# Digital Predistortion Equalizer using a Finite Impulse Response (FIR) Filter Implemented on FPGA

Bassam Nima, Yanan Li and Zhenyu Zhang

***Abstract*—Nonlinear frequency responses are a common problem in radio frequency (RF) components. Specifically, in wireless communication systems, signals are often unequally amplified or attenuated across a specified frequency band. Common communications components such as filters, amplifiers, and mixers can lead to nonlinear frequency responses, which can cause errors in transmitting and receiving. This article outlines the design and demonstration of a static and dynamic finite impulse response (FIR) digital equalizer circuit. Using predistortion topology with a coupled feedback loop, the adaptive LMS algorithm was implemented. The FIR filter was simulated in MATLAB and Vivado and then implemented onto an Eclypse Z7 FPGA evaluation board with no timing errors. Simulations showed that the custom RTL module gave the same frequency response that was produced in MATLAB calculations. The filter was able to dynamically equalize the frequency responses of different nonlinear boards that were used as the devices under test (DUT). Measurements showed that the equalizer was able to compensate for system distortion from 0.2 to 0.8 Nyquist frequency. The phase response remained relatively linear across the band of interest, with a group delay flatness less than 10ns.**

***Index Terms*—Digital Predistortion, finite impulse response (FIR) digital equalizer filter, Least-Mean Square (LMS) optimization algorithm, FPGA**

## I. Introduction

Radio frequency (RF) signals transmitted over long distances typically have RF interference and lose power in their signals, which is largely determined by path lengths, path angles, and medium attenuation factors [1]-[3]. This affects signals in many technologies in wireless communications, from Bluetooth and WLAN to earth-satellite radio-paths and radio detection of cosmic rays [4], [5]. To compensate for the attenuation and RF interference, improved transmitter and receiver antennas, analog filters, amplifiers, and mixers are implemented [6]. Low noise amplifiers are typically implemented to decrease the noise factor (to the limiting thermal noise of the FETs) of the low power input signals, however often these components have large phase and amplitude response errors in their frequency response [7]-[9], compounding with other frequency response errors in the RF transmission path from passive and active devices. Although improvements to flatten the frequency response could be made with analog devices easily implemented in the RF path, digital filters can be much more versatile and adaptive [6], [10], [11].

Digital filters can be classified as either recursive filters with an infinite impulse response (IIR) or non-recursive filters with a finite impulse response (FIR).

In a non-adaptive, the coefficients of FIR filter must be determined before processing the signal. In an adaptive FIR filter, the coefficients are continuously "adapted to minimize the [...] error signal" [11]. with Least-Mean Square (LMS) optimization algorithm. This methodology was implemented on a DSP56200 by D. E. Borth et al. as an echo canceller [11]. Kollar et al. compared the use of FIR and IIR filters and found that low order FIR filters used in equalization had better quality than IIR filters in correcting anti-aliasing filters [18].

In this paper, for the first time, we proposed a digital predistortion equalizer using the LMS algorithm. Static and dynamically configured digital predistortion was implemented for devices under test, in this case, self-made nonlinear circuit boards cascaded with an off-the-shelf RF amplifier, to mimic the intermediate frequency (IF) response of a wireless communication system.

## II. Principle of Digital Predistortion Equalizer Realized by Adaptive Algorithm

### A. Equalizer Topology

There are various designs that implement an FIR filter in an RF system, such as with feedforward (running the filter post-DUT), or running the filter parallel to the DUT, However, these topologies do not consider the adaptive model.

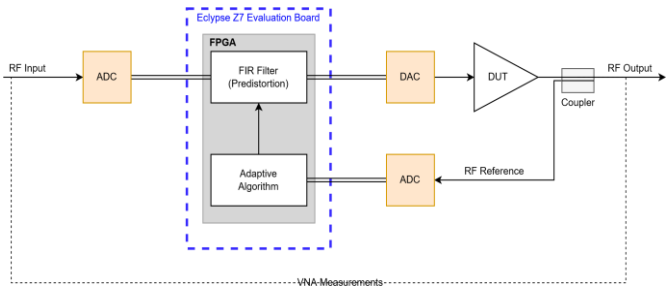



Fig. 1. The general topology of an adaptive FIR filter.

Submitted for review on October 15, 2022.

B. Nima is a student in the Department of Electrical and Computer Engineering at the University of Alberta (e-mail: bnima@ualberta.ca)

Y. Li is a student in the Department of Electronics and Information Engineering, Beijing-Dublin International College at Beijing University of Technology (e-mail: yanan.li@ucdconnect.ie)

Z. Zhang is an industrial professor in the Department of Electrical and Computer Engineering at the University of Alberta (e-mail: zhenyu15@ualberta.ca)

The most ideal case, is when the RF signal travels through the FIR filter first for predistortion, then through the DUT and then the RF reference is taken through a coupler. This is illustrated in the simplified block diagram in Fig. 1.

