mTunes: Efficient Post-Silicon Tuning of Mixed-Signal/RF Integrated Circuits Based on Markov Decision Process

Manzil Zaheer and Fa Wang
ECE Department, Carnegie Mellon University
Pittsburgh, PA 15213
{manzil, fwang1}@cmu.edu

Chenjie Gu
Strategic CAD Labs, Intel Corporation
Hillsboro OR 97124
chenjie.gu@intel.com

Xin Li
ECE Department, Carnegie Mellon University
Pittsburgh, PA 15213
xinli@cmu.edu

ABSTRACT

Uncertainty prevails in IC manufacturing and circuit operation. In particular, process variability has a huge impact on circuit performance, especially for mixed-signal/RF circuits, leading to unacceptable yields. Additionally, environmental uncertainties, such as temperature fluctuation and channel variation, further deteriorate performances in field. To combat variability, circuits are often made reconfigurable by adding tunable knobs to recover circuit performance in the post-manufacturing stage. However, as the number of knobs increases, knob tuning becomes challenging due to the huge search space. In fact, knob-tuning policies can have an observable impact on final performance and power consumption. In this paper, we propose mTunes, a method based on the Markov decision process for dynamically choosing the "right" knob tuning sub-routine from a pre-defined set achieving a balance between performance and power constraints. The proposed method has been applied to a reconfigurable RF front-end design, showing 60% improvement in yield compared to static tuning policies.

1. INTRODUCTION

Variability is prevalent during IC manufacturing and circuit operation. In particular, process variation keeps increasing as the transistor size scales [1]–[2]. This can be attributed to physical and fabrication limitations such as "under-etching uncertainties, variations of effective transistor dimensions, severe channel length modulation due to higher electric fields, and channel dopant fluctuations" [3]. Moreover, from the perspective of analog circuits, the technology scaling is accompanied by adverse effects such as lower transistor output impedances, reduced transistor linearity and limited voltage headroom. [4]. In post-manufacturing, the IC will be interconnected with a variety of peripherals (such as different boards, memory cards, wireless channels), each affecting the circuit performance in a different manner. Temporal variations such as temperature and voltage fluctuations further deteriorate circuit performance and power consumption.

This has made the design of high-performance analog/mixed-signal/RF circuits extremely challenging as their performances heavily depend on matched devices and differential signal paths, and are especially sensitive to even the slightest perturbation. Moreover, a major use of analog/mixed-signal/RF circuits is in wireless mobile products. In such mobile applications, circuit design with low power consumption is vital, thus compounding the problem. As a result, increasing process variation of smaller process nodes, together with environmental variation and noise, leads to diminishing yields and reliabilities of chip designs. Ensuring that the performances of an analog/mixed-signal/RF system meet the design specifications has become increasingly challenging.

To handle these challenges, circuit designers have traditionally resorted to conservative circuit design approaches, trading off some performance for higher yields and better variation tolerance. Recently, two methods have been proposed to combat these challenges, namely *post-production performance calibration* [5]–[7] and *self-healing* [8]–[10]. They allow more aggressive circuit specifications to be achieved, while maintaining expectations of high yield. Fundamentally, both methods employ a set of tunable components (or knobs) in the design for calibrating performance of the system. In post-production performance calibration, after manufacturing the knobs for failing dies are fine-tuned so that their performances meet the design specifications. This way, some dies that would have been directly thrown away under the traditional analog testing procedure can now be salvaged, thereby recovering yield. In self-healing, along with tunable knobs, on-chip sensors are also added. Such a design constantly monitors the system performance and employs subsequent on-chip correcting mechanisms by tuning the knobs using an optimisation algorithm. This not only allows the system to correct for process variability, but also provides enhanced resilience to environmental variations.

In both methods, the crux lies in the proper tuning of the knobs. With increasing system complexity, more knobs are added for often conflicting purposes (e.g., performance and power). Since the search space in the knob space is huge, global optimisation over all knob values is infeasible. Rather, a single knob or a group of small number of knobs are tuned sequentially as part of the system-level calibrations. In this case, the order in which tuning of knobs is carried out is very important as typically the objective function is not convex. Because it can happen that tuning one knob first can lead the system into a valley of local optima thus compromising achievable performance, power or pass/fail in the testing phase, whereas tuning in another order leads to a better optima. Thus proper selection of the order in which to tune the knobs is extremely crucial.

