A Framework for Design and Implementation of Adaptive Digital Predistortion Systems

Lin Li*, Peter Deaville*, Adrian Sapio*, Lauri Anttila†, Mikko Valkama†, Marilyn Wolf‡, Shuvra S. Bhattacharyya*†
*University of Maryland, College Park, ECE Department, College Park, MD 20742, USA
Email: {lli12311, deaville, asapio, ssb}@umd.edu
†Tampere University, Electronics and Communications Engineering, Finland
Email: {lauri.anttila, mikko.e.valkama}@tuni.fi
‡ Georgia Institute of Technology, Georgia, USA
Email: {wolf}@ece.gatech.edu

Abstract—**Digital predistortion (DPD) has important applications in wireless communication for smart systems, such as, for example, in Internet of Things (IoT) applications for smart cities. DPD is used in wireless communication transmitters to counteract distortions that arise from nonlinearities, such as those related to amplifier characteristics and local oscillator leakage. In this paper, we propose an algorithm-architecture-integrated framework for design and implementation of adaptive DPD systems. The proposed framework provides energy-efficient, real-time DPD performance, and enables efficient reconfiguration of DPD architectures so that communication can be dynamically optimized based on time-varying communication requirements. Our adaptive DPD design framework applies Markov Decision Processes (MDPs) in novel ways to generate optimized runtime control policies for DPD systems. We present a GPU-based adaptive DPD system that is derived using our design framework, and demonstrate its efficiency through extensive experiments.**

Keywords-**Smart systems, dataflow modeling, digital predistortion, Markov decision processes.**

I. Introduction

Smart systems are capable of sensing the environment and making decisions based on the available data in an adaptive manner. To gather useful information about the environment from sensors in real-time, smart systems may make use of Internet of Things (IoT) technology. The sensory information acquired by IoT devices can be transmitted to a base station for aggregation and analysis to make decisions.

IoT devices are often equipped with wireless interfaces. However, the quality of the wireless communication in smart systems can suffer from non-idealities. In particular, non-linearity of the power amplifier (PA) is a notorious source of signal distortion. To address this issue, digital predistortion (DPD) can be utilized to counteract PA non-linearities [1]. The implementation of an adaptive DPD system involves a complex optimization problem that affects the wireless communications quality, energy efficiency, and real-time performance of the associated devices.

Due to changes in the environment such as temperature and voltage, PA characteristics in general vary at run-time. Thus, for most efficient operation, a DPD system should be able to change the predistortion coefficients dynamically to ensure that its underlying model matches accurately with its operating environment. Moreover, for integration in smart systems with many types of connected devices and communication modes, it is critical for DPD systems to support efficient predistortion across time-varying operational requirements and modulation schemes. Thus, dynamic system control and reconfiguration are desirable compared to static design for DPD systems.

In this paper, we develop a general framework for deploying DPD implementations that address the need for efficient adaptivity in wireless communications devices. Implementations that are deployed using this framework are designed to autonomously make run-time decisions on DPD system configurations based on the current system state and operational state. The reconfiguration is driven by policies for dynamic system management that are derived at design time using Markov decisions processes (MDPs) [2]. The key novelty of this paper is the development of a general framework for MDP-based design and implementation of adaptive DPD systems.

We refer to the proposed framework as the *MDP framework for Adaptive DPD Systems* (*MADS*). Dynamic reconfiguration in MADS is performed with the objective of jointly optimizing Adjacent Channel Power Ratio (ACPR), system throughput, and power efficiency. ACPR is often viewed as the most critical metric for assessing the quality of DPD systems. ACPR is defined as the ratio of the mean power centered on the adjacent channels to the mean power centered on the desired channel.

To demonstrate the MADS Framework, we design an adaptive version of a state-of-the-art DPD algorithm from the literature — the DPD algorithm presented by Anttila in [1]. In particular, we apply the MADS Framework to develop an adaptive architecture that dynamically selects strategic configurations from the design space defined by Anttila's algorithm. We develop a hybrid CPU/GPU implementation of this adaptive architecture, and demonstrate its efficiency through extensive experiments.

