CHALMERS
UNIVERSITY OF TECHNOLOGY

Custom Precision Floating-Point Implementation of BAPS Algorithm for Hardware-Efficient Digital Predistortion

Master's thesis in Embedded Electronic System Design

ZIYI YU

Department of Microtechnology and Nanoscience
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Master's thesis 2025

Custom Precision Floating-Point Implementation of BAPS Algorithm for Hardware-Efficient Digital Predistortion

ZIYI YU

Department of Microtechnology and Nanoscience
Chalmers University of Technology
Gothenburg, Sweden 2025

Custom Precision Floating-Point Implementation of BAPS Algorithm for Hardware-Efficient Digital Predistortion
ZIYI YU

© ZIYI YU, 2025.

Supervisor: Per Larsson-Edefors, Department of Microtechnology and Nanoscience
Examiner: Lena Peterson, Department of Microtechnology and Nanoscience

Master's Thesis 2025
Department of Microtechnology and Nanoscience
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Custom Precision Floating-Point Implementation of BAPS Algorithm for Hardware-Efficient Digital Predistortion
ZIYI YU
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

Modern wireless communication systems demand improved spectral efficiency and power conservation, placing stringent requirements on radio frequency power amplifiers. Digital predistortion (DPD) techniques linearize power amplifiers while enabling efficient operation, with the basis-propagating selection (BAPS) algorithm offering computational efficiency through systematic reuse of basis functions. However, hardware implementation of BAPS remains underexplored, particularly regarding floating-point precision requirements that impact both linearization performance and hardware resource utilization.

This thesis systematically investigates floating-point precision requirements for BAPS-based DPD digital circuit implementations through analysis of 40 custom precision configurations spanning mantissa widths from 5 to 23 bits and exponent widths from 5 to 8 bits. A precision-configurable hardware architecture using parameterizable VHDL with Cadence ChipWare floating-point IP enables rigorous evaluation across four distinct BAPS configurations.

Results demonstrate that mantissa width critically determines performance, with 9-bit mantissa maintaining robust operation and 7 bits representing a critical threshold for degradation, while exponent width can be aggressively reduced from 8 to 5 bits without performance penalty. Hardware complexity analysis reveals approximately linear scaling with mantissa width, with a balanced configuration achieving 80% area reduction while maintaining linearization performance within 0.4 dB of single precision. These findings enable substantial hardware efficiency improvements for DPD implementations in resource-constrained 5G and emerging 6G wireless systems while maintaining required linearization performance.

Keywords: Digital Predistortion, BAPS Algorithm, Floating-Point, Digital Hardware Implementation, Wireless Communications

Acknowledgements

This project has taken eight months to complete, with several important research steps along the way. I would like to express my sincere gratitude to my supervisor, Per, for not only providing valuable knowledge related to this thesis but also for his guidance in research. I am also deeply thankful to Talemwa, who generously shared his research work with me and patiently answered my questions. My appreciation also goes to my examiner, Lena, for clarifying the overall procedure at the beginning and offering support throughout the process.

This has been a truly meaningful experience for me, both in shaping my research path and in deepening my understanding of myself: what concerns me, what sparks my curiosity, and where I can find happiness and fulfillment. I am also grateful to myself for persevering and staying committed to doing the right things.

Finally, I would like to thank my family and friends. As always, your love and support have been indispensable, and I could not have completed this work without you. This experience has given me the courage to move forward and embrace future challenges with optimism.

Ziyi Yu, Gothenburg, October 2025

Contents

List of Abbreviations

ACPR	Adjacent Channel Power Ratio
ADC	Analog-to-Digital Converter
AM/AM	Amplitude-to-Amplitude
AM/PM	Amplitude-to-Phase
ASIC	Application-Specific Integrated Circuit
BAPS	Basis-Propagating Selection
DAC	Digital-to-Analog Converter
DDR	Dynamic Deviation Reduction
DPD	Digital Predistortion
DSP	Digital Signal Processing
EVM	Error Vector Magnitude
FPGA	Field-Programmable Gate Array
GMP	Generalized Memory Polynomial
HDL	Hardware Description Language
HLS	High-Level Synthesis
LASSO	Least Absolute Shrinkage and Selection Operator
MP	Memory Polynomial
NMSE	Normalized Mean Square Error
NR	New Radio
OFDM	Orthogonal Frequency-Division Multiplexing
OMP	Orthogonal Matching Pursuit
PA	Power Amplifier
PAPR	Peak-to-Average Power Ratio
PCA	Principal Component Analysis
RF	Radio Frequency
RTL	Register-Transfer Level

1

Introduction

Modern communication systems like 5G New Radio (NR) and emerging 6G technologies demand increasingly higher data rates, spectral efficiency, and energy conservation while maintaining signal integrity [1]. This evolution has placed increased demands on radio frequency (RF) power amplifiers (PAs), which represent critical components in the signal chain of any wireless communication system.

Power amplifiers face an inherent trade-off between linearity and efficiency. When operated near saturation to maximize power efficiency, PAs introduce significant nonlinear distortion that degrades signal quality and creates spectral regrowth, leading to adjacent channel interference [2]. This challenge has become more acute with the complex modulation schemes and carrier aggregation techniques in advanced wireless standards, where peak-to-average power ratios (PAPRs) routinely become higher [3], forcing PAs to operate with significant backoff from their optimal efficiency point.

Digital predistortion (DPD) techniques have emerged as a critical solution for addressing these nonlinearities while enabling PAs to operate in their high-efficiency regions [4]. By applying a complementary nonlinear function to the input signal, DPD pre-compensates for PA distortion, allowing for improved spectral efficiency and enabling PAs to operate at higher efficiency points. The effectiveness of DPD implementations directly impacts the overall performance and efficiency of modern wireless transmitters.

Among the numerous DPD algorithms, the basis-propagating selection (BAPS) algorithm offers a promising balance between computational complexity and linearization accuracy [5]. Unlike conventional memory polynomial models, BAPS employs an adaptive basis function selection approach that optimizes model complexity while maintaining linearization performance. Despite the BAPS algorithm's hardware-friendly characteristics [5], its implementation in digital hardware remains underexplored, particularly regarding the optimization of floating-point precision and computational efficiency.

Modern digital signal processing implementations face critical decisions regarding numerical representation that directly impact both performance and resource utilization. While software implementations typically employ standard single or double precision, hardware implementations offer opportunities for precision optimization. By systematically analyzing precision requirements, hardware designers can achieve significant resource savings without compromising algorithmic effectiveness.

This thesis project addresses these challenges by developing a precision-configurable hardware implementation of the BAPS algorithm, focusing on systematic floating-point precision analysis. The implementation explores precision-performance trade-offs using parameterizable hardware description language (HDL) and specialized floating-point arithmetic units, providing quantitative insights into the relationship between numerical precision and linearization performance. The insights gained from this thesis can provide specific design trade-offs between precision and resource utilization that will benefit both academic research and industrial applications in current 5G and future 6G wireless networks.

1.1 Related Work

The intersection of DPD algorithm development and hardware implementation has evolved significantly, though most efforts have addressed software algorithms and hardware platforms separately rather than pursuing integrated optimization approaches.

DPD Algorithm Development

Memory polynomial (MP) models, based on the Volterra series nonlinear model, are widely used in DPD implementations due to their modeling capability and computational efficiency [6]. While efficient for moderate complexity scenarios, MP models often require higher orders to accurately capture complex nonlinearities, leading to polynomial growth in computational complexity. Enhanced variations, such as generalized memory polynomials [6], improve performance by incorporating cross-term effects between signals at different time instants, though at the cost of significantly increased computation.

The computational complexity challenge has motivated development of model reduction techniques. Pruning approaches using methods such as orthogonal matching pursuit (OMP) and least absolute shrinkage and selection operator (LASSO) reduce model size by eliminating less significant terms [7]. However, these techniques typically require large initial model structures and may sacrifice modeling generality.

More recently, neural network-based DPD models have emerged, leveraging data-driven learning to handle complex distortion effects [8]. Although these implementations demonstrate excellent linearization performance under specific conditions, their high training complexity, non-deterministic behavior, and substantial computational requirements present significant challenges for efficient hardware realization [9].

The BAPS algorithm addresses computational complexity limitations through a fundamentally different approach [5]. Rather than starting with large model structures and pruning terms, BAPS constructs basis functions iteratively through systematic reuse of previously computed elements. This approach creates a directed acyclic graph structure that naturally minimizes computational redundancy while maintaining the modeling capability of full Volterra series. The predetermined computational structure resulting from offline optimization makes BAPS promising for hardware implementation.

Hardware-Based DPD Implementations

The implementation of DPD algorithms in digital hardware has evolved in response to increasing signal-bandwidth and complexity requirements. Early implementations primarily employed fixed-point arithmetic with careful scaling to manage dynamic range limitations [10]. These approaches were effective for narrowband signals but struggled with the wide-dynamic-range requirements of modern wideband applications and the precision demands of complex DPD algorithms.

DPD techniques in general have seen various realizations in field programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) [11]. However, hardware implementations specifically targeting the BAPS algorithm remain limited. For conventional DPD models, optimization techniques have been applied to reduce computational complexity, such as lookup table-assisted gain indexing, time-division multiplexing for multiplier sharing, and principal component analysis (PCA) for model order reduction [12, 13].

The trend toward wider bandwidth signals and more complex PA models has increasingly favored floating-point implementations despite their higher resource requirements [14]. Hybrid approaches combining fixed-point arithmetic for less sensitive operations with floating-point for critical calculations have demonstrated effective resource utilization while maintaining numerical accuracy [15].

