Design and Implementation of Adaptive Signal Processing Systems Using Markov Decision Processes

Lin Li*, Adrian E. Sapio*, Jiahao Wu*, Yanzhou Liu*, Kyunghun Lee*, Marilyn Wolf‡, Shuvra S. Bhattacharyya*† *University of Maryland, College Park, ECE Department, College Park, MD 20742, USA

Email: {lli12311, asapio, jiahao, yzliu, leekh3, ssb}@umd.edu †Tampere University of Technology, Dept. Electronics and Communications Engineering, Finland

‡ Georgia Institute of Technology, Georgia, USA

Email: {wolf}@ece.gatech.edu

Abstract

In this paper, we propose a novel framework, called Hierarchical MDP framework for Compact System-level Modeling (HMCSM), for design and implementation of adaptive embedded signal processing systems. The HMCSM framework applies Markov decision processes (MDPs) to enable autonomous adaptation of embedded signal processing under multidimensional constraints and optimization objectives. The framework integrates automated, MDP-based generation of optimal reconfiguration policies, dataflow-based application modeling, and implementation of embedded control software that carries out the generated reconfiguration policies. HMCSM systematically decomposes a complex, monolithic MDP into a set of separate MDPs that are connected hierarchically, and that operate more efficiently through such a modularized structure. We demonstrate the effectiveness of our new MDP-based system design framework through experiments with an adaptive wireless communications receiver. [1]

[1]This paper has been accepted for publication in the Proceedings of the 2017 IEEE International Workshop on Signal Processing Systems. The official/final version of the paper is published on http://ieeexplore.ieee.org/Xplore.

I. INTRODUCTION

Modern signal processing applications impose increasing demands of adaptivity, flexibility and reconfigurability. On one hand, adaptive signal processing systems must adjust to dynamically-changing environmental conditions, system status or user requirements; on the other hand, the systems must often satisfy stringent constraints on energy-efficiency and real-time performance.

In this work, we apply Markov decision processes (MDPs) to address such challenges. MDPs have been used in many application areas as a foundation for dynamic determination of system configurations in stochastic environments. For example, Boutilier et al. propose factored MDPs as a method for compact representation of large, structured MDPs [1]. Benini et al. introduce a finite-state, abstract system model for power-managed systems [2]. However, the complexity of the MDP algorithms in general grows exponentially with increases in the size of the state space. Jonsson and Barto present an algorithm that performs hierarchical decomposition of factored MDPs to help alleviate this growth in complexity [3]. However, their development of hierarchical MDPs is focused on algorithms and theoretical analysis for state abstraction and MDP computation, and the connection to implementation of the hierarchical MDPs and application to real-world systems is not addressed. One objective of this paper is to help bridge this gap in the context of embedded signal processing systems.

In particular, in this work, we integrate the MDP schemes presented in [4] and [3]. This results in a novel approach to formulating MDPs for policy optimization in embedded signal processing systems with complex state spaces, and stringent implementation constraints. In our proposed design framework, we apply hierarchical MDPs to decompose the modeling of the application and embedded processing system into multiple MDPs. Each smaller MDP is formulated using an approach similar to that developed in [2]. We refer to this hybrid MDP approach as the *Hierarchical MDP approach for Compact System-level Modeling* (*HMCSM*).

To promote systematic derivation of embedded implementations using the HMCSM approach, we integrate the approach into the framework of dataflow-based design of signal processing systems and develop a dataflow-based framework for design and implementation of adaptive signal processing systems using HMCSM.

To demonstrate the efficiency and flexibility of the proposed design framework and the corresponding libraries and tools, we implement an adaptive wireless communication receiver

that dynamically optimizes its system configuration in response to changes in different use cases. As a key part of the receiver, the *channelizer* accounts for most of the computational complexity and energy consumption [5]. By adapting the configuration of the channelizer based on the communication scenario, we seek to optimize its energy efficiency while ensuring its correct functionality at any given time. We design an HMCSM MDP to perform this adaptation, and apply our dataflow-based MDP implementation framework to realize the resulting adaptive signal processing system on a state-of-the-art embedded platform.

