CHALMERS
UNIVERSITY OF TECHNOLOGY

Exploration of Short Floating-Point Numbers for Hardware-Friendly Digital Predistortion

Exploration of Short Floating-Point Numbers for Hardware-Friendly Digital Predistortion

Ziyi Yu, Mubanda Talemwa, and Per Larsson-Edefors
Dept. of Microtechnology and Nanoscience, Chalmers University of Technology, Gothenburg, Sweden

***Abstract*—Digital predistortion (DPD) is used to reduce nonlinear and memory effects in power amplifiers (PAs). Practical DPD algorithms use variants of Volterra series by opportunistically removing terms to reduce hardware complexity with minimal loss in linearization performance. Although current DPD algorithms can reduce the number of computations in terms of floating-point operations, precision requirements have not been analyzed in depth. With the motivation to reduce hardware complexity, many shorter floating-point formats have recently emerged in response to new application domains, such as machine learning. To study the trade-off between linearization performance and floating-point dynamic range and precision, we implement the basis-propagating selection (BAPS) model in digital hardware. We consider two use-case scenarios for PA models with different memory effects, and find that the requirements on exponent and mantissa resolution can be significantly relaxed from a default single-precision 32-bit format, without any substantial loss in linearization performance.**

I. Introduction

The rapid evolution of wireless communication systems, particularly 5G networks and emerging 6G technologies, demands increasingly higher data rates and spectral efficiency while maintaining symbol fidelity [1]. The problem is that these wideband signals, which are characterized by high peak-to-average power ratios (PAPRs) and complex envelope variations, lead to significant spectral regrowth and adjacent channel interference in the inherently nonlinear power amplifier (PA) [2]. Digital predistortion (DPD) has emerged as a critical solution for handling these nonlinearities to enable PAs to operate in regions of high efficiency [3].

Recent advances in DPD algorithms have focused on Volterra-based approaches, with the basis-propagating selection (BAPS) model demonstrating good performance in terms of modeling accuracy and computational efficiency [4]. Unlike traditional models, such as memory polynomial (MP) and generalized memory polynomial (GMP), BAPS constructs basis functions through iterative reuse of previously computed terms, significantly reducing the number of floating-point operations (FLOPs), while maintaining good linearization performance. However, with the exception of [5] which studies energy-efficient composition of basis functions, existing BAPS implementations target software platforms with standard floating-point precisions (binary32 or binary64), limiting their optimization potential for power-constrained hardware systems where custom-precision implementations could offer significant advantages.

While hardware implementations of DPD algorithms preferably employ fixed-point arithmetic [6], modern FPGA and ASIC platforms provide flexible floating-point IP cores that support custom precisions with short exponent and mantissa widths. However, the precision requirements for maintaining satisfactory DPD performance in floating-point hardware implementations remain largely unexplored, particularly for computationally intensive algorithms like BAPS. Here we investigate the impact of custom floating-point formats on DPD systems, addressing the critical design trade-off between computational precision and linearization performance.

II. BAPS-Based DPD Systems

Digital predistortion compensates for nonlinearities by applying an inverse PA model upstream of the PA itself. Traditional Volterra-based models suffer from high computational complexity, making hardware implementation challenging [7]. The BAPS model addresses this through a sequential computational approach that constructs basis functions iteratively. Starting with the input signal, $\phi_1 = x(n)$, the model sequentially constructs each subsequent basis function ϕ_r by applying either a so-called Type I or Type II operation.

Type I operations generate delayed versions of previously computed basis functions

$$\phi_r = q^{-m}\phi_i, \quad i < r, \tag{1}$$

where q^{-1} is the unit delay operator and m represents the memory delay step. Type II operations create nonlinear combinations according to

$$\phi_r = \phi_i\phi_j\phi_k^*, \quad i, j, k < r, \tag{2}$$

where $*$ denotes complex conjugation. The final DPD model output is computed as

$$y(n) = \sum_{r=1}^{R} \theta_r\phi_r(n), \tag{3}$$

where θ_r are the model coefficients and R is the total number of basis functions.

As demonstrated in [4], a greedy search algorithm can iteratively select optimal basis functions offline by evaluating all potential candidates from those already generated and choosing those yielding maximum error reduction. This predetermined computational structure makes the BAPS model well-suited for hardware architectures.