For a digital FIR filter with $N$ taps and order $N-1$, the output is found by:

$$y[k] = \sum_{n=0}^{N-1} b_k[n] f[k-n] \tag{1}$$

where $k$ is the sample time, $y[k]$ is the output, $b_k$ is the coefficient of tap $n$ at time $k$, and $f[k]$ is the input [10]. The FIR filter is causal, and finite since the impulse response is given by (2). In a non-adaptive, $b_k[n]$ must be determined before processing the signal.

$$h[k] = \sum_{n=0}^{N-1} b_n \delta[k-n] = \begin{cases} b_k & 0 < n < N-1 \\ 0 & otherwise \end{cases} \tag{2}$$

Now, the FPGA has a reference to reduce the error through the adaptive algorithm using LMS as described in (3) to determine the new coefficients.

$$b_{k+1}[n] = b_k[n] + K \cdot e[k] \cdot f[k-n] \tag{3}$$

where $e[k]$ is the error signal between the target $y'[k]$ and the output $y[k]$, and $K$ is the rate of adaptation [11]. A general block diagram of this adaptive method is illustrated in Fig. 2.

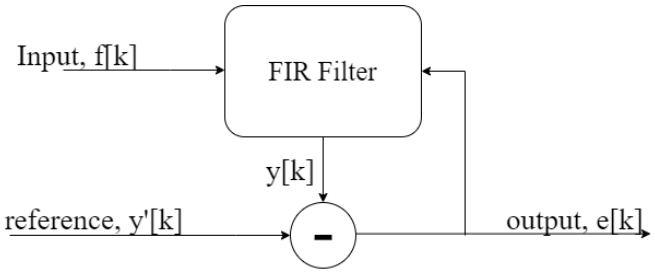



Fig. 2. The general topology of an adaptive FIR filter.

*B. LMS Coefficients Derivation*

There are several methods for determining the coefficients to an N-tap FIR Filter as outlined by Proakis, Digital Signal Processing [23].

The most straightforward method is by directly taking the inverse discrete Fourier transform (IFT). For an absolutely summable BIBO frequency response given by a real and imaginary value, the coefficients can be found directly by taking the Fourier transform of either the real or imaginary components.

A better method that requires no guesswork is to find the desired frequency response using Least Mean Squares to solve the system of equations, which can also be weighted, similar to in low-pass and high-pass FIR filters where the passband is given a higher weight.

An FIR filter of order $N-1$ (length $N$), the frequency response is:

$$H(\omega) = \sum_{n=0}^{N-1} b_n e^{-i\omega n} \tag{4}$$

Here, $\omega = 2\pi f/fs$ is the normalized frequency from DC to Nyquist. If the frequency response is discretized with frequencies $\omega_0, \omega_1, \omega_2 \ldots, \omega_{M-1}$ then the matrix equivalent for an M-point frequency

Response $H(\omega M)$ is given by (5).

Let the matrix be labeled as $\boldsymbol{A}$ and the LHS and RHS vectors be labelled as $\boldsymbol{H}$ and $\boldsymbol{b}$ respectively. $\boldsymbol{A}$ requirement is that the coefficients $\boldsymbol{b}$ are real, however $\boldsymbol{A} \in \mathbb{C}_{M\times N}$ and $\boldsymbol{H} \in \mathbb{C}_{M\times 1}$ and therefore $\boldsymbol{b} \in \mathbb{C}_{N\times 1}$.

Since this is an absolutely summable system, the real and imaginary components can be taken independently by extending the matrix resulting in real components only, such as:

$$\begin{bmatrix} \Re(\boldsymbol{H}) \\ \Im(\boldsymbol{H}) \end{bmatrix} = \begin{bmatrix} \Re(\boldsymbol{A}) \\ \Im(\boldsymbol{A}) \end{bmatrix}_{\boldsymbol{b}} \tag{6}$$

Now, the new vector to the LHS and matrix on the RHS can be labelled as $\boldsymbol{H}^{ext}$ and $\boldsymbol{A}^{ext}$, respectively. In other words, the complex $\boldsymbol{H} = \boldsymbol{A}\boldsymbol{b}$ becomes $\boldsymbol{H}^{ext} = \boldsymbol{A}^{ext}\boldsymbol{b}^{ext}$. Furthermore, $\boldsymbol{A}^{ext} \in \mathbb{R}_{2M\times N}$ and $\boldsymbol{H}^{ext} \in \mathbb{R}_{2M\times 1}$ and the entire system of equations is in the real domain [24].