II. Background

In this section, we elaborate on background in two key foundations of the contributions in this paper, MDP methods and dataflow-based design, that is relevant to development of our proposed HMCSM design framework.

A. MDP Methods

In Benini's work on MDP-based methods for system-level power management, the service provider, service requester, and power manager are defined as key system components. The policy is composed of a finite discrete sequence of decisions taken by the power manager. A generic deterministic stationary policy can be represented as a table with the rows representing all possible states and the columns representing all possible actions. The size of the policy table grows geometrically when the number of system states increases. As a result, the policies derived from the MDP techniques proposed in [2] are practical only for problems with relatively small numbers of system states.

A factored MDP only requires specification of the conditional probabilities with respect to dependent state variables, in contrast with traditional MDPs where the probabilities with respect to independent variables must be specified. The resulting modeling components are smaller in size and their policies are more compact compared to traditional MDPs. A more detailed and systematic introduction to factored MDPs can be found in [1].

As mentioned in Section I, hierarchical factored MDPs are explored by Jonsson and Barto [3]. They use a dynamic Bayesian network and a causal graph to identify relationships among state variables and construct a hierarchical MDP for a given policy optimization problem. In this

work, we build on these theoretical foundations of hierarchical factored MDPs, and apply this class of MDPs to design and implementation of adaptive signal processing systems.

B. Dataflow-based Modeling and Design

An important contribution of this work is the integration of MDP-based design methods into a model-based design framework based on dataflow models of computation. In the form of dataflow that we apply, signal processing applications are modeled as directed graphs, called dataflow graphs, in which vertices (actors) represent computations of arbitrary complexity, and edges represent first-in, first-out (FIFO) communication channels between actors.

In signal processing oriented dataflow models, special attention is given to the rates at which actors produce and consume data to and from their ports, respectively. These rates are referred to as the production rates and consumption rates of the associated actor ports or incident edges. Collectively, production rates and consumption rates are referred to as *dataflow rates*.

In this paper, we apply a form of dataflow called *parameterized synchronous dataflow* (*PSDF*) to demonstrate the model-based integration of the proposed MDP-based design techniques [6]. We use PSDF because it is useful in modeling dataflow graphs that have dynamically varying parameters. Quasi-static scheduling techniques have also been developed for these graphs that systematically derive *parameterized looped schedules* [7]. A parameterized looped schedule involves loops that iterate across subsets of actors, and have iteration counts that can be symbolic expressions in terms of static or dynamically-varying actor, edge or graph parameters. We apply parameterized dataflow due to its natural match with our objective of developing a model-based framework for adaptive signal processing. However, we envision that the framework can be adapted into modeling environments that are based on other parametric dataflow models, such as Boolean parametric dataflow [8] and the parameterized and interfaced dataflow meta-model [9]. Investigating such adaptations along with application of the distinguishing tools and analysis techniques available for such alternative models is an interesting direction for future work.

To implement PSDF-based models of adaptive signal processing systems, we apply the Lightweight Dataflow Environment (LIDE) [10]. LIDE is a software tool for dataflow-based design and implementation of signal processing systems. LIDE is based on a compact set of application programming interfaces (APIs) that is used for constructing and connecting dataflow actors, edges, and graphs. These APIs have been implemented in a variety of implementation languages.

In this work, we apply LIDE-C, which is based on C language implementation of the LIDE APIs.

III. Hierarchical MDP Approach for Compact System-level Modeling

The HMCSM framework is illustrated in Figure 1. At design time, the application functionality is modeled using dataflow techniques, as illustrated in the top right region of the figure. The design process also involves modeling the environmental and system-level dynamics, and reconfiguration process in the form of a hierarchical MDP, as illustrated by the part of Figure 1 that is labeled Hierarchical MDP Subsystem. As part of this modeling process, Markov models are created of both the processing demands imposed on the system by the application, and the dynamics of the processing system components. In a classical MDP formulation, these elements are combined into a single MDP. In this work, we additionally explore the use of Hierarchical MDPs in comparison to a single MDP to address the scaling problems that are well known to be a major weakness of classical MDPs (see Section I).

