL

3.1.9   Block RAM

The block RAM or commonly called BRAM is viewed as memory. To simplify the design implementation for storing large quantities of binary data, BRAM is needed. Storing large quantities of binary data can also be done by using flip-flops or latches, but the drawback is that it takes a lot of time to implement the design and a lot of FPGA resources will be consumed. Using BRAM in the design removes these issues, but a new function must be made to communicate with the BRAM. This module is called block RAM Control. The timing diagram for the BRAM operation in Figure 6 shows that the mode of read and write operations is write-first. This means that the inputted data are written first before outputting the stored data.

Three BRAMs are used in the system as defined below:

1. Code Count – used for storing the number of actual hits per code
2. DNL Count – used for storing the computed DNL per code
3. INL Count – used for storing the computed INL per code
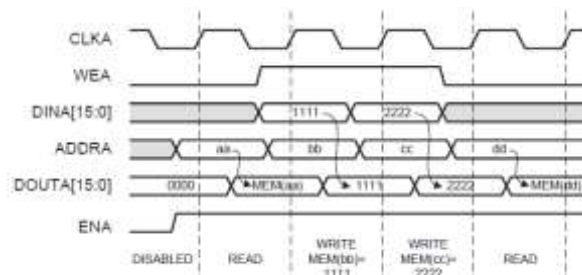
Figure 6   Write-first mode in BRAM.

3.2   Machine Noise Detection and Filtering

Dynamic sparkle codes are unwanted spikes or machine noise that are generated by the ATE. To have accurate results, these unwanted spikes must be filtered out so they are not included in measuring the non-linearity errors of the ADC. To filter out these unwanted spikes, a feature for machine noise detection was added to the process module.

3.3   Experimental Configuration

Figure 7 shows the experimental configuration. The OFCM was tested with ideal and non-ideal 12-bit ADC output with 32 hits per code. It was also tested with random hits per code and machine noise.
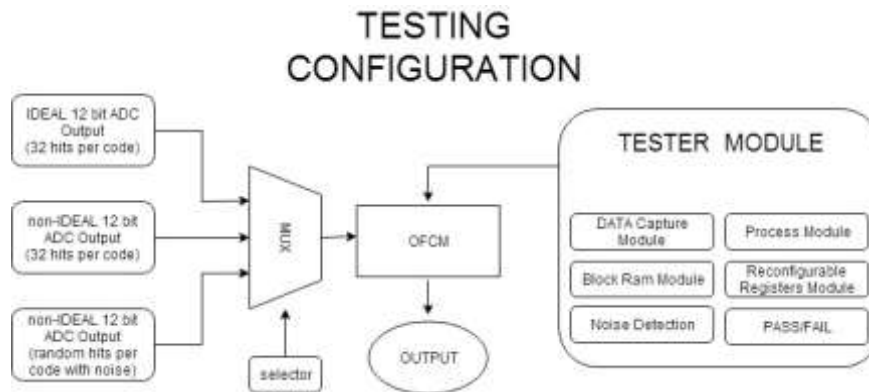
Figure 7   Testing configuration.

4   Results

4.1   Accuracy of OFCM in FPGA

The OFCM in FPGA was first tested using an ideal ADC output with 32 hits per code. Both the behavioral and the post-route simulation had an accurate result with no timing issues at a 50MHz clock.

Another test was done using a non-ideal ADC output with 32 hits per code. Both the behavioral and the post simulation had an accurate result with no timing issues at a 50 MHz clock.

4.2   Verification of OFCM in FPGA

The 4 major modules were tested using behavioral and post-route simulations to ensure that all modules were working properly with no timing issues at a clock frequency of 50 MHz.