Floating-Point Computation in Digital Hardware

The evolution of floating-point arithmetic in digital hardware has been driven by the competing demands of numerical accuracy and resource efficiency [16]. While IEEE 754 standard formats provide well-characterized behavior and broad compatibility, specialized applications often benefit from custom precision configurations that optimize bit allocation for specific algorithmic requirements [17].

Modern hardware development tools support parameterizable floating-point implementations that enable systematic exploration of precision requirements. Vendor-provided IP cores such as Xilinx LogiCORE offer configurable precision with optimization settings for latency, throughput, or resource utilization [18]. Additionally, open-source libraries like FloPoCo provide hardware-optimized implementations with customizable precision characteristics [19].

High-level synthesis (HLS) tools have emerged as an important methodology for translating algorithmic descriptions into efficient hardware implementations. These tools enable automatic conversion of floating-point algorithms while providing control over precision configuration and optimization objectives [20]. However, most HLS applications focus on general-purpose computing rather than the specific requirements of advanced DPD algorithms.

Research on precision optimization for digital signal processing (DSP) applications has established important foundations for application-specific floating-point design [21]. Studies have demonstrated that many signal processing algorithms can tolerate significant precision reduction without performance degradation, particularly when precision requirements are analyzed systematically across different computational stages.

This thesis builds upon these foundations by applying precision optimization specifically to the BAPS algorithm, leveraging its unique computational structure to achieve optimal precision-performance trade-offs for DPD applications.

1.2 Purpose and Goal

The purpose of this thesis is to advance the field of DPD by creating an efficient hardware implementation that bridges the gap between theoretical algorithm development and practical deployment in next-generation wireless systems. By demonstrating the viability of implementing sophisticated DPD algorithms with optimized floating-point precision, this work contributes to the broader goal of enabling more energy-efficient and computationally practical wireless communications.

The specific goal is to implement a precision-configurable BAPS algorithm in digital hardware, delivering a floating-point implementation that achieves linearization performance comparable to software-based simulations while optimizing resource utilization for practical DPD applications. This work quantifies key performance metrics, including adjacent channel power ratio (ACPR) and normalized mean square error (NMSE), across a systematic range of floating-point precision configurations.

Based on these goals, this thesis addresses the following research questions:

- How does the performance of hardware-based BAPS implementation compare with traditional software approaches?
- How do different BAPS configurations with varying computational complexity respond to precision reduction?
- What is the optimal floating-point precision for implementing BAPS in digital hardware that balances linearization performance with resource utilization?

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 presents the theoretical background of power amplifier nonlinearities, digital predistortion, and the BAPS algorithm, along with fundamentals of floating-point representation and hardware arithmetic relevant to this work.

Chapter 3 details the methodology employed in this project, including the workflow, tool selection rationale, and verification methodology used in this project.

Chapter 4 describes the hardware implementation architecture, including design decisions, modular decomposition, and the approach to precision-configurable floating-point arithmetic throughout the system.

Chapter 5 presents the results of the precision exploration and discusses the implications of the findings, analyzing the trade-offs between resource utilization, numerical precision, and linearization performance metrics across different BAPS configurations.

Chapter 6 concludes the thesis with a summary of contributions and suggestions for future research directions.

2

Technical Background

This chapter presents the theoretical foundations and key concepts necessary to understand the design and implementation of this project. The discussion begins with an overview of power amplifier characteristics and nonlinearities, followed by digital predistortion techniques with emphasis on the BAPS algorithm. Subsequently, floating-point representation and hardware architecture considerations relevant to DPD implementation are examined.

2.1 Power Amplifier

Power amplifiers are critical components in wireless communication systems that amplify signal power to levels sufficient for transmission over long distances. In modern wireless communication systems, PAs typically consume the majority of power in the transmitter chain, making their performance characteristics essential to overall system efficiency and performance [2].

2.1.1 Linearity and Efficiency Trade-off

An ideal PA would perfectly amplify input signals without distortion while converting DC power to RF power with 100% efficiency. In practice, however, there exists a fundamental trade-off between linearity and efficiency [2]. Linearity refers to the PA's ability to produce an output signal that is a scaled replica of the input signal, while efficiency relates to how effectively the PA converts DC power to RF output power.

Power amplifiers are categorized by their biasing conditions, which directly influence the linearity-efficiency trade-off [22]. For example, class-A amplifiers offer excellent linearity but poor efficiency (typically 25-30%), while class-B amplifiers improve efficiency (up to 78.5% theoretical maximum) at the cost of increased distortion.

This trade-off becomes particularly challenging in modern communication systems employing complex modulation schemes with high PAPRs, such as orthogonal frequency division multiplexing (OFDM) signals used in 5G and emerging 6G standards [23]. These wideband signals force PAs to operate with significant backoff from peak efficiency to maintain linearity, resulting in poor power efficiency. This fundamental conflict between linearity and efficiency motivates the need for linearization techniques that can compensate for PA nonlinearities while enabling operation closer to high-efficiency regions.

2.1.2 PA Nonlinear Behavior and Memory Effects

When signals pass through a nonlinear PA, several undesirable effects occur. Nonlinear amplification creates new frequency components outside the original signal bandwidth, causing spectral regrowth that interferes with adjacent channels. Additionally, nonlinearities distort the signal within its intended bandwidth, leading to increased error vector magnitude (EVM) and reduced system performance [24].

The nonlinear behavior of PAs manifests primarily as amplitude-to-amplitude and amplitude-to-phase (AM/AM and AM/PM) distortion [25]. AM/AM distortion describes the nonlinear relationship between input and output signal amplitudes, while AM/PM distortion represents the phase shift of the output signal as a function of input amplitude.

Beyond static nonlinearities, PAs also exhibit memory effects, where the output depends not only on the current input but also on past inputs [4]. These effects arise from various physical phenomena including thermal effects, semiconductor trapping effects, and unintentional electrical memory effects [2]. Thermal memory effects arise from temperature variations in the semiconductor junction due to envelope power fluctuations, with time constants ranging from microseconds to milliseconds. Electrical memory effects result from bias network components, charge trapping in semiconductor devices, and parasitic reactive elements, typically exhibiting faster dynamics in the nanosecond to microsecond range. These effects become increasingly significant as signal bandwidths exceed the reciprocal of the dominant memory time constants.

In 5G and emerging 6G systems, where signal bandwidths often exceed hundreds of megahertz, memory effects significantly complicate the linearization challenge and increase computational requirements for accurate modeling [23].

PA Behavioral Modeling Accurately characterizing PA behavior is essential for effective linearization. Behavioral models capture the input-output relationship of the PA without requiring detailed knowledge of the internal circuit components. These models serve as the foundation for DPD techniques [4]. The complexity of PA behavioral modeling directly impacts DPD algorithm design. While static nonlinearities can be compensated using memoryless techniques, memory effects require sophisticated behavioral models that capture both the nonlinear transformation and the dynamic response characteristics.

The Volterra series provides the most comprehensive mathematical framework for modeling complex nonlinear systems with memory [26]. A discrete-time complex baseband representation of the Volterra series is given by:

$$y(n) = \sum_{p=1}^{P} \sum_{\mathbf{m} \in \mathcal{M}_p} h_{2p-1}(\mathbf{m}) \prod_{i=1}^{p} x(n - m_i) \prod_{j=p+1}^{2p-1} x^*(n - m_j) \tag{2.1}$$

where $h_{2p-1}(\mathbf{m})$ denotes the kernel coefficients of order $2p-1$, and $\mathcal{M}_p$ is the index set defining valid memory taps. The products include both input signal terms and

their complex conjugates, capturing the complex nonlinear interactions essential for accurate PA modeling.

Due to the complexity of full Volterra series, simplified models are commonly used, such as memory polynomial models [6], generalized memory polynomial models [6], and dynamic deviation reduction models [27]. These behavioral models form the foundation for digital predistortion techniques, which can compensate for PA nonlinearities by applying an inverse nonlinear function before the amplifier.

2.2 Digital Predistortion

Digital predistortion is a signal processing technique that pre-compensates for PA nonlinearities by applying an inverse nonlinear function to the input signal before amplification. This approach enables PAs to operate closer to saturation while maintaining acceptable linearity, thereby improving the overall efficiency-linearity trade-off in wireless transmitters.

2.2.1 Principles of DPD

The fundamental concept of digital predistortion is to apply a nonlinear transformation to the input signal that is complementary to the PA's nonlinearity [4]. When the predistorted signal passes through the PA, the cascaded response approximates a linear system. Mathematically, if the PA's transfer function is denoted as $G(\cdot)$, the DPD function $F(\cdot)$ is designed such that:

$$G(F(x)) \approx Kx \tag{2.2}$$

where K is a constant gain factor, and x is the desired input signal. The effectiveness of this approach depends critically on the accuracy of the inverse function $F(\cdot)$ and its stability under varying operating conditions. In practice, the DPD function is typically implemented as a behavioral model trained using captured PA input-output data.

DPD effectiveness is fundamentally limited by several factors: modeling accuracy, adaptation speed, signal bandwidth constraints, and the stability of PA characteristics over temperature and time. Additionally, DPD cannot compensate for PA limitations such as insufficient output power or hard clipping at saturation [28].

2.2.2 DPD System Architecture

A typical DPD system comprises several key components working in coordination to achieve linearization, as illustrated in Figure 2.1. The baseband input signal x_{in} first passes through the digital predistorter, which applies the inverse nonlinear transformation based on the PA's characteristics to produce the predistorted signal x_{out_DPD}. The predistorted signal is then converted to analog domain via digital-to-analog converters (DACs), upconverted to RF frequency, and amplified by the PA.

