Hardware Software Codesign of Applications on the Edge: Accelerating Digital PreDistortion for Wireless Communications

Zhaoyang Han
Department of ECE
Northeastern University
Boston, United States
zhhan@coe.neu.edu

Yiyue Jiang
Department of ECE
Northeastern University
Boston, United States
jiang.yiy@northeastern.edu

Rahul Mushini
Department of Electronic Engineering
Maynooth University
Maynooth, Ireland
RAHUL.MUSHINI.2019@mumail.ie

John Dooley
Department of Electronic Engineering
Maynooth University
Maynooth, Ireland
John.Dooley@mu.ie

Miriam Leeser
Department of ECE
Northeastern University
Boston, United States
mel@coe.neu.edu

***Abstract*—We present a real-time adaptive Digital PreDistortion (DPD) system developed on a System-on-Chip (SoC) platform with integrated RF front end, namely the AMD/Xilinx RFSoC. The design utilizes the heterogeneity of the RFSoC and is carefully partitioned. The control logic and training algorithm are implemented on the embedded ARM processor, while the predistorter module is placed on the FPGA fabric. To better coordinate both the hardware and software implementations, the training algorithm has been optimized for a shorter training time which results in a system that adapts to current environmental conditions with a shorter latency. Specifically, the number of signal samples used in training are reduced by applying the probability distribution information from the input signal in order to reduce the training time while retaining the important data samples. Results show that this reduced training set maintains the accuracy of the full data set. The implemented design balances the processing on the ARM processor and FPGA fabric resulting in a computationally efficient solution which makes good use of the different resources available. It has been experimentally validated on an AMD/Xilinx Gen3 RFSoC board with an exsternal GaN Power Amplifier (PA).**

***Index Terms*—Digital pre-distortion, power amplifiers, FPGA.**

I. Introduction

Wireless implementations for 5G and beyond require processing with low latency, hence much of this processing needs to happen on the edge. As the number of transmitter paths in base stations increase, a growing demand is placed on the baseband resources needed to implement digital correction techniques. More computationally efficient solutions that are demonstrated to work in practice on embedded hardware are needed. In this research we target a specific wireless application, namely Digital PreDistortion (DPD).

As with most cellular standards, the primary requirement is that the RF front-ends do not generate any unintended frequency components which would interfere with other users of the frequency spectrum. However, the desire for lower carbon footprints requires the use of more power efficient hardware, which in turn requires the use of non-linear devices that are more likely to generate unwanted frequency components. In order to remove these, digital signal processing can be applied to pre-distort the signal at baseband. The net effect results in the RF output signal being linear and not containing any unwanted frequency components.

The evolution of cellular wireless communications has seen its largest changes proposed for 5G. Multiple competing objectives are set for the wireless communication hardware for these systems. Higher data-rates, higher capacity and ultra-reliable low latency are simultaneously desired. Low latency of the signal through the hardware has in the past focused research on minimizing the total number of computations in the DPD hardware block. However, as the number of transmitter hardware paths increases more sophisticated designs are required, and a growing demand is placed on the baseband resources needed to implement digital correction techniques. Thus, computationally efficient solutions that are demonstrated to work in practice on embedded hardware are needed. The focus of our research is to identify how best to implement DPD such that it reacts rapidly to the current wireless environment.

Power Amplifiers (PAs) are one of the most important devices in modern wireless communication systems. They draw a great deal of power and also suffer from non-linearities when transmitting signals. DPD has been widely studied to alleviate the non-linearities introduced by the PA. We target an

This research was partially supported by MathWorks and AMD/Xilinx as well as by Science Foundation Ireland (SFI) under Grant Number 13/RC/2077_P2 and 18/CRT/622.

AMD/Xilinx System on Chip (SoC); specifically the RFSoC, which integrates RF frontend, FPGA fabric, and embedded ARM cores all on the same package [1]; to implement DPD in real time. The challenge is how best to use the available resources. In particular, which portions of the implementation should be mapped to the embedded processor, and which belong on the FPGA fabric? This paper addresses these issues as well as balancing the processing on each type of resource.

