Adaptive Post-Silicon Tuning for Analog Circuits: Concept, Analysis and Optimization

Xin Li, Brian Taylor, YuTsun Chien and Lawrence T. Pileggi
Department of ECE, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
{xinli, briant, yutsunc, pileggi}@ece.cmu.edu

ABSTRACT

The well-known Pelgrom model [14] has demonstrated that the variation between two devices on the same die due to random mismatch is inversely proportional to the square root of the device area: $\sigma \sim 1/\mathrm{sqrt}(Area)$. Based on the Pelgrom model, analog devices are sized to be large enough to *average out* random variations. Importantly, with CMOS scaling, variations due to random doping fluctuations are making it exceedingly difficult to control device mismatches by sizing alone; namely, the devices have to be made so large that the benefits of CMOS scaling are not realized for analog and RF circuits. In this paper we propose a novel *post-silicon tuning* methodology to reduce random mismatches for analog circuits in sub-90nm CMOS. A novel *dynamic programming* algorithm is incorporated into a fast Monte Carlo simulation flow for statistical analysis and optimization of the proposed tunable analog circuits. We apply the proposed post-silicon tuning methodology to several commonly-used analog circuit blocks. We demonstrate that with the post-silicon tuning, device mismatch *exponentially* decreases as area increases: $\sigma \sim \exp(-\alpha \cdot Area)$.

1. INTRODUCTION

As integrated circuit (IC) technologies scale to 65nm and beyond, process variations become increasingly critical and make it continually more challenging to create a reliable, robust design with high yield [1]. Process variations can be classified into two broad categories: inter-die variations and intra-die variations. Inter-die variations model the common/average variations across the die, while intra-die variations model the individual, but spatially correlated, local variations (e.g., random device mismatches) within the same die. Among all sources of variations, the random mismatches due to doping fluctuations are expected to become dominant within the next few technology generations [2], as shown in Figure 1. Such large-scale variations must be carefully considered within today's IC design flow.

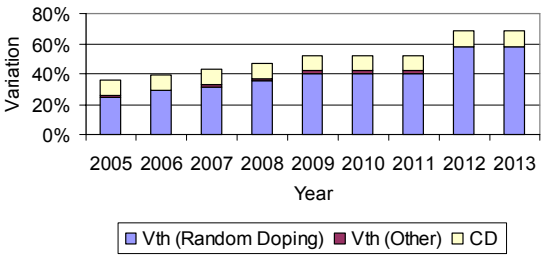

Figure 1. Process variations in future IC technologies [2].

During the past two decades, various statistical design methodologies have been proposed for analog circuits [3]-[7]. The key idea of these methods is to accurately predict random performance distributions and then leave sufficient performance margins to accommodate large-scale process variations. With scaling of CMOS below 90nm, the traditional statistical design methodologies attempt to reserve larger performance margins than ever before, thereby making it extremely difficult (or even infeasible) to achieve a high-performance circuit design. For this reason, the idea of post-silicon tuning has been proposed and successfully applied to various applications. For example, adaptive supply voltage and adaptive body bias are two widely-used techniques to reduce delay and leakage variations for digital circuits [8]-[10].

Analog circuits, however, are substantially different in nature. Most analog circuit designs (e.g., differential pair, switched-capacitor amplifier, etc.) are ratio-based [23]; namely, their behaviors depend on the ratio between two analog devices. These analog circuits are designed to be robust to inter-die variations, but they are extremely sensitive to device mismatches. Moreover, analog layouts are based on regular structures, such as concentric layout [23], which control systematic variations and make random fluctuations the dominant source of mismatch. Therefore, reducing random mismatches for analog devices (not only for transistors but also for resistors, capacitors, etc.) is a top priority for today's analog IC design [11].

The traditional approach for reducing random mismatches is to utilize large devices. According to the well-known Pelgrom model [12]-[13], the standard deviation of random mismatch is inversely proportional to the square root of the device area: $\sigma \sim 1/\mathrm{sqrt}(Area)$. Namely, if device area is increased by 100x, mismatch is only reduced by 10x. This fundamental limitation prevents analog circuits from further scaling to achieve smaller area, faster speed and lower power. The challenging problem is how to more effectively reduce random mismatches such that smaller devices can be used to achieve better performance.

In this paper we propose an adaptive post-silicon tuning approach to reduce random device mismatches. Instead of over-sizing analog devices, we propose a methodology that would decompose each device into N fingers and adaptively select the best-matched M ($M \leq N$) fingers based on post-silicon measurement. Such a post-silicon tuning was previously applied to several analog design examples [14]-[15] where simple brute-force search is used to find the optimal configuration. The objective of this paper is to develop a generalized methodology for post-silicon tuning that can be applied to a broad range of application domains. To do so, we will systematically analyze and optimize tunable analog designs, and demonstrate the substantial benefit offered by post-silicon tuning as compared to simple sizing following the Pelgrom model.

An important contribution of this paper is to propose a *dynamic programming* (DP) algorithm to select the best-matched fingers based on post-silicon measurement. To optimally select M fingers out of N ($M \leq N$) candidates, the number of possible combinations increases exponentially with N, thereby making such a discrete selection problem non-trivial to solve. The proposed dynamic programming partitions the complicated optimization problem into multiple, interacted sub-problems.

Instead of directly searching all N fingers, our sub-problem is defined to optimally select j fingers out of i ($j \le i$) candidates where i is initially set to 1 and it is iteratively increased to N. The sub-problem is solved once and its answer is saved, thereby avoiding the work of re-computing the answer every time when the sub-problem is encountered.

In addition, we propose to utilize *quantization* to efficiently lump many similar configurations together as a single DP state, thereby further reducing the computational complexity. As will be demonstrated by the numerical examples in Section 5, even for small-size problems (N = 10~14), the proposed dynamic programming algorithm achieves 10~20x speed-up compared with a brute-force search.

We further incorporate the proposed dynamic programming algorithm into a fast Monte Carlo analysis flow to efficiently predict the performance variations of tunable analog circuits. Note that such a statistical analysis problem cannot be easily solved using most existing techniques [16]-[19]. These existing methods assume continuous variations of uncertain parameters, while our proposed adaptive post-silicon tuning is discrete in nature.