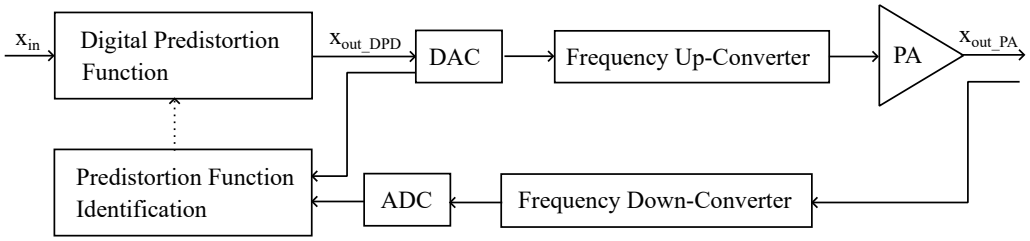

Figure 2.1: Simplified functional diagram of digital predistortion system architecture [29].

A crucial feedback path captures the PA output x_{out_PA}, downconverts it to baseband, and digitizes it through analog-to-digital converters (ADCs) for model parameter estimation and adaptation. Various learning architectures have been developed for DPD coefficient identification, with indirect learning approaches being most commonly implemented due to their computational simplicity and reliable convergence characteristics [30].

2.2.3 Volterra-Based DPD Models

The effectiveness of digital predistortion critically depends on the accuracy and computational efficiency of the underlying behavioral models. Several modeling techniques have been developed to capture PA nonlinear behavior with memory effects:

Memory Polynomial (MP) Model: A computationally efficient subset of the Volterra series that captures both nonlinearity and memory effects:

$$y(n) = \sum_{k=1}^{K} \sum_{m=0}^{M} a_{km} x(n-m)|x(n-m)|^{k-1} \tag{2.3}$$

where K represents the nonlinearity order, M is the memory depth, and a_{km} are the model coefficients. This model offers good linearization performance with moderate computational complexity [6].

Generalized Memory Polynomial (GMP) Model: Extends the MP model by incorporating cross-terms between signals at different time instants to better capture complex memory effects:

$$\begin{aligned} y(n) = &\sum_{k=1}^{K_a} \sum_{m=0}^{M_a} a_{km} x(n-m)|x(n-m)|^{k-1} \\ &+ \sum_{k=1}^{K_b} \sum_{m=0}^{M_b} \sum_{l=1}^{L_b} b_{kml} x(n-m)|x(n-m-l)|^{k-1} \\ &+ \sum_{k=1}^{K_c} \sum_{m=0}^{M_c} \sum_{l=1}^{L_c} c_{kml} x(n-m)|x(n-m+l)|^{k-1} \end{aligned} \tag{2.4}$$

The GMP model provides improved modeling accuracy for wideband signals through lagging and leading cross-terms, but at the cost of significantly increased computational complexity [6].

Advanced Modeling Approaches: More recent developments include dynamic deviation reduction (DDR) models that organize Volterra kernels based on dynamic order [27], and machine learning-based approaches using neural networks [8]. However, these traditional Volterra-based models often present significant computational complexity challenges for real-time hardware implementation, particularly for wideband signals with strict PAPR requirements.

2.2.4 Performance Metrics

The effectiveness of DPD systems is quantified using several metrics that capture different aspects of linearization performance:

ACPR quantifies spectral regrowth suppression by measuring the power ratio between adjacent channels and the main signal channel:

$$\text{ACPR} = 10\log_{10}\left(\frac{P_{\text{adjacent}}}{P_{\text{main}}}\right) \text{ [dBc]} \tag{2.5}$$

NMSE measures the time-domain accuracy between the desired linear response and actual PA output:

$$\text{NMSE} = 10\log_{10}\left(\frac{\sum_n |y_{\text{desired}}(n) - y_{\text{actual}}(n)|^2}{\sum_n |y_{\text{desired}}(n)|^2}\right) \text{ [dB]} \tag{2.6}$$

EVM quantifies constellation accuracy by measuring the RMS deviation between ideal and actual symbol positions:

$$\text{EVM} = \sqrt{\frac{\sum_n |s_{\text{ideal}}(n) - s_{\text{actual}}(n)|^2}{\sum_n |s_{\text{ideal}}(n)|^2}} \times 100 \tag{2.7}$$

These metrics provide comprehensive assessment across frequency domain (ACPR), time domain (NMSE), and modulation domain (EVM) performance. In this project, only ACPR and NMSE are utilized for quantitative evaluation.

2.3 Basis-Propagating Selection Algorithm

Traditional Volterra-based DPD models, while effective for linearization, present significant computational challenges that limit their practical implementation in real-time systems. The computational complexity grows exponentially with nonlinear order and memory depth, making hardware implementation limited by resource and power constraints. The BAPS algorithm addresses these limitations through a fundamentally different approach to basis function generation [5].

2.3.1 Basis-Propagating Selection Principle

The fundamental innovation of BAPS lies in recognizing that many Volterra terms share common computational elements. Rather than computing each basis function

independently, BAPS systematically identifies and reuses these shared computations. This dynamic adaptation reduces computational requirements by selecting only the most significant basis functions for each specific operating condition [5].

The algorithm begins with the input signal as the first basis function:

$$\phi_1(n) = x(n) \tag{2.8}$$

Subsequent basis functions are constructed using either Type I or Type II operations:

Type I Operation (Delay): This operation applies a delay to an existing basis function:

$$\phi_r = q^{-m}\phi_i \tag{2.9}$$

where q^{-m} represents a delay of m samples applied to a previously computed basis function ϕ_i.

Type II Operation (Nonlinear): This operation creates a new basis function through multiplication:

$$\phi_r = \phi_i \phi_j \phi_k^* \tag{2.10}$$

Here $i, j, k < r$ and $*$ denotes complex conjugation.

The final BAPS model output combines all basis functions linearly:

$$y(n) = \sum_{r=1}^{R} \theta_r \phi_r(n) \tag{2.11}$$

where θ_r are model coefficients determined through least-squares estimation, and R represents the total number of basis functions.

The specific sequence of operations is determined through an offline greedy search algorithm that iteratively selects basis functions providing maximum error reduction [31]. This optimization process needs to be performed only once for a given PA type, after which the predetermined operation sequence can be implemented directly in hardware.

2.3.2 Computational Advantages for Hardware Implementation

BAPS offers several critical advantages that make it particularly suitable for precision-configurable hardware implementations. The sequential construction approach eliminates redundant calculations present in traditional Volterra models. While a full Volterra series of order P and memory depth M requires $O(M^P)$ operations, BAPS achieves comparable performance with $O(R)$ complexity, where R is the number of basis functions typically much smaller than M^P [5].

Type I operations require only delay elements implemented as shift registers or memories, consuming minimal hardware resources. Type II operations reuse previously computed values, eliminating the need for storing large lookup tables or precomputed basis functions. The number of basis functions can be adjusted to

balance linearization performance against hardware complexity, providing flexibility for different application requirements and resource constraints.

These characteristics make BAPS particularly well-suited for hardware implementations where computational efficiency directly impacts power consumption and chip area.

2.4 Floating-Point Arithmetic in Digital Signal Processing

Digital predistortion algorithms require careful consideration of numerical representation to maintain both computational accuracy and hardware efficiency. While traditional DSP implementations have favored fixed-point arithmetic due to hardware constraints, the increasing complexity of DPD algorithms and the stringent precision requirements of modern communication systems have driven interest in floating-point implementations [32].

2.4.1 IEEE-754 Standard Overview

The IEEE-754 standard [17] defines a floating-point representation comprising a sign bit (S), a w-bit exponent (E), and a t-bit mantissa (M), with the normalized numerical value given by

$$\text{Value} = (-1)^{\text{S}} \cdot (1 + \text{M}) \cdot 2^{(\text{E}-\text{Bias})} \tag{2.12}$$

where Bias equals $2^{(w-1)} - 1$, and the mantissa M represents the fractional part with an implicit leading 1. Single precision (binary32) allocates 32 bits as 1 sign + 8 exponent + 23 mantissa bits, providing approximately 7 decimal digits of precision with a dynamic range from approximately 10^{-38} to 10^{38}. Double precision (binary64) extends this to 64 bits with enhanced precision and range.

The increasing demand for hardware-efficient DSP has motivated custom precision formats that optimize the allocation of bits between exponent and mantissa fields. Half-precision (binary16) reduces storage requirements by 50% while maintaining sufficient accuracy for many applications. More specialized formats like BFloat16 preserve the dynamic range of single precision through an 8-bit exponent while reducing mantissa precision to 7 bits, enabling memory savings with minimal impact on algorithm performance [33].

2.4.2 Floating-Point vs. Fixed-Point Arithmetic

Power amplifier signals exhibit wide dynamic variations with PAPR often exceeding 10 dB, requiring sufficient exponent bits to represent both small-signal and large-signal conditions without overflow or underflow. The iterative nature of many DPD algorithms, including BAPS, can accumulate numerical errors through successive computations, necessitating adequate mantissa precision to maintain modeling accuracy.

Fixed-point arithmetic represents numbers using a predetermined allocation of integer and fractional bits, providing deterministic behavior and simpler hardware implementation [34]. However, fixed-point systems require careful scaling management to prevent overflow and maintain precision across the signal's dynamic range [11]. The scaling factors must be predetermined based on worst-case signal conditions, often resulting in suboptimal precision utilization for typical operating conditions.