Implementation of digital pre-distortion (DPD) can be achieved by forming a series of basis functions using permutations of input signal samples, which are then multiplied by a set of weights and summed. The resulting signal, now pre-distorted, can be applied as the updated stimulus to the PA. The basis functions are commonly a subset of the Volterra series and are capable of linearizing a nonlinear dynamic system such as an RF power amplifier. Training the values for the weights in DPD is an iterative process, where in each iteration a calculated error is aimed to be progressively minimised until convergence is achieved [2]. Least squares has been employed successfully to calculate these weights for the DPD block. To minimize the total computational effort to calculate the weights on each iteration, a suitable strategy should be adopted to minimize the dimensions of the arrays and matrices used.

The output signal from the power amplifier is measured to identify how close it is to a linear output. Researchers [3], [4] have investigated the viability of using a bandlimited or low-rate ADC solution. Limiting the performance of the analog or mixed-signal components can save money and enable the use of lower frequency hardware, but requires the implementation of additional signal processing techniques such as compressive sampling [5] to reduce the number of signal samples used in the DPD weight calculation.

One approach targets the reduction of the total number of weights used in the pre-distortion function, to reduce the size of the matrix used in the least squares calculation in one dimension [6]. The second dimension depends on the number of time domain training signal samples used. Traditionally, a bandwidth which is a multiple of the original input signal is used, and the output signal is sampled at a rate equivalent to the rate of the original input signal. A large set of data points at this high sample rate need to be recorded to collect a training dataset covering a broad range of input signal levels and transitions between levels. By removing the unnecessary samples, a suitable sample rate can be chosen to minimize the mutual information between these consecutive time domain data points [7]. While this improves the accuracy of DPD, it requires a large number of consecutive sample points. An alternative is to take advantage of a-priori knowledge about the signal levels that will be most or least commonly experienced by the power amplifier. By only recording a predetermined smaller number of sample points at the various signal levels, enough information can be gathered on the relative performance of the PA, and the dimension of the matrix used in the least squares calculation of the DPD weights can be greatly reduced. Previous attempts [8] have used the probability density function of the signal sample values as a guide for how many samples at the various levels should be used. However, the solutions presented do not provide detail on how this can be realised in practice in an embedded system.

In this work, the complete DPD solution is demonstrated using only a Xilinx RFSoC FPGA and a high power RF GaN power amplifier. Using the FPGA programmable logic resources and embedded ARM processor on the RFSoC, a DPD solution is successfully validated through experimental measurement. Novel dimension reduction strategies are employed in the construction of the matrix used to calculate DPD weights to more heavily weight samples in non-linear regions. Removal of training sample points in this way results in improved DPD performance, which is quantified using Normalized Mean Squared Error (NMSE).

The contributions of this research are:

- Targeting the Xilinx RFSoC for a complete DPD solution which uses FPGA fabric, embedded ARM processor, and RF front end.
- Reducing the number of input samples used for training to accelerate training time without sacrificing performance. The algorithm selects the training samples according to the probability characteristics of the input signal in a manner that can efficiently be implemented in FPGA hardware.
- Experimentally validating DPD on the Xilinx RFSoC with training implemented on the ARM processor and DPD on the FPGA hardware making use of the reduced training dataset.

II. Background

A. Memory Polynomial Model

The Memory Polynomial (MP) model is a simplified form of Volterra series which is used to model a non-linear system. It consists only of products with the same time shift, see Eq. (1), and thus requires fewer coefficients compared with alternative models such as the Volterra Series.

$$y_{MP}(n) = \sum_{p=0}^{P} \sum_{m=0}^{M} a_{pm} x(n-m)|x(n-m)|^p \qquad (1)$$

Here a_{pm} are the model coefficients and n is the time index. P and M represent the degree of PA non-linearity and the memory depth of the model, respectively. x and y represent the input and output signals respectively.