In this paper, we propose to borrow the idea of *Markov Decision Process* (MDP) [11]–[14] from the machine learning community and develop a new algorithm, *mTunes* to overcome the aforementioned technical challenges. In particular, we consider a self-adaptive design where the tuning knobs are adjusted sequentially, and we focus on the problem of how to order the knob tuning sequence for minimum power consumption under performance constraints. This is achieved by first modelling the uncertainty (e.g., from process variation) and the dynamics of

DAC '15 June 07 - 11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2744873

tuning individual knobs using a Markov model. With this, mTunes dynamically selects the "best" subsequent knob to tune based on configuration of current knob settings, using a reward function specifically designed to balance the conflicting performance specifications and power consumption. Mathematically, mTunes selects the knob to optimise the reward function, in expectation, and this can be efficiently done using a dynamic programming formulation. Such design is different from conventional methods for system-level calibration which employs a fixed, deterministic ordering of the tuning knob selection. It is also important to note that mTunes introduces few design overhead because it does not require any additional sensor/information other than value of all the current knob settings. Our experimental results in Section 4 demonstrate that mTunes can increase the yield by up to 60% while maintaining similar power requirements, compared to a fixed order knob tuning.

Outline: The rest of the paper is organized as follows. We begin by formally describing the problem statement and provide an overview of MDP for tackling the problem in Section 2. This lays the foundation for the proposed knob tuning algorithm selection method, for which detailed implementation details are given in Section 3. In Section 4, we demonstrate that efficacy of the proposed approach through simulation showing more than 2× increase in yield. Finally, we conclude in Section 5.

2. PROPOSED APPROACH

2.1 Problem Formulation

In this paper, we target the problem of knob tuning in a reconfigurable circuit so that it meets the design specification. Let there p performance metrics in the specification; denote by m_j the value of performance metric and by Λ_j specification of that metric. Also by m_0 denote the power consumed by the circuit. Also let there be l knobs in the circuit and denote by k_i the value taken by knob i from some set Ω_i for each $i = 1, 2, ..., l$.

The goal is to tune all the knobs $(k_1, k_2, ..., k_l)$, such that all performance metrics meet the specifications, i.e. $\forall j : m_j \in \Lambda_j$, while minimizing the power. As stated earlier, global optimisation of all the knobs $(k_1, k_2, ..., k_l)$ is infeasible due to extremely large search space. Rather the designers provide a bag of algorithm $\mathcal{A}$ for taking the knobs and each algorithm $a \in \mathcal{A}$ tunes one knob or some small number of knobs at a time. Also we have a budget, T, of how many algorithms we can choose to run. So the objective becomes to pick these algorithm in correct order - the importance of correct order under limited budget can be understood by considering the following example.

Consider a simple problem where we have two knobs and two tuning algorithms - the first (second) tuning algorithm sweeps the first (second) knob, keeping the second (first) knob constant to find the configuration which minimizes some cost function. Also we are given a budget of running 2 algorithms at most. Suppose the cost function has contour plot as shown in Fig. 1. Also assume that we start by setting knobs to $(1, 1)$. Now if our policy is to tune knob 1 first, then we will get stuck at a local optima. Whereas by first tuning the knob 2, we can reach the state of global optima. With larger number of knobs and tuning algorithm, the problem enters the realm of high-dimensionality and selection of correct tuning algorithms order becomes non-trivial.

2.2 Assumptions

In order to tackle the problem described, we make an assumption that for a given circuit, all the performance metrics depend only on the knob configuration, i.e. for each chip there exists function $f_j : \bigotimes_i \Omega_i \to \mathbb{R}$ such that:

$$m_j = f_j(k_1, k_2, ..., k_l). \tag{1}$$

This means knowledge of $(k_1, k_2, ..., k_l)$ completely describes the state of the system. Under this setting notice that running any tuning algorithm $a \in \mathcal{A}$ will result in the following transition:

$$(k_1, k_2, ..., k_l) \xrightarrow{a} (k_1^+, k_2^+, ..., k_l^+). \tag{2}$$

So the next state of the system only depends on the present knob setting and the action taken, not on any historical configuration or operations. Thus the setup obeys Markov property perfectly. We can exploit this Markovity, in selecting the best order in which to execute the tuning algorithms.