The trade-off between floating-point and fixed-point arithmetic is particularly relevant for hardware implementations [35]. Floating-point provides wide dynamic range, maintaining relative precision across a wide range of values and eliminating overflow concerns, which simplifies algorithm development. Fixed-point, in contrast, offers lower hardware complexity, faster computation, and reduced power consumption, but at the cost of a limited dynamic range, fixed absolute precision, and the need for careful scaling to avoid overflow or underflow [35]. For algorithms like BAPS, which involve sequential basis function construction, floating-point arithmetic can simplify implementation and maintain numerical stability across the wide dynamic range characteristic of PA operations.

2.4.3 Custom Precision Trends in Hardware DSP

Modern hardware design tools increasingly support custom floating-point formats that enable application-specific optimization of the precision-complexity trade-off. Rather than using standard IEEE formats, designers can allocate bits between exponent and mantissa fields based on algorithm-specific requirements [36]. This capability becomes particularly valuable for DPD applications where different computational stages may have varying precision sensitivities.

The emergence of parameterizable floating-point IP cores enables systematic exploration of precision requirements without fundamental architectural changes [18, 37]. This flexibility supports precision-aware design methodologies where arithmetic accuracy is balanced against hardware resource utilization to achieve optimal implementation efficiency for specific performance targets.

Empirical evaluation of precision requirements is essential for determining whether single-precision floating-point is necessary for a specific DPD implementation or if custom precision is sufficient for BAPS operations [19]. This evaluation involves analyzing the numerical stability of the algorithm under different bit-width configurations and measuring the impact on linearization performance metrics, as will be demonstrated in Chapter 5.

2.5 Hardware Implementation Considerations

The implementation of digital predistortion algorithms in hardware has evolved significantly with advances in both algorithmic sophistication and hardware capability. Contemporary approaches encompass both dedicated ASIC implementations and configurable platform solutions, each offering distinct advantages for different application requirements.

ASIC implementations provide the highest level of optimization for DPD algorithms through custom arithmetic units and specialized architectures [38]. Recent developments include dedicated hardware blocks for coefficient adaptation, basis function generation, and predistortion computation, often integrated into pipeline architectures. Modern RF DACs with integrated DPD support have achieved ACPR performance exceeding -70 dBc in production systems [38], achieving superior power efficiency and processing throughput compared to general-purpose solutions.

FPGA-based implementations offer reconfigurability advantages that facilitate algorithm development across diverse PA types and operating conditions. Modern FPGA development environments provide comprehensive support for DPD implementations. Xilinx Floating-Point Operator IP offers parameterizable implementations with configurable precision and optimization settings, allowing designers to prioritize latency, throughput, or resource utilization based on application needs [18]. High-level synthesis tools like Xilinx Vitis HLS enable automatic conversion of floating-point algorithms to efficient hardware implementations, facilitating rapid prototyping and design space exploration [20]. Additionally, open-source libraries such as FloPoCo provide FPGA-optimized implementations with customizable precision and latency characteristics that complement vendor-provided IP [19].

Both ASIC and FPGA implementations of DPD algorithms must address the computational complexity of floating-point operations, which typically require deep pipelines to achieve high clock frequencies. This necessitates careful pipeline balancing and synchronization to maintain data coherency while sustaining the high throughput required for real-time operation [39].

3

Methods

This project employed a hardware-software co-design methodology to implement and evaluate the BAPS algorithm for digital predistortion with variable floating-point precision. The approach systematically explored precision-performance trade-offs across multiple BAPS configurations through MATLAB algorithm validation, parameterized register-transfer level (RTL) implementation, post-synthesis simulation, and comprehensive verification.

3.1 Experimental Workflow

The implementation followed a structured three-phase approach. The first phase established MATLAB implementations of all BAPS configurations as software reference models, generating comprehensive input-output test vectors using OFDM signals with controlled statistical properties. The second phase involved decomposing the BAPS algorithm into distinct parameterized RTL modules with configurable precision propagated through VHDL generics. Finally, a complete MATLAB–RTL–MATLAB verification flow was developed where identical input vectors were processed through both software reference and synthesized hardware implementations for numerical comparison.

3.2 Tool Selection

The project toolchain integrated Cadence Genus Synthesis Solution for RTL synthesis for both unconstrained and timing-constrained approach, targeting the ASAP7 7-nm FinFET technology [40]. Cadence Xcelium Parallel Simulator provided functional verification and post-synthesis simulation. MATLAB R2024b served as the primary platform for algorithm development, precision analysis, and performance evaluation with custom floating-point libraries. Remote PA hardware (RF Weblab [41]) enabled practical DPD performance measurement.

The implementation utilized Cadence design tools rather than FPGA-specific flows to enable comprehensive precision exploration. While FPGA vendor IP cores (e.g., Xilinx LogiCORE) provide configurable floating-point operations, Cadence ChipWare IP libraries offered finer-grained control over arithmetic parameters essential for systematic precision analysis [37]. The CW_fp_mult and CW_fp_addsub components provided IEEE 754-compliant arithmetic operations with configurable pre-

cision parameters. IP licensing constraints limited minimum exponent and mantissa widths to 5 bits each.

Specialized MATLAB functions were developed to convert decimal values to custom floating-point binary strings and vice versa, ensuring exact correspondence between software analysis and hardware arithmetic behavior. Performance metrics including NMSE and ACPR were computed using specialized MATLAB library functions with proper floating-point handling.

3.3 Exploration Strategy

Signal Generation and BAPS Configurations

Multi-carrier OFDM signals with 10.39 dB PAPR were generated to represent practical wideband communication scenarios. Each test comprised 79,280 complex baseband samples at 20 MHz bandwidth, providing sufficient statistical diversity for precision analysis while maintaining computational tractability for hardware verification.

Four BAPS configurations were evaluated: BAPS8-mem1, BAPS8-mem5, BAPS12-mem1, and BAPS12-mem5, where the numbers represent basis function counts (8 or 12) and memory depth parameters (mem1 or mem5). The memory depth parameter controls the maximum delay considered during offline greedy search optimization, resulting in BAPS structures with different ratios of Type I to Type II operations and varying computational complexity. This design enables assessment of precision requirements across varying computational complexity profiles and PA memory effect characteristics.

Precision Configuration Space

Floating-point precision was systematically varied across 40 configurations combining ten mantissa widths ($t = 5, 6, 7, 8, 9, 10, 11, 15, 19, 23$) and four exponent widths ($w = 5, 6, 7, 8$). This parameter space spans from minimal 11-bit (5,5) to standard 32-bit (8,23) formats, enabling characterization of hardware efficiency versus DPD performance across a wide complexity range. The complete evaluation matrix comprised 160 scenarios (40 precision configurations $\times$ 4 BAPS variants).

PA Hardware Validation

DPD linearization performance was validated using RF WebLab [41], a remote-accessible PA measurement platform. Figure 3.1 illustrates the validation workflow connecting the implemented DPD system to actual PA hardware.

The validation process operates as follows: predistorted output samples from the BAPS DPD implementation (either MATLAB or hardware) are transmitted to RF WebLab, a remote PA measurement platform, which applies these signals to a physical PA. The signal is sampled at 200 MHz with 160 MHz usable bandwidth. The PA

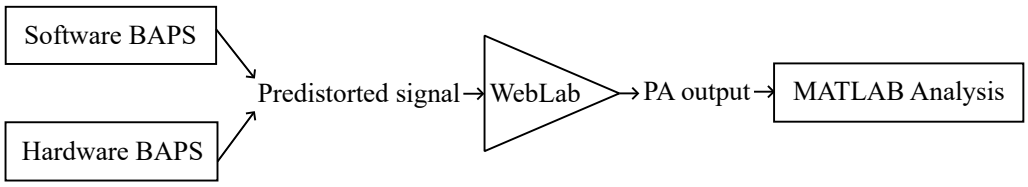

Figure 3.1: PA validation workflow using RF WebLab remote hardware.

output is captured and returned to MATLAB for spectral analysis. NMSE quantifies time-domain accuracy by comparing the linearized PA output against the ideal linear response, while ACPR measures frequency-domain spectral regrowth suppression in adjacent channels. This remote PA connection enables practical linearization performance assessment under real operating conditions, validating both software and hardware implementations against identical PA behavior.

4

Design and Implementation

This chapter details the hardware implementation of a precision-configurable BAPS DPD system, addressing three critical challenges: achieving high linearization performance comparable to practical systems, optimizing floating-point precision for resource efficiency, and enabling scalable complexity through configurable basis function counts.

4.1 System Architecture Overview

The BAPS DPD hardware implementation adopts a modular architecture designed to enable systematic exploration of floating-point precision effects across diverse algorithmic configurations. As illustrated in Figure 4.1, the system operates within a software-hardware co-design framework where MATLAB handles signal generation and performance analysis, while custom floating-point conversion functions provide bit-exact interface with the hardware implementation.

