CHALMERS
UNIVERSITY OF TECHNOLOGY

DSP Hardware Tradeoffs for Digital Predistortion based on Pruned Volterra Series

DSP Hardware Tradeoffs for Digital Predistortion based on Pruned Volterra Series

Mubanda Talemwa*, Jorge Muñoz Bautista†, Thomas Eriksson†, and Per Larsson-Edefors*
*Dept. of Microtechnology and Nanoscience, Chalmers University of Technology, Gothenburg, Sweden
†Dept. of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden

***Abstract*—This paper presents an investigation into the hardware implementation of a pruned Volterra series based predistorter using floating-point arithmetic. Reduced-complexity pruned Volterra series have been shown to deliver good linearization performance for Digital Predistortion (DPD) in RF Power Amplifiers (PAs). However, this performance has been demonstrated mostly in software applications such as Matlab. In contrast, we explore DSP hardware design tradeoffs where we balance system performance and DPD power consumption. Our investigation is centered on the BAsis Propagating Selection (BAPS) algorithm that has been shown to deliver better performance compared to other pruned Volterra algorithms, such as memory polynomial or generalized memory polynomial, with fewer parameters. We develop a DPD processor prototype and perform a comparison for different sets of BAPS basis functions to study their effect on both DPD power consumption and performance in terms of PA linearization expressed as Adjacent Channel Power Ratio (ACPR). We present a set of basis functions with 37 % lower power consumption and less than 2 dB performance loss compared to BAPS baseline solution.**

***Index Terms*—Linearization Techniques, Digital Predistortion, Power Amplifiers, Energy Efficiency, Digital Signal Processing**

I. Introduction

Modern wireless communication systems rely on high-bandwidth signals to meet increasing data-rate demands, leading to non-negligible memory effects in the output of the RF Power Amplifiers (PAs). Moreover, the need to improve RF PA efficiency, by increasing output power, results in nonlinear effects. Digital Predistortion (DPD) is a technique used to linearize the PA's output and to minimize memory effects [1]. PA nonlinearity and memory effects can be modeled using Volterra series [2]. Increasing the nonlinearity order or memory depth of the Volterra model results in a better approximation of the RF PA behavior [3]. However, when the full Volterra series expands to higher nonlinearity or memory orders, it rapidly becomes complex and computationally infeasible for real-time DPD algorithm implementation. Thus, to reduce model dimensionality, a common solution for practical Volterra series based DPD algorithm implementation is pruning the series to maintain high accuracy, in terms of Normalized Mean Square Error (NMSE) and Adjacent Channel Power Ratio (ACPR), while reducing computational complexity expressed in floating point operations (FLOPs) [4].

Measuring the complexity of DPD algorithms by mainly using the FLOP metric may poorly estimate the power consumption of the DPD algorithm hardware implementation. For example, it may not take into account potential energy-efficiency improvements, which could reduce the overall number of operations necessary for a particular result. There are assumptions on how computationally expensive multiply and accumulate (MAC) operations are [5]. However, the overall computational cost is intricately linked to the final hardware implementation [6]. This is because logic synthesis tools perform numerous optimizations to remove redundancies and share resources, and because signal and clock gating creates a nonlinear mapping of algorithm to circuit power dissipation.

In this work, we implement a low-complexity DPD processor as a *baseline case* and use it to evaluate power consumption for different sets of Volterra series bases. The choice of design for the predistorter is a floating-point serial architecture because it eases the identification of the parameters that impact accuracy and power. The design also greatly reduces computational resources by recursively calculating basis functions. A Matlab script generates the basis functions' dictionary and coefficients for each basis function. The synthesized predistorter calculates the bases at runtime and performs the filtering. Parameter identification is performed using the indirect learning architecture (ILA) [7], [8]. Both NMSE and ACPR measurements are performed on the output from RF Weblab [9] PA to evaluate performance.