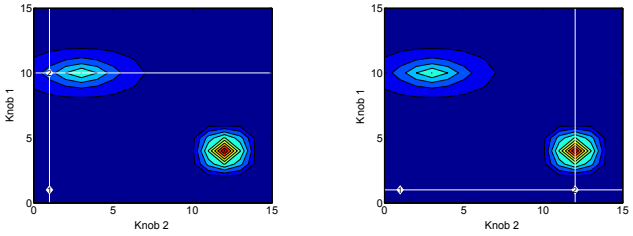
However, note that the functions $\{f_j\}$ will be different for each chip, attributed to many sources of variability as enlisted in the introduction. Due to this variability, the next state into which the system will transition upon application of a tuning algorithm becomes uncertain across chips. For this purpose we borrow the idea of MDP from machine learning community. A MDP is simply a tool to plan efficiently if the results of actions executed are uncertain. which we will describe next.

2.3 Markov Decision Process

Firstly, MDP is a Markov model. A Markov modelling scheme is applied to scenarios where the environment evolves in a memoryless fashion. In other words, it can be used to model random environments that change states according to some transition rule that only depends on the current state. Second aspect of MDP is the possibility to make decisions. Basically, in many situations we have the flexibility to respond to the random environment by taking suitable actions. However, the environments response to the action can have randomness in it. In this setting, the Markov property translates to the state transition rule/function probabilistically specifying the next state of the environment to depend on its current state *and* the action.

Formally, MDP is a 4-tuple $\Xi = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$, defined as:

- $\mathcal{S}$ is the set of states in the system. At every instant, the system must be one of the state $s \in \mathcal{S}$. For the problem in consideration configuration of tuning knobs will form the state space as will be explained in Section 3.
- $\mathcal{A}$ is set of actions which the agent can execute and at each time step we can choose to take one of the action. This will correspond to the bunch of tuning algorithms at our disposal.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a set of reward functions $\{r_1, ..., r_t\}$. They specify the expected instantaneous reward as a function of the state transition and action taken.



(a) Tuning knob 1 first (b) Tuning knob 2 first

Figure 1: A synthetic example to motivate the fact that ordering of tuning algorithm is important. Here an exemplary cost function is shown, and for the given starting point if we first tune the knob 2 then we can reach a global optima whereas tuning knob 1 first will lead us to get stuck at a local optima.

- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ is the conditional transition probability of moving to a new state $S_{t+1} \in \mathcal{S}$ given current state $S_t \in \mathcal{S}$ and action $A_t \in \mathcal{A}$. This can be represented as a tensor $\tau(s, a, s^+) = \mathbb{P}(S_{t+1} = s^+ | S_t = s, A_t = a)$.

Under assumptions of this model, the system evolves as follows: at each time point, the system is in a particular state, s, an action a is taken and there is a transition to another state s^+. However, we require that the state transition depend only upon s and a. In addition, s and a only give probabilistic information about what the resulting state will be according to $\mathcal{T}$. Also suppose the system has a finite lifetime T, in which case it is termed as a finite horizon MDP. Having described all the quantities of interest, the intention is to find a policy, mapping states to actions, that will maximizes our over-all rewards. So, starting from some state $S_0 = s$ at time $t = 0$, we want to find the policy $\pi^* = (\pi_0, \pi_1, ..., \pi_{T-1}), \pi_t \in \mathcal{A}^{|\mathcal{S}|}$, that indicates the algorithm to run when we are in state S_t at time t so as to maximize our expected rewards, i.e.

$$\pi^* = \underset{\pi \in \mathcal{A}^{|\mathcal{S}|T}}{\operatorname{argmax}} \mathbb{E}\left[\sum_{t=0}^{T-1} \mathcal{R}_t(S_t, \pi_t(S_t), S_{t+1}) \middle| S_0 = s, \pi\right]. \tag{3}$$

We will next describe a technique for efficiently determining this optimal policy given the correct model Ξ.

2.4 Value Iteration

We will provide here details for finite horizon value iteration for Markov Decision Process for completeness, because its details are not as easily accessible as its infinite horizon counterpart.

Begin by defining, the Q-function as:

$$Q_i(s, \pi_{i:T-1}) = \mathbb{E}\left[\sum_{t=i}^{T} r_t(S_t, \pi_t(S_t), S_{t+1}) \,\middle|\, S_i = s, \pi_{i:T-1}\right]. \tag{4}$$

By simple manipulations and using linearity of expectation, we can get the recursion:

$$\begin{aligned} Q_i(s, \pi_{i:T-1}) &= \mathbb{E}\left[r_i(s, \pi_i(s), S_{i+1}) + Q_{i+1}(S_{i+1}, \pi_{i+1:T-1})\right] \\ &= \sum_{s^+} \tau(s, \pi_i(s), s^+) \left(r_i(s, \pi_i(s), s^+) + Q_{i+1}(s^+, \pi_{i+1:T-1})\right). \end{aligned} \tag{5}$$