II. Related Work

Optimization problems for DPD systems have been widely studied over the years. For example, the work in [3]–[5] applies genetic algorithms to optimize DPD filter coefficients while assuming fixed filter and polynomial orders. In [6], both filter coefficients and polynomial orders are jointly optimized; however, this optimization is performed with respect to only a single objective — ACPR. Ghazi et al. propose a data-parallel implementation of reconfigurable DPD on a mobile Graphics Processing Unit (GPU) [7]. The implementation allows the DPD parameters to fit various transmission scenarios by selecting a set of candidate profiles based on desired linearization performance. However, this work does not provide any control policy for optimized run-time reconfiguration. In [8], Pareto-optimized DPD parameters are derived subject to multidimensional constraints to support dynamically reconfigurable DPD systems that are adaptive to changes in the employed modulation schemes and operational constraints. However, this work does not take into account reconfiguration costs nor statistics of the environment and system states.

In comparison to the related work referenced above, our contribution in this paper is novel in its development of a general framework for integrating dynamic reconfiguration systematically into a broad class of DPD algorithms. To the best of our knowledge, the proposed framework is the first that integrates MDP algorithms for the derivation of dynamic DPD system parameters, and optimizes multiple objectives including ACPR, power consumption and throughput.

III. MDP Framework for Adaptive DPD Systems

The MADS Framework is illustrated in Fig. 1. The framework is designed so that many kinds of DPD algorithms can be plugged in to generate MDP-integrated, adaptive systems that are based on those algorithms. When the MADS Framework is applied to a DPD algorithm X, we refer to X as the *base algorithm*, and the adaptive system produced by the MADS Framework is referred to as MADS-X. The base algorithm is assumed to have two stages: (1) an estimation stage, where the DPD coefficients are estimated according to the input and output signals of the PA, and (2) a filtering stage, where the input signal is filtered based on the coefficients estimated from the estimation stage.

In Fig. 1, boxes with black-colored borders represent general design processes that are involved in applying the MADS Framework, while boxes with blue-colored borders represent design processes that are specific to the base algorithm.

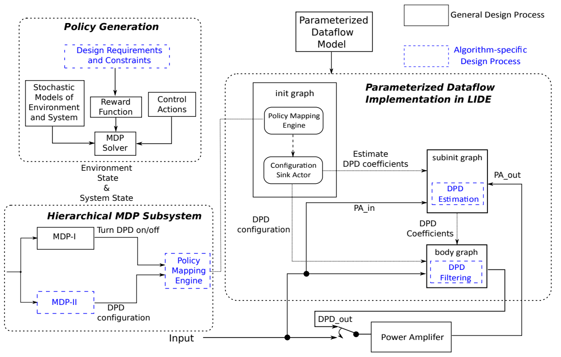

Fig. 1: An illustration of the MADS Framework.

Based on both general features of DPD applications and architecture-specific system behaviors, the MADS Framework models environmental and system dynamics in the form of *hierarchical MDPs* [9]. This modeling approach is illustrated in the block in Fig. 1 that is labeled Hierarchical MDP Subsystem. The Hierarchical MDP Subsystem consists of two smaller MDPs, MDP-I and MDP-II. MDP-I generates a policy that determines when to turn the DPD system on and off, while MDP-II determines key DPD parameter configurations that should be used at a given time when the DPD is on. MDP-I may turn predistortion off, for example, if due to current channel conditions or quality of service requirements, predistortion is expected to not be needed for some significant amount of time.

DPD parameters that can be configured by MDP-II include the polynomial orders, filter orders, and filter coefficients. The exact set of parameters that MDP-II optimizes in general differs between different choices of the base algorithm. Thus, when applying the MADS design methodology, MDP-II should be formulated specifically for the given base algorithm.

The block in Fig. 1 labeled Policy Generation illustrates the application of an MDP solver that is used to derive reconfiguration policies from MDP-I and MDP-II. As with the base algorithm, the MADS Framework is not specialized to any specific MDP solver. In our current implementation of the framework, we use the MDPSOLVE solver [10].