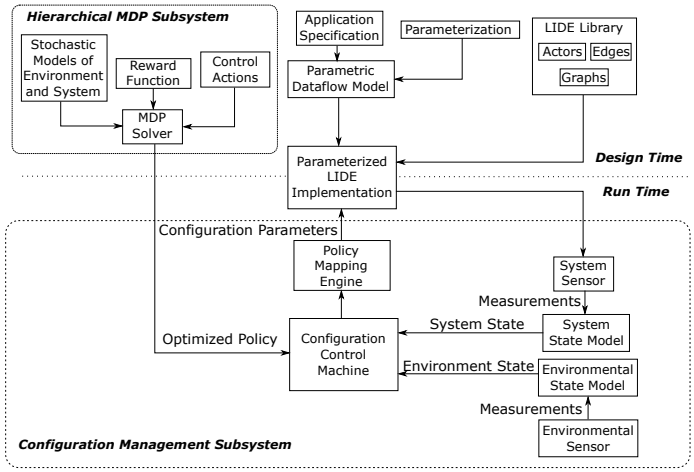

Fig. 1: An illustration of the HMCSM framework for design and implementation of adaptive signal processing systems.

The single MDP is transformed into a hierarchy of multiple MDPs by first factoring the elements in the MDP based on their stochastic interdependencies. Once the MDP has been factored, it can be decomposed into sub-problems that can be independently solved by multiple MDPs arranged in a hierarchy. For details on the processes involved in factoring and transforming the original MDP, we refer the reader to [3]. In the HMCSM framework, the factoring and

decomposition are carried out by hand, using knowledge of the application domain and the processing system. An interesting future direction for this work is the development of tools to help automate the factoring and decomposition process.

The Configuration Control Machine (CCM), shown in the lower (run time) portion of Figure 1, is used to manage dynamic system-level reconfiguration throughout operation of the embedded signal processing system. Here, by *dynamic reconfiguration*, we mean changes to any system-level parameters, including software, platform, and algorithmic parameters, that can be manipulated while the system is executing. At run-time, the CCM executes periodically and we refer to each periodic execution of the CCM as a *reconfiguration round*. During a reconfiguration round, the CCM determines, based on the current environmental state and system state, whether or not to perform a dynamic reconfiguration operation and the specific reconfiguration operation if reconfiguration is to be applied to the system. The blocks labeled System Sensor, System State Model, Environmental Sensor, and Environmental State Model represent measurements and models that are used by the CCM to determine the system and environmental state during a given reconfiguration round.

The block labeled *Control Actions*, in the design time portion, encompasses the set of possible reconfiguration operations that can be applied by the CCM in a given reconfiguration round. If A denotes the set of control actions, then the CCM can be viewed as an implementation of a function $P : S_e \times S_s \rightarrow A$, where S_e is the set of environmental states, and S_s is the set of system states. This function P is referred to as the *reconfiguration policy* in an adaptive signal processing system that is developed using the HMCSM framework. In HMCSM, the policy is applied at run-time, and derived at design-time. It is derived using an MDP Solver, which is a software module that automatically generates optimized policies from MDP model specifications.

The Policy Mapping Engine, shown near the center of Figure 1, translates control actions into updates to dynamic parameters in the embedded software that achieve the intended actions. These parameter updates are made by setting appropriate variables in an implementation of the application dataflow graph that is developed using LIDE (see Section II-B). This dataflow graph implementation is represented by the block labeled Parameterized LIDE Implementation, and the software tool that we use to construct this implementation is represented by the block labeled LIDE Library.

In the HMCSM framework, each control action is formulated in terms of specific changes

to specific parameters in the parameterized dataflow application model M. Execution of the control action A at run-time involves setting each parameter to the corresponding value such that subsequent operation of M will be performed using these new parameter settings. Operation continues with the new parameter settings until a new control action is applied to the system (in some subsequent reconfiguration round).