Now, given a desired frequency response $\boldsymbol{D}^{ext}$, the LMS algorithm can be applied to the equation:

$$\boldsymbol{D}^{ext} = \boldsymbol{A}^{ext}\boldsymbol{b} \tag{7}$$

The solution to the LMS problem is given by:

$$\boldsymbol{b} = \left(\boldsymbol{A}^{\text{ext}^T}\boldsymbol{A}^{\text{ext}}\right)^{-1}\boldsymbol{A}^{\text{ext}^T}\boldsymbol{D}^{\text{ext}} \tag{8}$$

which minimizes:

$$\boldsymbol{error} = |\boldsymbol{D}^{\text{ext}} - \boldsymbol{A}^{\text{ext}}\boldsymbol{b}| \tag{9}$$

Furthermore, improvements can be made using a weight matrix as described before, similar to in low pass and high pass filters. Given a diagonal matrix $\boldsymbol{W}$ with elements corresponding to each frequency, the new solution to the LMS problem becomes:

$$\boldsymbol{b} = \left(\boldsymbol{A}^{\text{ext}^T}\boldsymbol{W}\boldsymbol{A}^{\text{ext}}\right)^{-1}\boldsymbol{A}^{\text{ext}^T}\boldsymbol{W}\boldsymbol{D}^{\text{ext}} \tag{10}$$

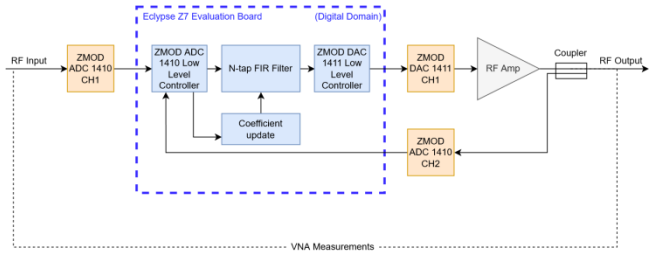
To find the most optimal weight matrix, the error was calculated recursively with looped values of weights and weight cutoffs. The weight cutoff and weight matrix that minimized this error was chosen to find the coefficients.

$$\begin{bmatrix} H(\omega_0) \\ H(\omega_1) \\ \cdots \\ H(\omega_{M-2}) \\ H(\omega_{M-1}) \end{bmatrix} = \begin{bmatrix} 1 & e^{-i\omega_0} & \ldots & e^{-i\omega_0(N-2)} & e^{-i\omega_0(N-1)} \\ 1 & e^{-i\omega_1} & \ldots & e^{-i\omega_1(N-2)} & e^{-i\omega_1(N-1)} \\ \ldots & \ldots & \ldots & \ldots. & \ldots \\ 1 & e^{-i\omega_{M-2}} & \ldots & e^{-i\omega_{M-2}(N-2)} & e^{-i\omega_{M-2}(N-1)} \\ 1 & e^{-i\omega_{M-1}} & \ldots & e^{-i\omega_{M-1}(N-2)} & e^{-i\omega_{M-1}(N-1)} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \cdots \\ b_{N-2} \\ b_{N-1} \end{bmatrix} \tag{5}$$

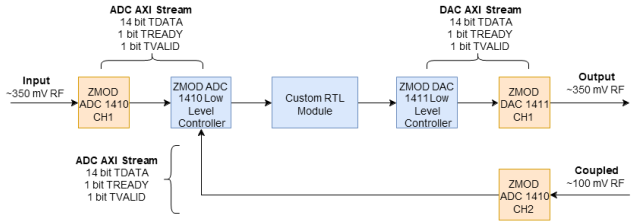
Since LMS is able to include real and imaginary desired values, a desired phase can be implemented such that we obtain a constant group delay. For this, we approximately add a group delay of $N/2\ ns$ to the desired frequency response $\boldsymbol{D}^{ext}$ firstly, and find the best group delay recursively.

## III. Implementation of Digital Predistortion Topology

The proposed design uses the predistortion topology as shown in Fig. 1. The chosen components in Fig. 3 are described in the following sections.




(a)




(b)

Fig. 3. Block diagram of the connections on an Eclypse Z7 FPGA board to implement the predistortion equalizer with an adaptive algorithm; (a) illustrates the connections and topology; (b) shows where the AXI protocol takes place and the use of 14 -bit data streams with TREADY and TVALID AXI handshakes.

### A. MATLAB Coefficient Generation

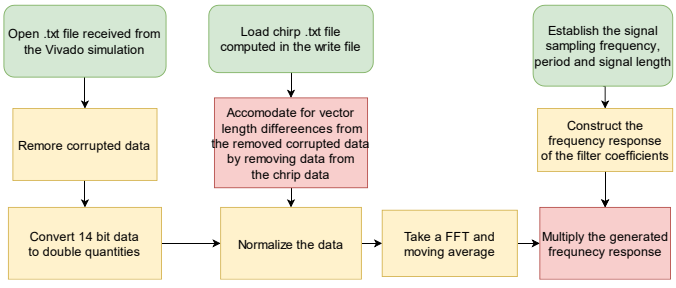



Fig. 4. Flowchart of calculating FIR coefficients using MATLAB.