Our statistical analysis demonstrates that if the adaptive post-silicon tuning is applied, device mismatch *exponentially* decreases as area increases: $\sigma \sim \exp(-\alpha \cdot Area)$. For example, a 1.4μm (width) x 50nm (length) NFET with post-silicon tuning shows the same mismatch variation as a 4×10^5μm (width) x 50nm (length) NFET without post-silicon tuning in a commercial 65nm CMOS process!

The remainder of this paper is organized as follows. In Section 2 we propose two analog circuit examples for adaptive post-silicon tuning. In Section 3, we develop a dynamic programming algorithm to optimally select M fingers out of N candidates for mismatch minimization. The proposed dynamic programming is further incorporated into a fast statistical analysis flow in Section 4. The efficacy of the proposed post-silicon tuning methodology is demonstrated by several numerical examples in Section 5. Finally, we conclude in Section 6.

2. ADAPTIVE POST-SILICON TUNING

We use two analog design examples (i.e., a differential pair and a switched-capacitor amplifier) to illustrate the basic concept of our proposed adaptive post-silicon tuning. These two circuit examples rely on transistor matching and capacitor matching, respectively. It should be noted, however, that the proposed post-silicon tuning methodology can be applied to many other analog applications where device matching is critical.

2.1 Tunable Differential Pair

Shown in Figure 2 is the simplified circuit schematic of a traditional differential pair [23]. It utilizes the symmetric topology to make the performance (e.g., offset voltage) insensitive to inter-die variations. However, random device mismatches make the circuit asymmetric and, hence, introduce offset voltage. In general, the transistors of a differential pair must be sufficiently large so that the offset voltage can be minimized.

An example of tunable differential pair is shown in Figure 3. The entire differential pair is decomposed into N branches, where each branch can be independently turned on/off by applying the proper digital controlling signal to switch the tail current. Based on post-silicon measurement, M ($M \le N$) branches will be adaptively selected to minimize the random mismatch.

If M branches $\{S_1, S_2, ..., S_M\}$ are selected where S_i is the index of the i-th selected branch, the input-referred offset voltage can be represented as [23]:

$$V_{OS} = \frac{1}{M} \cdot \sum_{i=1}^{M} V_{OS,Si} \quad (1)$$

where $V_{OS,Si}$ denotes the input-referred offset voltage of the S_i-th branch.

Since $\{V_{OS,i};\ i = 1, 2, ..., N\}$ are caused by random mismatches, they are typically modeled as independent, zero-mean random variables [12]-[13]. In this case, if all branches are selected without post-silicon tuning (i.e., $M = N$), it is easy to verify that the standard deviation of the offset voltage is inversely proportional to the square root of N: $\sigma_{OS} \sim 1/\mathrm{sqrt}(N)$ [22]. This result is referred to as the well-known Pelgrom model [12]-[13]. In Section 3, we will show how one can achieve a much smaller offset voltage by adaptively selecting M ($M \le N$) branches via post-silicon tuning.

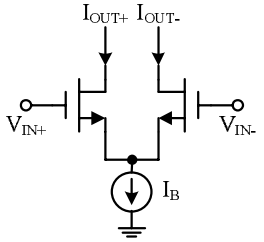

Figure 2. Simplified schematic of a traditional differential pair.

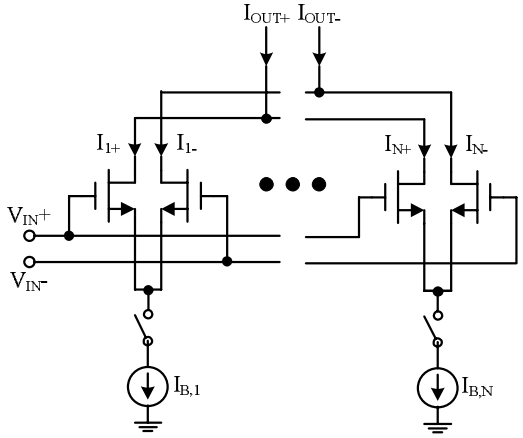

Figure 3. Simplified schematic of a tunable differential pair.

2.2 Tunable Switched-Capacitor (SC) Amplifier

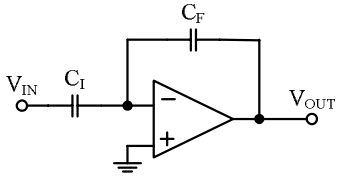

Figure 4. Simplified schematic of a traditional SC amplifier.

Figure 4 shows the simplified circuit schematic of a traditional SC (switched-capacitor) amplifier. For simplicity we assume that the operational amplifier in Figure 4 is ideal. The gain of the SC amplifier is determined by the ratio between the two capacitors: C_I and C_F. We further assume that C_I equals C_F and the SC amplifier has a unit gain. The random mismatch between these two capacitors is one of the major sources of gain error.

Our proposed tunable SC amplifier is shown in Figure 5. Both

C_I and C_F are decomposed into N fingers. Based on post-silicon measurement, M ($M \le N$) fingers will be adaptively selected to minimize the random mismatch.

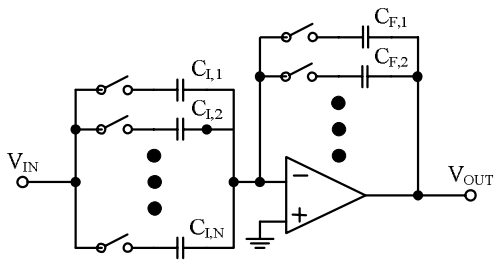

Figure 5. Simplified schematic of a tunable SC amplifier.

If M fingers $\{S_I,1, S_I,2, ..., S_I,M\}$ and $\{S_F,1, S_F,2, ..., S_F,M\}$ are selected for C_I and C_F respectively, the variations (i.e., the deviation from the mean value) of C_I and C_F can be expressed as:

$$\Delta C_I = \sum_{i=1}^{M} \Delta C_{I,Si} \tag{2}$$

$$\Delta C_F = \sum_{i=1}^{M} \Delta C_{F,Si} \tag{3}$$

where $\Delta C_{I,Si}$ and $\Delta C_{F,Si}$ denote the capacitance variations of the S_i-th selected fingers for C_I and C_F respectively. The average capacitor mismatch is defined as:

$$\Delta C_{MIS} = \frac{1}{M} \cdot \left(\sum_{i=1}^{M} \Delta C_{I,Si} - \sum_{i=1}^{M} \Delta C_{F,Si} \right) \tag{4}$$

Note that Eqn. (4) has a slightly different form than that in (1). What remains is the modeling and optimization method that can be used to minimize the mismatch as expressed in (4).