Besides the aforementioned Control Actions, the Stochastic Models of Environment and System and Reward Function are the other two main components in the formulation of an MDP in HMCSM (see Figure 1). These two components are developed by hand using well-established foundations of MDP modeling, along with domain knowledge of the targeted application. The Stochastic Models of Environment and System include, for each of the two models, the definition of the state space and the state transition matrix (STM). The STM is a stochastic matrix that defines the probability of the transition to the next state given the existing state, and conditioned on a given action. The Reward Function maps state-action pairs into *scores* that assess the utility of performing the associated control action during the given state. The approach for incorporating multidimensional design objectives into the scores produced by the reward function is provided in [4]. In our wireless communications case study, which we present in Section IV, the specific design objectives that we target are (a) energy consumption and (b) probability of packet collisions.

In summary, the HMCSM framework presented in this section provides a comprehensive methodology and supporting tools for design and implementation of adaptive embedded signal processing systems. The specific tools that we apply in our demonstration of the framework are MDPSOLVE [11] and LIDE, which correspond to the blocks labeled MDP Solver and LIDE Library, respectively, in Figure 1. However, the framework is not intended to be specific to these tools, and can readily be adapted to other tools for solving MDPs and implementing parametric dataflow graphs, respectively.

IV. Case Study

A. Background

In this section, we describe a detailed case study as an illustrative example of the methodology introduced in Section III. In addition to providing a demonstration of our proposed HMCSM

design framework on a practical, adaptive signal processing application, the case study also serves as a demonstration of a novel, application-specific, MDP-based system design.

B. Adaptive Receiver Architecture

Our case study is a wireless receiver for a Low Power Wide Area Network (LPWAN) used in a "Smart Cities" Internet of Things (IoT) application [12]. We propose an adaptive LPWAN receiver that dynamically adjusts the system bandwidth continually, and periodically transmits the new bandwidth setting to end nodes through the use of a downlink beacon. This case study implements the physical layer signal processing for such an adaptive receiver. The implemented architecture consists of reconfigurable channelizer and baseband processing algorithms.

In this work, we build on the MDP-based dynamically reconfigurable channelizer presented in [4]. Our work on this case study and the underlying HMCSM framework developed in this paper goes beyond the developments of [4] by (1) our application of hierarchical MDP techniques; (2) integrating MDP-based control with model-based methods for signal processing system implementation; (3) expanding the role of the MDP to control not just the channelizer configuration, but also the system bandwidth; and (4) comparing a baseline MDP framework with our proposed hierarchical MDP approach that reduces the model size.

C. Application Specification

Figure 2 shows a block diagram of the adaptive receiver architecture that is investigated in this case study. The channelizer is used to separate the incoming wideband signal into multiple data streams, where each of the data streams is associated with a distinct channel. Each channel is then oversampled for symbol timing recovery, and then processed by a Generalized Likelihood Ratio Test (GLRT) detector, which looks for the transmission preamble. Once a detection is successful, a matched filter demodulator recovers the transmitted data and confirms it with an error detection function (e.g., CRC32).

We compare the relative merits of two separate MDP schemes, labeled MDP-I and MDP-II. These two schemes are illustrated together in Figure 2. MDP-I consists of a single MDP employed for control of both the dynamic bandwidth as well as the channelizer processing configuration. MDP-II splits the modeling into two MDPs arranged in a hierarchy. These two MDPs are labeled MDP-II-a and MDP-II-b.

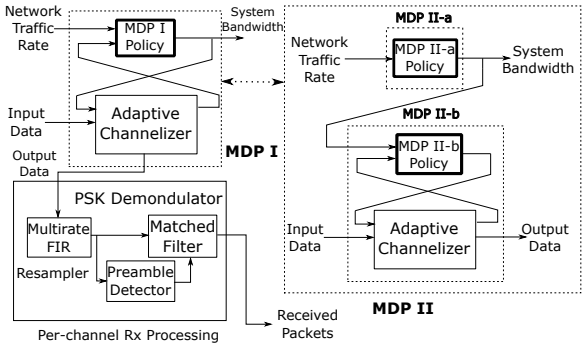

Fig. 2: Block diagram of receiver signal processing and two MDP schemes.