B. DPD Coefficient Estimation

A commonly used architecture for training the coefficients of the digital predistortion model is the indirect learning architecture (ILA) using the theory in [9], shown in Fig. 1. The predistorter is an inverse model of the memory polynomial (Eq. 1), expressed by Eq. 2.

$$\tilde{x}(n) = \sum_{p=0}^{P} \sum_{m=0}^{M} d_{pm} x(n-m)|x(n-m)|^p \qquad (2)$$

The post-distorter is an exact copy of the predistorter. In other words, the $\tilde{x}(n)$ and $x(n)$ are swapped by $s(n)$ and $y(n)$ as shown in Eq. 3 in matrix form.

$$S = Yd \tag{3}$$

Therefore, using a linear estimator such as least squares, the coefficients matrix d can be obtained by matching $s(n)$ with $\tilde{x}(n)$ [10].

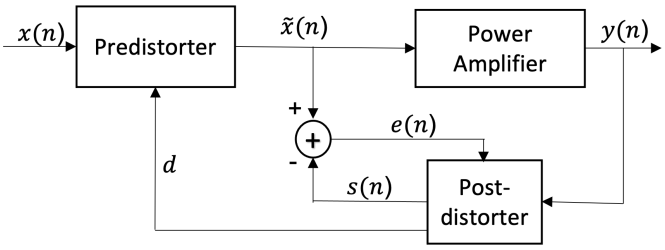

Fig. 1. Indirect Learning Architecture

The coefficients d_{pm} of the DPD model can be estimated through different methods. As there are many more observations of y and x than the number of coefficients, an overdetermined system can be formed. The optimal coefficients can be found by direct matrix inversion. The Least Squares algorithm is used to estimate the coefficients by minimizing the squared error J below, where N is the number of observations and x is the input signal. The matrix Y represents representations of the past output signal samples. The size of the matrix is M by N.

$$J = \frac{1}{N}\sum_{t=1}^{N}(x(t) - \hat{Y}^T(t)d)^2 \tag{4}$$

C. RFSoC Architecture

To achieve real-time processing of the DPD, an FPGA based System-on-Chip (SoC) board is used. It consists of three parts, the FPGA Programmable Logic (PL) the Processing System (PS) which is an embedded ARM processor, and a high-speed integrated RF-front end with 16 pairs of 16-bit DACs and 14-bit ADCs. The PL is used for high throughput, computationally expensive algorithms. The ARM processor is good at control processing such as task scheduling and can deal with high-precision floating point calculations, although at a relative low speed. To effectively make use of the PL and PS, a proper hardware-software co-design scheme is required as presented in Sec. III.

D. Related Work

The FPGA based SoC chip and boards containing such chips have been used for several implementations of DPD designs [11]–[14]. None of this prior research, however, implements real-time processing on the chip. Previous work makes use of offline processing on a host computer to run MATLAB scripts for modelling the power amplifiers (PAs). In [15], the author proposes an improved LUT-based DPD design on FPGA and also implements an estimation block for coefficient updating. However, details of the estimation block are not presented. In addition, the training block is implemented on the FPGA fabric; however, the training algorithm includes floating-point division which slows down the FPGA clock rate.

Recent research [16], [17] proposes parallel processing architectures for the predistorter model on an FPGA in order to meet the requirements of current high sampling rate RF front ends. In [16], the coefficients are computed externally instead of on the board. In [17], the ARM processor on the ZCU102 is mentioned for coefficient training and updating. However, the main focus of the paper is the novel architecture of the predistorter, and the implementation of the training algorithm is not described.

Others [18] propose a complete real-time, FPGA-based digital predistortion design and implementation on Xilinx's ZC706 SoC which also utilizes the embedded ARM processor for model training. However, due to the processing limitations and memory bandwidth of the ARM processor, the coefficients are only updated when the feedback signal's Normalized Mean Squared Error (NMSE) does not meet a required target.

In contrast, this work describes and implements a real-time DPD design. Our implementation focuses on both the training and DPD. Specifically, we examine the design choices of the training algorithm and the effect of the number of training signal samples used. An improved DPD design with compressed training sets for the adaptation algorithm is built completely on the Xilinx RFSoC.