II. Pruned Volterra series

A pruned Volterra model like the one shown in equation (1) consists of a set of basis functions denoted ϕ_r and corresponding coefficients θ_r:

$$y(n) = \sum_{r=1}^{R} \theta_r \phi_r(n) \tag{1}$$

There are numerous examples of pruned Volterra series in literature, e.g., the memory polynomial (MP) [10], the generalized memory polynomial (GMP) [11], and BAsis Propagating Selection (BAPS) [12]. In this paper, we will use BAPS for our predistorter, but the implementation of the predistorter is such that any other pruned Volterra series can be used.

A. BAsis Propagating Selection (BAPS)

The BAPS model described in [12] is an iterative greedy search algorithm that generates an optimized set of basis functions. measured by least NMSE. The complexity in basis calculation is reduced by reusing previously generated basis functions, meaning that the algorithm is inherently serial. Furthermore, the complexity in filtering is reduced as BAPS

needs less bases to meet the desired NMSE and ACPR compared to many other pruned Volterra series [12].

B. Constructing Volterra series basis recursively

As demonstrated in [12], any Volterra series basis function can be represented recursively using two types of operations: *type I* and *type II*. A type I operation entails a delay of a previous basis function, while a type II operation involves a multiplication of three previous basis functions. The input sample $x(n)$ is the first basis function $\phi_1(n) = x(n)$, in iteration $r = 1$. For subsequent iterations r, the basis function generation operations are given by equation (2)

$$\begin{aligned} \text{Type I}: \quad & \phi_r(n) = q^{-m}\phi_i(n) \\ \text{Type II}: \quad & \phi_r(n) = \phi_i(n)\phi_j(n)\phi_k^*(n) \end{aligned} \tag{2}$$

where, for type I operations; m denotes the delay of a basis function from previous iterations i, while for type II operations; $\phi_i\phi_j\phi_k^*$ denotes the product of basis functions from previous iterations $i, j,$ and $k \leq r$, and * denotes the complex conjugate.

A dictionary can be used to encode the construction of basis functions [12]. Then each entry in the dictionary represents an iteration and contains three fields. For type II operations, the fields are the indices i, j, and k, whereas for type I operations, the fields are index i, the delay m and 0.

III. DPD Processor Design

The 32-bit floating-point prototype of the predistorter processor takes as inputs: a complex-valued signal, a list of complex coefficients, and a basis function construction dictionary. Then it outputs a complex predistorted signal. The basis function construction dictionary is referred to as an *indices table*, simply because the iteration at which a basis function occurs is taken as its index in the dictionary.

A. Architecture and Operation

The DPD processor decodes the indices table to iteratively reconstruct basis functions using operations of type I and type II as shown by equation (2), then performs a MAC operation on the coefficients and constructed basis functions as shown by equation (1). In [12], type I operations can comprise both causal and non-causal terms, i.e., positive and negative time delays, respectively. However, for simplicity, this prototype design only supports causal terms. Figure 1 shows the major design blocks and general operation.

1) Operation: At runtime, the user sets the `SAMPLE READY` bit when an input sample is available, and subsequently the design raises the `REQUEST SAMPLE` flag when the output is available. The `REQUEST SAMPLE` flag is lowered when an input sample has been registered for processing. This *request-acknowledge* process continues until halted by the user when the `SAMPLE READY` bit is left unset. The `Next Entry` control signal reads an entry from the indices table and sets a corresponding coefficient to the MAC block. The `Set Operation Type` configures the type of operation to be evaluated by the Basis Function