Then the value function defined as:

$$\mathbf{v}_t^*(s) = \max_{\pi \in \mathcal{A}^{|\mathcal{S}|(T-t)}} Q_t(s, \pi_{t:T-1}), \tag{6}$$

along with (5) yields the famous Bellman equation [11]:

$$\mathbf{v}_t^*(s) = \max_{\pi_t(s) \in \mathcal{A}} \sum_{s^+} \tau(s, \pi_t(s), s^+) \left(r_t(s, \pi_t(s), s^+) + v_{t+1}^*(s^+)\right). \tag{7}$$

Now to solve for the value function, a dynamic program can be constructed as shown in

Thus we can see that computational complexity of the value-iteration algorithm is in general quadratic in the number of states and linear in the number of actions. Commonly, the tensor τ is sparse and if there are on average a constant number of next states with non-zero probability (as will be in our case) then the cost per iteration is linear in the number of states and linear in the number of actions. We will now use this machinery in finding the optimal order for executing tuning knob algorithm.

3. IMPLEMENTATION DETAILS

In this section, we describe the proposed mTunes method. In particular, we first explain the modelling using MDP and obtaining the optimal policy. Subsequently, we delve into integration of this policy for knob tuning algorithm selection.

Algorithm 1 Finite Horizon Value Iteration

Input: The Markov model Ξ
Output: Optimal policy π^*

```
Initialize v*_T(s) ← 0 for all s ∈ S.
for t = T ↓ 1 do
    for s ∈ S do
        for a ∈ A do
            Q(s, a) = Σ_{s+} τ(s, a, s+) (r_t(s, a, s+) + v*_{t+1}(s+))
        end for
        v*_t(s) = max_{a∈A} Q(s, a)
        π*_t(s) = argmax_{a∈A} Q(s, a)
    end for
end for
```

3.1 Markov Modelling

Often circuits, especially analog and mixed signal circuits, are represented by response surface models (RSM) and thus abstracting away the internal circuit details. This enables us to expedite design and analysis by providing a fast and accurate way to evaluate system performance. There are several known methods to build RSM for circuit performance, e.g. empirical model, regression model (polynomial,etc.), neural nets, fuzzy logic. [15]–[20]

In our proposed approach we first of all pick any desired RSM for each nominal performance metric of interest. Then using simulation data, we fit the nominal case model for all performance metrics, i.e. for each $j = 1, 2, ..., p$ we obtain:

$$m_j = \bar{f}_j(k_1, ..., k_l). \tag{8}$$

From this we can see that performance of the system can be completely described by the knob values, according to our assumption. Thus we can denote the state s of the system by $s = (k_1, ..., k_l)$ and $\mathcal{S} = \bigotimes_i \Omega_i$.

Now under "perfect" nominal conditions, application of any tuning algorithm will result in a deterministic behaviour. In other words, if the system is any state $s \in \mathcal{S}$ at time t, then application of any tuning algorithm $a \in \mathcal{A}$ will lead to some new state s^+ deterministically, i.e.

$$\mathbb{P}(S_{t+1} | S_t = s, A_t = a) = \begin{cases} 1 & \text{if } S_{t+1} = s^+ \\ 0 & \text{else.} \end{cases} \tag{9}$$

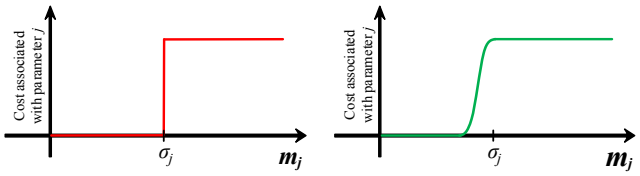
However, perfect nominal condition is never a reality, the chip will be subject to process and environmental variability. So, ideally each chip will have its own function $\tilde{f}_j$ for each performance metric. We approximate this to first order by claiming that, for all chips we have:

$$\tilde{f}_j = \bar{f} + \epsilon_j, \tag{10}$$

where ϵ_j are independent random variables capturing the variability. The statistics of ϵ_j can be obtained by running Monte-Carlo simulation and designer's advise can be used to make it more accurate.

Under this scenario, application of tuning algorithm will not result in deterministic behaviour. Because, different dies under different process variation and environmental condition can have different response $\tilde{f}_j$ and so the underlying optimisation in the tuning algorithm can lead to different knob configurations.