4.2.1   Data Capture Module

The SPI protocol is a popular method of transferring data. The advantage of SPI is that only four pins are needed for transferring data. With an FPGA it is

possible to capture the data serially with an SPI clock frequency of 50 MHz and convert these data to parallel format. Figure 8 shows the behavioral and the post-route simulation of the data capture module. This module uses shift registers for capturing the serial data.
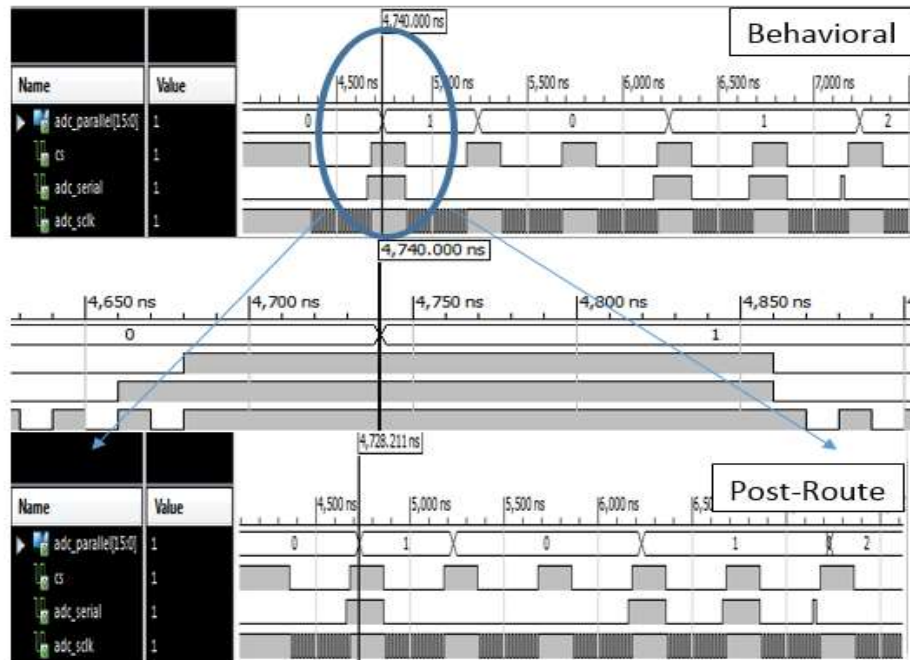
Figure 8   Behavioral and post-route simulation results for data capture.

This test tried to capture serial data with a value of **1**, **0**, **0**, **1**, **1**, and **2**. Data capturing and conversion was successful, as shown in Figure 8, for both the behavioral and the post-route simulation. Data capturing started when the chip select (CS) was high and triggered the capture of all values in the shift registers. After capturing, the program waited three clock cycles before the serial data was converted to parallel data. The test equipment and the FPGA board used two different clocks, which is the reason why a minimum of three clock cycles was needed to synchronize the data from the test equipment to the FPGA board. This method ensures that the signal becomes stable. Adding a synchronizer module to synchronize two data with different clocks is a good design technique to avoid a meta-stable state. The data captured using behavioral and post-route simulations were correct and identical with no timing issues at a clock frequency of 50 MHz.

4.2.2 Reconfigurable Registers Module

Reconfigurable registers are used for the purpose of storing the parameters needed to control the operation of the linearity test using the SPI protocol. These parameters are key factors in fully controlling the operation of the FPGA. Using these parameters, only one program is needed to handle all the operations. Taking advantage of the SPI protocol for reconfiguring the registers during initialization can make the system fully customizable. As shown in Figure 9 (behavioral simulation), the serial data were captured when CS was high and were converted to parallel data after 3 clock cycles. The purpose for this is to avoid a meta-stable state and to make sure that the signal is stable.