III. Floating-Point Arithmetic

The IEEE-754 standard [8] defines a floating-point representation comprising a sign bit (S), a w-bit exponent (E), and a t-bit mantissa (M), with the numerical value given by

$$\text{Value} = (-1)^{\text{S}} \cdot 2^{(\text{E}-\text{Bias})} \cdot (1 + \text{M}) \tag{4}$$

for normalized numbers, where Bias equals $2^{(w-1)} - 1$.

While IEEE-754 defines standard formats, application-specific requirements have driven development of custom-precision configurations. Table I summarizes industry-standard formats, reflecting different design priorities: machine learning applications favor reduced precision for higher throughput, while graphics and DSP applications require balanced precision-range trade-offs.

TABLE I
Examples of Custom Floating-Point Formats

Format	Sign	Exponent	Mantissa	Total
IEEE FP32	1	8	23	32
AMD FP24	1	7	16	24
Pixar PXR24	1	8	15	24
NVIDIA TF32 [9]	1	8	10	19
IEEE FP16	1	5	10	16
Google bfloat16 [10]	1	8	7	16
IBM DLFloat [11]	1	6	9	16

This demonstrates that precision optimization is a recognized strategy across multiple domains, with each format targeting specific computational requirements. For instance, DLFloat uses 6 exponent bits and 9 mantissa bits optimized for deep learning applications [11], while bfloat16 maintains the same exponent width as FP32 but reduces mantissa width for improved throughput [10]. This flexibility in exponent and mantissa allocation enables fine-grained control over dynamic range and numerical precision, making custom formats particularly attractive for specialized applications, of which DPD systems are an example.

Hardware floating-point arithmetic can be implemented using different methodologies, each suited for different optimization objectives and constraints. FPGA platforms offer rapid prototyping capabilities through vendor IP cores such as Xilinx LogiCORE [12] and Intel FPGA IP, supporting both standard IEEE formats and parameterizable precisions. Commercial EDA vendors for ASICs provide synthesis-ready floating-point IP libraries, such as Synopsys DesignWare and Cadence ChipWare [13], which provide parameterizable floating-point units supporting custom-precision configurations.

IV. Exploration Platform Implementation

Our exploration platform adopts a modular architecture consisting of three modules: the BAPS basis function builder, the DPD computation engine, and a coordinating wrapper module, as illustrated in Figure 1. We evaluate two BAPS configurations (BAPS8-mem1, BAPS8-mem5) with the same basis function count of eight but capturing different PA memory effect characteristics, where mem1 and mem5 refer to memory depth parameters used during the greedy search algorithm. These configurations result in different combinations of Type I and Type II operations, enabling assessment of precision requirements across different computational complexity profiles.

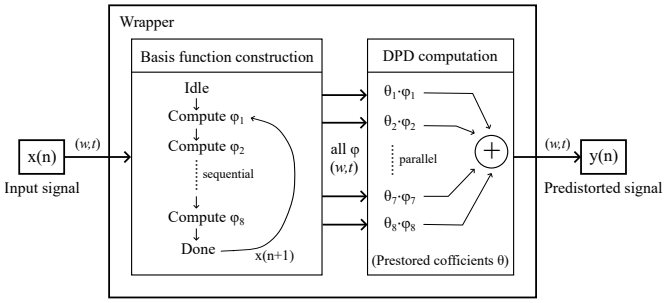

Fig. 1. Block diagram of the explored BAPS hardware implementation with a configuration for eight basis functions.

The basis function construction module implements sequential computation through a finite state machine that computes basis functions ϕ_1 through ϕ_r according to predetermined Type I or Type II operations. Type I delay operations are implemented using configurable shift register arrays, while Type II operations require custom floating-point complex multipliers to compute the nonlinear terms as defined in Eq. (2).

The computed basis function values are transferred to the DPD computation module, where parallel complex multipliers simultaneously compute $\theta_i \cdot \phi_i$ products using prestored coefficients θ. The weighted summation is then calculated using custom floating-point complex adders arranged in a balanced tree structure to minimize accumulation latency. All floating-point arithmetic operations utilize the same custom-precision configuration (w, t).

The custom floating-point arithmetic units utilize Cadence ChipWare IP components (`CW_fp_mult` and `CW_fp_addsub`) with parameterizable field widths, while maintaining IEEE-754 compliance where applicable. All units operate in round-to-nearest mode with configurable precision propagated through VHDL generics. Both minimum exponent and mantissa widths are constrained to 5 bits by the IP licensing limitations.

