# LARTTE: A Posynomial-Based Lagrangian Relaxation Tuning Tool for Fast and Effective Gate-Sizing and Multiple $V_t$ Assignment

Hsinwei Chou
Dept. of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI 53706
hsinweichou@wisc.edu

Yu-Hao Wang
Incentia Design Systems
Hsinchu Park, Taiwan
howard@incentia.com

Charlie Chung-Ping Chen
Dept. of Electrical Engineering
National Taiwan University
Taipei, 106, Taiwan
cchen@cc.ee.ntu.edu.tw

**Abstract— In this paper, we propose a novel method for fast and effective gate-sizing and multiple $V_t$ assignment using Lagrangian Relaxation (LR) and posynomial modeling. Our algorithm optimizes a circuit's delay and power consumption subject to slew rate constraints, and can readily take process variation into account. We first use SPICE to generate accurate delay and power models in posynomial form for standard cells, then formulate a large-scale, convex optimization problem based on these models. Finally, we perform LR to solve for the globally-optimal[1] set of transistor sizes and $V_t$s (with discretization) for each gate. Our key contribution is that we show for the first time that using posynomial models, LR-based circuit tuning can be carried out in a "generalized" or non-Gauss-Seidel manner for improved accuracy. Experimental results show that our implemented tuning tool, LARTTE, exhibits linear runtime and memory usage requirement, can effectively tune a circuit with over 15,000 variables and 8,000 constraints in under 7 minutes, and can minimize the probability of final delay variation by introducing a margin of separation between the worst output arrival time and all other outputs' arrival times.**

## I. Introduction

With the continuous scaling of CMOS technology and problem size/complexity explosion, the task of circuit or transistor-level tuning (for delay, power, noise, etc.) can be extremely time-consuming and even overwhelming for today's designers. In past works [1] [2], it was shown that the method of Lagrangian Relaxation (LR) can be used to solve the circuit tuning problem efficiently and effectively. However, there are several flaws with these earlier works, such as the lack of power consumption consideration as well as the use of the simple-yet-inaccurate Elmore delay model. Furthermore, as the magnitude of leakage power is quickly catching up to that of dynamic power [3], the important task of using multiple $V_t$ levels in a design [4] [5] for leakage power reduction must be addressed in the circuit tuning process.

As CMOS scales into nanometer technology, another problem that must be dealt with in the tuning process is on-chip process variation [6]. From device geometry to device parameters, many things can vary to cause the final critical delay value to differ non-trivially from that calculated via static timing analysis. This in turn makes yield rate estimation difficult. This problem is especially severe in the case of a circuit after automated transistor size tuning, since a well-tuned circuit that neglects process variation during the tuning process typically exhibits a "wall-like" distribution in its primary outputs' arrival times [7]. This is because to squeeze the most performance benefits out of sizing, a standard tuner will size in such a way that most, if not all, of the primary outputs'(PO) arrival times end up with equally critical values. This exacerbates the issue of process variation and final delay uncertainty.

In this paper, we propose a novel method to perform efficient gate-sizing and multiple $V_t$ assignment using LR and posynomial modeling. Our algorithm optimizes a circuit's delay and power consumption subject to slew rate constraints, and can readily take process variation into account. We first use SPICE to generate accurate delay and power models in posynomial form [10] for standard cells, then formulate a large-scale, convex optimization problem based on these models. Finally, we perform LR to solve for the globally-optimal (with respect to the posynomial-based optimization problem, without discretization) set of transistor sizes and $V_t$s (with discretization) for each gate. Our main contribution is that we show for the first time that LR-based circuit tuning can be carried out in a 'generalized' or non-Gauss-Seidel manner. In previous works, the Elmore delay model was used in the LR framework, which consequently constrained the optimization flow to a serialized and ordered process (more on this in Section III-E). Our posynomial-based approach does not suffer from this limitation, and can thus tune much more accurately and faster. Experimental results show that our implemented tuning tool, LARTTE, exhibits linear runtime and memory usage requirement, can effectively tune a circuit with over 15,000 variables and 8,000 constraints in under 7 minutes, and can minimize the probability of final delay variation by introducing a margin of separation between the worst output arrival time and all other outputs' arrival times. Experiments also show that LARTTE compares favorably with SNOPT [11], a state-of-the-art general-purpose optimization problem solver. LARTTE is over 250x faster than SNOPT, but can achieve the same quality of results.

This paper is organized as follows. Background and posynomial modeling information are detailed in Section II, followed by the main LARTTE algorithm description in Section III. Modifications to LARTTE to guard against process variation is detailed in Section IV. Experimental results and concluding remarks follow in Section V and Section VI.

[1] Optimality is with respect to the posynomial approximation-based optimization problem, without discretization.

## II. Preliminaries and Posynomial Modeling

In this section, we first define the notations that we will be using throughout this paper. Then, we provide some background information on posynomial functions and optimization problems in general. Finally, we describe the method that we used to accurately characterize the various attributes of a gate (ie delay, dynamic power, leakage power, input slew, etc.) as posynomial functions.

### A. Notations

The following notations are used throughout this paper. Given a combinational circuit, we first introduce two auxiliary nodes, a sink and a source (see Figure 1). The sink has all of its fan-ins from the primary outputs, and the source has all of its fan-outs to the primary inputs. The nodes in the circuit are labeled in reverse topological order, with the sink having the index of 0 and the source having the index of N (assume N total nodes). Let $input(i)$ and $output(i)$ be the set of node indices that connect directly to the input(s) and output(s) of node $i$. Define $\mathcal{D}$ and $\mathcal{G}$ to be the set of primary inputs and internal gate components in the circuit, respectively. For $i \in \mathcal{G}$, $a_i$ is the arrival time at the output of gate $i$, $W_{g_i}$ is the width of the NMOS and PMOS (adjusted by a $\gamma$ ratio), $V_{tn_i}$ and $V_{tp_i}$ are the NMOS and PMOS threshold voltages, $C_{L_i}$ is the loading capacitance (expressed as a function of the widths of the loading gates), and $s_i$ is a designer-specified upper bound on the input slew rate of gate $i$[2]. Let $T_i$, $D_i$, $P_{dynamic_i}$, and $P_{leakage_i}$ denote the input slew rate, propagation delay, dynamic power, and leakage power posynomial functions of gate $i$, respectively. Lastly, define $L_{w_i}$ and $U_{w_i}$ to be the lower and upper bound of $W_{g_i}$, $L_{tn_i}$ and $U_{tn_i}$ to be the lower and upper bound of $V_{tn_i}$, and $L_{tp_i}$ and $U_{tp_i}$ to be the lower and upper bound of $V_{tp_i}$.