This test tried to initialize the reconfigurable registers by changing the total ramp count from **8 to 2**, the counts per code from **16 to 64**, and the window length from **5 to 4**. The test was successful. The behavioral and the post-route simulation for the 3 registers, total number of ramp counts, count per code and window length had the same results with no timing issues at a clock frequency of 50 MHz.
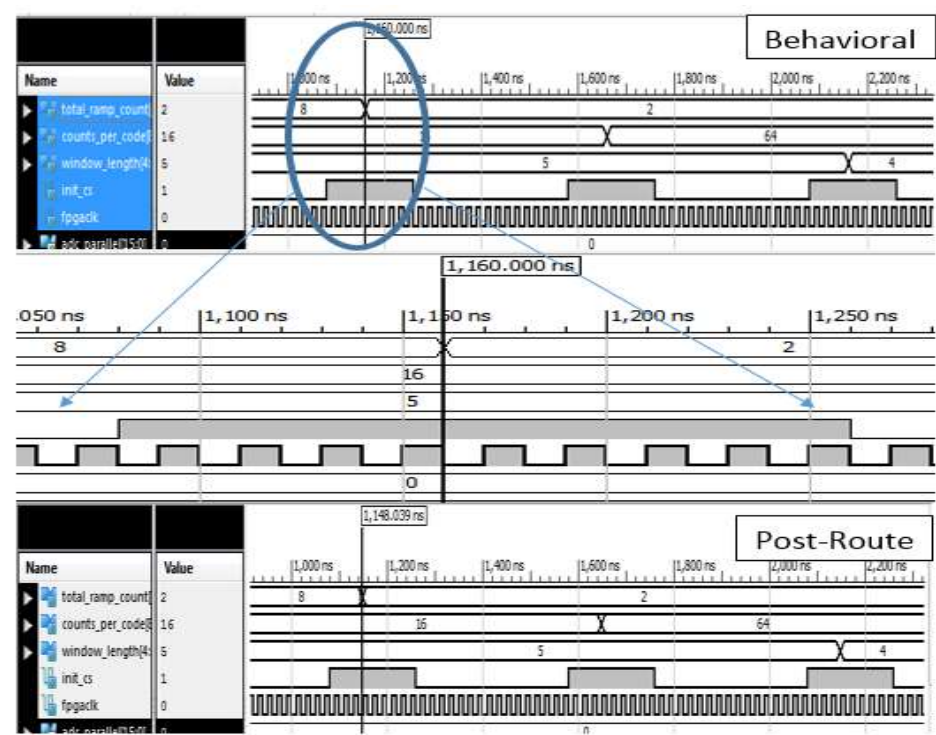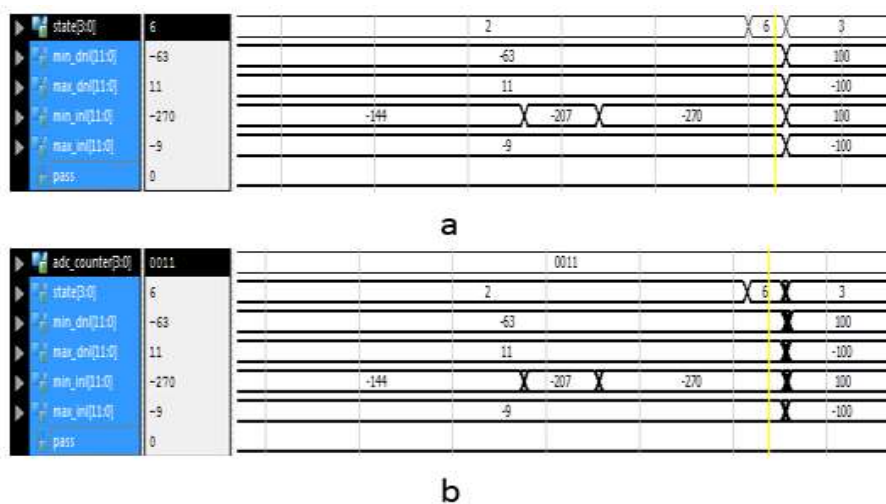

Figure 9 Behavioral and post-route simulation results for the reconfigurable register.

4.2.3   Process Module

In deciding whether the DUT passed or failed, a method to extract the minimum and maximum values for both DNL and INL was used. Using the parameters supplied by the ATE, the program can determine the allowable range for the DUT to pass the linearity test. Figure 10 shows the minimum and maximum simulation results for DNL and INL. If the computed value is beyond the minimum and maximum allowable range, then on that ramp the DUT fails the linearity test on that particular ramp. When the state value is equal to 2, it finalizes all the minimum and maximum values for both DNL and INL. When it is finished finalizing all the minimum and maximum values, the state will automatically change from 2 to 6. The latter is the state where the program decides if the chip passes or fails the linearity test based on the minimum and maximum values. The expected results for the minimum and maximum DNL are -63 and 11 while the expected results for the minimum and maximum INL are -270 and -9. As shown in Figure 10, both the minimum and maximum values for DNL and INL in state 6 were correct for both the behavioral and the post-route simulation with no timing issues at a clock frequency of 50 MHz.