The wrapper module manages data flow between the basis function construction and DPD computation modules, coordinating the sequential basis function generation with the parallel DPD calculation. The architecture enables pipelined operation where the next input sample $x(n + 1)$ can begin basis function computation while the current sample completes DPD computation. The overall design, together with parameterizable precision support, makes it a useful platform for systematic evaluation of floating-point precision effects in DPD implementations. However, this architecture was not developed for high-throughput scenarios where more parallel solutions would be required.

V. Exploration Methodology

We systematically evaluate 40 floating-point configurations through the combination of ten mantissa field widths ($t =$

$5, 6, 7, 8, 9, 10, 11, 15, 19, 23$) and four exponent field widths ($w = 5, 6, 7, 8$). This extensive parameter space enables the characterization of precision-performance trade-offs across a wide range of hardware complexity levels, from short 11-bit (5,5) to standard 32-bit (8,23) configurations.

Custom Matlab functions are used to convert decimal values to custom floating-point binary strings and vice versa. These functions ensure correspondence between software and hardware domains, enabling precision analysis that reflects actual hardware arithmetic behavior.

A total of 79,280 complex baseband samples are generated using a multi-carrier signal with PAPR of 10.39 dB, representative of practical communication waveforms. The input signals are processed with single-precision Matlab implementations of the BAPS algorithm and DPD computation to establish reference results.

Each test vector passes through a complete Matlab–RTL–Matlab verification flow. The input vectors are converted to the target custom floating-point formats using custom conversion functions in Matlab and applied to the synthesized netlist through a behavioral VHDL testbench. Post-synthesis simulation outputs are then converted back to Matlab decimal numbers using inverse functions, enabling direct numerical comparison with the reference.

Both the software reference and hardware-generated predistorted signals are further evaluated using a remote PA (RF Weblab [14]), ensuring that the assessment reflects overall DPD performance rather than numerical accuracy alone. The linearized PA outputs are measured using two key DPD metrics: normalized mean square error (NMSE) and adjacent channel power ratio (ACPR). NMSE quantifies the deviation between the desired and actual PA outputs, indicating linearization accuracy. ACPR measures spectral regrowth suppression, reflecting compliance with regulatory emission requirements.

VI. Results

The accuracy of the implemented BAPS system is validated through direct comparison with the Matlab reference model using single precision (8,23). Figure 2 demonstrates the spectral overlap between hardware and software (Matlab) implementations for the predistorted signal. Figure 3 shows that the final linearized PA output maintains this high level of correspondence, confirming that our hardware implementation preserves the DPD algorithm's effectiveness without introducing significant numerical artifacts.

The ensuing precision analysis evaluates performance degradation using normalized metrics referenced to single precision. Positive values indicate performance degradation, while negative values indicate performance improvement:

$$\Delta NMSE = NMSE(w,t) - NMSE(8,23) \tag{5}$$

$$\Delta ACPR = ACPR(w,t) - ACPR(8,23) \tag{6}$$

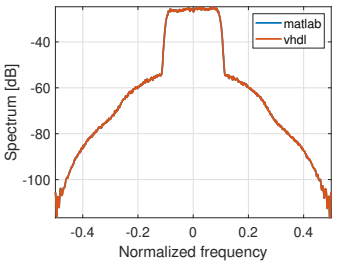

Fig. 2. Predistorted signal comparison between software reference and hardware implementation.

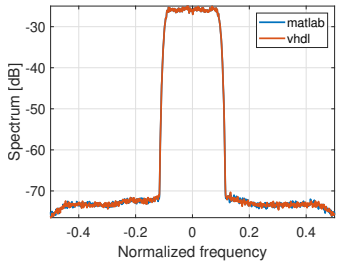

Fig. 3. Predistorted PA output comparison between software reference and hardware implementation.

A. Mantissa Width Analysis

BAPS8-mem1 Configuration: Table II presents NMSE and ACPR for several different precisions. The table reveals that there is a region where performance remains constant with precision: From 23 to 9 mantissa bits, performance degradation remains minor (less than 0.15 dB), indicating robust operation across this precision range. But then there is a transition region at 8–7 mantissa bits with modest degradation. Below 7 bits, degradation becomes critical, with noticeable performance loss at 6 bits and severe degradation at 5 bits ($\Delta NMSE$ = 4.17 dB, $\Delta ACPR$ = 3.34 dB).