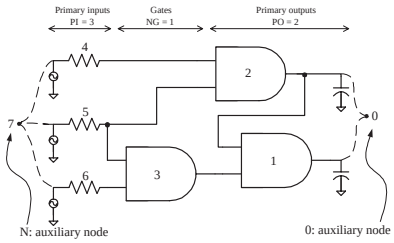



Fig. 1. A combinational circuit.

### B. Background: Optimization Problems and Posynomial Functions

In general, optimization problems [12] have the form:

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & g_i(x) \leq 0,\ i = 1, \ldots, m \\ & h_i(x) = 0,\ i = 1, \ldots, n \end{aligned} \tag{1}$$

where $x \in \Re^n$ is a $n$-vector of optimization variables and $f_0$, $g_i$, and $h_i$ are the objective function, inequality constraints, and equality constraints, respectively. If $f_0$, $g_i$, and $h_i$ are all convex functions, then the problem becomes a convex optimization problem. An important property of the convex optimization problem is that any locally-optimal solution is also guaranteed to be globally optimal.

A posynomial function has the form:

$$f(x) = \sum_{j=1}^{k} c_j x_1^{\alpha_{1j}} x_2^{\alpha_{2j}} \ldots x_n^{\alpha_{nj}} \tag{2}$$

where $f$ is a real-valued function whose domain $x \in \Re^n$ is non-negative, $c_j \geq 0$, and $\alpha_{ij} \in \Re$. A posynomial is a sum of monomials. It is well-known [10] that under a simple exponential transformation, a posynomial function can be converted into a convex function. Hence, if an optimization problem is expressed in terms of posynomial functions, then a global minimum can be easily found by searching for a local minimum, which can be done with any formal mathematical programming technique [12]. Thus, this is the main motivation in this work for using posynomials to model gate characteristics.

### C. The Posynomial Modeling Procedure

The posynomial modeling procedure is essentially done via least-square regression analysis on SPICE simulation data. Formally, we define the posynomial parametric regression problem as follows:

$$\begin{aligned} \textbf{Posyfit: minimize} \quad & \sum_{i=1}^{m} \left[ \left( \left( \sum_{j=1}^{k} c_j x_{1i}^{\alpha_{1j}} x_{2i}^{\alpha_{2j}} \ldots x_{ni}^{\alpha_{nj}} \right) - b_i \right)^2 \right] \\ \text{subject to} \quad & c_j \geq 0 \end{aligned} \tag{3}$$

where $x \in \Re^{m \times n}$ corresponds to m different sets of a n-vector of tunable parameter values, $b \in \Re^m$ is a vector of m different SPICE-simulated scalar results (each corresponding to one unique simulation run under the associated tunable parameter values in x), and k, $c \in \Re^k$, and $\alpha \in \Re^{k \times n}$ are the unknown parameters that we are trying to determine. The value of m is user-defined and corresponds to the number of SPICE simulations that will be run to generate the necessary $b_i$ values for posynomial-fitting. In general, a higher m leads to a greater accuracy in the final characterized posynomial, but in turn requires a longer pre-processing time (as SPICE simulations are inherently time-consuming). The value of n is the number of tunable parameters which affect the metric being approximated. For example, if the delay posynomial form is being determined, then n equals 5, for the delay of a gate depends on $W_g$[3], $C_L$, $V_{tn}$, $V_{tp}$, and s (input slew rate).

The posynomial-fitting procedure works as follows. First, we select m different sets of n-tunable parameter values and simulate each individually to find m different $b_i$ values. Then, after plugging these terms back into equation 3, we are left with 3 unknowns, k, c, and $\alpha$. To solve for these, we first guess a value for the vector $\alpha$ and its dimension k. Then, using $\alpha$ and k, we solve for the last remaining unknown, c, using CFSQP [13], a general-purpose unconstrained problem solver. If the resulting least-square value using the solved c is below an error tolerance level, we stop and return the characterized posynomial form

[2]For simplicity of presentation, $a_i$ and $s_i$ are assumed to be the same for both the rising and the falling transition.

[3]Size of NMOS. The PMOS width is adjusted by a $\gamma$ value. For ease of presentation, this is not shown in this paper.

(the inner summation term). Otherwise, we repeat the fitting-procedure for a different guess of $\alpha$ and k, and continue to do so until the least-square error is minimized.

To avoid excessive trial count in guessing the posynomial form, we employ the following heuristic when trying to find the right k, c, and $\alpha$. First, we guess a dominant monomial term by exploiting well-known dependence relationships. For example, for the delay posynomial, we start with a term that has $W_g^{-1}$ and $C_L^1$, since we know in general that the delay of a gate depends on its loading capacitance and its drive strength. Then, based on the resulting fitting error using this guess, we gradually adjust the power coefficients appropriately and add more monomial terms into the posynomial equation until we find a reasonably accurate approximation.

We give the following example to more clearly illustrate the posynomial-fitting procedure. Suppose that we are trying to determine the delay posynomial of a particular gate, say a CMOS inverter. Then, let m=2 and pick the following two sets of tunable parameter values: $\{W_g$=3, $V_{tn}$=0.7, $V_{tp}$=0.7, $C_L$=5, s=0.5$\}$ and $\{W_g$=4, $V_{tn}$=0.9, $V_{tp}$=0.8, $C_L$=2, s=0.7$\}$. Next, we simulate in SPICE the delay of an inverter under these two sets of parameters, and call the results $b_1$ and $b_2$. Assume for this example that $b_1$=15 and $b_2$=10. Given these data, the Posyfit problem is reduced to the following:

$$\begin{aligned} \text{minimize} \quad & \left(\left(\sum_{j=1}^{k} c_j 3^{\alpha_{1j}} 0.7^{\alpha_{2j}} 0.7^{\alpha_{3j}} 5^{\alpha_{4j}} 0.5^{\alpha_{5j}}\right) - 15\right)^2 + \\ & \left(\left(\sum_{j=1}^{k} c_j 4^{\alpha_{1j}} 0.9^{\alpha_{2j}} 0.8^{\alpha_{3j}} 2^{\alpha_{4j}} 0.7^{\alpha_{5j}}\right) - 10\right)^2 + \\ \text{subject to} \quad & c_j \geq 0 \end{aligned} \tag{4}$$