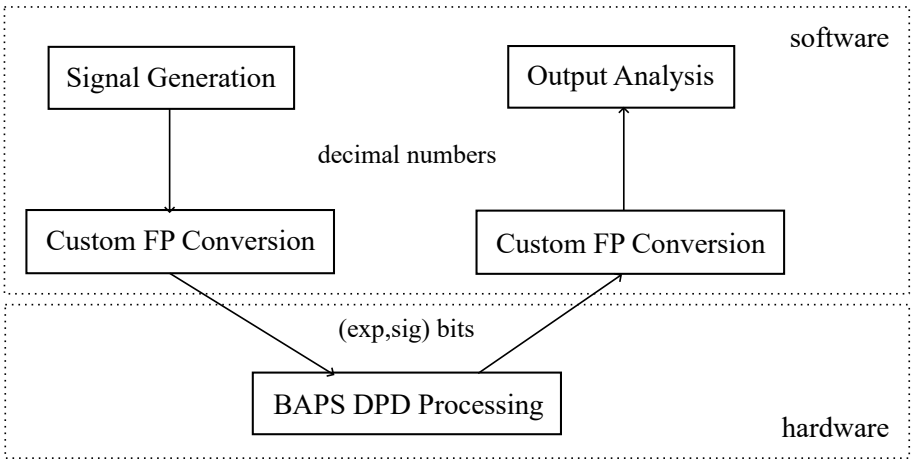

Figure 4.1: System overview of the BAPS DPD hardware implementation with software interface.

Verification Flow Each evaluation scenario follows a comprehensive verification flow designed to ensure both functional correctness and performance accuracy:

1. MATLAB reference computation using double-precision arithmetic as baseline from established BAPS implementation [5]

2. Custom floating-point conversion to target (w,t) configuration in Eq. 2.12
3. RTL simulation with unconstrained synthesized netlist using identical input vectors
4. Backward conversion to MATLAB decimal representation
5. Remote PA connection in MATLAB for practical performance assessment
6. NMSE and ACPR metrics calculations to quantify time-domain accuracy and frequency-domain performance degradation

The complete BAPS system uses unconstrained synthesis for all performance evaluations. Timing-constrained synthesis is applied only for hardware complexity analysis (Section 4.4) to enable fair area comparisons across precision configurations.

Configuration Management Strategy Rather than implementing specific basis function formulations, the architecture provides flexible computational primitives that accommodate varying Type I and Type II operation combinations across all evaluated configurations.

The four evaluated configurations (BAPS8-mem1, BAPS8-mem5, BAPS12-mem1, BAPS12-mem5) differ in basis function count and memory depth characteristics, resulting in distinct computational requirements. The memory depth parameter fundamentally affects the Type I to Type II operation ratio, with mem1 configurations favoring more intensive nonlinear operations and mem5 configurations incorporating additional delay operations that may exhibit different precision sensitivity.

The underlying hardware architecture remains consistent across all configurations, with configuration-specific parameters loaded at synthesis time to determine operation sequences and resource allocation. This approach enables fair comparison of precision effects across different algorithmic complexities while maintaining implementation consistency.

4.2 Modular Architecture Implementation

The block diagram of the RTL design is shown in Figure 4.2, illustrating the interaction between the three main computational modules and their precision-configurable interfaces.

Basis Function Builder

The basis function construction module implements a finite state machine that handles the specific operation sequence required by each BAPS configuration. The VHDL module progresses through sequential computation states, with each state executing the arithmetic operations necessary for computing the corresponding basis function according to the predetermined BAPS sequence.

The state machine architecture incorporates two distinct computational units. Type I operation units utilize configurable delay elements implemented through shift register arrays, providing memory depth capabilities that adapt to the specific con-

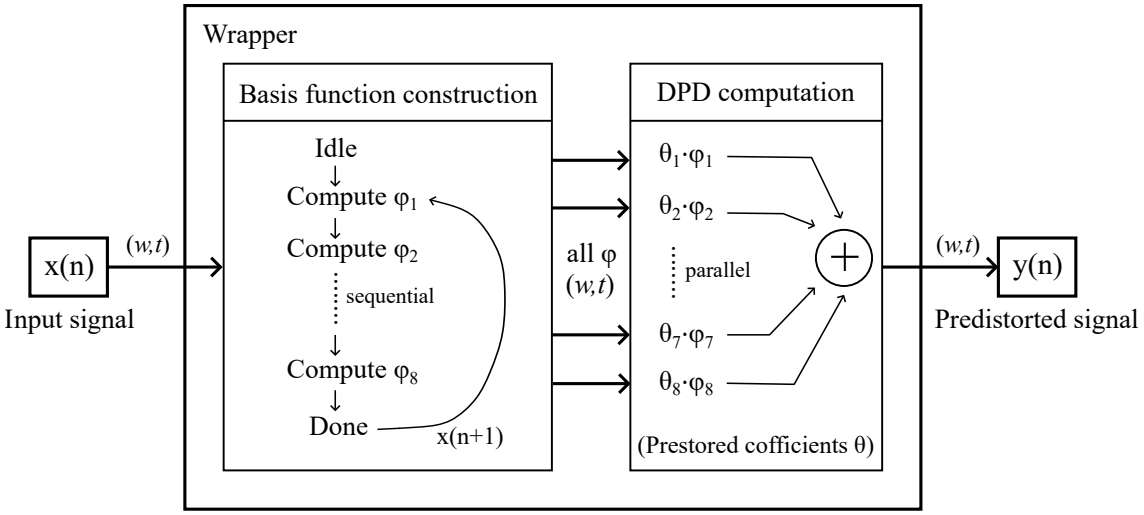

Figure 4.2: BAPS DPD processing system block diagram.

figuration requirements. These delay operations are inherently robust to precision reduction, as they perform no arithmetic computation beyond address manipulation.

Type II operation units represent the computationally intensive core of the BAPS algorithm, implementing complex floating-point multiplications using Cadence ChipWare IP components. These units compute products of the form $\phi_i \phi_j \phi_k^*$, where the complex conjugation and multiplication operations require careful precision management to maintain algorithmic accuracy.

A critical optimization involves strategic caching of frequently reused computational results, particularly magnitude-squared terms that appear in multiple basis function calculations. The hardware manages intermediate result storage based on the specific BAPS configuration's computational structure, reducing redundant calculations while maintaining precision consistency across reused values.

The state sequences are determined by the specific BAPS configuration requirements, with tailored finite state machine implementations for each variant to accommodate varying basis function counts and operation dependencies. The sequential nature of basis function dependencies naturally maps to this state machine approach, where each computation step can only proceed once all prerequisite basis functions have been calculated and validated.

The predetermined basis function definitions for the evaluated configurations are obtained through offline greedy search optimization. Tables 4.1 and 4.2 illustrate the specific operation sequences for 8 and 12 basis function variants respectively.

The comparison reveals distinct computational characteristics between mem1 and mem5 variants. Configurations of mem1 type exhibit more Type II operations, potentially creating greater sensitivity to mantissa precision. In contrast, configurations of mem5 type incorporate more Type I delay operations, which may demonstrate different precision requirements due to their inherently exact nature.

Table 4.1: 8-Basis Function Set Definition for mem1 and mem5 Configurations

Index	Basis Function (mem1)	Basis Function (mem5)
ϕ_1	$x(n)$	$x(n)$
ϕ_2	$\phi_1 \cdot \lvert\phi_1\rvert^2$	$\phi_1 \cdot \lvert\phi_1\rvert^2$
ϕ_3	$q^{-1}\phi_1$	$q^{-4}\phi_1$
ϕ_4	$q^{-1}\phi_3$	$\phi_2 \cdot \lvert\phi_1\rvert^2$
ϕ_5	$\phi_2 \cdot \lvert\phi_3\rvert^2$	$q^{-1}\phi_2$
ϕ_6	$q^{-1}\phi_4$	$\phi_3 \cdot \lvert\phi_1\rvert^2$
ϕ_7	$\phi_6 \cdot \lvert\phi_1\rvert^2$	$q^{-2}\phi_2$
ϕ_8	$\phi_2 \cdot \lvert\phi_1\rvert^2$	$q^{-1}\phi_5$

Table 4.2: 12-Basis Function Set Definition for mem1 and mem5 Configurations

Index	Basis Function (mem1)	Basis Function (mem5)
ϕ_1	$x(n)$	$x(n)$
ϕ_2	$\phi_1 \cdot \lvert\phi_1\rvert^2$	$\phi_1 \cdot \lvert\phi_1\rvert^2$
ϕ_3	$q^{-1}\phi_1$	$q^{-4}\phi_1$
ϕ_4	$q^{-1}\phi_3$	$\phi_2 \cdot \lvert\phi_1\rvert^2$
ϕ_5	$\phi_2 \cdot \lvert\phi_3\rvert^2$	$q^{-1}\phi_2$
ϕ_6	$q^{-1}\phi_4$	$\phi_3 \cdot \lvert\phi_1\rvert^2$
ϕ_7	$\phi_6 \cdot \lvert\phi_1\rvert^2$	$q^{-2}\phi_2$
ϕ_8	$\phi_2 \cdot \lvert\phi_1\rvert^2$	$q^{-1}\phi_5$
ϕ_9	$q^{-1}\phi_6$	$q^{-1}\phi_1$
ϕ_{10}	$\phi_1 \cdot \lvert\phi_3\rvert^2$	$q^{-3}\phi_9$
ϕ_{11}	$\phi_{10} \cdot \lvert\phi_1\rvert^2$	$\phi_{10} \cdot \lvert\phi_1\rvert^2$
ϕ_{12}	$\phi_1 \cdot \lvert\phi_2\rvert^2$	$\phi_9 \cdot \lvert\phi_9\rvert^2$

DPD Computation Engine

Once all basis functions are computed by the basis function builder, the DPD computation module performs the final linear combination of basis functions with predetermined coefficients, implementing the core DPD transformation through parallel coefficient multiplication and tree-structured summation. The parallel computation structure enables high-throughput operation while maintaining precision consistency across all computational paths.

The module utilizes parallel complex multipliers to compute $(\theta_i \cdot \phi_i)$ products using prestored coefficients loaded at initialization. Each multiplier utilizes identical floating-point precision configurations, ensuring consistent numerical behavior across all parallel computation paths. The coefficient storage is implemented using synthesizable memory structures that adapt to the basis function count requirements.