For generating suitable filter coefficients for our digital equalizer filter we use MATLAB to process the DUT's s-parameters, format the least-mean squares algorithm (or any other method we think might work) for complex quantities, execute the least-means square algorithm for generating filter coefficients (matrix operations), easily generate the frequency response created by a filter with those coefficients, compute the error relative to the ideal response and easily plot all of these results. Its design process is shown in Fig. 4.

### B. MATLAB Sweep Read and Write

In order to simulate the frequency response on an FPGA board, the chirp signal with a flat magnitude reponse in a wide range of frequencies is used as the input signal. To simulate the same conditions as the experiment, a sampling time of 10 ns was chosen. The frequency of the chirp signal was set from 100 kHz to 50 MHz, which represents the entire bandwidth for the dynamic equalization case. The signal was converted from a decimal float, which it defaults to in MATLAB, to a signed 14-bit binary number which is used in Verilog. The signal was then saved to a .txt file to be read in Verilog. The FFT of the chirp signal was also done in MATLAB to confirm the flatness of its response.

The processed signal from the FPGA is now read into MATLAB for analysis and verification. The output signal was read and cleaned, then converted from a signed 14-bit binary number into a double-precision decimal value. Additionally, the initial signal created in the "write" stage was read for comparison. A sampling time of 10 ns was used in order to calculate the corresponding frequencies of the signals. The output signal was padded with zeroes due to a size mismatch with the input signal, caused by the delayed processing of the signal in the FPGA. The Fourier transform of both signals was taken using the fast Fourier transform and then processed. A moving average of 300 data points was taken in order to minimize the effects of noise near the endpoints. Additionally, the frequency response due to the filter coefficients was imported.

### C. Verilog Design

The project board was set to the Eclypse Z7 board file, which was provided by Digilent's repository [27]. The constraints file was also obtained from Digilent's repository, which provided the pin assignments of the FPGA, that could then be subsequently used in the IP Integrator.

In the IP Integrator, a clocking wizard was used to set 4 separate clocks from the 125 MHz Zybo clock: 200 MHz for the ADC System Clock, 100 MHz for the DAC System Clock, 100 MHz for the ADC and DAC Sampling Clock, and 100 MHz for the custom RTL module. Furthermore, a Processor System Reset IP module was implemented for the reset of the Zybo Z7 to communicate with the custom RTL module, ADC, and DAC.

All input and output connections (other than the tdata, tready, tvalid streams) were automated by the IP Integrator automated connector as depicted in Fig. 5. As shown in Fig. 1, the FIR filter works through delays, multiplications and additions. The custom RTL module included a circular buffer register, where in an input was set to the first register and the previous registers were shifted. This results in a delay for each input by some integer N along the circular buffer. Since the FIR filter works

at 100 MHz each input is delayed by 10ns. Every cycle they are delayed again and so forth.

Using this circular buffer, delayed_data, each component is taken and multiplied by the fixed, signed 16-bit valued fir coefficients, fir_coefficient that were pre-calculated in MATLAB. Since the input is signed 14 bit, and the multiplier is signed 16 bit, the output results in signed 30-bit. Each term is summed together with an accumulator and then truncated to signed 14-bit. The DAC_tdata is set to this 14-bit value, which is the "FIR filter time data". This cycle repeats every 10 ns.

To testbench this module, different valued inputs were required. As such, MATLAB read and write sweep programs were used to generate a sweep signal. The input was taken every 10 ns (100 MHz sampling) similar to the sampling rate of the ADC. The reset and ADC_tready and DAC_tvalid were tested as well to determine if the AXI stream handshake worked properly.

Finally, dynamic equalization was implemented using the aforementioned LMS adaptive algorithm. Feedback was returned to ADC CH2 (from post DUT equalization). The error was determined by subtracting this value from a delayed input stream. For the $k_{th}$ FIR coefficient, the error was multiplied by the current input delayed by 10k ns, as shown in Equation (3) [11, 26].

## IV. Module Testing and Experimental Results

### A. Software Test

The two core design pieces that needed to be simulated were the filter coefficients through MATLAB and the comparison of the chirp signal FPGA simulations of the passthrough and the filter core artificially programmed into it.

The goal of the coefficient simulation in MATLAB was to alter the filter coefficients until its frequency response multiplied with the frequency response of the DUT that had no more than 2 dB ripple over the specified frequency band and maintained a mostly constant group delay. With the FPGA simulations in Vivado, it was required to generate a time domain signal that could test the filter for comparison with the frequency response shown from the MATLAB simulation. One way to perform this was by generating a chirp signal, this chirp signal can characterize the frequency response of the device since it can be tuned to predictably cover all the specified frequencies over a predictable period of time.

The primary goal of the Vivado simulations was to show that the filter frequency response shown in MATLAB simulations from the set of coefficients designed beforehand was very close to the frequency response shown in Vivado. In doing this it was expected to get somewhat of a better idea of what limitations of the FPGA may do to the filter performance and how the design could potentially be altered to compensate for them with more work.