It should be noted that the k, c, and $\alpha$ values are required to be the same across all m copies of the inner summation term, since we are trying to determine a posynomial model that would be accurate for any set of parameter values. With the reduced Posyfit problem, we can then carry out the iterative fitting procedure to find the unknown parameters (k, c, and $\alpha$), and thus the delay posynomial function, for the inverter. The returned posynomial is expressed as a function of $W_g$, $V_{tn}$, $V_{tp}$, $C_L$, and s. For illustration purpose, the following is the actual inverter delay posynomial form found in this work: $D_{inv}(W_g,V_{tn},V_{tp},C_L,s)$ = $0.39V_{tn}V_{tp}^{-1}$ + $2.14W_g^{-1}C_LV_{tp}$ + $623V_{tp}^{0.5}W_g^{0.5}$ + $12.2V_{tn}^3$ + $29W_g^{0.5}V_{tn}^{-1}V_{tp}^{0.5}$ + $0.14s^{0.5}$ + $1.07W_g^{-1}C_LV_{tn}^2V_{tp}^{-1}$.

In this work, we set the stopping criteria of the fitting procedure to be when 90% of the fitting samples, using the guessed posynomial form, agree numerically to within $\pm$ 10% of their corresponding SPICE results. Also, when generating the SPICE values, we assumed the worst case conditions (ie. for delay simulations, the input signal to the last transistor in the stack is set to arrive last, etc). Table I shows the model-fitting error mean and standard deviation for the characterized gates. Prefixes Inv, Na, and No in the table represent inverter, NAND, and NOR gates, and suffixes TP, PL, and PD represent delay, leakage, and dynamic power respectively. The unit for the entries in the table is the % difference (in either direction) between the samples' values using the final posynomial form and their corresponding SPICE results. For example, the leakage power posynomial of an inverter (InvPL) has a mean fitting error of 2.6%, and a standard deviation of 5.6%. For illustration purpose, the fitting error distribution for a NAND6 is also given in Figure 2. The unit for the x-axis in these figures is again the % difference.

TABLE I
MODEL FITTING ERROR MEAN AND STANDARD DEVIATION

| Gate | Mn. | Dev. | Gate | Mn. | Dev. | Gate | Mn. | Dev. |
|---|---|---|---|---|---|---|---|---|
| InvPD | -0.1 | 3.5 | Na6TP | -0.2 | 4.7 | No4PL | -2.7 | 6.0 |
| InvPL | -2.6 | 5.6 | Na7PD | -0.2 | 4.4 | No4TP | -0.1 | 3.2 |
| InvTP | -0.1 | 2.5 | Na7PL | -0.1 | 1.8 | No5PD | -0.1 | 2.1 |
| Na2PD | -1.2 | 6.5 | Na7TP | -0.2 | 4.8 | No5PL | -0.0 | 1.8 |
| Na2PL | -0.0 | 1.6 | Na8PD | -0.2 | 4.5 | No5TP | -0.2 | 4.7 |
| Na2TP | -0.1 | 3.4 | Na8PL | -0.0 | 1.8 | No6PD | -0.1 | 2.2 |
| Na3PD | -0.4 | 6.7 | Na8TP | -0.3 | 4.9 | No6PL | -2.7 | 6.1 |
| Na3PL | -0.0 | 1.8 | Na9PD | -0.2 | 4.9 | No6TP | -0.1 | 3.2 |
| Na3TP | -0.2 | 4.2 | Na9PL | -0.0 | 1.9 | No7PD | -0.1 | 2.3 |
| Na4PD | -0.3 | 5.6 | Na9TP | -0.3 | 5.1 | No7PL | -2.8 | 6.5 |
| Na4PL | -0.0 | 1.8 | No2PD | -0.8 | 6.6 | No7TP | -0.1 | 3.0 |
| Na4TP | -0.2 | 4.5 | No2PL | -2.5 | 5.4 | No8PD | -0.1 | 2.4 |
| Na5PD | -0.2 | 4.9 | No2TP | -0.1 | 3.2 | No8PL | -2.8 | 5.6 |
| Na5PL | -0.0 | 1.8 | No3PD | -0.7 | 6.3 | No8TP | -0.1 | 3.0 |
| Na5TP | -0.2 | 4.7 | No3PL | -2.6 | 5.6 | No9PD | -0.1 | 2.6 |
| Na6PD | -0.2 | 4.5 | No3TP | -0.1 | 2.9 | No9PL | -2.8 | 5.5 |
| Na6PL | -0.0 | 1.8 | No4PD | -0.2 | 4.5 | No9TP | -0.1 | 3.1 |

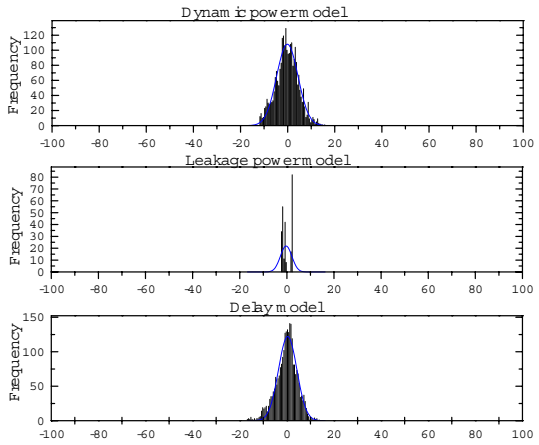



Fig. 2. Model-fitting error distribution for a NAND6 gate.

## III. POSYNOMIAL-BASED LAGRANGIAN RELAXATION

In this section, we derive a generalized Lagrangian Relaxation tuning algorithm which incorporates the use of posynomial delay and power models. The section is organized as follows. In III-A, we formally formulate the circuit tuning optimization problem, or the Primal Problem ($\mathcal{PP}$). III-B introduces the Lagrangian Subproblem, $\mathcal{LRS}/\lambda$. III-C states the first-order KKT condition which will be used in our algorithm to significantly speed up the tuning process. III-D outlines the Lagrange Multiplier adjustment scheme used in this work, while III-E describes the method by which $\mathcal{LRS}/\lambda$ can be solved optimally, efficiently, and accurately in a generalized manner. This will in turn solve our original problem ($\mathcal{PP}$) as well. Finally, in III-F, we discuss the necessary post-tuning $V_t$ discretization heuristic as well as give a summary of LARTTE.