III. Architecture

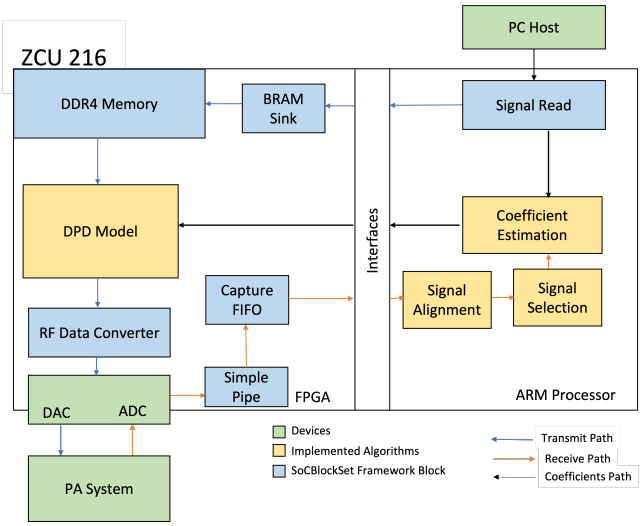

Fig. 2. DPD on RFSoC ZCU216

In this research all aspects of processing of a DPD algorithm are mapped onto a Xilinx RFSoC, which includes FPGA fabric, an embedded ARM processor and an RF frontend. Our architecture is flexible enough to support loading the signal to be transmitted from any protocol. It also can support many different DPD algorithms; in this research we use the memory polynomial as described above. The DPD system developed is

partitioned into two parts. The memory polynomial, as shown in Eq. (1) consists of hardware-friendly multiplications and accumulations that can be easily implemented in parallel and are mapped to FPGA fabric. The training algorithm, as described by Eq. (4), involves matrix inversion and control conditions. Therefore, the training algorithm and corresponding control logic are placed on the ARM processor.

A. Design Blocks

The implementation, including the division of processing blocks between the FPGA fabric and the ARM processor is shown in Fig. 2.

The RFSoC includes an embedded ARM processor, which is also referred to as the Processor System (PS). We map various tasks to the PS including training and handling of interfaces between the FPGA and the host. The software running on the ARM processor handles updating the coefficients used by the DPD model.

The signal to be transmitted is first sent from a host machine over Ethernet and stored on the processor side. The signal is then loaded onto the external DDR4 memory on the Programmable Logic (PL) side where it will be repeatedly sent through the DAC on the RFSoC. On the receive path, when the signal after going through the power amplifier is received, it will be sent back to the PS and used to retrain the DPD model. Revised coefficients are applied to the DPD processing on the FPGA fabric. This provides a very tight loop and allows the hardware to react to the environment with low latency.

At the embedded processor, the received signal is first aligned with the original signal within the Signal Alignment block and then the coefficients are determined for the DPD block by making use of selected training data and using a memory polynomial model. The ARM processor has two processing cores. In our design, one is used for transmitting the signal, while the other is used for receiving and post-processing the feedback signal.

On the hardware side, a DPD block with memory polynomial is implemented in the FPGA fabric along with memory sink and capture blocks for handling the data transfer between software and hardware. We currently process two samples per clock cycle. The blocks are enabled to process multiple input samples in parallel to meet the high RF-DAC sampling rates [17], where up to eight samples per clock cycle can be processed. Our design can easily be modified to handle this case.

B. Trade-off when Choosing Training Data Sets

In post-processing, one of the parameters to be determined is the size of the training set. While a larger training set increases the accuracy of the DPD model, it also burdens the processor and results in a longer training time. In addition, the bandwidth of the interfaces between the FPGA hardware and ARM processor is limited. In the interface generated by the MathWorks SoC blockset, one 32 bit AXI stream forms the connection between PS and PL. Note that Xilinx's DMA IP can support a maximum 300MB/s data rate for moving data across this boundary. While there is room to improve the implementation, this interface forms a bottleneck of the design.

In our implementation, on the FPGA side, the logic runs at 256 MHz while the ARM processor runs at 950 MHz, but processes data serially. The FPGA side will generate two 16-bit IQ samples in each clock cycle. However, since the training algorithm is complex, the ARM processor can not match the FPGA processing rate.