Figure 10 Minimum and maximum DNL and INL values: (a) behavioral (b) and post-route simulation results.

4.2.4   Block RAM Module

Without using block RAMs for storing all the computed values for DNL and INL, the compilation process is considerably slower: it would take a day or two to finish synthesizing the code. Taking advantage of using block RAMs for storing the values solved the issue: now it takes less than a minute to synthesize

the program. The only disadvantage of using block RAMs is that additional modules are needed to communicate with it. The mode used in reading and writing on the block RAMs is write-first mode. The system can work without storing the computed values but this function was added for debugging purposes. This way, if something goes wrong in the system it is easier to track the problem. The standard size of the block RAMs was a 12 x 4 12-bit address and a storing capacity of 4 bits per address. The project required 3 sets of 12 x 12 memory size so a total of 9 block RAMs were used in this project.

To verify if the block RAMs for storing the code count and the computed DNL and INL values were working properly a test was done using ideal and a non-ideal ADC output. Both the behavioral and the post-route simulation had accurate results. Another test was done by supplying a random amount of hits per code from code 1 to code 13. The number of hits per code and their corresponding expected DNL and INL values are shown in Table 1 in hexadecimal format.

Random values were inputted to see if the simulation values and expected values in the block RAM shown in Table 1 were the same. The result shows that both the behavioral and the post-route simulation were exactly the same compared to the expected values shown in Table 1 with no timing issues at a clock frequency of 50 MHz. This tells that the design implementation of the block RAM memory was successful.

Table 1   Expected block RAM results.

| Code | Block RAM Memory | | |
	Count	DNL	INL
1	028	FE8	FE8
2	045	005	FED
3	048	008	FF5
4	03C	FFC	FF1
5	033	FF3	FE4
6	04B	00B	FEF
7	048	008	FF7
8	024	FE4	FDB
9	014	FD4	FAF
10	001	FCI	F70
11	001	FCI	F31
12	001	FCI	EF2
13	001	FCI	EB3

4.3   Machine Noise Detection and Filtering

Sparkle codes or machine noise are unwanted spikes produced by the ATE. One of the major advantages of using OFCM is being able to detect and filter out these sparkle codes. Sparkle codes are only produced by the ATE and not by the DUT, so the latter can be excluded from the linearity performance measurement.

A sparkle code was inputted in the system to test if it could be detected. The input sequence for this test was 1, 0, 0, 1, 1, **20**, 2, and 1. To be successful for this test, the system should be able to detect a spike for code 20. Figure 11 shows both the behavioral and the post-route simulation for sparkle code detection. The result shows that in a simulation time of around 7us, the system was able to detect the sparkle code with no timing issues at a clock frequency of 50 MHz, thus making this test successful.
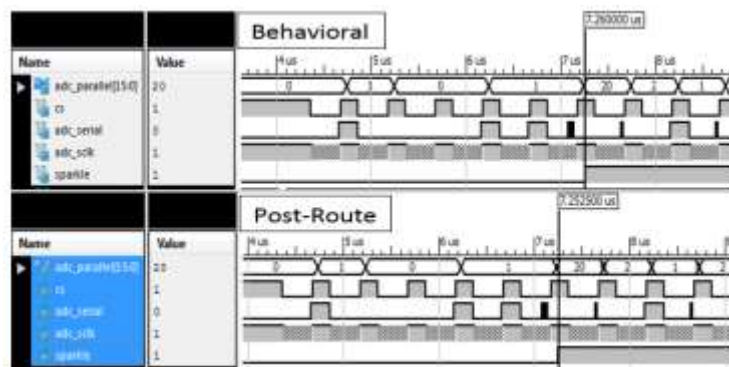
Figure 11 Behavioral and post-route simulation of machine noise detection and filtering.