### A. Primal Problem Formulation

In general, the problem of minimizing the maximum delay and power consumption (dynamic + leakage) subject to arrival time and slew constraints can be formulated as a large-scale, nonlinear programming problem. We call the following the Pri-

mal Problem ($\mathcal{PP}$):

$$\begin{aligned}
\mathcal{PP}: \quad \text{minimize} \quad & \alpha_1 a_0 + \alpha_2 P_{leakage}(Wg, Vtn, Vtp, s) \\
& + \alpha_3 P_{dynamic}(Wg, C_L, Vtn, Vtp, s) \\
\text{subject to} \quad & a_j \leq a_0,\ j \in input(0) \\
& a_j + D_i \leq a_i,\ i \in \mathcal{G} \cap \forall j \in input(i) \\
& D_i \leq a_i,\ i \in \mathcal{D} \\
& T_i \leq s_i,\ i \in (\mathcal{D} \cup \mathcal{G}) \\
& L_{w_i} \leq W_{g_i} \leq U_{w_i},\ i \in \mathcal{G} \\
& L_{tn_i} \leq V_{tn_i} \leq U_{tn_i},\ i \in \mathcal{G} \\
& L_{tp_i} \leq V_{tp_i} \leq U_{tp_i},\ i \in \mathcal{G}
\end{aligned} \tag{5}$$

where $\alpha_1$, $\alpha_2$ and $\alpha_3$ are the normalized weighting factors to the maximum delay of the circuit, $a_0$ (arrival time of the artificial sink node), total leakage power, $P_{leakage}$, and total dynamic power, $P_{dynamic}$[4]. The sum of $\alpha_1$, $\alpha_2$, and $\alpha_3$ is 1. The weighting factors are user-assigned based on the operating condition of the target application, such as how much time spent in idle mode, how critical is the timing of the design, etc. (Alternatively, a more sophisticated $\alpha$ assignment scheme could be applied, ie. iteratively invoking LARTTE and adjusting the $\alpha$ factors along the way based on the previous iteration's tuned results). For all other notations, see Section II-A.

From simple rearrangement, equation 5 can be transformed into the following:

$$\begin{aligned}
\text{minimize} \quad & \alpha_1 a_0 + \alpha_2 P_{leakage}(Wg, Vtn, Vtp, s) \\
& + \alpha_3 P_{dynamic}(Wg, C_L, Vtn, Vtp, s) \\
\text{subject to} \quad & \frac{a_j}{a_0} \leq 1,\ j \in input(0) \\
& \frac{a_j + D_i}{a_i} \leq 1,\ i \in \mathcal{G} \cap \forall j \in input(i) \\
& \frac{D_i}{a_i} \leq 1,\ i \in \mathcal{D} \\
& \frac{T_i}{s_i} \leq 1,\ i \in (\mathcal{D} \cup \mathcal{G}) \\
& L_{w_i} W_{g_i}^{-1} \leq 1,\ W_{g_i} U_{w_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& L_{tn_i} V_{tn_i}^{-1} \leq 1,\ V_{tn_i} U_{tn_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& L_{tp_i} V_{tp_i}^{-1} \leq 1,\ V_{tp_i} U_{tp_i}^{-1} \leq 1,\ i \in \mathcal{G}
\end{aligned} \tag{6}$$

In general, $\mathcal{PP}$ is not in the form of a convex optimization problem. However, posynomials can be readily transformed into convex form by the following simple exponential transformation of the variables [10]: Let $x$ represent the vector of all tunable parameters, and transform each entry $x_i$ in x to a new variable $y_i$, where $x_i = e^{y_i}$. Now, y represents the vector of tunable parameters, and it is substituted into the $\mathcal{PP}$ equation for x to form a convex optimization problem. Applying LR to the transformed $\mathcal{PP}$ will give us an optimal solution in terms of y, but we can easily recover the desired $x_i$s by exponentiating the $y_i$s.

### *B. Lagrangian Relaxation with Logarithmic Transformations*

From PP, after making the necessary exponential variable transformations (to both the posynomials as well as the arrival time terms), the next step is to make a Logarithmic transformation[5] on the non-simple constraints by taking the natural log of both sides. Since the logarithmic function is monotonically increasing, this can be done without affecting the final result. The newly transformed problem is the following:

$$\begin{aligned}
\text{minimize} \quad & \alpha_1 e^{a_0^*} + \alpha_2 P_{leakage}^*(Wg, Vtn, Vtp, s) \\
& + \alpha_3 P_{dynamic}^*(Wg, C_L, Vtn, Vtp, s) \\
\text{subject to} \quad & \ln\left(\frac{e^{a_j^*}}{e^{a_0^*}}\right) \leq 0,\ j \in input(0) \\
& \ln\left(\frac{e^{a_j^*} + D_i^*}{e^{a_i^*}}\right) \leq 0,\ i \in \mathcal{G} \cap \forall j \in input(i) \\
& \ln\left(\frac{D_i^*}{e^{a_i^*}}\right) \leq 0,\ i \in \mathcal{D} \\
& \ln\left(\frac{T_i^*}{e^{s_i^*}}\right) \leq 0,\ i \in (\mathcal{D} \cup \mathcal{G}) \\
& L_{w_i} W_{g_i}^{-1} \leq 1,\ W_{g_i} U_{w_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& L_{tn_i} V_{tn_i}^{-1} \leq 1,\ V_{tn_i} U_{tn_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& L_{tp_i} V_{tp_i}^{-1} \leq 1,\ V_{tp_i} U_{tp_i}^{-1} \leq 1,\ i \in \mathcal{G}
\end{aligned} \tag{7}$$

where parameters with a $*$ superscript represent those after an exponential change of variable.