Thus $\mathbb{P}(S_{t+1} | S_t, A_t)$ can now have multiple non-zero entries. Now to estimate the tensor $\tau(s, a, s^+) = \mathbb{P}(S_{t+1} = s^+ | S_t = s, A_t = a)$, we use the first order approximation of (10) and the learnt statistics of ϵ_j. With this information, each of the entries of τ can be estimated using direct maximum likelihood estimation (MLE). Also we can



(a) Fixed policy (b) State dependent policy

Figure 2: Two possible cost functions – step and smooth – associated with a performance metric is shown. It is assumed that according to design specification the performance metric should be less than σ_j.

assume our favourite distribution form, like Gaussian Mixture Model (GMM) for $\tau(s, a, \cdot)$ and fit that using the Monte-Carlo simulation data.

3.2 Reward Function Design

As explained in the introduction, raison d'être of tunable knobs is to improve the yield by adjusting knobs so as to meet the specifications. The tuning should be performed such that chip should meet the specifications at the end. In terms of MDP framework, we should reward transitions such that we end up in a state/knob configuration where performance metrics meet the specifications. Also during the intermediate steps we do not care if specifications are met or not. This hints at setting $r_t \equiv 0$ for $t = 0, 1, ..., T-2$. Otherwise, if we provide instantaneous rewards, it can greedily take steps which end up in a local optima.

Next, once the specifications are met, it is of not much value to exceed them. Rather, it is much more useful to reduce power consumption as much as possible, while respecting the specifications. So, we choose the following reward at the end:

$$r_{T-1}(s, a, s^+) = r(s^+) = -f_0(s^+) - C \sum_{j=1}^{p} \mathbb{I}\{f_j(s^+) \notin \Lambda_j\} \qquad (11)$$

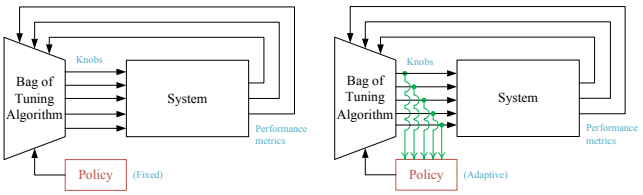
This reward function basically rewards more to enter a final state with a lower power requirement and penalize heavily if any of the performance metric does not meet the specification. Amount of penalty is controlled by the constant C.

However, such a reward function provides no indication about the system reaching close to specification boundary during the training procedure. By making the transition around specification from zero penalty to high penalty a smooth function, we get two fold benefits. Firstly, we can get an indication about reaching near the edge of feasible region. Secondly, this helps in leaving margin to accommodate fluctuations due to variability. For example, if for a performance metric m_j, the specification is $\Lambda_j = (-\infty, \sigma_j]$, i.e. the performance metric m_j has to be less than or equal to σ_j. In this example, the cost associated with m_j in (11) is depicted in Figure 2a. The corresponding smooth version is shown in Figure 2b. We will see that using the smoother variant leads to a better performance of mTunes in Section 4.

Now, we have specified the entire model Ξ for MDP. Using value iteration algorithm as described in Section 2.3, we can obtain the optimal policy π^*. This would conclude the off-line computations involved in mTunes. Next we discuss, how to use the obtained policy π^* in selection of sub-routines during the actual tuning of the chip.

3.3 System Integration

Our proposed method does not require much deviation from current paradigm of reconfigurable circuits. Currently, the reconfigurable systems are designed and tuned as depicted in



(a) Step (b) Smooth

Figure 3: A schematic representing the general architecture of reconfigurable circuits. We highlight that in proposed method, one can move from static fixed tuning policy to dynamic state dependent policy by just adding a feed-back from the present knob value.

Fig. 1a. The system will have l tunable knobs. There is a bunch of tuning algorithms, which control (a subset of) the knobs and receive feedback from the system in terms of some performance metrics, probably using some on-chip sensors. There is a pre-determined and fixed policy/order in which these algorithms are executed. The order is stored in the policy block. [8]–[9]

We propose to introduce a feedback to the policy block from the knob configurations and make the policy block contain the MDP based optimal policy π^*. This can be stored in as a look up table or implemented via digital logic. Our proposed approach work will work as follows. Based on the knob configuration and budget remaining, the policy block will look up the π_t^* to determine which tuning algorithm to run. Then this algorithm is executed. This will result in the knobs being changed. After the algorithm finishes, based on the new knob settings the process is repeated till the budget is exhausted. This procedure is summarized in Algorithm 2.