The channelizer can be implemented by one of two options, as detailed in [4] — a bank of M polyphase decimators and mixers (labeled as $\mathrm{DCM}[1], \mathrm{DCM}[2], \ldots, \mathrm{DCM}[M]$) or a Discrete Fourier Transform Filter Bank (labeled as DFTFB). Both options are capable of meeting the processing requirements of the channelizer subsystem. However, they do so using different algorithmic means and the relative efficiency of one option compared to the other is highly platform- and implementation-dependent. Using our proposed MDP-based approach, the reconfiguration policy is systematically optimized for the exact processing characteristics of a specific platform (e.g., measured power consumption).

D. PSDF Model

We implement the channelizer subsystem as a PSDF design with dynamic parameters. Our PSDF model of the channelizer system is illustrated in Figure 3. Here, M is a static parameter of the application that represents the total number of available channels.

At run-time, the MDP-generated reconfiguration policy determines how many channels to enable, based on the data rate, and whether to apply DCM processing or DFTFB processing. These policy decisions are used to manipulate a set of dynamic parameters $\{Y_1, Y_2, \ldots, Y_M\}$ that is associated with M distinct DCM actors $\mathrm{DCM}[1], \mathrm{DCM}[2], \ldots, \mathrm{DCM}[M]$, respectively. The policy decisions are also used to manipulate a parameter Z that is associated with an actor labeled DFTFB, which represents DFTFB processing on all of the enabled channels. Each of these $(M+1)$ parameters is binary valued. In particular, for each $i \in \{1, 2, \ldots, M\}$ if DCM processing is enabled, and the ith channel is enabled, then $Y_i = 1$ and $Z = 0$. Conversely, if DFTFB processing is enabled, then $Z = 1$, and $Y_i = 0$ for all i.

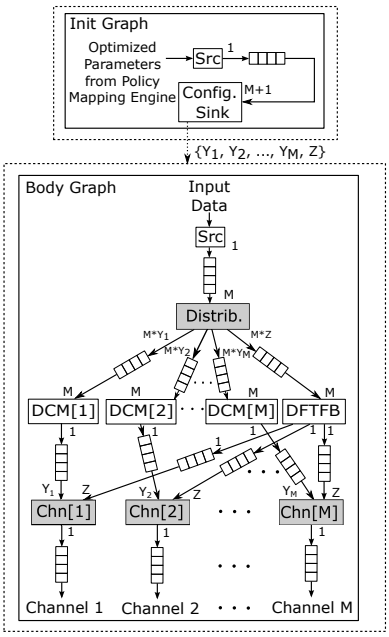

Fig. 3: PSDF specification of channelizer subsystem.

After each reconfiguration round, updated values of $\{Y_1, Y_2, \ldots, Y_M\}$ and Z are propagated to the adaptive channelizer flowgraph through the Init Graph, as illustrated in Figure 3. The channelizer flowgraph is parameterized in terms of these $(M+1)$ dynamic parameters and encapsulated within the Body Graph, as shown in the figure. The Init Graph and Body Graph are modeling constructs in PSDF that are used, respectively, for reconfiguration functionality (determination and propagation of new parameter values), and core signal processing functionality associated with an application.

The Config. (configuration) Sink actor in Figure 3 is a special actor in LIDE that is used to propagate parameter updates from the Init Graph to the Body Graph in PSDF-based implementations. The shaded actors in the Body graph are dynamically parameterized actors. The expressions next to the input and output ports represent the consumption and production rates associated with the ports, respectively. The Distrib. (distributor) actor takes its input data and distributes copies of it to the appropriate subset of DCM/DFTFB actors depending on the current values of the dynamic parameters in the graph.

The actors labeled $\mathrm{Chn}[1], \mathrm{Chn}[2], \ldots, \mathrm{Chn}[M]$ in Figure 3 copy data from their input ports to their output ports based on which (if any) input ports have nonzero consumption rate. These

actors generate samples on each of the M output channels of the channelizer subsystem.

V. Experiments

As mentioned in Section IV, we compare the relative merits of a conventional, monolithic MDP approach (labeled MDP-I) with our proposed hierarchical MDP approach (labeled MDP-II) to adaptive signal processing system design.

1) MDP-I: In MDP-I, the (single) MDP state space consists of the instantaneous rate of uplink packets generated by all end nodes, and the configuration of the basestation processor (number of channels enabled and channelizer configuration).