3. DISCRETE OPTIMIZATION FOR TUNABLE ANALOG CIRCUITS

The key problem posed by the proposed post-silicon tuning methodology is how to optimally select the best configuration to minimize the matching error. Such an optimization problem can be stated as follows: Given $\{V_{OS,i}; i = 1, 2, ..., N\}$ for the tunable differential pair or $\{\Delta C_{I,i}, \Delta C_{F,i}; i = 1, 2, ..., N\}$ for the tunable SC amplifier, select M branches/fingers out of N ($M \le N$) candidates such that the absolute value of the mismatch defined in (1) or (4) is minimized. For this optimization problem, M can be either pre-determined or variable. If M is a variable, we should select the optimal M value to achieve the minimal mismatch. This requires exploration of all possible values for M to select the optimal scenario. Therefore, in what follows, we focus on the situation where M is variable, since any other optimization with a pre-determined M is a sub-problem of this general case.

The aforementioned optimization problem is discrete in nature. To solve it, one straightforward approach is to enumerate all possible combinations (referred to as *brute-force search* in this paper). However, the total number of all possible configurations exponentially increases with N, thereby quickly making the computation infeasible. Theoretically, it can be proven that our discrete optimization problem for mismatch minimization is *NP-hard*. Namely, any algorithm that exactly solves the problem must require exponential runtime in worst case.

Motivated by this observation, we propose a *dynamic programming* [24] approach to search for the optimal configuration that yields the minimal matching error. The essence of the proposed dynamic programming is to partition the complicated discrete optimization problem into multiple, interacted sub-problems. Instead of directly searching all N branches/fingers, our sub-problem is defined to optimally select j branches/fingers out of i ($j \le i$) candidates where i is initially set to 1 and it is iteratively increased to N. The sub-problem is solved once and its answer is saved, thereby avoiding the work of re-computing the answer every time when the sub-problem is encountered. In addition, we propose to utilize *quantization* to lump many similar configurations together as a single DP state, thereby further reducing the computational complexity. In what follows, we first develop the dynamic programming algorithm for the tunable differential pair in Figure 3, and then extend it to optimize the tunable SC amplifier in Figure 5.

3.1 Dynamic Programming for Differential Pair

A. Mathematic Formulation

Our proposed dynamic programming follows the standard formulation described in [24]. It consists of two major components: (1) a finite set Ω that contains a number of quantized matching error values and (2) a three-dimensional table $T(i, j, k)$ that saves all possible DP states (i.e., matching errors in our application).

The finite set Ω is used to quantize matching errors such that similar error values are approximated as a single numerical number. It, in turn, allows us to lump many configurations with similar matching errors as a single DP state.

Given the offset voltage for each branch $\{V_{OS,i}; i = 1, 2, ..., N\}$ and M selected branches $\{S_1, S_2, ..., S_M\}$, we define the matching error as:

$$Err_{OS} = \sum_{i=1}^{M} V_{OS,Si} \ . \tag{5}$$

Eqn. (5) is identical to (1) except for a scaling factor M. It is easy to verify that for any $M \in \{1, 2, ..., N\}$ the matching error in (5) is bounded by:

$$B_L = \sum_{V_{OS,i}<0} V_{OS,i} \tag{6}$$

$$B_U = \sum_{V_{OS,i}>0} V_{OS,i} \tag{7}$$

where B_L and B_U represent the lower bound and the upper bound, respectively. Discretizing the interval $[B_L, B_U]$ yields the finite set:

$$\Omega = \{B_L, B_L + h, \cdots, B_U - h, B_U\} \tag{8}$$

where h is the step size. For example, if $B_L = -1$, $B_U = 1$ and $h = 0.5$, then $\Omega = \{-1, -0.5, 0, 0.5, 1\}$.

Given the set Ω in (8), we quantize $\{V_{OS,i}; i = 1, 2, ..., N\}$ by mapping each $V_{OS,i}$ to the nearest element in Ω. It, in turn, yields:

$$\{V_{OS,i}^{\Omega}; i = 1,2,\cdots,N\}. \tag{9}$$

For example, if $\{V_{OS,1} = -0.3, V_{OS,2} = 0.9\}$ and $\Omega = \{-1, -0.5, 0, 0.5, 1\}$, then $\{V_{OS,1}^{\Omega} = -0.5, V_{OS,2}^{\Omega} = 1\}$.

The second key component of the proposed dynamic programming is a three-dimensional table $T(i, j, k)$, where $i, j \in \{1, 2, ..., N\}$ and $k \in \Omega$. Note that the index k can be a rational or real (not integer) number, depending on the discretization in (8):

- $T(i, j, k) = 1$ (true) if and only if $\exists A \in 2^{\{1, 2, ..., i\}}$ such that:

$$|A| = j \tag{10}$$

$$\sum_{i \in A} V_{OS,i}^{\Omega} = k \tag{11}$$

where $|A|$ stands for the size of the set A and $2^{\{1, 2, ..., i\}}$ denotes the power set of $\{1,2,...,i\}$ (i.e., the collection of all subsets of $\{1,2,...,i\}$). For instance, $2^{\{1, 2\}} = \{\{\}, \{1\}, \{2\}, \{1,2\}\}$.

- $T(i, j, k) = 0$ (false) otherwise.

The three-dimensional table $T(i, j, k)$ contains all possible matching errors when selecting j branches from $\{1, 2, ..., i\}$. Starting from $i = 1$, we recursively find the best-matched j branches out of i ($j \le i$) candidates, save the answer in $T(i, j, k)$, and increase i until $i = N$. As i eventually reaches N, $T(N, j, k)$ provides all possible matching errors when selecting j branches out of N candidates. Similar tables have been widely used to solve many other dynamic programming problems [24]. Next, we will demonstrate how to efficiently fill the table $T(i, j, k)$ for our proposed post-silicon tuning problem.