Algorithm 2 MDP Based Tuning Algorithm Selection

1: Initialize knobs to $s = (k_1^0, ..., k_l^0)$.
2: **for** $t = 1 \rightarrow T$ **do**
3: Find the algorithm to run, i.e. $a \leftarrow \pi_t(s)$.
4: Execute the tuning algorithm a. Within this tuning algorithm a, optimal knob values are chosen based on some subset (possibly all) of performance metrics.

$$(k_1, k_2, ..., k_l) \xrightarrow{a} (k_1^+, k_2^+, ..., k_l^+)$$

5: This cause a state transition to a new state $s = (k_1^+, k_2^+, ..., k_l^+)$.
6: **end for**

Note that our proposed approach is indifferent to the implementation and details of the various tuning algorithms in $\mathcal{A}$. In other words, the working mTunes does not depend on how the tuning algorithms select the optimal knobs, what is the cost function they use or which subset of performance metrics do they measure. Thus, showing the generality of our proposed approach.

4. EXPERIMENTS

4.1 Setup

In this section as a proof-of-concept, two versions – high-performance and low-power – of a reconfigurable RF front-end example designed in a GPDK 45nm CMOS process is considered. It is used to demonstrate the efficacy of the proposed method by comparing against a baseline, with a fixed order of executing tuning algorithms. The schematic of the RF front-end is shown in Fig. 4. It consists of two components: a tunable LNA and a tunable down-conversion mixer. The tunable LNA is a

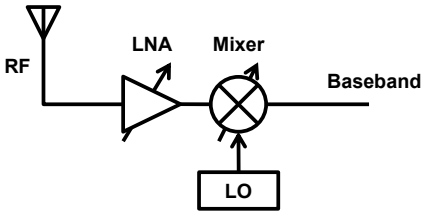


Figure 4: Schematic of a simple 2.4 GHz reconfigurable RF receiver

cascade of three stages where the bias current of each stage can be independently controlled by a 4-bit digital-to-analog converter (DAC). The tunable down-conversion mixer also includes two control knobs, each tuned by a 4-bit DAC. The two control knobs independently tune the bias current and load resistance of the mixer.

In total the system has $l = 5$ knobs, each taking values among 16 possibilities. There are $p = 3$ performances of interest for this RF front-end example, they are listed in Table 1 along with their design specifications for both low-power and high-performance versions.

We consider a budget of $T = 5$ tuning steps and $|\mathcal{A}| = 5$ tuning algorithms, one for each knob. The operation of each tuning sub-routine is straightforward: it sweeps one of the knobs, while keeping the rest fixed. Consequently, it selects the knob value that maximizes the following function of performance metrics of interest:

$$r(s) = -f_0(s) - 1000 \sum_{j=1}^{p} \mathbb{I}\{f_j(s) \notin \Lambda_j\}, \tag{12}$$

where $s = (k_1, ..., k_l)$. This function basically selects the knob value corresponding to lowest power consumption while meeting the specifications. If none of the knob values meet the specification, it adds a big penalty and still selects the knob setting consuming least power.

We compare the proposed approach, mTunes, with a baseline approach for knob tuning. The only difference in the two methods lies in deciding the next tuning step. In baseline approach, we assume that the tuning algorithm are invoked sequentially in the natural order, i.e. 1, 2, 3, 4, 5. While, in the proposed approach, to re-emphasize, we use an adaptive state dependent policy where the next tuning algorithm to invoke is decided based on the current state. Next we describe exactly how we learnt the policy for the circuit in consideration.

4.2 Fitting mTunes

For this example, we use a full quadratic RSM for all the performance metrics. Since there are $l = 5$ knobs, the model has $\binom{l+2}{2} = 21$ coefficients for each performance metric. We fit this model by carrying simulations under nominal conditions for 1315 knob configurations. These knob configurations are obtained by using D-optimal design of experiment.

Next, we run 1315 Monte-Carlo simulation with random knob configurations to obtain the statics of ϵ_j. Using this and first order approximation of the variations in circuit performance metric, we estimated the transition probability tensor $\hat{\tau}$ by direct maximum likelihood estimation. With this the MDP model, $\hat{\Xi}$ is complete, and we run the value iteration algorithm with reward function in (11) with $C = 1000$ and its smoothed counter part to obtain the optimal policies.