Instead of training with all the received signals, a subset of the received signals is used. Under the control of the ARM processor, only a small amount of the received signal samples are sent across the hardware-software boundary for training.

To further alleviate the burden on the ARM processor, the subset received by the ARM processor has been compressed according to its probability information. Some of the samples in the subset have been discarded in such a way as to ensure that non-linear samples are preserved, as described in Sec. IV.

By using this two-step compression approach (subsetting followed by intelligent compression), we show that a small subset of the received signals can be used for training without compromising accuracy.

C. Design Flow

The design is built using the SoC Blockset from Mathworks, which supports the Xilinx RFSoc. we use this for implementation of DPD on the RFSoC. The flow has the advantage of rapidly generating a solution and allows us to easily experiment with different implementations and tradeoffs. The design is entered as a Simulink model; the FPGA logic is generated from that model using HDL Coder while Embeddedd Coder produces the software implementation on the ARM processor. The SoC Blockset provides a framework for controlling the signal reads and writes and communication between hardware and software through tunable parameters. These parameters determine how frequently the training happens and how many samples are processed during each training. Determining the best parameters for our setup will be further investigated.

IV. Compressed Data Set for Training

To reduce the amount of computation on the processor without losing the nonlinear characteristics of the training dataset, the probability information of the input signal is used to guide the selection of a compressed training dataset [8]. Here we use a probability density function-based training data compression method to reduce the amount of data, which has several benefits. It better balances the processing speed between the ARM processor and FPGA fabric and lowers the requirement for data transfer between the two. Instead of using consecutive samples, we select samples based on the training signal's power density probability.

We choose the fraction of training signal samples for dataset compression. Then we choose a resolution for the probability density function and generate the histogram of the training signal. We randomly select samples from each bin of the histogram in proportion to the bin size and also keep the

previous samples according to the memory tap size. Finally, we use the compressed training signal to calculate the coefficients of the DPD. Using this pdf-based selection criteria, not only are the nonlinear characteristics of the training signal fully preserved, but the memory effects are also expressed, as shown in Fig. 3. The compressed signal (the orange histogram) is the output of the selected samples of the signal (the yellow histogram) with their previous samples according to memory depth, and it almost recovers the original signal (the blue histogram) with a smaller sample size. In the experiment presented here, the number of training samples is reduced from 40k to 16k with sample ratio of 0.1, which significantly lowers the burden on the data buses between hardware and software as well as speeding training.

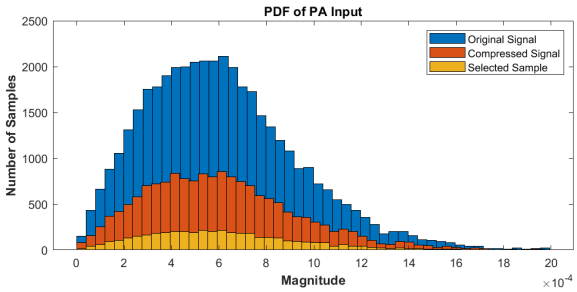

Fig. 3. Histogram of original signal and compressed signal

V. Hardware Implementation and Experimental Results

The experimental testbench setup, shown in Fig. 4, makes use of a Xilinx RFSoC ZCU216 board. The test power amplifier, an NXP AFSSC5G37D37 Doherty PA, is connected with a linear driver stage power amplifier, NXP BGA7210. The PA is connected back to the ADC receive path on the RFSoC ZCU216 through a 40dB attenuation chain. The center frequency for operating the PA is set to 3.7GHz, which is a widely used band for 5G signals.

Without DPD the NMSE of the received signal is -10.92dB. After DPD using training with 40,960 samples the NMSE improves to -20.48dB. When using the compressed data training set of 15,360 samples, the NMSE performance was essentially unchanged from the full training set, achieving an NMSE of -19.56dB. These results are illustrated in Figure.5.