As shown in Fig. 1, a reconfigurable DPD application system developed in MADS is modeled as a *parameterized dataflow* graph. Parameterized dataflow is a graph-based form of model-based design that is well-suited to design and implementation of signal processing systems that have dynamic parameters [11]. To implement the dataflow graph, we apply the *lightweight dataflow environment* (*LIDE*), which is a design tool for dataflow-based design and implementation of signal processing systems [12]. LIDE provides a compact set of application programming interfaces (APIs) for implementing signal processing applications as dataflow graphs. A useful feature of LIDE is that it facilitates the retargeting of designs to different implementation languages, such as C, CUDA, and Verilog/VHDL, and different platforms [13]. MADS inherits this useful feature of retargetability from LIDE.

A parameterized dataflow specification consists of three distinct graphs — the *init*, *subinit*, and *body* graphs. Intuitively, the init and subinit graphs are used to compute parameter updates for the body graph. The init graph can also modify parameters of the subinit graph. In the remainder of this section, we describe how these component graphs are applied in the MADS Framework; for general definitions on these and other parameterized dataflow concepts, we refer the reader to [11].

In our LIDE-based implementation of MADS, the init graph computes system hyperparameters that affect the overall DPD system architecture (i.e., the DPD specific subsystems that are used and their interconnections). We refer to these hyperparameters as *architecture configuration parameters*. The parameterized dataflow runtime system in LIDE propagates the parameter updates computed by the init graph to the subinit and body graphs.

In contrast to the init graph, the subinit graph computes system parameters that are used to configure a given DPD architecture. We refer to these architecture-specific parameters as *DPD coefficients* since they are constrained to being values associated with digital filter coefficients. The subinit graph encapsulates the base-algorithm-specific DPD estimation subsystem as a main component. The estimation subsystem is used to estimate new values for filter coefficients that are to be used in subsequent executions of the body graph. On the other hand, the body graph encapsulates the set of available DPD architectures, and performs the signal predistortion based on the most-recently selected architecture (selected by the init graph) and its most-recently configured coefficients (from the subinit graph). Execution control changes iteratively among the init, subinit, and body graphs based on coordination rules that are defined as part of parameterized dataflow semantics [11].

IV. Activation Control

In this section, we present the formulation of MDP-I. This MDP is designed in a general way to operate with arbitrary base algorithms. In general, the formulation of an MDP requires specification of four key components: the state space (SS), action space (AS), state transition matrix (STM), and reward function (RF). In the remainder of this section, we describe the formulation of these components for MDP-I. For general background on MDPs, including the role of their four general components, we refer the reader to [2].

SS: The SS for MDP-I is represented as: $s_1 = (T_x, on)$, where T_x is the current transmission power level, and on is a binary value representing whether the system is turned on (1) or off (0). The subscript 1 is used to indicate correspondence with MDP-I. We quantize the continuous values of transmission power to obtain a discrete SS.

AS: The AS consists of two actions: one to turn on (activate) the DPD system and the other to turn it off. We denote the action by a_1.

STM: We define two STMs, which specify the state transition probabilities for each of the two actions. The definition of these STMs exploits two properties: (1) T_x is independent of the action and the other state variable, and (2) the DPD system is fully under the control of the action, so the system transitions to the on/off state as requested by the action with probability 1. The STMs are defined by:

$$\begin{aligned} &P(s_1(t+1) = (j,y) | s_1(t) = (i,x), a_1) \\ &= P(T_x(t+1) = j, on(t+1) = y | T_x(t) = i, on(t) = x, a_1) \\ &= P(T_x(t+1) = j | T_x(t) = i) P(on(t+1) = y | a_1) \ , \end{aligned} \tag{1}$$

where $P(on(t+1) = y|a_1)$ is 1 if a_1 is to turn on the DPD and 0 otherwise.