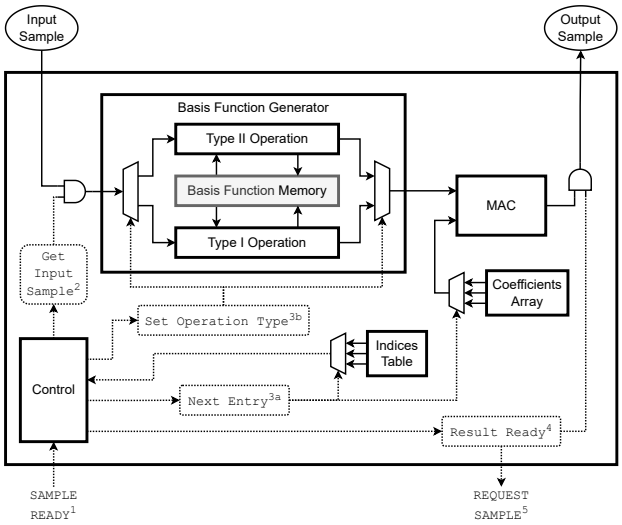

Fig. 1. General design and operation. The dotted lines show control operations, solid lines show data flow, and the thick solid lines are the blocks. The superscript numbers show the order of operations. Operations labeled 3a and 3b show the basis function generation iterations.

Generator (BFG) block. The controller continues this process until the last basis function is generated, then the final output is produced by the MAC block.

2) Timing: The design uses parallel 32-bit floating-point adders and multipliers, therefore, only one cycle is required to complete an addition or multiplication operation, i.e., one cycle is used by type II operations for a non-register access operation. Due to sequential retrieval of data from three memory addresses to perform double multiplication, type II operations require two more memory accesses than type I and three more cycles to complete.

The MAC operation always takes two cycles. Similarly, additional pipelining registers in the design always require a constant number of cycles to complete processing of every input sample. Therefore, the time it takes to process an input sample will vary depending on the operations in the indices table, but can be easily estimated from the type of operations in the indices table.

B. Basis Function Generation

Calculated basis functions are stored in a *shift register file* that is a two-dimensional memory; with rows representing the number of basis functions, and columns the memory depth. The first column stores the basis functions that are being constructed for the input sample that is being processed. Unlike pruned Volterra series such as GMP, the unconstrained BAPS does not allow for predetermining the type of operation at a given iteration. This is why we have two-dimensional memory even though several basis functions may not be type I. Depending on the indices table, registers not storing memory terms are dynamically clock gated to reduce power consumption. Figure 2 shows the BFG block in more detail, showing memory registers to construct six basis functions

and a memory depth of two; the first basis function is the input sample itself, and its construction is denoted as a *type 0* operation. Memory terms are stored by shifting each column by one step before a new input sample is read.

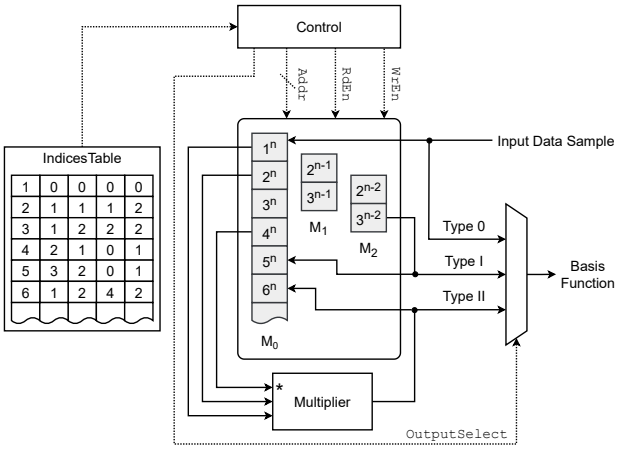

Fig. 2. General design and operation of the BFG block with a 3-input multiplier with one of the inputs conjugated, and a 3-column memory showing six rows. The construction of basis functions 5 (type I) and 6 (type II) are shown. The superscript on the memory registers denotes the column or memory delay (see equation (2)). Clock-gated registers for the other two memory columns (M_1, M_2) are not shown for brevity. Control signals are shown by dotted lines.