These results are achieved for a training dataset that was half the size of the original training set. As a result, this compressed training set reduces the time to transmit data from the FPGA to the ARM processor, as well as the time to compute new coefficients using the least squares algorithm. The time complexity of least squares is $O(k^2N)$, where k is the total number of coefficients in the DPD model and N is the number of samples. Since this is linear in the number of training samples, by compressing the training data the time for training is reduced by half. Note that the minor increase in complexity for binning and selecting the data is $O(N)$. Therefore, when the number of coefficients k increases and

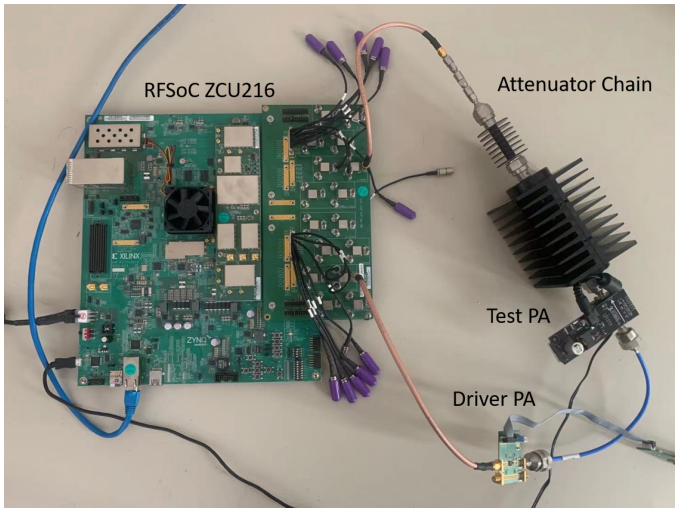

Fig. 4. DPD on RFSoC ZCU216

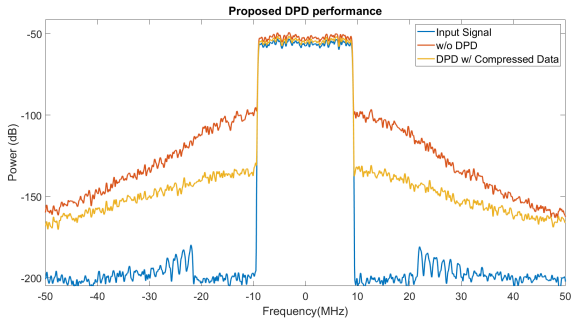

Fig. 5. Spectrum Analysis of Implemented DPD

thus the DPD model is more complex, the compressed data sets will reduce the training time even further. In our experiments, we reduced training data from 40,960 to 15,360 samples, which in theory should result in an improvement of 62.5% in training time. Our experimental results were very close to what was predicted in theory, and show an 61.08% improvement in training time.

VI. Conclusions

In this paper, a real-time adaptive digital pre-distortion architecture is designed and implemented on the state-of-the-art FPGA-based Gen3 Xilinx RFSoC. Our DPD implementation takes advantage of having both FPGA and embedded ARM processor on the same chip, and carefully partitions and places different parts of the system on different processing units. Specifically, a memory polynomial DPD is implemented on the FPGA fabric while training is performed on the ARM processor. The DPD training speed has been greatly improved by reducing the number of training samples. By selecting specific signal samples for the training set, the burden on the ARM processor is reduced, and the DPD model can be updated more frequently. The signals are selected in order to ensure that the characteristics of the original signals are preserved. This was achieved with minimum additional hardware resources

and achieves the same quality and improved performance as using a larger set of training samples. The result is a DPD implementation experimentally validated on the RFSoC that can rapidly adapt to changing conditions in its environment.

In the future, additional trade-offs between hardware and software implementations will be investigated, as well as bottlenecks in the design, to further improve the performance of the overall system.

REFERENCES

[1] Xilinx, "Xilinx RFSoC," 2022. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html

[2] R. N. Braithwaite, "A comparison of indirect learning and closed loop estimators used in digital predistortion of power amplifiers," in *IEEE MTT-S International Microwave Symposium*, May 2015.