RF: We formulate the RF R_1 as a linear combination of four competing metrics: the averaged DPD power consumption M_P, ACPR M_A, throughput M_T, and switching cost M_S incurred by turning on the DPD system from an off status.

$$\begin{aligned} R_1(s_1(t), a_1) \ &= c_1 M_P(s_1(t)) + c_2 M_A(s_1(t)) + \\ &\quad c_3 M_S(a_1, s_1(t)) + c_4 M_T(s_1(t)) \ , \end{aligned}$$

where c_1, c_2, c_3 and c_4 are the weights of the optimization objectives. Determination of these weights is a design issue that influences operational trade-offs of the DPD system. The values of c_1, c_2, and c_3 should be negative and c_4 should be positive since the MDP is formulated to *maximize* the reward function.

V. Demonstration and Experiments

In this section, we demonstrate the MADS Framework by applying it to Anttila's algorithm [1] as the base algorithm. We refer to the integration of MADS with Anttila's algorithm as *MADS-A*.

In Anttila's algorithm, the DPD system is split into two parts (*branches*): direct and conjugate predistortions. We denote the maximum orders of the polynomials for the direct and conjugate branches by p and q, respectively. The parameters p and q influence trade-offs among throughput, ACPR, and power consumption. Strategic control of these parameters is the target of MDP-II in MADS-A. In Anttila's algorithm, only odd-order polynomials are used. Thus, the sets of polynomial orders are $I_p = \{1, 3, 5, \ldots, p\}$ for the direct branches and $I_q = \{1, 3, 5, \ldots, q\}$ for the conjugate branches. For more details about Anttila's algorithm, we refer the reader to [1].

A. MDP-II Formulation

As discussed in Section III, the MDP-II component of MADS is application-specific in that it needs to be specialized to the base algorithm. In this section, we present our MDP formulation for MDP-II in MADS-A. In MADS-A, the base algorithm parameters that are controlled by MDP-II are the polynomial orders p and q. The components of MDP-II are summarized as follows.

SS: The SS consists of the current transmission power level (as in MDP-I) and the deployed DPD configuration (p-q combination), where $p \in \{1, 3, 5, 7, 9\}$ and $q \in \{1, 3, 5, 7, 9\}$. In MDP-II, we represent the state as $s_2 = (T_x, p, q)$.

AS: An action in MDP-II corresponds to determining the p-q combination for the next MDP time step. Thus, the AS can be represented as the set of all possible p-q combinations, and the AS contains $5 \times 5 = 25$ elements. We denote the AS by a_2.

STM: We define 25 STMs corresponding to the 25 actions in MDP-II. The state transitions for transmission power are controlled by MDP-I, and are independent from both the action and the system configuration. Similar to our development of MDP-I, we assume that given a particular action, there is a deterministic transition to the target configuration. The STMs can be expressed in a form similar to that of Equation 1. We omit the details due to space limitations.

RF: Similar to MDP-I, the reward function is a linear combination of M_P, M_A, M_T, and a switching cost M_S. The metrics M_P, M_A, M_T are the same as in MDP-I and have the same weighting coefficients. However, the switching cost M_S for MDP-II is different. For MDP-II, M_S refers to the cost of reconfiguration from the p-q combination in the current time step to the combination to be used in the next time step (based on a_2). The weight of M_S for MDP-II is generally determined separately from the corresponding weight for MDP-I.