Creating $T(i, j, k)$ involves two major steps: (1) initialization and (2) recursive iteration. The first initialization step is to fill in all table entries for $j = 1$. This step is trivial, since we only consider the cases where a single branch is selected.

$$T(i,1,k) = 1 \quad \left(\begin{array}{c} i = 1,2,\cdots,N \\ k = V_{OS,1}^{\Omega}, V_{OS,2}^{\Omega}, \cdots, V_{OS,i}^{\Omega} \end{array} \right). \tag{12}$$

Next, during the second step, we need to create a recurrence relation that allows us to iteratively fill in all other entries of the table $T(i, j, k)$. Note that $T(i, j, k) = 1$ if and only if any of the following two conditions is satisfied:

- $T(i–1, j, k) = 1$, i.e., selecting j branches from $\{1, 2, ..., i–1\}$ yields the error value k and the i-th element $V_{OS,i}^{\Omega}$ will not be selected for $T(i, j, k)$ to be true.
- $T(i–1, j–1, k–V_{OS,i}^{\Omega}) = 1$, i.e., selecting $j–1$ branches from $\{1, 2, ..., i–1\}$ yields the error value $k–V_{OS,i}^{\Omega}$ and the i-th element $V_{OS,i}^{\Omega}$ will be selected for $T(i, j, k)$ to be true.

Based on this observation, we conclude the following recurrence relation:

$$T(i,j,k) = T(i-1,j,k) \vee T\left(i-1, j-1, k - V_{OS,i}^{\Omega}\right) \tag{13}$$

where $\vee$ stands for the logic operator *OR*. Given (12) and (13), one can iteratively fill in the three-dimensional table $T(i, j, k)$, thereby yielding the matching error values for all possible configurations. During this process, a list of index values $\{S_1, S_2, ..., S_j\}$ can be saved for each table entry that is 1 (true), if one wants to know which branches are selected for $T(i, j, k)$ to be true.

After the table $T(i, j, k)$ is available, the final step is to search for all entries that are 1 (true) and scale the matching error in (5) back to the offset voltage in (1). For example, if $T(N, j, k) = 1$, meaning that selecting j branches out of N candidates results in the matching error k, the corresponding offset voltage is k/j. Algorithm 1 summaries the proposed dynamic programming algorithm for the tunable differential pair.

Algorithm 1: dynamic programming for differential pair

(1) Start from a given set of $\{V_{OS,i}; i = 1, 2, ..., N\}$ and a given step size h.
(2) Calculate the lower bound and upper bound using the equations (6)-(7).
(3) Create the finite set Ω in (8).
(4) Map $\{V_{OS,i}; i = 1, 2, ..., N\}$ to $\{V_{OS,i}^{\Omega}; i = 1, 2, ..., N\}$ in (9).
(5) Initialize the table $T(i, j, k)$ based on (12).
(6) Iteratively fill in all other entries of the table $T(i, j, k)$ using the recurrence relation in (13).
(7) For any $j \in \{1, 2, ..., N\}$ and $k \in \Omega$, if $T(N, j, k) = 1$, calculate the corresponding offset voltage $V_{OS} = k/j$ based on the definition in (1).
(8) Select the best configuration that yields the minimal offset voltage.

Algorithm 2: adaptive control for the quantization step size *h*

(1) Start from a given step size h.
(2) Set $r = 1$.
(3) Apply Algorithm 1 to estimate the minimal offset voltage V_{OS}^{r}, where the superscript r stands for the estimation result from the r-th iteration.
(4) If the estimated offset voltage value is unchanged between two successive iterations, i.e.,

$$\frac{\left|V_{OS}^{r} - V_{OS}^{r-1}\right|}{\left|V_{OS}^{r}\right|} \le \varepsilon \tag{14}$$

where ε is a pre-defined error tolerance, then stop. Otherwise, $r = r+1$, $h = h/2$ and return Step (3).

Algorithm 1 is based on a given step size h. In practice, the value of h can be adaptively controlled for a given accuracy requirement. Starting from a large step size h, h should be iteratively reduced (e.g., divided by 2) if the error is not sufficiently small. Algorithm 2 outlines a simplified algorithm for adaptive step control.

In summary, we have proposed a dynamic programming algorithm to optimally select M branches out of N candidates such that the random mismatch is minimized for the tunable differential pair in Figure 3. The proposed dynamic programming applies quantization to *approximate* the solution of the original discrete optimization problem that is NP-hard. In what follows, we will theoretically analyze the computational complexity of the proposed algorithm and demonstrate why it is much more efficient than a simple brute-force search.

B. Computational Complexity

The computational complexity of the proposed dynamic programming is mainly determined by the size of the table $T(i, j, k)$, where

$$\left|T(i,j,k)\right| = N^2 \cdot \left(B_U - B_L\right)/h \tag{15}$$

denotes the size of the table $T(i, j, k)$. To determine the relation between $|T(i, j, k)|$ and N, we need to further know how B_U, B_L and h depend on the value N.

Studying (6), one can easily notice that the lower bound B_L is bounded by:

$$B_L \ge -\sum_{i=1}^{N} \left|V_{OS,i}\right|. \tag{16}$$

Since the offset voltages $\{V_{OS,i}; i = 1, 2, ..., N\}$ are typically modeled as independent random variables [12]-[13], the standard deviation of the random variable $|V_{OS,1}| + |V_{OS,2}| + ... + |V_{OS,N}|$ is proportional to the square root of N [22], yielding:

$$B_L \sim -\sqrt{N}. \tag{17}$$

The relation between B_U and N can be similarly derived as:

$$B_U \sim \sqrt{N}. \tag{18}$$

On the other hand, for a given accuracy requirement, the step size h depends on the final matching error Err_{OS} defined in (5). For example, if Err_{OS} is small, a small h should be automatically selected by Algorithm 2 to keep the relative error smaller than ε in (14). As will be demonstrated by the numerical examples in Section 5.1, for the tunable differential pair in Figure 3, the

matching error exponentially decreases as N increases. Therefore, given a fixed error tolerance ε in (14), the step size h is an exponential function of N:

$$h \sim e^{-\alpha \cdot N} \tag{19}$$

where α is a positive real number. Substituting (17)-(19) into (15) gives the final computational complexity of the proposed dynamic programming:

$$O\left(N^2 \cdot \sqrt{N} \cdot e^{\alpha \cdot N}\right). \tag{20}$$

The computational complexity in (20) exponentially increases with N. However, as will be demonstrated by our numerical examples in Section 5.1, the proposed dynamic programming is still much faster than the brute-force search that has a complexity of $O(2^N)$. We observe that the proposed dynamic programming algorithm achieves 10x speed-up, even if N is as small as 14. We expect that the efficiency of the proposed dynamic programming will be more pronounced as N further increases.