[3] L. Ding, F. Mujica, and Z. Yang, "Digital predistortion using direct learning with reduced bandwidth feedback," in *2013 IEEE MTT-S International Microwave Symposium Digest (MTT)*, Jun. 2013.

[4] H. Huang, P. Mitran, and S. Boumaiza, "Digital Predistortion Function Synthesis using Undersampled Feedback Signal," *IEEE Microwave and Wireless Components Letters*, vol. 26, no. 10, pp. 855–857, Oct. 2016.

[5] E. J. Candes and M. B. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[6] D. Byrne, R. Farrell, and J. Dooley, "Sample Pruning Based on Leverage for Digital Pre-Distortion," in *Int. Workshop on Integrated Nonlinear Microwave and Millimetre-Wave Circuits (INMMiC)*, Jul. 2020, pp. 1–3.

[7] M. B. Kennel and H. D. I. Abarbanel, "False neighbors and false strands: A reliable minimum embedding dimension algorithm," *Phys. Rev. E*, vol. 66, p. 026209, Aug. 2002.

[8] Z. Wang, J. Dooley, K. Finnerty, and R. Farrell, "Selection of Compressed Training Data for RF Power Amplifier Behavioral Modeling," *10th European Microwave Integrated Circuits Conference*, 2015.

[9] M. Schetzen, "Theory of pth-order inverses of nonlinear systems," *IEEE Transactions on Circuits and Systems*, vol. 23, no. 5, pp. 285–291, 1976.

[10] R. M. Elaskary, A. H. Mehana, Y. Fahmy, and M. El-Ghoneimy, "Closed-form analysis for indirect learning digital pre-distorter for wideband wireless systems," *Physical Communication*, vol. 50, 2022.

[11] H. Rezgui, F. Rouissi, and A. Ghazel, "Fpga implementation of the predistorter stage for memory polynomial-based dpd for ldmos power amplifier in dvb-t transmitter," in *Conference on Advanced Systems and Electric Technologies (IC_ASET)*. IEEE, 2017.

[12] S. Juárez-Cázares, A. Meléndez-Cano, J. Cárdenas-Valdez, J. Galaviz-Aguilar, C. Vazquez-Lopez, P. Roblin, and J. Núñez-Pérez, "Fpga-based modeling and design methodology of a digital pre-distortion system for power amplifier linearization," in *Mechatronics, Electronics and Automotive Engineering (ICMEAE)*. IEEE, 2016.

[13] G. C. L. Cunha, S. Farsi, B. Nauwelaers, and D. Schreurs, "An fpga-based digital predistorter for rf power amplifier linearization using cross-memory polynomial model," in *2014 International Workshop on Integrated Nonlinear Microwave and Millimetre-wave Circuits (INMMiC)*, 2014, pp. 1–3.

[14] Z. Han, M. Loughman, Y. Jiang, R. Mushini, M. Leeser, and J. Dooley, "Computationally efficient look-up-tables for behavioral modelling and digital pre-distortion of multi-standard wireless systems," in *International Conference on Cognitive Radio Oriented Wireless Networks, International Wireless Internet Conference*. Springer, 2022, pp. 39–55.

[15] J. Ren, "A new digital predistortion algorithms scheme of feedback fir cross-term memory polynomial model for short-wave power amplifier," *IEEE Access*, vol. 8, pp. 38 327–38 332, 2020.

[16] W. Li, E. Guillena, G. Montoro, and P. L. Gilabert, "Fpga implementation of memory-based digital predistorters with high-level synthesis," in *2021 IEEE Topical Conference on RF/Microwave Power Amplifiers for Radio and Wireless Applications (PAWR)*, 2021, pp. 37–40.

[17] H. Huang, J. Xia, and S. Boumaiza, "Novel parallel-processing-based hardware implementation of baseband digital predistorters for linearizing wideband 5g transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 68, no. 9, pp. 4066–4076, 2020.

[18] N. Kumar, M. Rawat, and K. Rawat, "Software-defined radio transceiver design using fpga-based system-on-chip embedded platform with adaptive digital predistortion," *IEEE Access*, vol. 8, 2020.