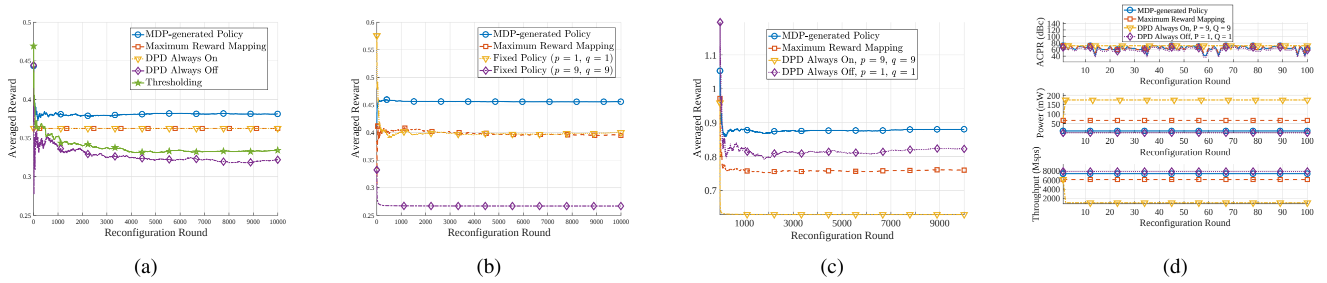

Fig. 2: Measured results for (a) MDP-I, (b) MDP-II, (c) MADS-A (Hierarchical MDPs), (d) Averaged ACPR (dBc), power (mW) and throughput (Msps) of hierarchical MDPs.

B. *Experimental Results*

We implemented the MADS-A system using LIDE on a hybrid CPU/GPU platform composed of an Intel i7-2600K (CPU) running at 3.4 GHz, and an NVIDIA GeForce GTX 1080 (GPU). The body graph (see Fig. 1) of MADS-A is mapped to the GPU and the subinit and init graphs are mapped to the CPU. The system throughput and power consumption data that we collected to calibrate the reward functions for the MDP formulations in MADS-A are based on the NVIDIA GeForce 1080.

Metrics under different DPD configurations are obtained from a wireless transmission simulator identical to that used in [1]. The simulator that we used consists of a WiFi signal generator, pulse shaping filter, DPD, and Wiener PA with the same parameters as used in [1]. Signal bandwidth is 20 MHz with 64 subcarriers and QPSK modulation. For each system configuration (p, q, P_{TX}), the simulator is executed and the results are used later by the MDP solver.

The key component for the STM is the transmit power transition matrix (TPTM). To obtain the transition matrix, we collect WiFi packets with a bandwidth of 20 MHz on the 5 GHz band with a laptop in a building within the University of Maryland campus. A software called *libpcap* is used on the laptop to capture the transmit power levels of WiFi packets from nearby devices. The TPTM is derived from the transmit power data that is collected in this way.

In the remainder of this section, we present experimental results for MDP-I, MDP-II and the hierarchical combination of both MDP-I and MDP-II. We compare the the three resulting MDP-generated policies among themselves to assess the utility of using the proposed hierarchical MDP. We also compare the MDP approaches with a number of simple, static policies for configuration management. The hierarchical combination represents the MADS-A implementation, while the other reconfiguration policies are implemented using the same CPU/GPU testbed by adding/disabling appropriate functionality.

In each of the experiments, we simulate the DPD application system for $10,000$ MDP time steps (*reconfiguration rounds*), where the interval between steps is 10 milliseconds (ms). This is the average reception interval between two packets that was measured in the WiFi experiments described above.

The simulation is carried out with MATLAB using reward functions that are computed based on profiled execution time and power consumption characteristics that are measured from the targeted CPU/GPU platform.

The results are summarized in in Fig. 2(a)–2(c) with $(c_1, c_2, c_3, c_4) = (-0.4, -0.3, -0.2, 0.1)$. Here, the curves labeled "Maximum Reward Mapping" represent the policy that selects the action with highest reward in the current state without considering the potential impact of the action on the future. The curve labeled "Thresholding" represents the performance of a policy that turns off the system when the transmission power is smaller than a certain pre-defined value (5 dBm). The curves labeled with the prefix "Fixed Policy" represent policies that simply fix the DPD configuration as specified.

The results in Fig. 2(a)–2(c) clearly demonstrate the capability of MADS-A to significantly outperform the individual MDPs used in isolation as well the more conventional (static) configuration management schemes. A similar observation can be made for different (c_1, c_2, c_3, c_4).

In Fig. 2(d), we demonstrate the ACPR, power consumption, and throughput under different policies. These results show the effectiveness of the proposed framework — in particular, its ability to strike a balance among different DPD metrics.