Table 1: Specifications

Metric	Low-Power	High-Performance
Voltage gain (VG)	≥ 35 dB	≥ 36 dB
Noise Figure (NF)	≤ 7 dB	≤ 6 dB
Input referred 1 dB compression point (1dBCP)	≥ -39 dB	≥ -36 dB

Table 2: Results for Low-Power Version

Metric	Baseline	mTunes I	mTunes II
Voltage gain	38.12 dB	36.46 dB	36.59 dB
Noise figure	6.910	6.867	6.887
1dBCP	-39.09 dB	-33.45 dB	-33.85
Power	22.23 mW	20.98 mW	20.97 mW
Yield	40.04%	100%	100%

Table 3: Results for High-Performance Version

Metric	Baseline	mTunes I	mTunes II
Voltage gain	38.17 dB	36.06 dB	36.88 dB
Noise figure	5.921	5.784	5.782
1dBCP	-38.86 dB	-32.64 dB	-34.01 dB
Power	25.74 mW	26.36 mW	26.41 mW
Yield	0.00%	86.07%	91.52%

4.3 Evaluation

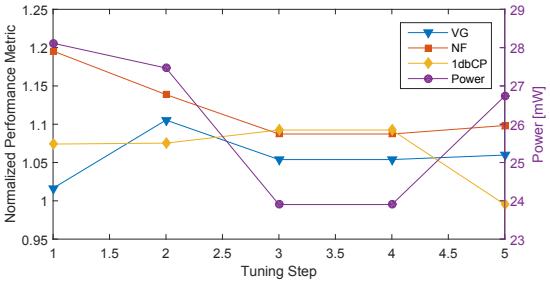
The two methods are evaluated on 512 dies by Monte-Carlo simulations for both low-power and high-performance versions. Additionally, for mTunes we have two implementations, first one using policy obtained by the step reward function (11) and the second using its smoothed variant.

The tuning procedure is carried out as follows: We begin by setting initially all knob values to the minimum possible setting. Then we have to decide which knob to be tuned. In case of baseline, we simply start by tuning the first knob, while for mTunes we use the policy to determine the knob to be tuned. Then as described in the setup, the tuning is carried out by sweeping through all the values of the chosen knob and obtaining the performance metrics at each setting. We try to find a knob value that minimizes power while meeting the specifications on other performance metrics. If none of them meets the specification, add large penalties as can be seen from (12). By choosing the best knob value thus, we transition into a new state. The process is repeated 5 times.

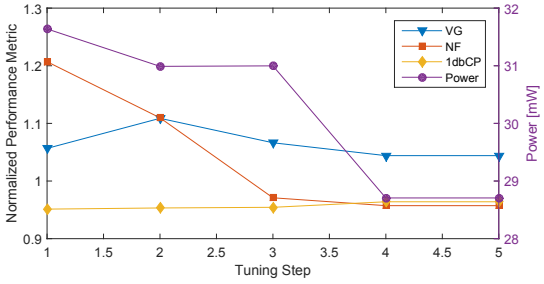
The average performance metrics, power consumption and yield obtained at the end of tuning procedure are reported in Table 2 and 3. Note that we selected objective function for tuning algorithms to be same as the reward function of MDP for a fair comparison between the two.

4.4 Analysis

Studying Table 2 and 3 reveals several important observations. First, it appears that baseline unnecessarily exceeds design specifications for some of the metrics and for others cannot even meet the specifications. In particular, the baseline achieves more than required gain and thus struggle in obtaining high input referred 1 dB compression point. Also, such behaviour leads to higher power consumption and low yield. To explore this further we plotted a representative the trajectory of performance metrics evolution after each training step in Figure 5 for both the baseline and proposed mTunes. Second, but more importantly, the proposed mTunes is able to balance between all the performance metric and achieve high yield, upto 100%, while at the same time reducing



(a) Baseline



(b) mTunes

Figure 5: As an example, a plot showing trajectory of performance metric evolution with each tuning step using baseline and mTunes for one chip is presented. Here normalized version is plotted to fit different performance metrics in the same scale. The voltage gain (VG) should be larger than 1 while noise figure (NF) and 1dBCP should be smaller than 1 to meet the specifications.

power consumption by upto 5% compared to the baseline.

Finally, it is worth mentioning that all the computations for mTunes, e.g. the model training, evaluating the optimal policy etc. are done in advance. Hence, it does not increase the tuning/test cost after the chip is manufactured.