The output file was used to save the corresponding FIR waveforms so that the Fourier Transform could then be taken in the Sweep Read MATLAB.

### B. RTL Module Simulation

Simulations were done over a 10 ms period to fit the 10 kHz to 50 MHz chirp signal. The filtered analog signal is read out into a text file. After data processing, the frequency response is plotted in Fig. 6. There is a negligible error (due to the chirp frequency response) between the expectation (blue) and the simulation (red).

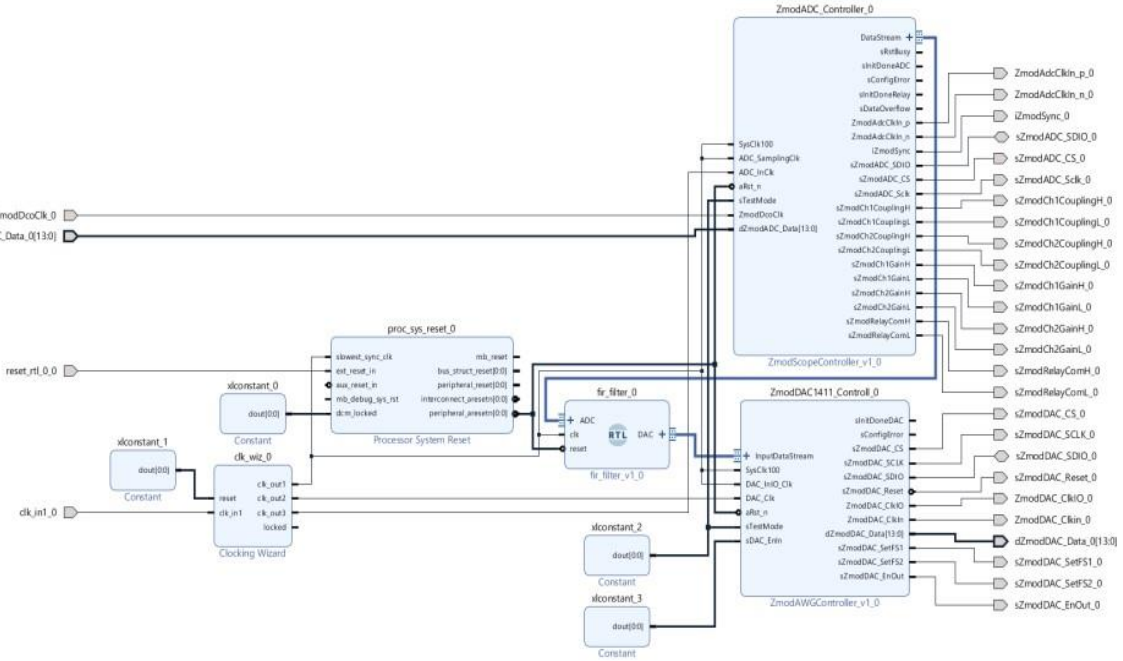



Fig. 5. IP block diagram of the IP Cores and custom RTL module in Vivado.

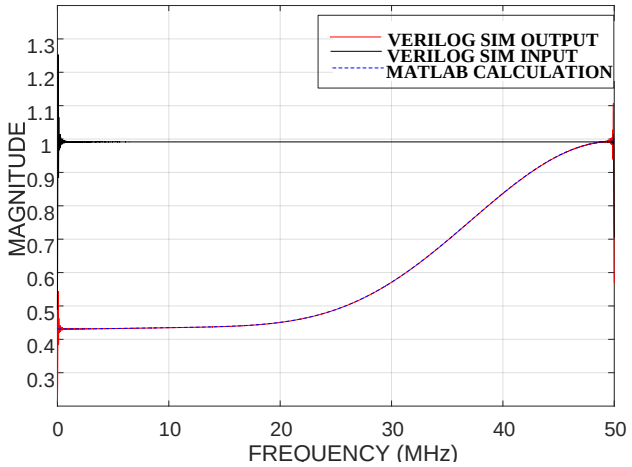
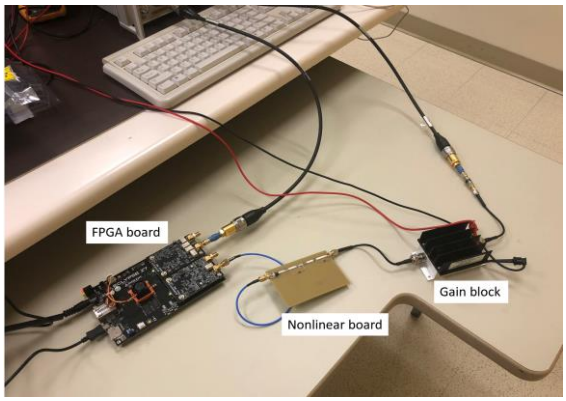