The action space consists of the next configuration of the basestation processor (number of channels enabled and channelizer configuration).

The STM for the packet generation rate is computed a priori from the traffic rate measured in a design-time simulation. The STM for the processing system is obtained from the dynamics of the implementation on the reference platform.

The MDP reward metric is a linear combination of two competing metrics: the probability of packet collision and the power consumption expended in basestation processing. This linear combination of conflicting metrics tasks the MDP with finding an optimal trade-off at any time based on relative weightings that are provided by the designer for the two metrics.

2) MDP-II: In MDP-II, the control task is split across two hierarchically arranged MDPs: MDP-II-a and MDP-II-b. MDP-II-a is used to determine the optimal number of channels to enable at a given time, while being agnostic to what processing configuration is actually used in the receiver to implement that configuration. MDP-II-b is used to determine the optimal processing configuration for an exogenously specified number of channels.

A. Implementation

The configurable components of the processing system were implemented using the method detailed in Section III, and deployed to a Raspberry Pi 3 Model B computing platform. Each of the available configurations was run in a test mode, and the average processing power consumed was measured and tabulated as shown in Table I. The device we used for measuring power consumption is the Tektronix Keithley Series 2280 Precision Measurement DC Power Supply. The measurements were then provided as input to the MDP models, and used to generate control policies using these empirically measured characteristics of the processing system.

TABLE I: Platform measurements.

Processing Configuration	Number of Channels Processed	Average Power
DCM	1	1.4406 W
DCM	2	1.4781 W
DCM	3	1.5203 W
DCM	4	1.5660 W
DCM	5	1.6025 W
DCM	6	1.6524 W
DCM	7	1.7013 W
DCM	8	1.7453 W
DFTFB	8	1.6754 W

B. Simulation Results

We performed a physical layer signal processing simulation in MATLAB to generate uplink traffic from $1{,}000$ end nodes. The simulation compared the use of the two (adaptive) MDP Schemes and also (non-adaptive) cases where only the fixed channelizer configurations were used. Each run of the simulation generated results of the form shown in Figure 4. In this figure, the MDP (MDP-I) is in control of the number R of receiver channels enabled. Note that when the traffic rate is high, the system increases R in order to reduce packet collisions.

The two competing MDP schemes, MDP-I and MDP-II, produced the same control policy. However, a key difference is that the hierarchical MDP reduced the model size from 1.63MB to 265kB, which is a factor of 6.1 times smaller than the original size. This reduction becomes especially relevant when housing the MDP model and policy generation code on the processing platform itself. Such an *embedded MDP* realization is useful because it allows the MDP and generated policy to adapt dynamically based on learned characteristics of the operating environment. Integration of embedded MDP techniques into the HMCSM framework is a useful direction for future work.

MDP-II also provides a benefit in the execution time required for the MDP solver to compute the optimal policy. We applied the MATLAB-based open source solver called MDPSOLVE [11]. The execution times were measured as 294ms for MDP-I, 50.8ms for MDP-II-a and 41.5ms for MDP-II-b. As a result, in a deployment with a fixed processing system that periodically recomputes the control policy in response to a changing external environment, the hierarchical MDP scheme reduces the solver time from 294ms to 50.8ms, which is a factor of over 5.7X

smaller.

All of the simulation runs that we carried out are compared in Figure 5. Different simulations for the MDP-generated policy were carried out using different relative weightings for the two performance metrics. Simulations were also carried out for fixed signal processing configurations. The square-shaped points in Figure 5 correspond to DCM-based channelizer operation with the number of enabled channels ranging from 1 to 8. The fixed configuration DFTFB case is represented by the triangle-shaped point in the top left region of the figure. In the context of all of the design points evaluated and the two performance metrics, we see from the figure that the MDP-based approach generates a Pareto front. This demonstrates that the adaptive reconfiguration capability provided by the MDP leads to better performance in comparison with fixed configurations for each of the available signal processing algorithms.