3.2 Dynamic Programming for SC Amplifier

A. Mathematic Formulation

Algorithm 3: dynamic programming for SC amplifier

(1) Start from a given set of $\{\Delta C_{I,i}, \Delta C_{F,i}; i = 1, 2, ..., N\}$ and a given step size h.
(2) Apply the dynamic programming described in Algorithm 1 to build the three-dimensional tables $T_I(i, j, k_I)$ and $T_F(i, j, k_F)$ for ΔC_I in (2) and ΔC_F in (3), respectively.
(3) For any possible values of j, k_I and k_F, if $T_I(N, j, k_I) = 1$ and $T_F(N, j, k_F) = 1$, calculate the corresponding capacitor mismatch $C_{MIS} = (k_I - k_F)/j$ based on the definition in (4).
(4) Select the best configuration that yields the minimal capacitor mismatch.

Algorithm 1 can be extended to solve the capacitor matching problem in (4). The basic idea is to first apply the same dynamic programming to calculate the capacitance variations for ΔC_I in (2) and ΔC_F in (3) respectively. Next, all possible combinations of ΔC_I and ΔC_F are checked and the optimal configuration with the smallest mismatch C_{MIS} (defined in (4)) is selected. Algorithm 3 summarizes the major steps for the dynamic programming of the SC amplifier. Although Algorithm 3 assumes a given step size h, the value of h can be iteratively determined by an adaptive control scheme similar to Algorithm 2.

B. Computational Complexity

The computational cost of Algorithm 3 is dominated by three major tasks: (1) creating the table $T_I(i, j, k_I)$, (2) creating the table $T_F(i, j, k_F)$, and (3) checking all combinations between $T_I(N, j, k_I)$ and $T_F(N, j, k_F)$ to calculate all possible values of the capacitor mismatch. As discussed in Section 3.1B, the computational complexities of the first two tasks are respectively determined by:

$$\left|T_I\left(i, j, k_I\right)\right| \sim N^2 \cdot \sqrt{N}/h \tag{21}$$

$$\left|T_F\left(i, j, k_F\right)\right| \sim N^2 \cdot \sqrt{N}/h \tag{22}$$

where $|T_I(i, j, k_I)|$ and $|T_F(i, j, k_F)|$ denote the size of the tables $T_I(i, j, k_I)$ and $T_F(i, j, k_F)$, respectively.

The computational complexity of the third task is determined by:

$$\left|T_I\left(N, j, k_I\right)\right| \cdot \left|T_F\left(N, j, k_F\right)\right| \sim \left(N \cdot \sqrt{N}/h\right)^2 = N^3/h^2 \tag{23}$$

where $T_I(N, j, k_I)$ and $T_F(N, j, k_F)$ are both two-dimensional tables, since their first-dimension index is fixed to N.

On the other hand, given a fixed error tolerance ε in (14), the step size h exponentially decreases as N increases, similar to the case discussed in Section 3.1B. If h is expressed as the exponential function in (19), the overall computational complexity of Algorithm 3 is dominated by (23):

$$O\left(N^3 \cdot e^{2\alpha \cdot N}\right). \tag{24}$$

As will be demonstrated by the numerical examples in Section 5.2, the computational complexity of Algorithm 3 is much lower than the complexity of the brute-force search which is close to 2^{2N} in this particular application. The proposed dynamic programming algorithm achieves 20x speed-up, even if N is as small as 10.

4. STATISTICAL ANALYSIS FOR TUNABLE ANALOG CIRCUITS

To quantitatively demonstrate the substantial benefit offered by the proposed post-silicon tuning, the statistical performance distribution with post-silicon tuning must be estimated and compared with the well-known Pelgrom model when no post-silicon tuning is applied. For this purpose, we propose a fast statistical analysis flow for tunable analog circuits in this section. The proposed statistical analysis flow is facilitated by a combination of controlled random sampling and dynamic programming.

Note that our statistical analysis problem cannot be easily solved using most existing techniques [16]-[19]. These existing methods assume continuous variations of uncertain parameters, while the proposed adaptive post-silicon tuning is discrete in nature.

The proposed fast Monte Carlo analysis flow is shown in Algorithm 4. Instead of directly drawing random samples from a random number generator, the proposed fast Monte Carlo analysis creates sampling points from a controlled random sequence (i.e., Latin hypercube sampling) such that high estimation accuracy can be achieved by using a small number of sampling points [20]-[21]. The key idea of Latin hypercube sampling is to fill the high-dimensional random space based on the given probability density function $pdf(\bullet)$ and make the sampling point distribution close to $pdf(\bullet)$ as much as possible. Next, for each Latin hypercube sampling point, the dynamic programming algorithm proposed in Section 3 is applied to find the minimal mismatch. Finally, the mismatch values calculated from all random samples are utilized to estimate the probability distribution. The proposed fast Monte Carlo analysis flow will be applied to several circuit examples in Section 5.