From equation 7, we can form the general Lagrangian function [12] by introducing non-negative Lagrange multipliers to relax each arrival time and slew constraint into the objective function. Simple bounds on the transistor widths and $V_t$s are not relaxed. For example, for $j \in input(0)$, let $\lambda_{j0}^A$ denote the multiplier for the constraint $\ln(\frac{e^{a_j^*}}{e^{a_0^*}}) \leq 0$. For $i \in \mathcal{G} \cap \forall j \in input(i)$, let $\lambda_{ji}^A$ denote the multipliers for the constraint $\ln(\frac{e^{a_j^*} + D_i^*}{e^{a_i^*}}) \leq 0$, and for $i \in (\mathcal{D} \cup \mathcal{G}) \cap \forall j \in input(i)$, let $\lambda_{ji}^S$ denote the multipliers for the constraint $\ln(\frac{T_i^*}{e^{s_i^*}}) \leq 0$. For $i \in \mathcal{D}$, let $\lambda_{mi}^A$ denote the multipliers for the constraint $\ln(\frac{D_i^*}{e^{a_i^*}}) \leq 0$. Finally, let $\lambda$ be the vector of all the multipliers introduced. Then, the general Lagrangian function can be written as:

$$\begin{aligned}
\mathcal{L}(Wg, Vtn, Vtp, a, s, \lambda) = {} & \alpha_1 e^{a_0^*} + \alpha_2 P_{leakage}^*(Wg, Vtn, Vtp, s) \\
& + \alpha_3 P_{dynamic}^*(Wg, C_L, Vtn, Vtp, s) \\
& + \sum_{j \in input(0)} \lambda_{j0}^A \ln\left(\frac{e^{a_j^*}}{e^{a_0^*}}\right) \\
& + \sum_{i \in \mathcal{G}} \sum_{j \in input(i)} \lambda_{ji}^A \ln\left(\frac{e^{a_j^*} + D_i^*}{e^{a_i^*}}\right) \\
& + \sum_{i \in (D \cup G)} \sum_{j \in input(i)} \lambda_{ji}^S \ln\left(\frac{T_i^*}{e^{s_i}}\right) \\
& + \sum_{i \in \mathcal{D}} \lambda_{mi}^A \ln\left(\frac{D_i^*}{e^{a_i^*}}\right)
\end{aligned} \tag{8}$$

The Lagrangian relaxation subproblem associated with a particular fixed Lagrange multiplier value $\lambda$ ($\mathcal{LRS}/\lambda$) is:

$$\begin{aligned}
\mathcal{LRS}/\lambda: \quad \text{minimize} \quad & \mathcal{L}_\lambda(Wg, Vtn, Vtp, a, s) \\
\text{subject to} \quad & L_{w_i} W_{g_i}^{-1} \leq 1,\ W_{g_i} U_{w_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& L_{tn_i} V_{tn_i}^{-1} \leq 1,\ V_{tn_i} U_{tn_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& L_{tp_i} V_{tp_i}^{-1} \leq 1,\ V_{tp_i} U_{tp_i}^{-1} \leq 1,\ i \in \mathcal{G}
\end{aligned} \tag{9}$$

From the theory of the Lagrangian function, it is known that there exists a vector value of $\lambda$ for which the optimal solution

[4]For simplicity of presentation, the activity factor is not shown in the parameter list of the $P_{dynamic}$ term because it is not a tunable parameter.

[5]We perform the Logarithmic transformation because empirically, we found that it resulted in exceptional runtime improvement.

of $\mathcal{LRS}/\lambda$ is equal to the optimal solution of the original $\mathcal{PP}$ problem. Hence, if we can find this $\lambda$ value, then we can find the optimal solution to the original problem (by solving $\mathcal{LRS}/\lambda$).

Before we discuss our strategy for finding the correct $\lambda$ value, we first present a key part of our algorithm which is largely responsible for the excellent runtime of LARTTE.

### C. First-Order KKT Necessary Condition For The Lagrangian Function Solution

For a given Lagrangian function that we are interested in solving, The theory of the Lagrangian tells us that for a particular vector value $\lambda$ to be the correct, optimal solution multiplier, the first-order Kuhn-Karush-Tucker (KKT) necessary condition must hold. Under the first-order KKT condition, the gradient of the Lagrangian function with respect to all variable parameters must be equal to 0. That is, $\nabla_{W^*_{g_i}}\mathcal{L}_\lambda$=0, $\nabla_{V^*_{tn_i}}\mathcal{L}_\lambda$=0, and $\nabla_{V^*_{tp_i}}\mathcal{L}_\lambda$=0 for $1 \leq i \leq$ NG+PO. Also, $\nabla_{a^*_i}\mathcal{L}_\lambda$=0 and $\nabla_{s^*_i}\mathcal{L}_\lambda$=0 for $1 \leq i \leq$ PI+NG+PO. Therefore, in trying to find out what the correct, optimal multiplier value $\lambda$ should be, we need only consider cases where the above conditions are satisfied. This 'filtering' process is the key to dramatic runtime reduction.

By taking $\nabla_{a^*_i}\mathcal{L}_\lambda$=0 and $\nabla_{s^*_i}\mathcal{L}_\lambda$=0 to the Lagrangian, we obtain the following required optimality condition on the arrival time and slew constraint multipliers:

$$\sum_{j\in input(0)} \lambda^A_{j0} = \alpha_1 e^{a^*_0}$$

$$\sum_{j\in input(i)} \lambda^A_{ji} = \sum_{k\neq 0\in output(i)} \frac{\lambda^A_{ik}\cdot e^{a^*_i}}{e^{a^*_i}+D^*_k},\ i\in(\mathcal{D}\cup\mathcal{G})$$

$$\sum_{j\in input(i)} \lambda^S_{ji} = \sum_{k\neq 0\in output(i)} \left(\frac{\lambda^A_{ik}}{e^{a^*_i}+D^*_k}\frac{\partial D^*_k}{\partial s^*_i} + \frac{\lambda^S_{ik}}{T^*_k}\frac{\partial T^*_k}{\partial s^*_i}\right) + \alpha_2\frac{\partial P^*_{leakage}}{\partial s^*_i} + \alpha_3\frac{\partial P^*_{dynamic}}{\partial s^*_i},\ i\in(\mathcal{D}\cup\mathcal{G}) \quad (10)$$

Note that each line in 10 applies to an individual set of components of $\lambda$ and is independent to the other lines. For example, if a particular vector value $\lambda^*$ is to be deemed a candidate for the correct, optimal multiplier $\lambda$, then all of its outgoing primary output (PO) multiplier components must sum up to be $\alpha_1 e^{a^*_0}$. Furthermore, for all gates in $\mathcal{D}\cup\mathcal{G}$, all of their incoming multipliers (from fan-in gates) must sum up to their outgoing multipliers (multiplied by $\frac{e^{a^*_i}}{e^{a^*_i}+D^*_k}$). In considering only those values of $\lambda^*$ which satisfy equation 10 as solution candidates for the correct, optimal multiplier $\lambda$, our tuning process can significantly cut down on runtime by avoiding unnecessary computation involving impossible $\lambda$ candidates.