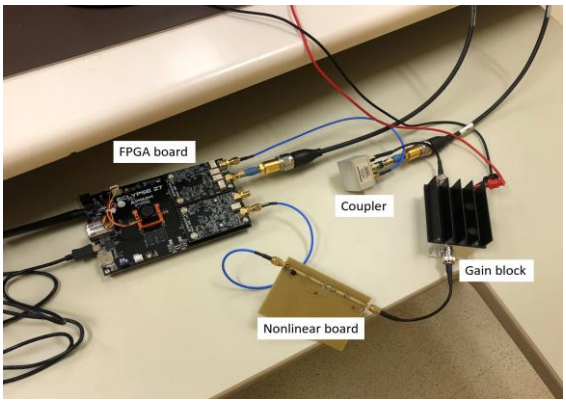

Fig. 6. Frequency response comparison of expected (blue) and simulated (red).

*C. Hardware Test*

The hardware testing was divided into three main parts: testing the DUT which consists of a self-made nonlinear board and an off-the-shelf gain block, testing the static digitally equalized system prototype, and testing the dynamical digitally equalized system prototype.




(a)




(b)

Fig. 7. Hardware connection of (a) Static predistortion equalization and (b) Adaptive predistortion equalization

The purpose of the static prototype testing was to see whether or not the performance of the filter coefficients (derived beforehand in the algorithm) that were programmed into the FPGA followed the frequency performance that was expected on a VNA. Once the single set of coefficients was performing equalization of the RF system to an acceptable level it was time to test the performance of the dynamic Verilog algorithm programmed on the FPGA.

The filter coefficients found from the development of the static equalizer were used as the initial coefficient guess into the dynamic equalizer. The performance baselines used in all the equalized prototype system tests were to show that the magnitude frequency performance didn't change (ripple) more than 2 dB and that the group delay flatness remained within 10ns error of the mean (almost entirely constant) for as much of the chosen frequency band as possible.

*D. Static Equalization Results*

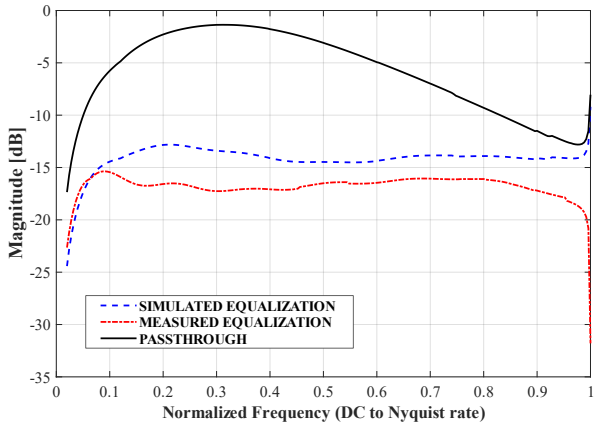



Fig. 8. The magnitude response of the system in the static mode: without equalization (black), simulated equalization (blue), and measured equalization (red).

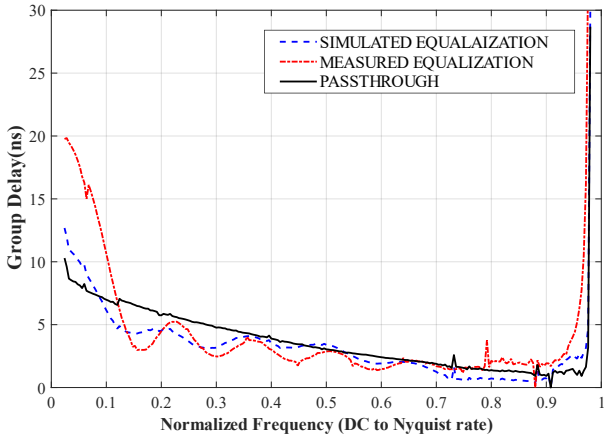



Fig. 9. The group delay of the system in the static mode: without equalization (black), simulated equalization (blue), and measured equalization (red).

With the aid of MATLAB, a 15-tap filter was found with coefficients. The frequency response was simulated and then the coefficients were implemented into the RTL. Fig. 7(a) shows the system connection. The response of the entire system was measured. The simulations and measurements were plotted in Fig. 8. Additionally, the group delay were measured as in Fig. 9.

### *E. Dynamic Equalization Results*

After the static equalization found promising results, the filter coefficients were used as the starting point for the program was synthesized, implemented, and a bitstream was generated that was programmed onto the Eclypse Z7. The system was connected as pictured in Fig 7 (b).

The frequency response of unequalized settings was measured using CH2 of the DAC. Then the frequency response of equalization was measured using CH1 of the DAC for comparison. Differing nonlinear boards cascaded by the same gain block were used as DUT #1 and #2. The magnitude response and group delays of the corresponding situations were plotted in Fig. 10 - Fig. 13. It can be found that our proposed adaptive digital equalizer exhibited excellent performance in the 0.2 -0.8 Nyquist frequency range.