Algorithm 4: fast Monte Carlo analysis for tunable circuits

(1) Generate L random samples $\{V_{OS,i}^{(l)}; i = 1, 2, ..., N; l = 1, 2, ..., L\}$ or $\{\Delta C_{I,i}^{(l)}, \Delta C_{F,i}^{(l)}; i = 1, 2, ..., N; l = 1, 2, ..., L\}$ using Latin hypercube sampling [20]-[21], where the subscript i denotes the i-th branch/finger and the superscript l stands for the l-th sampling point.
(2) For each random sample $l \in \{1, 2, ..., L\}$, apply dynamic programming (Algorithm 1 or Algorithm 3) to estimate the minimal mismatch ($V_{OS}^{(l)}$ or $C_{MIS}^{(l)}$).
(3) Given the L samples for the performance of interest (i.e., $V_{OS}^{(l)}$ or $C_{MIS}^{(l)}$), estimate the statistical characteristics (e.g., standard deviation, probability distribution, etc.).

5. NUMERICAL EXAMPLES

We demonstrate the efficacy of the proposed post-silicon tuning methodology using two circuit examples: a differential pair

and a switched-capacitor amplifier. Both circuit examples are implemented with a commercial 65nm CMOS process. All numerical experiments are run on a Linux 2.6GHz server. Two major observations will be concluded from our numerical experiments.

- When applying the adaptive post-silicon tuning, the standard deviation of random mismatch exponentially decreases as N (the number of total branches/fingers) increases. This result is dramatically better than the well-known Pelgrom model when no post-silicon tuning is applied.
- The proposed dynamic programming significant reduces the computational cost compared with the brute-force search (i.e., simply enumerating all possible configurations). We demonstrate 10~20x speed-up even if N is as small as 10~14.

5.1 Tunable Differential Pair

We applied the proposed post-silicon tuning methodology to the tunable differential pair in Figure 3. All transistor fingers in the differential pair have the size of 100nm (width) x 50nm (length), which is the minimal feature size of this technology.

A. Offset Voltage Characterization

We characterized the offset voltage $\{V_{OS,i};\ i = 1, 2, ..., N\}$ of one branch by transistor-level Monte Carlo simulation. The device model provided by the foundry contains statistical information to model both inter-die variations and local device mismatches. Our Monte Carlo simulation result verifies that the offset voltage $V_{OS,i}$ is almost independent of inter-die variations. For this reason, $\{V_{OS,i};\ i = 1, 2, ..., N\}$ of different branches are modeled as independent random variables. In addition, our Monte Carlo analysis shows that the offset voltage $V_{OS,i}$ can be approximated as a zero-mean Normal distribution, as shown in Figure 6. Note that we have normalized $V_{OS,i}$ in Figure 6 such that its standard deviation is equal to 1, as required by our non-disclosure agreement with the foundry. $\{V_{OS,i};\ i = 1, 2, ..., N\}$ are modeled as N independent standard Normal distributions (i.e., zero mean and unit variance) in this example.

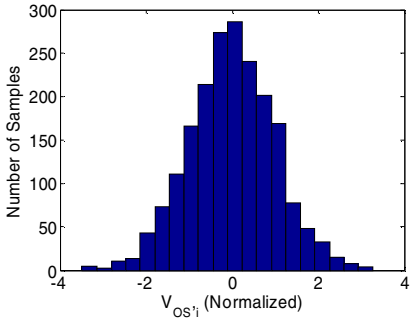

Figure 6. Random offset voltage ($V_{OS,i}$) of one branch estimated by 2000 transistor-level Monte Carlo simulation samples.

B. Post-Tuning Offset Estimation

Given the normalized offset voltages $\{V_{OS,i};\ i = 1, 2, ..., N\}$, we ran Monte Carlo analysis (Algorithm 4) with 10^4 samples to estimate the standard deviation σ_{OS} of the offset voltage V_{OS} defined in (1) after post-silicon tuning is applied. For testing and comparison, both the proposed dynamic programming (DP-MC) and the brute-force search (BS-MC) are utilized within the Monte Carlo flow to search for the optimal configuration that yields the minimal mismatch. We set the error tolerance $\varepsilon = 1\%$ in (14) to adaptively select the quantization step size h for dynamic programming. Figure 7 shows the values of the estimated σ_{OS} as the total number of branches N varies from 1 to 14.

From the result in Figure 7, we find that the proposed post-silicon tuning methodology achieves $\sigma_{OS} \sim \exp(-0.54N)$, while the well-known Pelgrom model predicts an improvement of only $\sigma_{OS} \sim 1/\mathrm{sqrt}(N)$. To further demonstrate the substantial benefit offered by the proposed post-silicon tuning, Table 1 outlines the required transistor sizes to achieve the same σ_{OS} when post-silicon tuning is and is not applied. Note that a 1.4μm (width) x 50nm (length) transistor with post-silicon tuning offers the same mismatch variation as a 4×10^5μm (width) x 50nm (length) transistor without post-silicon tuning in this differential pair example!

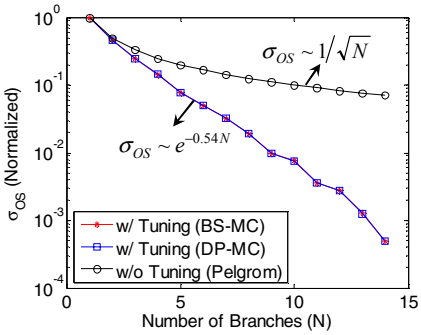

Figure 7. The standard deviation of the offset voltage (σ_{OS}) decreases as the total number of branches (N) increases.

Table 1. Required total gate width to achieve the given σ_{OS} (assuming minimal gate length 50nm for all cases)

σ_{OS} (Normalized)	/w Tuning		w/o Tuning
	# of Branches	Width (μm)	Width (μm)
1.00×10^{0}	1	0.1	1.00×10^{-1}
4.63×10^{-1}	2	0.2	4.67×10^{-1}
1.44×10^{-1}	4	0.4	4.79×10^{0}
5.08×10^{-2}	6	0.6	3.88×10^{1}
1.91×10^{-2}	8	0.8	2.74×10^{2}
7.63×10^{-3}	10	1.0	1.72×10^{3}
2.77×10^{-3}	12	1.2	1.30×10^{4}
4.85×10^{-4}	14	1.4	4.24×10^{5}

It is important to note that the proposed post-silicon tuning methodology requires control and measurement circuitries for adaptive configuration. The area overhead for these additional circuitries is not included in Table 1. However, we expect that the additional cost for post-silicon configuration is easily warranted based on the significant area reduction shown in Table 1.