Following coefficient multiplication, a balanced tree structure of floating-point adders performs the final summation. This tree-structured approach provides logarithmic latency scaling with basis function count while maintaining numerical stability through systematic accumulation order. The tree structure adapts to different basis function counts: 8 parallel multipliers with 7 adders for BAPS8 configurations,

scaling to 12 multipliers with 11 adders for BAPS12 variants.

System Wrapper and Coordination

The wrapper module provides essential coordination between the sequential basis function construction and parallel DPD computation phases, managing data flow and timing relationships to maximize system throughput while respecting computational dependencies.

The wrapper implements the pipeline control enabling overlapped execution between consecutive input samples. While the current sample $x(n)$ completes DPD computation, the next sample $x(n+1)$ can simultaneously begin basis function computation, provided that the basis function builder has completed its computational sequence for the previous sample.

Pipeline depth varies dynamically with configuration complexity, as BAPS12 variants require additional computation cycles compared to BAPS8 configurations. The wrapper accommodates this variation through configuration-specific coordination logic that adapts timing relationships based on the loaded BAPS parameters. Standardized input and output interfaces ensure consistent behavior across all precision configurations, with proper handshaking protocols managing the transition between sequential and parallel computation phases. The wrapper also provides essential control signals for initialization, reset, and configuration loading during system startup.

4.3 Floating-Point Arithmetic Implementation

All arithmetic operations utilize Cadence ChipWare IP components to ensure IEEE 754 compliance and enable systematic precision exploration across the full parameter space. The implementation strategy leverages specialized IP cores optimized for different computational requirements while maintaining consistent precision propagation throughout the system.

Multiplication operations utilize the CW_fp_mult component configured for complex arithmetic, while addition and subtraction operations employ the component of CW_fp_addsub. These components can then handle complex multiplication operations, specifically, the four real multiplications and two real additions required for $(a + jb)(c + jd) = (ac - bd) + j(ad + bc)$, while automatically managing intermediate precision and rounding behavior. The precision parameters (w,t) are propagated consistently through all arithmetic operations via VHDL generics, enabling evaluation of the complete 40-configuration precision space without requiring design modifications.

All components operate in round-to-nearest mode, consistent with IEEE 754 default rounding behavior and matching the software reference implementation. This consistency enables direct numerical comparison between hardware and software results, facilitating accurate assessment of precision-induced variations.

4.4 Additional Hardware Complexity Analysis

The hardware complexity analysis adopts a focused approach targeting the most computationally intensive components to establish precision-performance scaling characteristics. This concentrates on representative computational modules that capture the essential complexity scaling trends across different floating-point precision configurations.

Type II operation module, containing complex floating-point multiplications and conjugate or magnitude-squared calculations, represents the most computationally intensive operations within the BAPS algorithm. This module can provide critical insights into how precision configuration affects hardware complexity, as it incorporates the majority of the system's arithmetic operations and exhibits the greatest sensitivity to precision parameters.

Area analysis utilizes the ASAP7 7-nm FinFET process design kit with a 5-ns clock period constraint across all precision configurations. The ASAP7 PDK requires a 16× area downscaling factor due to library cell normalization characteristics. All reported area values apply this correction factor for accurate representation of physical implementation.

The analysis methodology isolates individual Type II modules for synthesis and area evaluation, enabling clear attribution of resource scaling to precision effects rather than system-level architectural variations. This component-level approach establishes the foundation for understanding system-wide resource trends while maintaining analytical clarity.

5

Results and Discussions

This chapter presents the performance evaluation results for the BAPS DPD implementation across different floating-point precision configurations. The analysis systematically examines precision-performance trade-offs through comprehensive measurement of NMSE and ACPR metrics. Additionally, this chapter presents the results of the hardware complexity analysis on the area of the Type II module.

5.1 System Validation

The hardware implementation was validated through comprehensive comparison with MATLAB reference models. The validation process converted all MATLAB signals (inputs, outputs, and intermediate results) from double-precision to single-precision floating-point to match the hardware arithmetic precision. The complete signal chain—DPD output, remote PA processing, and PA output—operates at single precision for both software and hardware paths.

Figure 5.1 demonstrates excellent spectral correspondence between software reference and hardware implementations for the predistorted signal, confirming that the RTL implementation preserves algorithmic functionality without introducing significant numerical artifacts. Figure 5.2 presents the linearized PA output measurements from RF Weblab, further validating system performance under practical operating conditions. The PA output shows substantial spectral regrowth suppression, with both software and hardware implementations achieving comparable adjacent channel power reduction.

These validation results confirm the hardware implementation maintains functional correctness, supporting reliable precision analysis in the following sections. Numerical tests also demonstrate that the NMSE and ACPR values of the hardware implementation align with software methods, with an error margin within 0.1 dB. This suggests the solution maintains the effectiveness of DPD when interfacing with actual PA hardware.

5.2 Precision-Performance Analysis

5.2.1 Mantissa Width

Analysis of mantissa width effects reveals distinct precision regions across all BAPS configurations, with consistent patterns emerging from systematic precision sweeps.

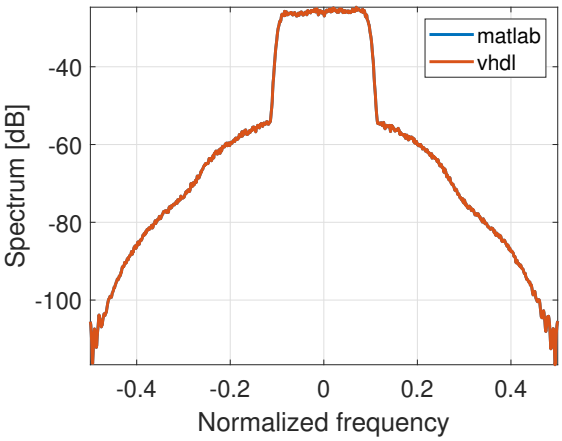

Figure 5.1: Predistorted signal comparison between software reference and hardware implementation.

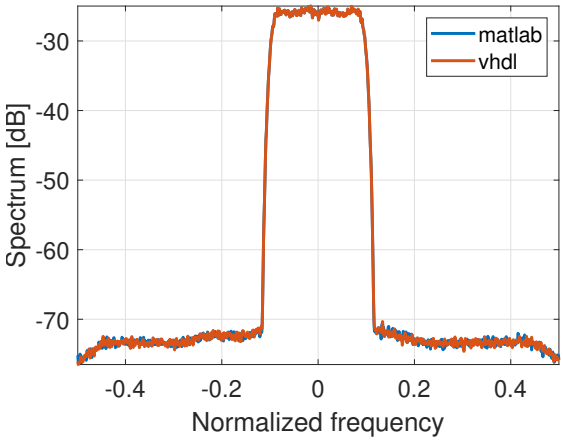

Figure 5.2: PA output comparison between software reference and hardware implementation.

Using BAPS8-mem1 as the primary case study, Table 5.1 summarizes the quantitative performance degradation across the mantissa width sweep.

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:

$$\Delta NMSE = NMSE(w,t) - NMSE(8,23) \tag{5.1}$$

$$\Delta ACPR = ACPR(w,t) - ACPR(8,23) \tag{5.2}$$

Three operational regions can be clearly identified based on performance degradation characteristics. For mantissa bit widths of 23–9, performance degradation remains minimal, with performance difference below 0.15 dB relative to single precision. This region demonstrates that substantial precision reduction is possible without compromising DPD performance. For the transition region of 8–7 bit widths, moderate degradation begins to appear. The critical degradation region occurs when mantissa bit widths are less than 7, with 5 mantissa bit width producing $\Delta NMSE = 4.17$ dB and $\Delta ACPR = 3.34$ dB. This dramatic degradation renders these configurations possibly unsuitable for practical DPD applications.

The BAPS8-mem5 configuration exhibits similar precision characteristics, as shown in Table 5.2. The mem5 variant demonstrates slightly improved robustness to precision reduction, with comparable degradation thresholds but marginally better performance at reduced precision levels.

Figure 5.3 illustrates the progressive emergence of spectral regrowth as mantissa width decreases from 8 to 5 bits using single precision as the baseline. The spectral plots clearly demonstrate how precision reduction progressively compromises linearization effectiveness, with significant adjacent channel power increases becoming visible at 6 mantissa bits and severe degradation at 5 bits.

Table 5.1: 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

Table 5.2: 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

5.2.2 Exponent Width

In contrast to mantissa width effects, exponent width reduction demonstrates negligible impact on DPD performance across all evaluated configurations. Systematic reduction from 8 to 5 exponent bits produces performance degradation below 0.1 dB for both NMSE and ACPR metrics, indicating that exponent width is not a limiting factor for typical DPD signal levels and dynamic ranges.

Figures 5.4 and 5.5 demonstrate this insensitivity to exponent width reduction across different mantissa configurations. Even when mantissa precision is reduced to 6 bits, exponent width variation produces minimal additional degradation, confirming that exponent requirements are independent of mantissa precision constraints.