VI. Conclusions

In this paper, we have motivated the relevance of adaptive digital predistortion (DPD) to smart systems, and we have proposed a general framework that applies Markov decision processes (MDPs) for design and implementation of adaptive DPD systems. Our framework, called the MDP framework for Adaptive DPD Systems (MADS), is designed so that many kinds of DPD algorithms can be plugged in to generate MDP-integrated, adaptive systems that are based on those algorithms. We demonstrate MADS by plugging into it a state-of-the-art DPD algorithm, and implementing the resulting adaptive DPD system on a hybrid CPU/GPU platform. Through extensive experiments, we have demonstrated the utility of the resulting implementation — in particular, of the hierarchical MDP approach that is at the core of the MADS Framework.

Acknowledgments

This research was sponsored in part by the US National Science Foundation.

References

[1] L. Anttila, P. Händel, and M. Valkama, "Joint mitigation of power amplifier and I/Q modulator impairments in broadband direct-conversion transmitters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 4, pp. 730–739, 2010.

[2] O. Sigaud and O. Buffet, Eds., *Markov Decision Processes in Artificial Intelligence*. Wiley, 2010.

[3] R. Sperlich, J. A. Sills, and J. S. Kenney, "Power amplifier linearization with memory effects using digital pre-distortion and genetic algorithms," in *Proceedings of the Radio and Wireless Conference*, 2004, pp. 355–358.

[4] C. Çiflikli and A. Yapící, "Genetic algorithm optimization of a hybrid analog/digital predistorter for RF power amplifiers," *Analog Integrated Circuits and Signal Processing*, vol. 52, no. 1, pp. 25–30, 2007.

[5] K. Freiberger, M. Wolkerstorfer, H. Enzinger, and C. Vogel, "Digital predistorter identification based on constrained multi-objective optimization of wlan standard performance metrics," in *Proceedings of the International Symposium on Circuits and Systems*, 2015, pp. 862–865.

[6] A. H. Abdelhafiz, O. Hammi, A. Zerguine, A. T. Al-Awami, and F. M. Ghannouchi, "A PSO based memory polynomial predistorter with embedded dimension estimation," *IEEE Transactions on Broadcasting*, vol. 59, no. 4, pp. 665–673, 2013.

[7] A. Ghazi, J. Boutellier, O. Silvén, S. Shahabuddin, M. Juntti, S. S. Bhattacharyya, and L. Anttila, "Model-based design and implementation of an adaptive digital predistortion filter," in *Proceedings of the IEEE Workshop on Signal Processing Systems*, 2015, pp. 1–6.

[8] L. Li, A. Ghazi, J. Boutellier, L. Anttila, M. Valkama, and S. S. Bhattacharyya, "Evolutionary multiobjective optimization for adaptive dataflow-based digital predistortion architectures," *EAI Endorsed Transactions on Cognitive Communications*, vol. 3, no. 10, pp. 1–9, 2017.

[9] A. Jonsson and A. Barto, "Causal graph based decomposition of factored MDPs," *Journal of Machine Learning Research*, vol. 7, pp. 2259–2301, 2006.

[10] P. L. Fackler, "MDPSOLVE a MATLAB toolbox for solving Markov decision problems with dynamic programming — user's guide," North Carolina State University, Tech. Rep., January 2011.

[11] B. Bhattacharya and S. S. Bhattacharyya, "Parameterized dataflow modeling for DSP systems," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2408–2421, October 2001.

[12] C. Shen, W. Plishker, H. Wu, and S. S. Bhattacharyya, "A lightweight dataflow approach for design and implementation of SDR systems," in *Proceedings of the Wireless Innovation Conference and Product Exposition*, Washington DC, USA, November 2010, pp. 640–645.

[13] S. Lin, Y. Liu, K. Lee, L. Li, W. Plishker, and S. S. Bhattacharyya, "The DSPCAD framework for modeling and synthesis of signal processing systems," in *Handbook of Hardware/Software Codesign*, S. Ha and J. Teich, Eds. Springer, 2017, pp. 1–35.