C. Comparison of Accuracy and Complexity

As discussed in Section 3.1B, the quantization step size h in Algorithm 1 should be decreased to satisfy the given relative error tolerance, as the total number of branches N increases and the offset variation σ_{OS} decreases. Given the error tolerance $\varepsilon = 1\%$ in (14), Algorithm 2 adaptively determine the step size h for each value of N, as shown in Figure 8. Note that h exponentially decreases as N increases.

We used the same set of Monte Carlo samples for both the proposed dynamic programming (DP-MC) and the brute-force search (BS-MC). It, in turn, allows us to compare the estimated

σ_{OS} values from DP-MC and BS-MC, and use their relative difference as a criterion to measure the error incurred by the quantization of the DP-MC flow. In this example, the relative estimation errors of σ_{OS} are well-controlled (<1%) for all values of N, as shown in Figure 9.

Figure 10 shows the computational time for both the proposed dynamic programming (DP-MC) and the brute-force search (BS-MC). The brute-force search has a complexity of $O(2^N)$, since it enumerates all possible configurations. In this example, even if the value of N is as small as 14, the proposed dynamic programming algorithm achieves 10x speed-up compared with the brute-force search. We expect that the efficiency of the dynamic programming will be more pronounced as N further increases.

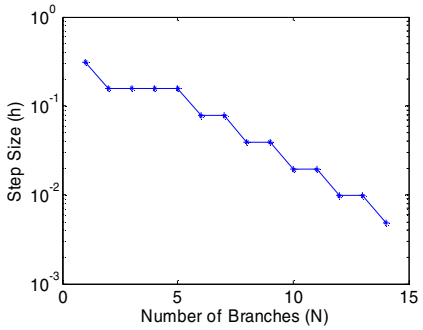

Figure 8. The quantization step size (h) decreases for dynamic programming as the total number of branches (N) increases.

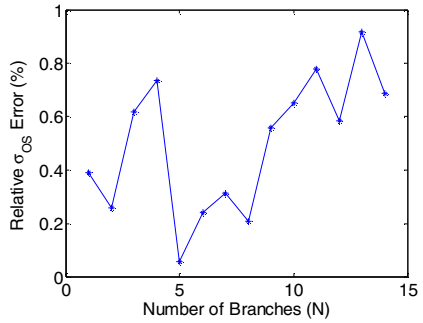

Figure 9. The relative estimation errors of σ_{OS} for DP-MC are smaller than 1% for all values of N.

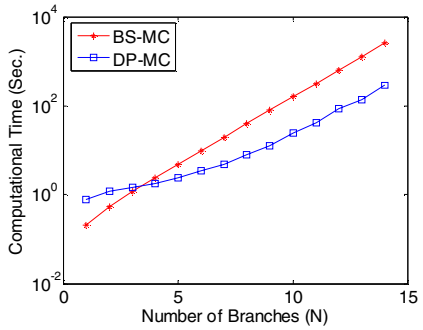

Figure 10. Computational time comparison of dynamic programming (DP-MC) and brute-force search (BS-MC).

5.2 Tunable SC Amplifier

We applied the proposed post-silicon tuning methodology to the tunable SC amplifier in Figure 5. The capacitor mismatch defined in (4) is independent of inter-die process variations; therefore, only local mismatches are considered in this example. The capacitor mismatches $\{\Delta C_{I,i}, \Delta C_{F,i}; i = 1, 2, ..., N\}$ of all fingers are normalized and modeled as independent standard Normal distributions.

A. Capacitor Mismatch Estimation

We ran Monte Carlo analysis (Algorithm 4) with 10^4 samples to estimate the standard deviation σ_{MIS} of the capacitor mismatch defined in (4) after post-silicon tuning is applied. The error tolerance in (14) is set to ε = 1% for adaptive step size control. Figure 11 shows the values of the estimated σ_{MIS} where both the proposed dynamic programming (DP-MC) and the brute-force search (BS-MC) are utilized to search for the optimal configuration with the minimal mismatch. Note that, in this example, the proposed post-silicon tuning methodology achieves $\sigma_{MIS} \sim \exp(-0.94N)$, while the well-known Pelgrom model predicts an improvement of only $\sigma_{MIS} \sim 1/\mathrm{sqrt}(N)$.

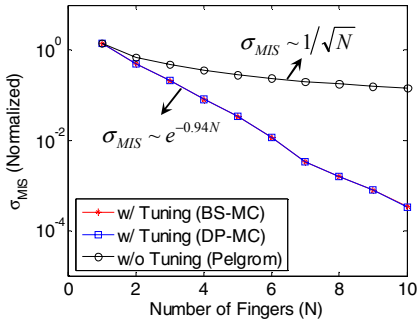

Figure 11. The standard deviation of the capacitor mismatch (σ_{MIS}) decreases as the total number of fingers (N) increases.

B. Comparison of Accuracy and Complexity

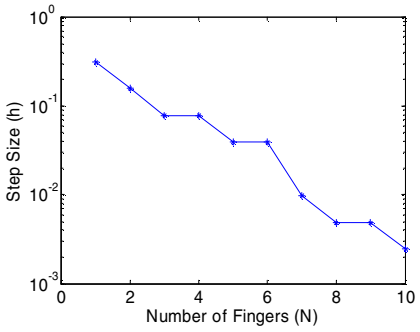

Figure 12. The quantization step size (h) decreases for dynamic programming as the total number of fingers (N) increases.

As shown in Figure 12 and Figure 13, the step size h is adaptively controlled for the proposed dynamic programming (DP-MC) such that the estimation error of σ_{MIS} is smaller than the given error tolerance ε = 1%. Figure 14 shows the computational time for both the proposed dynamic programming (DP-MC) and the brute-force search (BS-MC). The brute-force search enumerates all possible configurations and its complexity is close to $O(2^{2N})$. The proposed dynamic programming significantly reduces the computational cost in this example. Even if the value of N is as small as 10, the computational time is reduced from 7.5 hours (by BS-MC) to 25 minutes (by DP-MC) which is a 17x

speed-up. It should be noted that the computational time shown in Figure 14 is the total runtime for 10^4 Monte Carlo samples. The dynamic programming cost for configuring one chip (i.e., the computational cost of one Monte Carlo sample) is much smaller.