This behavior reflects the fundamental distinction between mantissa and exponent roles in floating-point arithmetic. While mantissa bits determine computational precision and directly affect numerical accuracy of arithmetic operations, exponent bits primarily control representable dynamic range. Since DPD signals typically operate within constrained amplitude ranges well within the dynamic range of reduced expo-

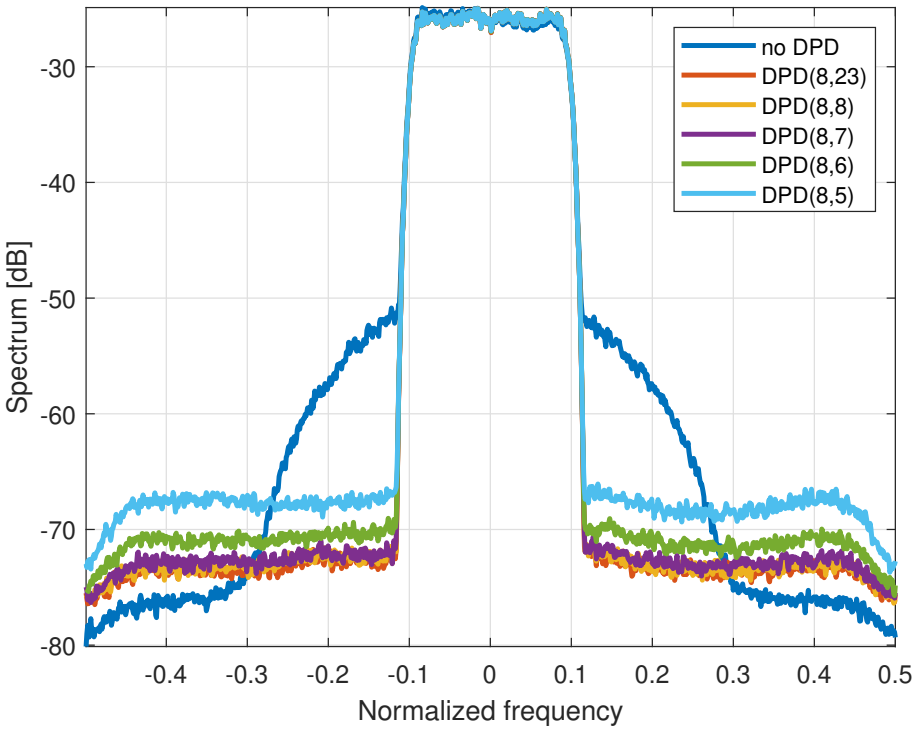

Figure 5.3: Predistorted PA output: mantissa width sweep.

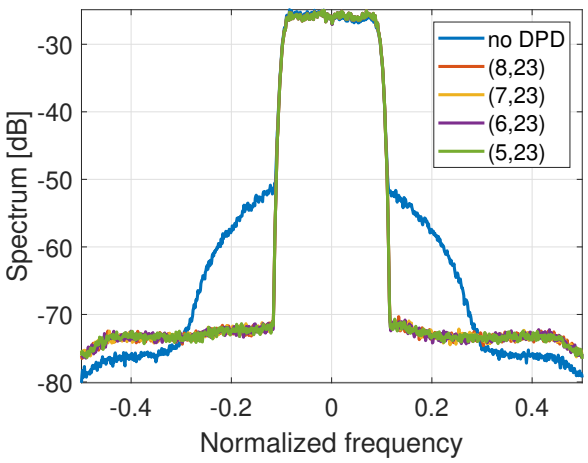

Figure 5.4: PA output with varying exponent bits (23 mantissa bits).

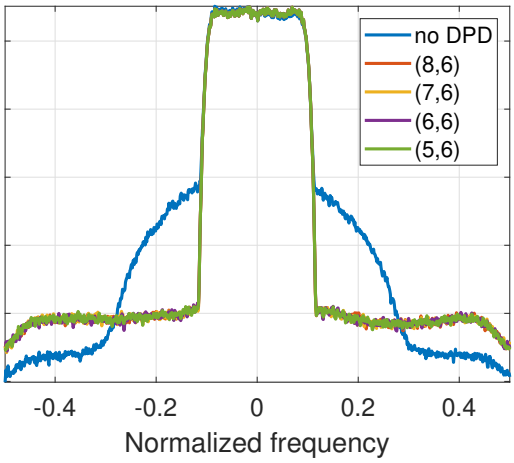

Figure 5.5: PA output with varying exponent bits (6 mantissa bits).

nent fields, the shortened exponent representation does not compromise algorithmic effectiveness.

The practical implication is that exponent width can be aggressively reduced to minimize hardware complexity without much performance degradation, enabling significant bit-width savings through precision optimization that prioritizes mantissa bits over exponent bits.

5.3 Configuration Scalability Analysis

Comparison of BAPS configurations with varying basis function number counts reveals that precision requirements remain remarkably consistent despite significant differences in computational demands. Specifically, the BAPS8 and BAPS12 systems, which differ by 50% in basis function numbers, exhibit identical critical thresholds at 6–7 mantissa bits, with performance degradation patterns following nearly identical trajectories.

Table 5.3 presents single-precision (8,23) measurements across all four configurations, establishing reference performance levels. The configurations achieve comparable linearization effectiveness, with performance differences below 0.5 dB for both NMSE and ACPR metrics. BAPS12-mem1 consistently achieves the best performance, while mem5 configurations show slightly reduced effectiveness.

Table 5.3: Single-precision baseline measurements for four configurations.

Configuration	NMSE (dB)	ACPR (dB)
BAPS8-mem1	-38.13	-43.64
BAPS8-mem5	-38.03	-43.31
BAPS12-mem1	-38.40	-43.79
BAPS12-mem5	-37.94	-43.46

Similar measurement methods are also employed for all other 40 precision configurations. All four variants share identical mantissa width requirements: the robust operation region extends to 9 mantissa bits with performance degradation below 0.15 dB relative to single precision, performance degradation becomes noticeable at 8 bits (0.15–0.4 dB), reaches a critical threshold at 7 bits, and becomes severe below 6 bits.

For example, BAPS12-mem1 maintains NMSE within 0.1 dB of single precision down to 9 mantissa bits, experiences 0.3 dB degradation at 7 bits, and shows critical performance loss at 6 bits. These thresholds match those observed for BAPS8-mem1 almost exactly, despite the 50% difference in computational complexity. Similarly, the mem5 variants exhibit the same critical thresholds as their mem1 counterparts, though with marginally better resilience in the transition region.

This consistency across different BAPS configurations suggests that the precision requirements are basically fundamental to the BAPS computational structure rather than dependent on specific Type I/Type II operation ratios resulting from different memory depth settings.

5.4 Hardware Complexity Trade-offs

Area analysis from constrained synthesis reveals approximately linear scaling with total floating-point bitwidth. Figure 5.6 illustrates the linear relationship between arithmetic unit area and precision configuration. The scaling behavior shows that

mantissa width contributes more significantly to area requirements than exponent width, consistent with the internal architecture of floating-point arithmetic units where mantissa processing requires more extensive logic resources [42].

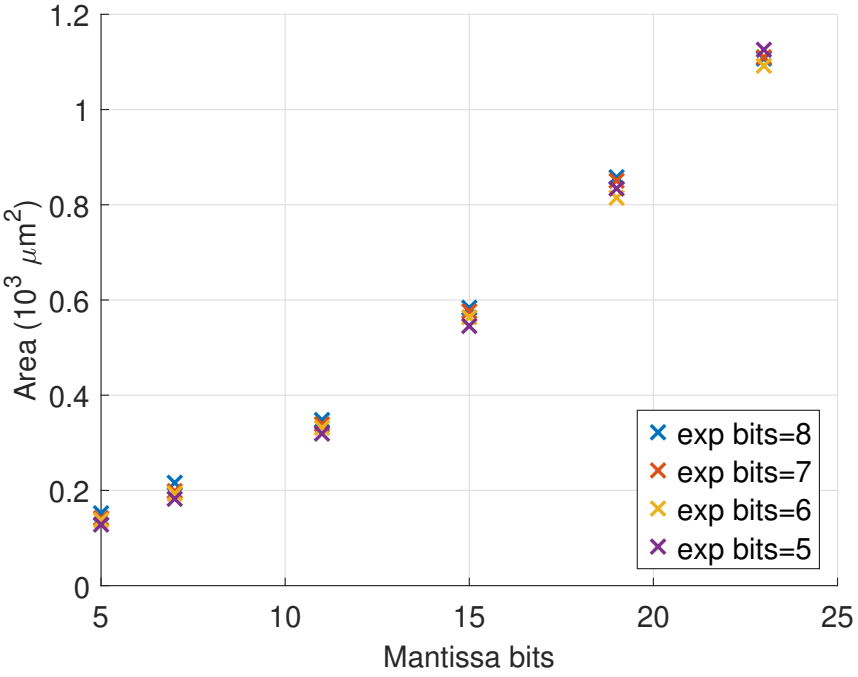

Figure 5.6: Synthesized area of Type II module for different floating-point bitwidths.

The area scaling relationship enables predictable complexity-performance trade-offs for design optimization. Reducing from 23 to 7 mantissa bits achieves approximately 80 % area reduction (from ∼1.1 to ∼0.2 $\times 10^3\,\mu m^2$) while maintaining performance within 0.4 dB of single precision. This represents substantial hardware savings with acceptable performance compromise for resource-constrained applications.

5.5 Discussions

The systematic precision exploration reveals that mantissa width fundamentally limits BAPS DPD performance, with 7 bits representing a critical threshold beyond which linearization effectiveness degrades substantially. In contrast, exponent precision can be aggressively reduced to 5–6 bits without performance penalty, reflecting the constrained dynamic range of typical PA signals. This sensitivity difference enables focused optimization: prioritize mantissa precision while minimizing exponent bits for maximum hardware efficiency. The findings also suggest that fixed-point arithmetic with appropriately sized word lengths could be an alternative, particularly given the constrained dynamic range requirements demonstrated by the minimal exponent sensitivity.