IV. Results

The predistorter processor design is synthesized using Cadence Genus [13] in the predictive ASAP7 cell library [14] for a 220-MHz clock rate. Logic design validation and netlist power simulation are performed using Cadence Xcelium [15], based on a data trace from measurements on a fabricated RF PA using the RF Weblab interface [9]. The Weblab system allows a user to upload input signal samples to a remote PA, and receive the amplified signal. While Weblab has a class AB configured PA operating under stable conditions, the PA's transfer function may drift slightly between characterizations.

A. Power Simulation

Power consumption is determined by not only the indices table operation types, but also operation dependencies, input data, coefficients, and actual implementation of operation execution blocks.

We analyze power consumption by extracting dynamic power, focusing on the BFG block. Table I shows the percentage average dynamic power consumption of the major design blocks; dynamic power is obtained by taking an average from simulation with 27 different sets of basis functions, each of which represents a possible architectural design. The BFG block includes type II complex multiplication and registers that store memory terms for type I operations. The BFG block's dynamic power dominate overall dynamic power consumption, with *approx.* 95 % of its power being consumed by type II operation product. Power analysis begins when the rest of the system power is in a steady state. The steady state is reached several cycles after initialization, when data fills up all pipeline stages so that other blocks, beside the BFG, consume approximately constant power to process a nonzero input sample.

TABLE I
Percentage dynamic power consumption per block

Block	**Power (%)**
Design	100
BFG	65
- Delay registers (type I)	3
- Complex multiplier (type II)	62
MAC	17
Other	18

Delay registers in the design are to a significant degree clock gated, which reduces their power consumption. Similarly, the MAC block will always perform the same number of operations for every input sample and consumes fairly constant power for all samples when compared to the BFG block. Optimization of the design is, therefore, steered towards minimizing type II operations to lower overall power consumption. This can be done by finding a set of basis functions, i.e., an indices table configuration, that reduces power consumption without compromising accuracy.

B. Indices Table Configuration

In order to investigate power consumption for basis function generation, we performed power simulations on the design using different sets of basis functions. Different indices tables will lead to different order and dependencies of type I and type II operations, and this in turn creates different bit patterns in the design that affect power consumption. Therefore, having different indices tables is useful in *exercising* our design to extract information about its power consumption. Indices table *configuration* refers to the order of operations, while *composition* refers to the dependencies of operations.

An initial indices table is obtained by running an unconstrained BAPS algorithm in Matlab. The unconstrained BAPS algorithm will generate basis functions depending on the PA's characteristics and operating conditions. This initial configuration acts as a benchmark, and we denote it the *baseline*. In addition to the baseline indices table, we created 27 indices table configurations by *biasing* the BAPS system identification algorithm to favor a desired table configuration. Generally, desired indices tables should have at least one type II operation, but type II operations should not comprise more than half of all operations. After system identification, the runtime performance, in terms of NMSE and ACPR, is calculated from the linearized PA output, and together with the design's power consumption are used to find a configuration with a good tradeoff.

For every configuration, system identification with BAPS is performed to generate thirteen basis functions and coefficients, using the Weblab PA setup and ILA for parameter identification. The PA input is a complex baseband signal generated

as bandlimited Gaussian noise. To reduce the variability in results, power simulations are performed with the same input signal and the same number of operations for all indices table configurations. Power analysis is performed for 32 input samples and a total of $13 \cdot 32 = 416$ operations. For each configuration, system identification is performed on five different signals, generating five runs of the same configuration. Here, the *average* NMSE, ACPR, and dynamic power, of the five runs for each configuration, are retrieved.

Five of the 27 configurations have a good power *vs.* accuracy tradeoff, i.e., less than 2 dB loss in performance and lower power consumption, compared to baseline. Table II shows the configurations and the number of type I and II operations in each. Configuration CFG_1 has type II operations followed by type I operations. Configuration CFG_2 has type I in the middle of the indices table with type II operations above and below. Configuration CFG_3 has type I operations followed by a type II operation in alternating fashion. Configurations CFG_4, and CFG_5 are arranged similarly to CFG_2. CFG_6 has only type I operations, while CFG_7 has only type II operations. CFG_6 and CFG_7 have the least and highest power consumption respectively, and have been added for comparison purposes.