4.4   Evaluation of Economic Impact of using OFCM

Because of the expected machine noise, computing time was added to ensure that the system remains stable. This computing time is measured from the end of the ramp to the state when it is ready to receive a new ramp. The measured additional computing time was 350ns for the behavioral simulation and 359ns for the post-route simulation, as shown in Figure 12. Table 2 shows a computation summary to evaluate the economic impact of employing OFCM for measuring the linearity performance of ADCs.
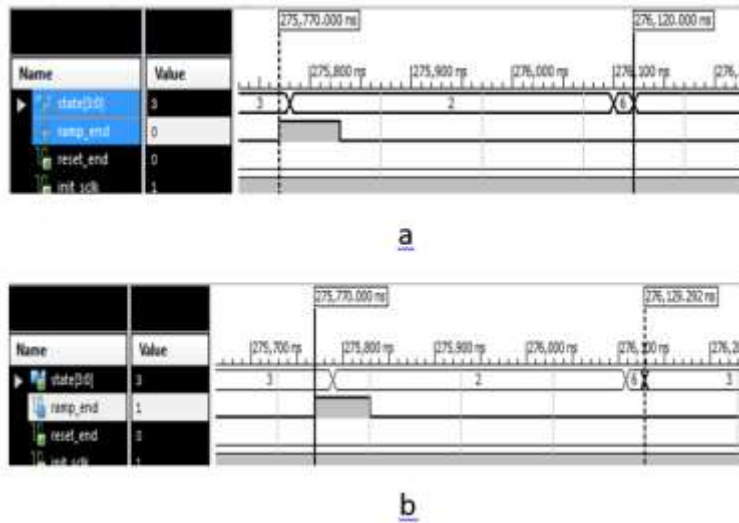
Figure 12 OFCM additional computing time: (a) behavioral and (b) post-route simulation.

Table 2 Test time, time-saving, labor and electricity cost computation summary.

Method	Test Time (ms)	Time-saving (%)	Labor Cost Savings per 100M 12 BIT ADC (Php)	Electricity Cost Savings per 100M 12 BIT ADC (Php)
Conventional	186	0	0.00	0.00
Background	136	26.8	51,911.00	84,488.00
Adaptive	149.04	19.9	38,546.00	62,735.00
OFCM (theoretical)	103.304	44.4	86,000.00	139,972.00
OFCM (actual)	103.304	44.4	86,000.00	139,972.00

5   Conclusion

A design implementation of the OFCM was successfully implemented using Verilog code. Then behavioral and post-route simulations were used to verify the full functionality of the OFCM in an FPGA. It is as follows:

1. Capture data in SPI format
2. Calculate DNL and INL accurately in real time and store the values in the block RAM
3. Calculate the minimum and maximum values of DNL and INL
4. Modify the reconfigurable registers
5. Accept non-monotonic ramp signals with positive slope

Finally, the OFCM in the FPGA was also tested with input signals from machine noise. The behavioral and the post-route simulation results showed that the OFCM was able to identify and filter out the machine noise. Lastly, the economic impact was evaluated. The actual computation result showed that using the OFCM in an FPGA can obtain a time-saving of 44.4% compared to the conventional process.

Acknowledgments

References

[1]   Jin L., Panthasarathy, K., Kuyel, T., Geiger, R. & Chen, D., *High-Performance ADC Linearity Test Using Low-Precision Signals in Non-Stationary Environment,* Test Conference, Proceedings ITC 2005, IEEE International, pp. 1191, 2005.

[2]   Azais, F., Bernard, S., Bertrand, Y. & Renovell, M., *A Low Cost BIST Architecture for Linear Histogram Testing of ADCs*, Journal of Electronic Testing: Theory and Applications, **17**, pp. 139-407, 2001.

[3]   Xing, H., Jiang, H., Chen, D. & Geiger, R., *High-resolution ADC Linearity Testing Using a Fully Digital-Compatible BIST Strategy*, IEEE Transactions on Instrumentation and Measurement, pp. 2697-2705, 2009.

[4]   Duan, J., Chen, D. & Geiger, R., *Cost Effective Signal Generators for ADC BIST*, Circuits and Systems, ISCAS IEEE International Symposium on, pp. 13-16, 2009.