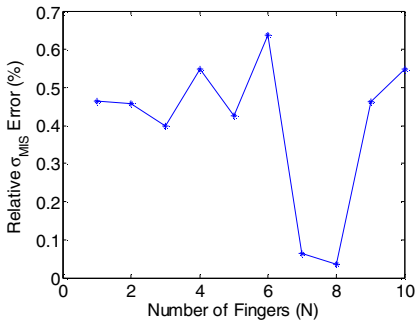

Figure 13. The relative estimation errors of σ_{MIS} for DP-MC are smaller than 1% for all values of N.

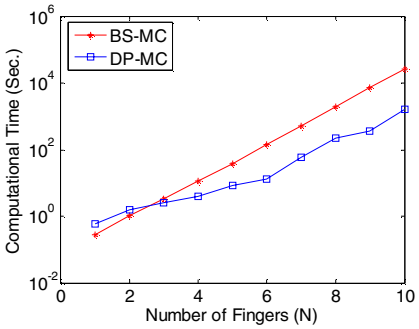

Figure 14. Computational time comparison of dynamic programming (DP-MC) and brute-force search (BS-MC).

6. CONCLUSIONS

In this paper, an adaptive post-silicon tuning methodology has been proposed to effectively reduce random device mismatches for analog circuits. Two tunable analog examples (i.e., a differential pair and a switched-capacitor amplifier) were discussed in detail. A novel dynamic programming algorithm was proposed to efficiently determine the optimal tuning configuration that yields the minimal mismatch. The proposed dynamic programming achieves significant (10~20x) speed-up compared with a brute-force search. The dynamic programming technique was further incorporated into a fast Monte Carlo analysis flow for efficient statistical analysis of the proposed tunable analog circuits. Our numerical results demonstrate that if the adaptive post-silicon tuning is applied, device mismatch exponentially decreases as area increases: $\sigma \sim \exp(-\alpha \cdot Area)$.

7. ACKNOWLEDGEMENT

This work has been supported by the National Science Foundation (NSF) under contract CCF-0702278.

8. REFERENCES

[1] S. Nassif, "Modeling and analysis of manufacturing variations," *IEEE CICC*, pp. 223-228, 2001.

[2] Semiconductor Industry Associate, *International Technology Roadmap for Semiconductors*, 2005.

[3] G. Gielen and R. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825-1852, Dec. 2000.

[4] G. Debyser and G. Gielen, "Efficient analog circuit synthesis with simultaneous yield and robustness optimization," *IEEE ICCAD*, pp. 308-311, 1998.

[5] A. Seifi, K. Ponnambalam and J. Vlach, "A unified approach to statistical design centering of integrated circuits with correlated parameters," *IEEE Trans. CAS – I*, vol. 46, no. 1, pp. 190-196, Jan. 1999.

[6] F. Schenkel, M. Pronath, S. Zizala, R. Schwencker, H. Graeb and K. Antreich, "Mismatch analysis and direct yield optimization by spec-wise linearization and feasibility-guided search," *IEEE DAC*, pp. 858-863, 2001.

[7] X. Li, P. Gopalakrishnan, Y. Xu and L. Pileggi, "Robust analog/RF circuit design with projection-based performance modeling," *IEEE Trans. CAD*, vol.26, no. 1, pp. 2-15, Jan. 2007.

[8] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE JSSC*, vol. 27, no. 11, pp. 1396-1402, Nov. 2002.

[9] T. Chen and S. Naffziger, "Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation," *IEEE Trans. VLSI*, vol. 11, no. 5, pp. 888-899, 2003.

[10] M. Mani, A. Singh and M. Orshansky, "Joint design-time and post-silicon minimization of parametric yield loss using adjustable robust optimization," *IEEE ICCAD*, pp. 19-26, 2006.

[11] P. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE JSSC*, vol. 40, no. 6, pp. 1212-1224, Jun. 2005.

[12] M. Pelgrom, A. Duinmaijer and A. Welbers, "Matching properties of MOS transistors," *IEEE JSSC*, vol. 24, no. 5, pp. 1433-1440, Oct. 1989.

[13] P. Drennan and C. McAndrew, "Understanding MOSFET mismatch for analog design," *IEEE JSSC*, vol. 38, no. 3, pp. 450-456, Mar. 2003.

[14] S. Ray and B. Song, "A 13b linear 40MS/s pipelined ADC with self-configured capacitor matching," *IEEE ISSCC*, pp. 852-861, 2006.

[15] K. Chan and I. Galton, "A 14b 100MS/s DAC with fully segmented dynamic element matching," *IEEE ISSCC*, pp. 2390- 2399, 2006.

[16] A. Dharchoudhury and S. Kang, "Worse-case analysis and optimization of VLSI circuit performance," *IEEE Trans. CAD*, vol. 14, no. 4, pp. 481-492, Apr. 1995.

[17] E. Felt, S. Zanella, C. Guardiani and A. Sangiovanni-Vincentelli, "Hierarchical statistical characterization of mixed-signal circuits using behavioral modeling," *IEEE ICCAD*, pp. 374-380, 1996.

[18] M. Sengupta, S. Saxena, L. Daldoss, G. Kramer, S. Minehane and J. Chen, "Application-specific worst case corners using response surfaces and statistical models," *IEEE Trans. CAD*, vol. 24, no. 9, pp. 1372-1380, Sep. 2005.

[19] X. Li, J. Le, P. Gopalakrishnan and L. Pileggi, "Asymptotic probability extraction for nonnormal performance distributions," *IEEE Trans. CAD*, vol. 26, no. 1, pp. 16-37, Jan. 2007.

[20] M. Mckay, R. Beckman and W. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239-245, May. 1979.

[21] E. Pebesma and G. Heuvelink, "Latin hypercube sampling of Gaussian random fields," *Technometrics*, vol. 41, no. 4, pp. 303-312, Nov. 1999.

[22] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, 2001.

[23] B. Razavi, *Design of Analog CMOS Integrated Circuits*, McGraw, 2001.

[24] D. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 2005.