Using equation 10, we now present our method for solving for the correct, optimal $\lambda$ value (and consequently the optimal solution of our original problem as well).

### D. Iterative Multiplier Adjustment for Determining Optimal $\lambda$

We employ an iterative, modified sub-gradient method for finding the desired $\lambda$ vector. First, we arbitrarily pick a starting lambda value which satisfies equation 10. For example, we can start by assigning each of the $\lambda^A_{j0}$ to be $\frac{\alpha_1 e^{a^*_0}}{N}$, where N is the number of inputs to sink node 0. The assignment of all other multiplier components can be done in a similar manner (in reverse topological order). After forming an initial $\lambda^*$ guess, we then iteratively update $\lambda^*$ using a modified sub-gradient approach shown in Table II, line 3, to form a new guess at every iteration. $\theta_k$ is a step size value which is initialized to 1 and gradually updated over iterations using a Trust-Region approach [14]. We continue to iterate and make new guesses for the correct, optimal value of $\lambda$ until our $\mathcal{LRS}/\lambda^*$ value converges to that of the PP value. When this occurs, we will have found our desired multiplier $\lambda$, which is just equal to the $\lambda^*$ at the stopped iteration.

### E. Solving $\mathcal{LRS}/\lambda$ in a Generalized Manner

Our LARTTE algorithm terminates when the solution of $\mathcal{LRS}/\lambda$ converges to that of $\mathcal{PP}$. In order to do this, we must have a method for solving $\mathcal{LRS}/\lambda$ for the optimal (with respect to the given $\lambda$) tunable parameter vector, x. In previous works [1] [2], due to the use of the Elmore delay model, this procedure had to be carried out in a Gauss-Seidel-like or serialized manner. The reason is as follows. In the Elmore model, the delay at a node is characterized as a function of the resistance and capacitance values along the path, not as a function of the tunable parameters in the circuit (specifically, the widths of the gates). Therefore, the process of solving for the tunable vector x in $\mathcal{LRS}/\lambda$ cannot be done in one single iteration (in a generalized manner). This is why in [1] and [2], the authors resorted to a greedy, serialized approach, where the tunable parameters ($x_i$s) are solved in topological order and one-at-a-time so that after solving for one $x_i$, its corresponding downstream resistance and capacitance values can be updated appropriately for the next gate's $x_i$ to be solved correctly. The order-dependent and serial nature of this solving-procedure makes the tuning process inaccurate and time-consuming, and this is the key drawback to these previous works.

In this work, we overcome the above drawback by using posynomial-based models in the LR framework for the first time. Since the posynomials are characterized with respect to only the tunable parameters of the circuit, $\mathcal{LRS}/\lambda$ is expressed completely in terms of x, and the entire problem can be solved optimally in a generalized manner (in one single iteration) using any formal mathematical programming technique. Thus, the serialization/order restriction is removed entirely in our posynomial-based method, leading to a much more accurate and faster tuning process. The accuracy, efficiency, and elegance of our generalized solving-procedure is the key to LARTTE's performance, and is the main contribution of this work.

We resort to an off-the-shelf solver in L-BFGS-B [15] to solve $\mathcal{LRS}/\lambda$. L-BFGS-B implements the well-known, Limited-Memory BFGS method [12], which has been proven to be exceptional for handling large-scale, unconstrained problems. This method belongs to the class of quasi-Newton methods, which uses a Hessian approximation of the objective function (instead of the exact Hessian) to compute the Newton search direction for the minimum. However, unlike the standard BFGS method, the Limited-Memory approach uses only the curvature information from the most recent iterations to construct the Hessian approximation. This is beneficial for large problems whose Hessian matrices cannot be computed at a reasonable cost or are too dense to be manipulated easily. To avoid any confusion, we leave out

**ALGORITHM** LARTTE:
**Output**: optimal gate-sizing and $V_t$ allocation solution
1. $k := 1$ /* iteration number */
   $\lambda :=$ arbitrary initial vector of constraint multipliers satisfying (10)
   Initialize all optimization tunable parameters
2. Solve $\mathcal{LRS}/\lambda$ by calling L-BFGS-B to minimize $\mathcal{L}_\lambda(Wg, Vtn, Vtp, a, s, \lambda)$ until optimal solution found and then compute $a_1, \ldots, a_{\text{PI+NG+PO}}$ and $s_1, \ldots, s_{\text{PI+NG+PO}}$
3. /* Adjust multipliers $\lambda$ */
   for $i := 0$ to PI+NG+PO do
   foreach $j \in input(i)$ do

$$\lambda_{ji}^{NEW} := \begin{cases} \lambda_{ji}^{A} * \left( \frac{e^{a_j^*}}{e^{a_0^*}} \right)^{\theta_k} & \text{if } i = 0 \\ \lambda_{ji}^{A} * \left( \frac{e^{a_j^* + D_i^*}}{e^{a_i^*}} \right)^{\theta_k} & \text{if } i \in \mathcal{G} \\ \lambda_{ji}^{A} * \left( \frac{D_i^*}{e^{a_i^*}} \right)^{\theta_k} & \text{if } i \in \mathcal{D} \\ \lambda_{ji}^{S} * \left( \frac{T_i^*}{e^{s_i^*}} \right)^{\theta_k} & \text{if } i \in (\mathcal{D} \cup \mathcal{G}) \end{cases}$$

   Project $\lambda_{ji}^{NEW}$ to the nearest point satisfying (10)
4. $k := k + 1$
5. Goto step 2 until the cost functions of $\mathcal{PP}$ and $\mathcal{LRS}/\lambda$ converge to within a specified tolerance
6. Discretize the $V_t$ solutions
7. Solve $\mathcal{LRS}/\lambda$ by calling L-BFGS-B to find the optimal solution

TABLE II
LARTTE ALGORITHM.

the internal details of this method and refer interested readers to [12].

### F. Vt Discretization and LARTTE Summary

Up to now, we have treated $V_t$ as a tunable parameter in $\Re$. This was done because LR is a technique for optimizing continuously-differentiable problems. Obviously, this is a problem because in practice, there is usually only a limited number of $V_t$ levels available for use. Hence, in order to rectify this situation, we must discretize our $V_t$ solutions in the end to the nearest allowable $V_t$ value. For example, if we find that after tuning, one of our transistors has an optimal $V_t$ solution value of 0.176V, but we can only choose between a device with 0.24V $V_t$ and a device with 0.16V $V_t$, then we would discretize this transistor's $V_t$ solution to be 0.16V instead. This discretization step is carried out at the end of the tuning process for all transistors and their corresponding $V_t$ solutions. If needed, the gate sizes can also be discretized to suit an ASIC synthesis flow.