TABLE II
NMSE and ACPR Results for BAPS8-mem1 Configuration

FP precision			NMSE		ACPR	
Exp	**Mant**	**Total**	dB	ΔdB	dB	ΔdB
8	23	32	-38.18	0.00	-43.69	0.00
8	19	28	-38.05	0.13	-43.68	0.01
8	15	24	-38.10	0.08	-43.68	0.01
8	11	20	-38.07	0.10	-43.60	0.09
8	10	19	-38.05	0.13	-43.60	0.08
8	9	18	-38.09	0.08	-43.63	0.05
8	8	17	-37.95	0.23	-43.52	0.17
8	7	16	-37.79	0.38	-43.41	0.28
8	6	15	-36.50	1.68	-42.35	1.33
8	5	14	-34.01	4.17	-40.35	3.34

Figure 4 visualizes the spectral performance degradation caused by mantissa width reduction from 8 to 5 bits, highlighting the emergence of spectral regrowth as precision decreases. Comparisons with the nonlinearized output and single-precision reference linearized output are also shown. The critical threshold at 6–7 mantissa bits reflects that in this region the accumulated errors in repeated complex multiplications begin to compromise the algorithm's effectiveness.

BAPS8-mem5 Configuration: Table III demonstrates that precision requirements remain consistent across BAPS configurations. Despite the increased memory depth, the critical degradation threshold remains at 6–7 mantissa bits. The configuration of mem5 shows slightly reduced sensitivity to reduced precision, likely due to its different composition of Type I and Type II operations, but the overall precision requirements remain unchanged.

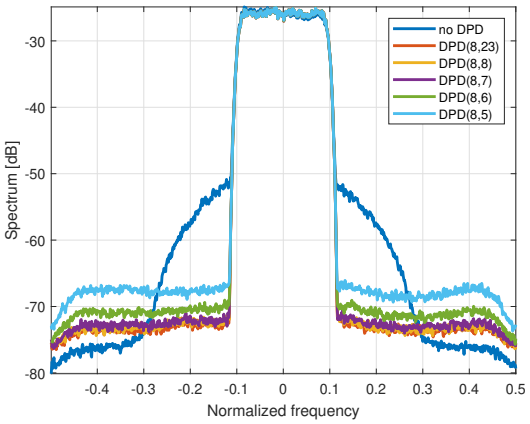

Fig. 4. Predistorted PA output: mantissa width sweep.

TABLE III
NMSE AND ACPR RESULTS FOR BAPS8-MEM5 CONFIGURATION

FP precision			NMSE		ACPR	
Exp	Mant	Total	dB	ΔdB	dB	ΔdB
8	23	32	-37.96	0.00	-43.39	0.00
8	19	28	-37.96	0.00	-43.31	0.08
8	15	24	-37.96	-0.01	-43.36	0.03
8	11	20	-37.95	0.00	-43.27	0.12
8	10	19	-37.92	0.03	-43.28	0.11
8	9	18	-37.97	-0.02	-43.26	0.13
8	8	17	-37.89	0.07	-43.20	0.19
8	7	16	-37.64	0.31	-43.00	0.39
8	6	15	-36.69	1.27	-42.30	1.09
8	5	14	-34.14	3.81	-40.14	3.25

B. Exponent Width Analysis

The reduction in exponent width exhibits fundamentally different behavior compared to mantissa width reductions. Figures 5 and 6 demonstrate that exponent width reduction has minimal impact on DPD performance for typical signal levels. Even aggressive reduction to 5 exponent bits yields negligible performance degradation below 0.1 dB, indicating that exponent width is not a limiting factor for typical DPD signal levels.

This behavior stems from the nature of floating-point arithmetic: the mantissa determines computational precision, while the exponent determines the representable dynamic range. Since DPD signals typically operate within a constrained amplitude range, reducing the number of exponent bits has negligible impact on algorithm effectiveness.

C. Hardware Complexity Analysis

Figure 7 presents the synthesized area results for a single Type II nonlinear computational module under a 5-ns timing constraint, using Cadence Genus with the ASAP7 7-nm FinFET process design kit [15]. The area scales approximately linearly with total bitwidth, with the mantissa bits dominating the complexity.

This scaling relationship enables predictable complexity-performance trade-offs: decreasing the number of mantissa bits from 23 to 7 leads to an 80 % area saving at a performance level that remains within 0.4 dB of single precision.