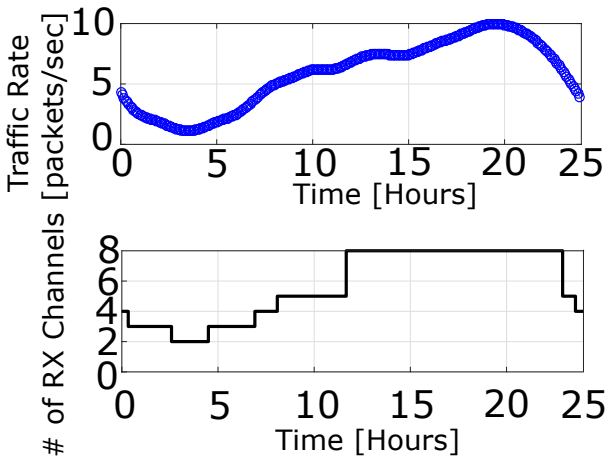

Fig. 4: Simulation results for MDP-I.

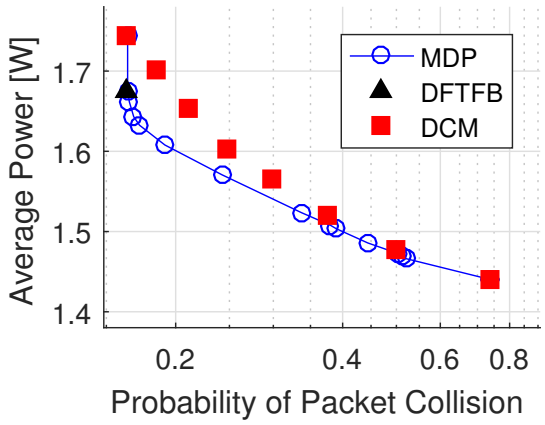

Fig. 5: Comparison among MDP-generated policies and fixed-configuration designs.

VI. Conclusions

In this paper, we have developed the Hierarchical MDP framework for Compact System-level Modeling (HMCSM) and its application to design and implementation of adaptive embedded signal processing systems. HMCSM provides a structured design methodology that integrates model-based design for embedded signal processing in terms of dataflow methods; MDP formulation using compact, hierarchical models; optimal policy generation from these models at

design time; and dynamic, system-level reconfiguration at run time. The framework enables systematic derivation of system-level reconfiguration policies that are based on application-specific functional requirements and operational constraints. Useful directions for future work include adapting the HMCSM framework for use with other parametric dataflow models (beyond PSDF), development of tools to help automate the factoring and hierarchical decomposition of MDPs, and integration of embedded MDP techniques into HMCSM.

ACKNOWLEDGMENTS

This research was sponsored in part by the US National Science Foundation (CNS1514425 and CNS151304).

REFERENCES

[1] C. Boutilier, R. Dearden, and M. Goldszmidt, "Exploiting structure in policy construction," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995, pp. 1104–1111.

[2] L. Benini, A. Bogliolo, G. A. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 6, pp. 742–760, June 1999.

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

[4] A. Sapio, M. Wolf, and S. S. Bhattacharyya, "Compact modeling and management of reconfiguration in digital channelizer implementation," in *Proceedings of the IEEE Global Conference on Signal and Information Processing*, December 2016.

[5] S. J. Darak, A. P. Vinod, R. Mahesh, and E. M.-K. Lai, "A reconfigurable filter bank for uniform and non-uniform channelization in multi-standard wireless communication receivers," in *Proceedings of the International Conference on Telecommunications*, 2010, pp. 951–956.

[6] 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.

[7] ——, "Quasi-static scheduling of reconfigurable dataflow graphs for DSP systems," in *Proceedings of the International Workshop on Rapid System Prototyping*, Paris, France, June 2000, pp. 84–89.

[8] V. Bebelis, P. Fradet, A. Girault, and B. Lavigueur, "BPDF: A statically analyzable dataflow model with integer and Boolean parameters," in *Proceedings of the International Workshop on Embedded Software*, 2013, pp. 1–10.

[9] K. Desnos, M. Pelcat, J. Nezan, S. S. Bhattacharyya, and S. Aridhi, "PiMM: Parameterized and interfaced dataflow meta-model for MPSoCs runtime reconfiguration," in *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2013, pp. 41–48.

[10] 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.

[11] 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.

[12] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.