TABLE II
INDICES TABLES' CONFIGURATIONS

Label	Type of Operations	
	Type I	Type II
CFG_1	9	3
CFG_2	8	4
CFG_3	6	6
baseline*	6,6,6,5,6	6,6,6,7,6
CFG_4	9	3
CFG_5	10	2
CFG_6	12	0
CFG_7	0	12

* table composition for five runs in corresponding order

Table III shows dynamic power and performance of the different configurations compared to the baseline configuration.

TABLE III
AVERAGE DYNAMIC POWER AND PERFORMANCE

Configuration	Power (µW)	NMSE (dB)	ACPR (dB)
CFG_1	885	-37.9	-43.0
CFG_2	1125	-38.5	-43.3
CFG_3	1249	-38.9	-43.3
baseline	1269	-38.9	-43.4
CFG_4	971	-37.7	-42.9
CFG_5	791	-37.6	-42.8
CFG_6	495	-21.1	-30.4
CFG_7	1349	-26.2	-40.9

Figures 3 and 4 plot dynamic power versus performance, for configurations that have lower power consumption than baseline. CFG_5 has the best tradeoff: a 37 % reduction in dynamic power consumption with 1.3 dB and 0.6 dB performance penalty for the NMSE and ACPR respectively. Even with a tighter accuracy requirement of 1 dB compared to baseline, a 30 % reduction in power consumption is achieved with CFG_1.

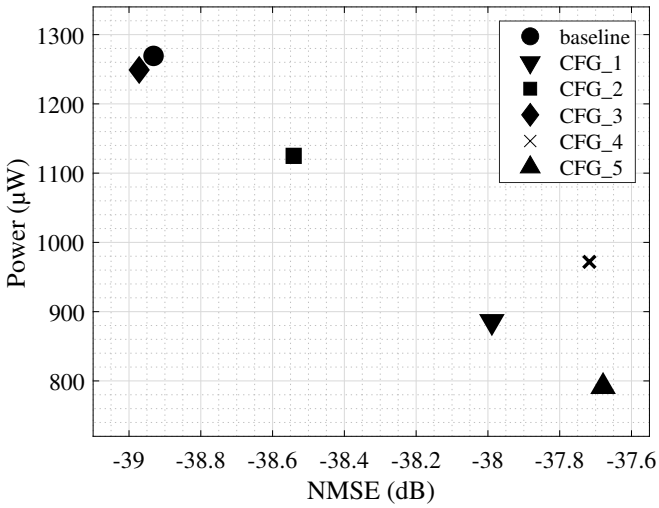

Fig. 3. Dynamic power *vs.* NMSE.

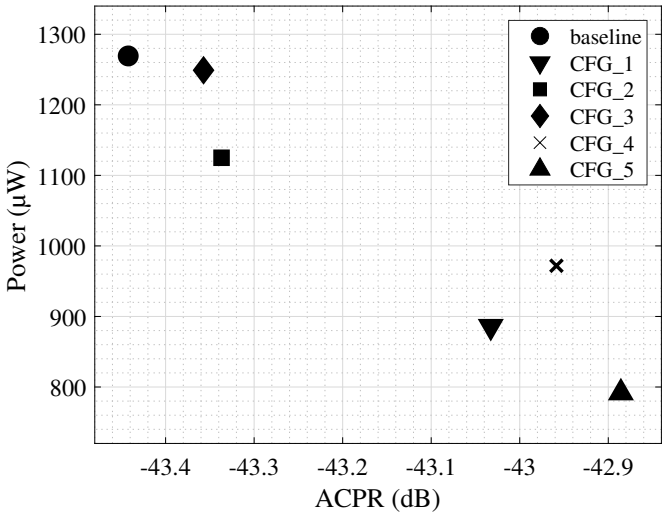

Fig. 4. Dynamic power *vs.* ACPR.