Since the discretization step is a heuristic, the quality/optimality of the solution after applying discretization seems questionable at first. However, we have empirically found that as long as the number of $V_t$ levels available for use is around 4 or more, the solution after discretization will typically be not too far off from the original, un-discretized solution (as it will be shown in our experimental results section). Hence, our LR technique is still reasonable under mild assumptions.

A summary of LARTTE is given in Table II.

## IV. LARTTE WITH PROCESS VARIATION GUARD

Recall that if process variation were not taken into account during the tuning process, then a "good" tuning tool will size in such a way that many of the outputs end up having the same critical arrival time in the end. This in turn creates a high probability that the final delay value (subject to process variation) will differ from that calculated via static timing analysis, since any of the critical output's arrival time can increase from variation. Therefore, any chip yield rate projection that was done can end up being highly inaccurate, which is undesirable. To improve yield estimation accuracy, process variation (specifically final delay variation) can be taken into account during the LARTTE tuning process as follows: First, LARTTE is invoked normally and the critical-path delay and its corresponding output pin is recorded. Then, using this recorded information, the same critical output's arrival time constraint is modified/relaxed to derive a new $\mathcal{PP}$ formulation, which is then subsequently solved by LARTTE again. Thus, we have the following:

$$\begin{aligned} \text{minimize} \quad & \alpha_1 a_0 + \alpha_2 P_{leakage}(Wg, Vtn, Vtp, s) \\ & + \alpha_3 P_{dynamic}(Wg, C_L, Vtn, Vtp, s) \\ \text{subject to} \quad & a_j \le a_0,\ j \in input(0),\ j \ne c \\ & a_c \le (1.0 + \eta) a_0,\ c = \text{critical PO}, 0 \le \eta \le 1 \\ & a_j + D_i \le a_i,\ i \in \mathcal{G} \cap \forall j \in input(i) \\ & D_i \le a_i,\ i \in \mathcal{D} \\ & T_i \le s_i,\ i \in (\mathcal{D} \cup \mathcal{G}) \\ & L_{w_i} \le W_{g_i} \le U_{w_i},\ i \in \mathcal{G} \\ & L_{tn_i} \le V_{tn_i} \le U_{tn_i},\ i \in \mathcal{G} \\ & L_{tp_i} \le V_{tp_i} \le U_{tp_i},\ i \in \mathcal{G} \end{aligned} \tag{11}$$

It can be seen from equation 11 that the only thing that has changed from the original $\mathcal{PP}$ formulation is that the old constraint on the original critical output has been modified/relaxed by a user-specified ratio, $\eta$. This is done to explicitly introduce a margin of separation between the most critical arrival time and all other outputs' arrival times. By doing so, the probability of final delay variation can be minimized. The value of $\eta$ ranges between 0 to 1, with a larger $\eta$ leading to a greater margin of separation.

## V. EXPERIMENTAL RESULTS

We implemented LARTTE in C++ and ran all of our experiments on a 1.0GHz P4 machine with 1.0Gb of RAM. The stopping criterion of LARTTE was set to when $\mathcal{PP}$ and $\mathcal{LRS}/\lambda$ agreed to within 1.0%. Lower and upper bounds of the transistor width were $0.2\mu m$ and $1.1\mu m$, respectively. For $V_t$, the lower and upper bounds were 0.14V and 0.26V. $V_{DD}$ was 1.0V. Input slew ranged from $30ps$ to $150ps$. Four $V_t$ levels were made available for discretization: 0.14V, 0.18V, 0.22V, and 0.26V. Appropriate activity factors were assigned to the posynomials throughout the circuit using PowerMill. All SPICE simulations were carried out in $0.1\mu m$ technology. We conducted our experiments on the ISCAS85 benchmark circuits, where the number of gates ranged from 214 to 3,512, and the total number of tunable parameters from 654 to 15,198. Table III shows the LARTTE optimization results. Only the $V_t$s were discretized. To illustrate the convergence property of LARTTE, we show in Figure 3 the convergence sequence for a 12-bit ALU controller. As it can be seen, the duality gap is closing each step along the way as desired. This behavior was observed in all of our experiments.

In Table III, the 'optimize delay' columns show the maximum delay before and after tuning, with only timing involved in the objective function ($\alpha_1$=1, $\alpha_2$=$\alpha_3$=0). All transistors have a nominal $V_t$ value of 0.18V. After obtaining the best possible delay

value from sizing optimization alone, we then try to optimize the total power consumption subject to that same delay value. Hence, the solution obtained from tuning the power consumption is guaranteed to have a critical path delay not exceeding the corresponding delay value shown in the 'optimize delay' column. For power tuning, the dynamic and leakage power terms were arbitrarily assigned equal weights. The resulting optimized-power solution from tuning both the transistor width and $V_t$ are shown in the 'optimize total power' columns. Compared to the power consumption of the circuit with delay-tuning only, this shows an average of 58% improvement in total power reduction. The table also shows that LARTTE exhibits linear runtime and memory usage requirement (see Figure 4 as well). Lastly, we show in the table the leakage power consumption before and after $V_t$ discretization. As expected, the discretized solution is always inferior to the original solution. However, it can be seen that the drop in leakage savings is relatively trivial in all cases. This suggests that with 4 levels of $V_t$ available, the discretization heuristic works reasonable well. This is also shown through Figure 5, which analyzes the degree of power reduction which can be achieved as a function of the number of $V_t$s available for use.

To gauge the effectiveness and runtime of LARTTE, we used a state-of-the-art, general convex problem solver in SNOPT to solve the same primal problem (with discretization as well). The runtime results are tabulated in Table III, where it can be seen that our LR method is over 250x faster. Furthermore, we verified that our LARTTE solution agreed with the SNOPT solution to within 1% in all cases.