The consistency of precision requirements across BAPS8 and BAPS12 configurations indicates that accuracy bottlenecks arise from individual arithmetic operations

rather than cumulative error propagation. This finding suggests the identified precision thresholds may generalize to other Volterra-based DPD algorithms with similar computational structures, though algorithm-specific validation remains necessary.

5.6 Limitations and Future Work

The current study focuses on four specific BAPS configurations with a single PA model and signal type. While the consistent trends across evaluated configurations suggest broader applicability, validation with diverse PA characteristics and signal types would strengthen the generalizability of these findings.

The investigation was constrained by IP licensing limitations requiring minimum 5-bit field widths. Future work could explore even more aggressive precision reduction, particularly for exponent fields where current results suggest substantial further reduction may be possible, and investigate fixed-point arithmetic implementations that eliminate exponent overhead entirely while maintaining sufficient dynamic range for DPD operations.

Finally, extending the precision exploration to alternative DPD algorithms would provide broader insights into floating-point requirements for digital predistortion applications, potentially revealing algorithm-specific precision sensitivities that could inform future DPD development.

5.7 Ethical Considerations

This thesis was developed with AI assistance (Claude, ChatGPT, Grammarly) for text refinement and revision.

AI support was specifically used for improving clarity and grammatical correctness of technical descriptions such as restructuring paragraphs for better logical flow and refining transition sentences between sections. Additionally, AI tools assisted with formatting consistency checks, including LaTeX syntax verification and reference formatting. Some prompts are like "Revise this paragraph for clearer flow" or "Help with LaTeX table structure".

All technical content, experimental design, data analysis, results interpretation, and conclusions are the original work without AI assistance.

6

Conclusion

This thesis developed and validated a precision-configurable hardware implementation of the BAPS algorithm, establishing the first comprehensive characterization of floating-point precision requirements for this computationally efficient DPD approach. A parameterizable VHDL architecture utilizing Cadence ChipWare floating-point IP enables systematic precision exploration across the complete design space while maintaining bit-accurate correspondence with software reference models. The modular structure, separating basis function construction and DPD computation stages with configurable Type I and Type II operations, provides a reusable framework for implementing BAPS variants with different complexity-performance targets.

Through analysis of 40 custom precision configurations across four distinct BAPS variants, the key finding is that mantissa width critically determines performance, with robust operation maintained down to 9 bits, a transition region at 7–8 bits, and severe degradation below 7 bits. In contrast, exponent width has negligible impact, enabling aggressive reduction from 8 to 5 bits without performance penalty. Notably, precision requirements remain consistent across different BAPS configurations despite substantial variations in computational complexity, suggesting that precision sensitivities are intrinsic to the BAPS algorithm structure rather than dependent on implementation details. Hardware complexity analysis confirms approximately linear scaling, with mantissa width dominating resource usage.

This research demonstrates that sophisticated DPD algorithms can be efficiently implemented at reduced precision, far below standard floating-point formats, enabling substantial hardware savings while maintaining linearization effectiveness. The systematic methodology provides a framework applicable to other DPD algorithms and establishes that careful precision optimization can deliver the hardware efficiency needed for deployment in resource-constrained wireless systems.

Future work should investigate diverse PA characteristics and signal types, extension to alternative DPD algorithms, and fixed-point solutions. These findings enable more efficient implementation of DPD techniques in next-generation wireless networks while maintaining the performance required for modern communication systems.

Bibliography

[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] S. Cripps, *RF Power Amplifiers for Wireless Communications*, 2nd ed. Norwood, MA, USA: Artech House, 2006.

[3] Y. Rahmatallah and S. Mohan, "Peak-to-average power ratio reduction in OFDM systems: A survey and taxonomy," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1567–1592, 2013.

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

[5] 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.

[6] 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.

[7] J. Reina-Tosina, M. Allegue-Martínez, C. Crespo-Cadenas, C. Yu, and S. Cruces, "Behavioral modeling and predistortion of power amplifiers under sparsity hypothesis," *IEEE Transactions on Microwave Theory and Techniques*, vol. 63, no. 2, pp. 745–753, 2015.

[8] T. Kobal, Y. Li, X. Wang, and A. Zhu, "Digital predistortion of RF power amplifiers with phase-gated recurrent neural networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. 70, no. 6, pp. 3291–3299, 2022.

[9] R. Hongyo, Y. Egashira, T. M. Hone, and K. Yamaguchi, "Deep neural network-based digital predistorter for Doherty power amplifiers," *IEEE Microwave and Wireless Components Letters*, vol. 29, no. 2, pp. 146–148, 2019.

[10] Y. Ma, Y. Yamao, Y. Akaiwa, and C. Yu, "FPGA implementation of adaptive digital predistorter with fast convergence rate and low complexity for multi-channel transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 11, pp. 3961–3973, 2013.

[11] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Jan. 2007.

[12] L. Guan and A. Zhu, "Low-cost FPGA implementation of Volterra series-based digital predistorter for RF power amplifiers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 4, pp. 866–872, 2010.

[13] P. L. Gilabert, G. Montoro, D. López, N. Bartzoudis, E. Bertran, M. Payaró, and A. Hourtane, "Order reduction of wideband digital predistorters using principal component analysis," in *2013 IEEE MTT-S International Microwave Symposium Digest (MTT)*, 2013, pp. 1–7.

[14] H. Cao, A. Soltani Tehrani, C. Fager, T. Eriksson, and H. Zirath, "Dual-input nonlinear modeling for I/Q modulator distortion compensation," in *2009 IEEE Radio and Wireless Symposium*, 2009, pp. 39–42.

[15] R. N. Braithwaite, "Digital predistortion of an RF power amplifier using a reduced Volterra series model with a memory polynomial estimator," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 10, pp. 3613–3623, 2017.

[16] C. Mythile, S. Jaisiva, A. Arunadevi, and K. Sudhapriya, "Examining floating point precision in contemporary FPGAs," in *2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, 2024, pp. 1–7.

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

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

[19] F. de Dinechin and B. Pasca, "Designing custom arithmetic data paths with FloPoCo," *IEEE Design Test of Computers*, vol. 28, no. 4, pp. 18–27, 2011.

[20] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, "A survey and evaluation of FPGA high-level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1591–1604, 2016.

[21] C. H. Ho, C. W. Yu, P. Leong, W. Luk, and S. J. E. Wilton, "Floating-point FPGA: Architecture and modeling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 12, pp. 1709–1718, 2009.

[22] F. Raab, P. Asbeck, S. Cripps, P. Kenington, Z. Popovic, N. Pothecary, J. Sevic, and N. Sokal, "Power amplifiers and transmitters for RF and microwave," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 3, pp. 814–826, 2002.

[23] C. Fager, T. Eriksson, F. Barradas, K. Hausmair, T. Cunha, and J. C. Pedro, "Linearity and efficiency in 5G transmitters: New techniques for analyzing efficiency, linearity, and linearization in a 5G active antenna transmitter context," *IEEE Microwave Magazine*, vol. 20, no. 5, pp. 35–49, 2019.

[24] L. Guan and A. Zhu, "Green communications: Digital predistortion for wideband RF power amplifiers," *IEEE Microwave Magazine*, vol. 15, no. 7, pp. 84–99, 2014.

[25] A. Saleh, "Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers," *IEEE Transactions on Communications*, vol. 29, no. 11, pp. 1715–1720, 1981.

[26] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. New York: Wiley, 1980.

[27] A. Zhu, J. C. Pedro, and T. J. Brazil, "Dynamic deviation reduction-based Volterra behavioral modeling of RF power amplifiers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 12, pp. 4323–4332, 2006.

[28] Y. Guo, C. Yu, and A. Zhu, "Power adaptive digital predistortion for wideband RF power amplifiers with dynamic power transmission," *IEEE Transactions on Microwave Theory and Techniques*, vol. 63, pp. 1–13, Oct. 2015.

[29] S. Ahmed, M. Ahmed, S. Bensmida, and O. Hammi, "Power amplifier predistortion using reduced sampling rates in the forward and feedback paths," *Sensors*, vol. 24, no. 11, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/11/3439

[30] 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.

[31] P. B. Nair, A. Choudhury, and A. J. Keane, "Some greedy learning algorithms for sparse regression and classification with Mercer kernels," *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 781–801, 2002.

[32] 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.

[33] 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

[34] R. Lyons, *Understanding Digital Signal Processing*, ser. Prentice Hall professional technical reference. Prentice Hall/PTR, 2004.

[35] C. Inacio and D. Ombres, "The DSP decision: fixed point or floating?" *IEEE Spectrum*, vol. 33, no. 9, pp. 72–74, 1996.

[36] M. Langhammer and B. Pasca, "Design and implementation of an embedded FPGA floating point DSP block," in *2015 IEEE 22nd Symposium on Computer Arithmetic*, 2015, pp. 26–33.

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

[38] C. Erdmann, E. Cullen, D. Brouard *et al.*, "A 330mw 14b 6.8GS/s dual-mode RF DAC in 16nm FinFET achieving -70.8dBc ACPR in a 20MHz channel at 5.2GHz," in *IEEE ISSCC*, 2017, pp. 280–281.

[39] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.

[40] 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.

[41] 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.

[42] P. Larsson-Edefors, "Energy-efficient computation of TensorFloat32 numbers on an FP32 multiplier," in *IFIP/IEEE 33rd Int. Conf. on Very Large Scale Integration (VLSI-SoC)*, 2025, to be published.