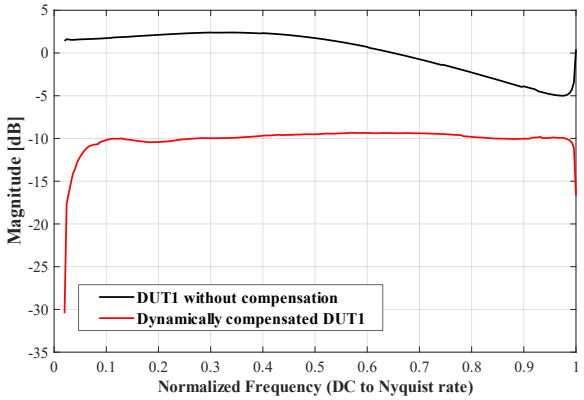



Fig. 10. The magnitude response of the system with DUT #1 in the dynamic mode

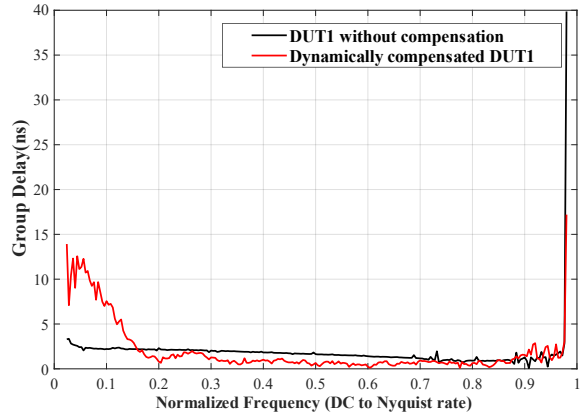



Fig. 11. The group delay of the system with DUT #2 in the dynamic mode.

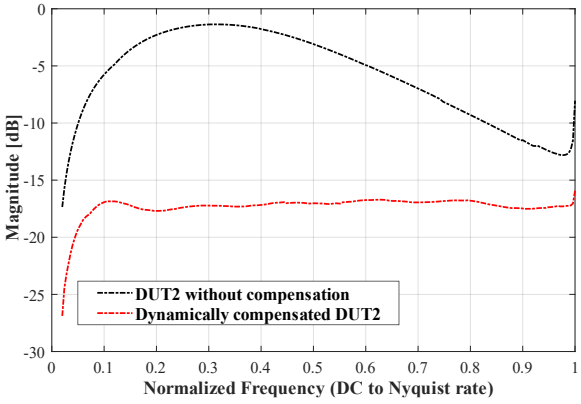



Fig. 12. The magnitude response of the system with DUT #2 in the dynamic mode.

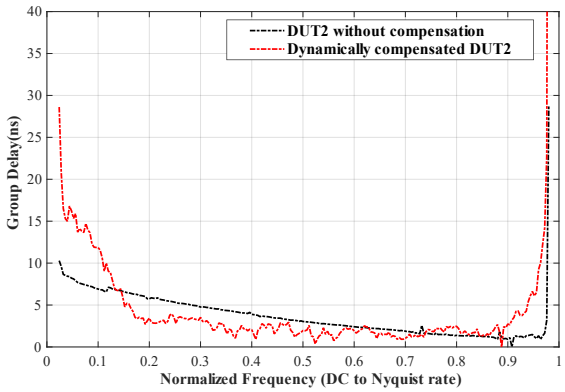



Fig. 13. The group delay of the system with DUT #2 in the dynamic mode.

## V. Conclusion

In this paper, we proposed a predistortion equalizer by using an adaptive LMS algorithm and implemented it onto an Eclypse Z7 FPGA platform that used a Zmod DAC and Zmod ADC. The prototype used a predistortion topology with a coupled feedback loop. In this way, adaptive equalization within 2 dB bandwidth was demonstrated in the 0.2 - 0.8 Nyquist frequency range (10 - 40 MHz in our case). At this range, the equalizer kept the phase response relatively linear with a group delay flatness less than 10ns. Different DUT configurations were dynamically examined to validate the design concept. It is foreseeable that our proposed adaptive digital equalizer will be widely used in wireless communication systems.

## References

[1] K. S. Choi, J. H. Kim, D. Ahn, N. H. Jeong and J. K. Pack, “Trends in rain attenuation model in satellite system,” 13th International Conference on Advanced Communication Technology (ICACT2011), 2011, pp. 1530-1533.

[2] D. G. Dudley, “Wireless propagation in circular tunnels,” in IEEE Transactions on Antennas and Propagation, vol. 53, no. 1, pp. 435-441, Jan. 2005, doi: 10.1109/TAP.2004.836407.