V. CONCLUSION

We have introduced a hardware implementation of a DPD predistorter based on the BAPS algorithm. A serial, 32-bit floating-point prototype is implemented to achieve high accuracy and low power, and to act as a reference for future designs. We have explored some power optimizations related to different sets of basis functions. BAPS type II operations consume most power, however, through special arrangement of type I and type II operations in a basis functions set, power consumption is reduced by a third with minimum loss in linearization performance. This demonstrates that algorithm-circuit codesign has a great potential to reduce power dissipation in digital predistorter designs.

ACKNOWLEDGEMENT

This project is financially supported by the Swedish Foundation for Strategic Research (SSF).

REFERENCES

[1] M. F. Haider, F. You, S. He, T. Rahkonen, and J. P. Aikio, "Predistortion-based linearization for 5G and beyond millimeter-wave transceiver systems: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2029–2072, 2022.

[2] C. Cheng, Z. Peng, W. Zhang, and G. Meng, "Volterra-series-based nonlinear system modeling and its engineering applications: A state-of-the-art review," *Mechanical Systems and Signal Processing*, vol. 87, pp. 340–364, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327016304393

[3] S. Boyd and L. Chua, "Fading memory and the problem of approximating nonlinear operators with Volterra series," *IEEE Transactions on Circuits and Systems*, vol. 32, no. 11, pp. 1150–1161, 1985.

[4] A. S. Tehrani, H. Cao, S. Afsardoost, T. Eriksson, M. Isaksson, and C. Fager, "A comparative analysis of the complexity/accuracy tradeoff in power amplifier behavioral models," *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 6, pp. 1510–1520, 2010.

[5] J. Wood, "System-level design considerations for digital pre-distortion of wireless base station transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 5, pp. 1880–1890, 2017.

[6] Y. Li, X. Wang, and A. Zhu, "Reducing power consumption of digital predistortion for RF power amplifiers using real-time model switching," *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 3, pp. 1500–1508, 2022.

[7] R. N. Braithwaite, "A comparison of indirect learning and closed loop estimators used in digital predistortion of power amplifiers," in *2015 IEEE MTT-S International Microwave Symposium*, 2015, pp. 1–4.

[8] C. Eun and E. Powers, "A new Volterra predistorter based on the indirect learning architecture," *IEEE Transactions on Signal Processing*, vol. 45, no. 1, pp. 223–227, 1997.

[9] P. N. Landin, S. Gustafsson, C. Fager, and T. Eriksson, "Weblab: A web-based setup for PA digital predistortion and characterization [application notes]," *IEEE Microwave Magazine*, vol. 16, no. 1, pp. 138–140, 2015.

[10] J. Kim and K. Konstantinou, "Digital predistortion of wideband signals based on power amplifier model with memory," *Electronics Letters*, vol. 37, pp. 1417–1418, Nov. 2001.

[11] D. Morgan, Z. Ma, J. Kim, M. Zierdt, and J. Pastalan, "A generalized memory polynomial model for digital predistortion of RF power amplifiers," *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3852–3860, 2006.

[12] W. Cao, S. Wang, P. N. Landin, C. Fager, and T. Eriksson, "Complexity optimized digital predistortion model of RF power amplifiers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 3, pp. 1490–1499, 2022.

[13] *Cadence® Genus®, v. 18.14*, Cadence Design Systems, Inc., 2019.

[14] V. Vashishtha, M. Vangala, and L. T. Clark, "ASAP7 predictive design kit development and cell design technology co-optimization," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 992–998.

[15] *Cadence® Xcelium®, v. 24.03*, Cadence Design Systems, Inc., 2024.