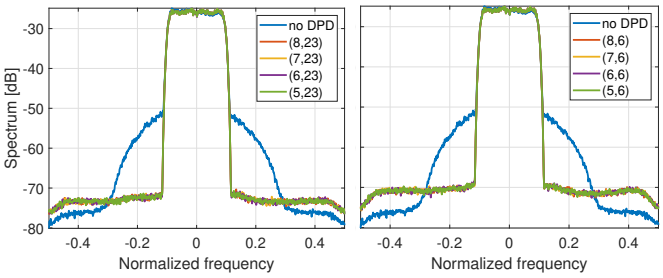

Fig. 5. PA output with varying exponent bits (23 mantissa bits).

Fig. 6. PA output with varying exponent bits (6 mantissa bits).

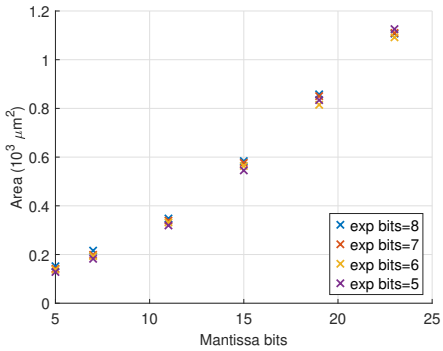

Fig. 7. Synthesized area of Type II module for different floating-point bitwidths.

VII. CONCLUSION

This work presents a systematic exploration of floating-point precision requirements for hardware-based BAPS digital predistortion systems. Using a parameterizable VHDL framework with configurable arithmetic IP components, we evaluated a wide range of custom floating-point formats and quantified their impact on linearization performance using NMSE and ACPR metrics on RF Weblab outputs. The consistent precision requirements observed across the tested BAPS configurations suggest that the findings may extend to other Volterra-based DPD implementations that involve similar complex multiplication and accumulation operations.

The results show that mantissa width is the critical factor for accuracy, with performance stable down to 7–8 bits but degrading rapidly below 6 bits. In contrast, exponent width has negligible effect, with as few as 5 bits proving to be sufficient across all cases. These findings point to useful precision-performance trade-offs where reductions in mantissa width can yield an 80 % area reduction, while maintaining performance within 0.4 dB of single precision. Additionally, as we can establish there are limited requirements on dynamic range, our evaluations show that fixed-point representations may be sufficient for implementation of DPD systems.

ACKNOWLEDGEMENT

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

REFERENCES

[1] T. S. Rappaport, Y. Xing, O. Kanhere, S. Ju, A. Madanayake, S. Mandal, A. Alkhateeb, and G. C. Trichopoulos, "Wireless communications and applications above 100 GHz: Opportunities and challenges for 6G and beyond," *IEEE Access*, vol. 7, pp. 78 729–78 757, 2019.

[2] A. Katz, J. Wood, and D. Chokola, "The evolution of PA linearization: From classic feedforward and feedback through analog and digital predistortion," *IEEE Microwave Magazine*, vol. 17, no. 2, pp. 32–40, 2016.

[3] F. M. Ghannouchi and O. Hammi, "Behavioral modeling and predistortion," *IEEE Microwave Magazine*, vol. 10, no. 7, pp. 52–64, 2009.

[4] 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.

[5] M. Talemwa, J. Muñoz Bautista, T. Eriksson, and P. Larsson-Edefors, "DSP hardware tradeoffs for digital predistortion based on pruned Volterra series," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2025, to be published.

[6] N. Dwivedi, V. A. Bohara, M. A. Hussein, and O. Venard, "Fixed point digital predistortion system based on indirect learning architecture," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014, pp. 1376–1380.

[7] 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.

[8] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[9] M. Fasi, N. J. Higham, M. Mikaitis, and S. Pranesh, "Numerical behavior of NVIDIA tensor cores," *PeerJ Computer Science*, vol. 7, p. e330, 2021. [Online]. Available: https://doi.org/10.7717/peerj-cs.330

[10] Intel Corporation, "BFLOAT16 – Hardware Numerics Definition White Paper," Tech. Rep., 2018. [Online]. Available: https://software.intel.com/sites/default/files/managed/40/8b/bf16-hardware-numerics-definition-white-paper.pdf

[11] A. Agrawal, S. M. Mueller, B. M. Fleischer, X. Sun, N. Wang, J. Choi, and K. Gopalakrishnan, "DLFloat: A 16-b floating point format designed for deep learning training and inference," in *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, 2019, pp. 92–95.

[12] Xilinx, *Floating-Point Operator v7.1 LogiCORE IP Product Guide*, Xilinx Inc., San Jose, CA, Nov. 2019.

[13] *Genus ChipWare IP Components Guide*, Cadence Design Systems, January 2024, product Version 23.1.

[14] 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.

[15] 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.