5. CONCLUSION

In this paper, a novel mTunes approach based on MDP is proposed for efficient knob tuning in reconfigurable analog and mixed-signal circuits. Built upon MDP, mTunes can generate a dynamically optimal tuning scheme to improve parametric yield with low tuning/testing cost. The proposed mTunes approach has been validated on a 2.4GHz reconfigurable RF receiver front-end, through extensive simulations. Our experimental results demonstrate that mTunes consistently out-performs conventional fixed order tuning approach by more than 2x improvement in yield. The aforementioned yield increase can be directly translated to a valuable cost reduction for high volume analog and mixed-signal circuit production.

As future works we wish to reduce number of samples required by employing tricks like Bayesian Model Fusion [6] to incorporate data from past measurement or from similar system obtained through previous simulations or previous stepping of similar products. Also we would like to handle the problem of state space explosion due to curse of dimensionality. One possible way could be through embedding into reproducing kernel Hilbert space [21].

6. ACKNOWLEDGEMENTS

This work has been supported in part by Intel Corporation and National Science Foundation under contract CCF-1316363.

7. REFERENCES

[1] X. Li, J. Le, and L. T. Pileggi, *Statistical Performance Modeling and Optimization*. Now Publishers, 2007.

[2] "International Technology Roadmap for Semiconductors," Semiconductor Industry Associate, Tech. Rep., 2011.

[3] M. Onabajo and J. S.-Martinez, *Analog circuit design for process variation-resilient systems-on-a-chip*. Springer, 2012.

[4] A.-J. Annema, B. Nauta, R. van Langevelde, and H. Tuinhout, "Analog circuits in ultra-deep-submicron CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 1, pp. 132–143, 2005.

[5] N. Kupp, H. Huang, P. Drineas, and Y. Makris, "Post-production performance calibration in analog/RF devices," in *IEEE ITC*, pp. 1–10, 2010.

[6] X. Li, F. Wang, S. Sun, and C. Gu, "Bayesian Model Fusion: A statistical framework for efficient pre-silicon validation and post-silicon tuning of complex analog and mixed-signal circuits," in *IEEE/ACM ICCAD*, pp. 795–802, Nov 2013.

[7] J. Liaperdos, A. Arapoyanni, and Y. Tsiatouhas, "A test and calibration strategy for adjustable RF circuits," *Analog Integrated Circuits and Signal Processing*, vol. 74, no. 1, pp. 175–192, 2013.

[8] S. M. Bowers, K. Sengupta, K. Dasgupta, and A. Hajimiri, "A fully-integrated self-healing power amplifier," in *RF Integrated Circuits Symposium*, pp. 221–224, 2012.

[9] K. Sengupta, K. Dasgupta, S. M. Bowers, and A. Hajimiri, "On-chip sensing and actuation methods for integrated self-healing mm-wave CMOS power amplifier," in *Microwave Symposium Digest (MTT)*, pp. 1–3, 2012.

[10] B. Sadhu, M. A. Ferriss, A. S. Natarajan, *et al.*, "A linearized, low-phase-noise VCO-based 25-GHz PLL with autonomic biasing," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 5, pp. 1138–1150, 2013.

[11] R. Bellman, "A Markovian decision process," *Journal Of Mathematics And Mechanics*, vol. 6, pp. 679–684, 1957.

[12] R. A. Howard, "Dynamic programming and Markov processes," 1960.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *arXiv preprint cs/9605103*, 1996.

[14] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2009.

[15] A. I. Khuri and J. A. Cornell, *Response surfaces: designs and analyses*. CRC press, vol. 152, 1996.

[16] C. Guardiani, P. Scandolara, J. Benkoski, and G. Nicollini, "Yield optimization of analog ics using two-step analytic modeling methods," *Solid-State Circuits, IEEE Journal of*, vol. 28, no. 7, pp. 778–783, Jul 1993.

[17] J. Wang, *Response surface modeling for analog and mixed-signal design*. ProQuest, 2008.

[18] X. Li and H. Liu, "Statistical regression for efficient high-dimensional modeling of analog and mixed-signal performance variations," in *ACM DAC*, pp. 38–43, 2008.

[19] A. Khawas, A. Banerjee, and S. Mukhopadhyay, "A response surface method for design space exploration and optimization of analog circuits," in *VLSI (ISVLSI), IEEE Computer Society Annual Symposium on*, pp. 84–89, July 2011.

[20] M. Farooq Anjum, I. Tasadduq, and K. Al-Sultan, "Response surface methodology: A neural network approach," *European Journal of Operational Research*, vol. 101, pp. 65–73, 1997.

[21] S. Grünewälder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton, "Modelling transition dynamics in MDPs with RKHS embeddings," in *ICML*, vol. 1, pp. 535–542, 2012.