[5]   Chao, A., Chang, S. & Ting, H., *A SAR ADC BIST for Simplified Linearity Test*, IEEE, pp. 146-149, 2011.

[6]   Yong, W., Jin, W., Feng, L. & Yi-zheng, Y., *Optimal Schemes for ADC BIST Based on Histogram,* Test Symposium, Proceedings 14th Asian, pp. 52-57, 2005.

[7]   Balakrishna, Y., Nareshbabu, M. & Kumar, G., *Design and Testing of High Speed Multipliers by Using Linear Feedback Shift Register*, International Journal of Emerging Trends in Engineering and Development, pp. 373-381, 2013.

[8]   Erdogan, E. & Ozev, S., *An ADC-BIST Scheme Using Sequential Code Analysis*, Design, Automation & Test in Europe Conference & Exhibition, DATE '07, pp. 1-6, 2007.

[9]   Huang, J., Ong, C. & Cheng, K., *A BIST Scheme for on-chip ADC and DAC Testing*, Design, Automation and Test in Europe Conference and Exhibition, Proceedings IEEE, pp. 216-220, 2000.

[10]   Parthasarathy, K., Kuyel, T., Price, D., Jin L., Chen, D. & Geiger, R., *BIST and Production Testing of Adcs Using Imprecise Stimulus*, ACM Trans. on Design Automation of Electronics Systems, 2003.

[11]   Renovell, M., Azais, F., Bernard, S. & Bertrand, Y., *Hardware Resource Minimization for Histogram-Based ADC BIST*, VLSI Test Symposium, 2000. Proceedings. 18th IEEE, pp. 247-252, 2000.

[12]   Ting, H., Chao, I., Lien, Y., Chang, S. & Liu, B., *Low Cost Output Response Analyzer Circuit for ADC BIST*, Testing and Diagnosis, IEEE International Conference on Circuits and Systems, pp. 1-4, 2009.

[13]   Duan, J., Chen D., & Geiger, R., *A Low Cost Method for Testing Offset and Gain Error for ADC BIST,* 2012 IEEE International Symposium, pp. 2023-2026, 2012.

[14]   Sunter, S.K. & Nagi, N., *A Simplified Polynomial-Fitting Algorithm for DAC and ADC BIST*, Test Conference, Proceedings International, pp. 389-395, 1997.

[15]   Gines, A.J., Peralias, E.J. & Rueda, A., *An Adaptive BIST for INL Estimation of ADCs Without Histogram Evaluation*, Mixed-Signals, Sensors and Systems Test Workshop, 2010 IEEE 16th International, pp. 1-6, 2010.

[16]   Lee, D., Yoo, K., Kim, K. & Han, G., *Code-width Testing-Based Compact ADC BIST Circuit,* Circuits and Systems II: Express Briefs, IEEE Transactions, **51**(11), pp. 603-606, 2004.

[17]   Xing, H., Jiang, H., Chen, D. & Geiger, R.L., *High-resolution ADC Linearity Testing Using a Fully Digital-compatible BIST Strategy*, IEEE Transactions on Instrumentation and Measurement, pp. 2697-2705, 2009.

[18]   Bating, C., Lim, P. & Reyes, C., Blitz Linearity – *Real Time ADC Linearity Testing, Analog Devices,* 2014.

[19]   STMicroelectronics Application Note 2015, *SPC56ELxx/RPC56ELxx ADC Built-in Self-tests ADC Working in CPU Mode (AN4371)*, <https://www.st.com/content/ccc/resource/technical/document/applicatio n_note/cc/7d/12/60/aa/a8/46/53/DM00096576.pdf/files/DM00096576.pdf /jcr:content/translations/en.DM00096576.pdf>, October 2015.

[20]   Xilinx Product Document 2013, *Introduction to FPGA Design with Vivado High-Level Synthesis*, UG998 (v1.0) 2 July 2013, <https://www.xilinx.com/support/documentation/sw_manuals/ug998-vivado-intro-fpga-design-hls.pdf>, October 2015.

[21]   National Instruments White Paper 2012, *Introduction to FPGA Technology: Top 5 Benefits*, published 16 April 2012, <http://www.ni.com/white-paper/6984/en/>, October 2015.