[3] J. Pingenot, R. Rieben and D. White, “Full wave analysis of RF signal attenuation in a lossy cave using a high order time domain vector finite element method,” IEEE/ACES International Conference on Wireless

Communications and Applied Computational Electromagnetics, 2005., 2005, pp. 658-661, doi: 10.1109/WCACEM.2005.1469674.
[4] G. A. Siles, J. M. Riera and P. Garcia-del-Pino, "Atmospheric Attenuation in Wireless Communication Systems at Millimeter and THz Frequencies [Wireless Corner]," in IEEE Antennas and Propagation Magazine, vol. 57, no. 1, pp. 48-61, Feb. 2015, doi: 10.1109/MAP.2015.2401796.
[5] G. A. Dulk, W. C. Erickson, R. Manning, and J. L. Bougeret, "Calibration of low-frequency radio telescopes using the galactic background radiation," Astronomy and Astrophysics, vol. 365, no. 2, pp. 294–300, Jan. 2001.
[6] S. Winder, Analog and digital filter design. Amsterdam: Newnes, 2007.
[7] A. Der Ziel, "Thermal noise in field-effect transistors," Proceedings of the IRE, vol. 50, no. 8, pp. 1808–1812, 1962.
[8] Y. Lin et al., "Analysis and Design of a CMOS UWB LNA With Dual-RLC-Branch Wideband Input Matching Network," in IEEE Transactions on Microwave Theory and Techniques, vol. 58, no. 2, pp. 287-296, Feb. 2010, doi: 10.1109/TMTT.2009.2037863.
[9] Y. Lin et al., "Analysis and Design of a CMOS UWB LNA With Dual-RLC-Branch Wideband Input Matching Network," in IEEE Transactions on Microwave Theory and Techniques, vol. 58, no. 2, pp. 287-296, Feb. 2010, doi: 10.1109/TMTT.2009.2037863.
[10] B. P. Lathi, Principles of signal processing and Linear Systems. Oxford University Press, 2009.
[11] D. E. Borth, I. A. Gerson, J. R. Haug and C. D. Thompson, "A flexible adaptive FIR filter VLSI IC," in IEEE Journal on Selected Areas in Communications, vol. 6, no. 3, pp. 494-503, April 1988, doi: 10.1109/49.1917.
[12] Mini-Circuits, "Coaxial Amplifier ZHL-6A+" REV. H M162646 datasheet, Jun. 2017.
[13] Mini-Circuits, "Coaxial Directional Coupler ZMDC-30-1+" REV. D M151107 datasheet, Mar. 2008.
[14] Diligent, Eclypse Z7 Board Reference Manual, Online.
[15] Digilent, "Zmod ADC 1410: SYZYGY-compatible Dual-channel 14-bit Analog-to-Digital Converter Module," SKU 410-396.
[16] Digilent, "Zmod DAC 1411: SYZYGY-compatible Dual-channel 14-bit Analog-to-Digital Converter Module," SKU 410-397.
[17] A. S. Chauhan, V. Soni, "Design of FIR filter on FPGAs using IP cores," International Journal of Advancements in Technology, vol. 4, no. 1, Mar., pp. 50-57, 2013.
[18] I. Kollar and Y. Rolain, "Complex correction of data acquisition channels using FIR equalizer filters," in IEEE Transactions on Instrumentation and Measurement, vol. 42, no. 5, pp. 920-924, Oct. 1993, doi: 10.1109/19.252527.
[19] McNelles, P., Lu, L. "Field programmable gate array reliability analysis using the dynamic FLOWGRAPH methodology," in Nuclear Engineering and Technology, March 31, 2016.
[20] "Tips for preventing spectrum analyzer damage," Keysight, 21 Feb, 2019. [Online].
[21] A. Brown, "Zmod Scope Reference Manual," Diligent. [Online].
[22] A. Brown, "Zmod AWG reference manual,"Digilent. [Online].
[23] John G. Proakis, Dimitris G. Manolakis, Third Edition Digital Signal Processing Principles, Algorithms, and Applications, Prentice Hall, 1996.
[24] Julius O. Smith III, "Spectral Audio Signal Processing," W3 Publishing, 2011, [Online].
[25] A. Mutapcic, S. J. Kim, S. Boyd, "Robust Chebyshev FIR Equalization," Stanford University, 2007.
[26] Y. T. Lai, C. C. Kao, H. J. Chen, "Design and Implementation of an Adaptive FIR Filter Based on Delayed Error LMS Algorithm," 1999 IEEE Workshop on Signal Processing Systems. SiPS 99. Design and Implementation (Cat. No.99TH8461), 1999, pp. 704-712, doi: 10.1109/SIPS.1999.822378.
[27] Digilent, Github Repository, https://github.com/Digilent
[28] ARM Developer Staff, AMBA 4 AXI4-Stream Protocol Specification V1.0, ARM, 2010.