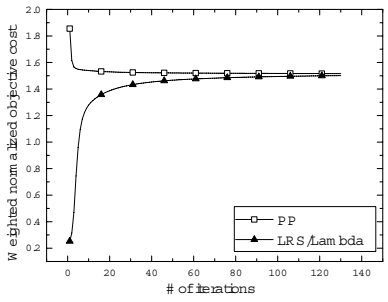
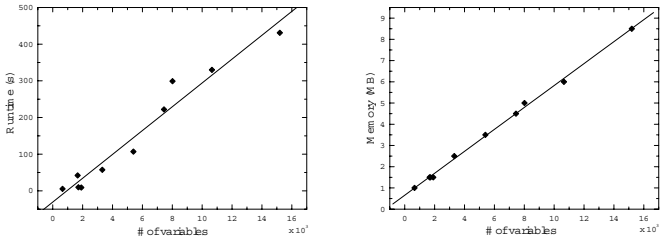


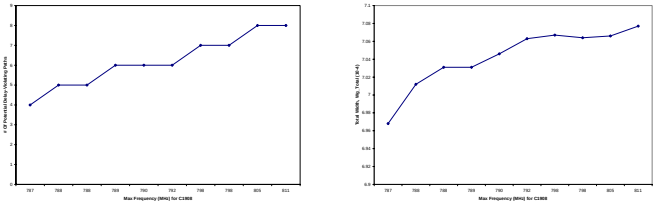

Fig. 3. The convergence sequence for a 12-bit ALU and controller.




(a) (b)

Fig. 4. The (a) runtime and (b) storage requirements of LARTTE vs. number of variables.

We next investigate LARTTE's effectiveness of guarding against final delay variation. An $\eta$ value of 0.5 was used in all of our tests. Shown in Table IV are all the circuit's critical-path delay before and after re-invoking LARTTE with the process variation modifications. Also shown are the next closest output arrival times before and after re-tuning. As it can be seen, LARTTE successfully creates a distance separation between these two values after re-tuning. We also show the relationship between $\eta$'s value and the resulting max frequency in Fig. 6(a). For the circuit (c1908) in that figure, it can be seen that increasing the value of $\eta$ bumps up the final critical delay value and decreases the maximum operating frequency. However, as the critical delay value rises, the number of 'potential' delay-violating paths directly decreases. A 'potential path' was arbitrarily defined as any non-critical path whose final delay value is within 10% of the final critical delay value. The tradeoff between max frequency and final delay variation probability is clear from this figure. Finally, for completeness, we also show the tradeoff between max frequency and total circuit area in Fig. 6(b). Area was calculated by summing all transistors' widths.



(a) (b)

Fig. 6. The (a) Number of Potential Delay-Violating Paths Vs. Max Frequency, and (b) Total Wg Vs. Max Frequency.

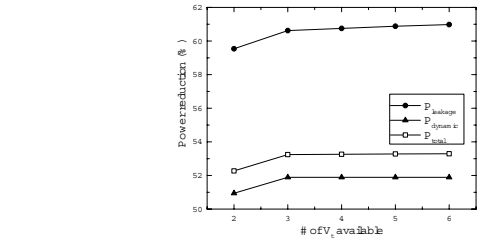



Fig. 5. Effect of varying the number of $V_t$ levels available on the potential of power reduction with LARTTE

## VI. Conclusion

In this paper, we presented a novel, effective, and fast way to perform simultaneous gate-sizing and multi-$V_t$ assignment using Lagrangian Relaxation and posynomial modeling. We made the key contribution of showing that a posynomial-based LR approach is generalized and accurate. Our technique is practical and versatile, as it can be used for both custom and ASIC design flow (with gate-size discretization). We also showed an easy way to modify the tuning algorithm to directly take process variation into account.

TABLE III
RESULTS OF OPTIMIZATION ON ISCAS'85 BENCHMARK CIRCUITS USING 4 LEVELS OF $V_t$

| Circuit Name | # of Gates | # of Var. | # of Constr. | Optimize Delay (ps) | | | Optimize Total Power (0.1mW) | | | | | | Leakage Power | | Memory (MB) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min. size nom.-$V_t$ | Sizing nom.-$V_t$ | % | Sizing nom.-$V_t$ | Sizing multi-$V_t$ | % | Runtime (s) | | Speed up | Before Discretize | After Discretize | |
| | | | | | | | | | | SNOPT | LARTTE | | | | |
| c432 | 214 | 654 | 473 | 1620 | 1230 | 24.1 | 1.25 | 0.59 | 52.9 | 31 | 5 | 5.9 | 7.66e-6 | 7.67e-6 | 1.0 |
| c499 | 514 | 1716 | 1059 | 1060 | 895 | 15.6 | 3.49 | 1.46 | 58.3 | 290 | 10 | 29.7 | 1.71e-5 | 1.74e-5 | 1.5 |
| c880 | 383 | 1665 | 987 | 1070 | 872 | 18.5 | 3.41 | 1.35 | 60.4 | 341 | 42 | 8.1 | 1.90e-5 | 1.91e-5 | 1.5 |
| c1355 | 546 | 1908 | 1227 | 1070 | 914 | 14.6 | 5.62 | 2.93 | 47.9 | 269 | 9 | 29.7 | 4.43e-5 | 4.47e-5 | 1.5 |
| c1908 | 880 | 3315 | 1781 | 1500 | 1220 | 18.7 | 7.22 | 3.07 | 57.5 | 1316 | 57 | 23.0 | 4.21e-5 | 4.24e-5 | 2.5 |
| c2670 | 1193 | 5397 | 2903 | 1860 | 1520 | 18.3 | 10.7 | 4.09 | 61.9 | 7915 | 107 | 74.0 | 3.93e-5 | 3.95e-5 | 3.5 |
| c3540 | 1169 | 7446 | 3824 | 2170 | 1800 | 17.1 | 14.7 | 6.02 | 58.9 | 20773 | 222 | 93.6 | 5.44e-5 | 5.48e-5 | 4.5 |
| c5315 | 2307 | 10656 | 5932 | 1900 | 1590 | 16.3 | 19.8 | 8.42 | 57.4 | 64424 | 330 | 195.2 | 9.28e-5 | 9.32e-5 | 6.0 |
| c6288 | 2416 | 8016 | 5120 | 6070 | 5170 | 14.8 | 15.8 | 4.66 | 70.4 | 25326 | 299 | 84.7 | 1.85e-5 | 1.89e-5 | 5.0 |
| c7552 | 3512 | 15198 | 8011 | 1520 | 1250 | 17.8 | 27.8 | 12.6 | 54.6 | 117067 | 431 | 271.6 | 1.35e-4 | 1.36e-4 | 8.5 |

TABLE IV
DELAY SEPARATION BEFORE AND AFTER RE-TUNING WITH PROCESS VARIATION MODIFICATIONS

| Circuit Name | Before Re-Tuning | | After Re-Tuning | |
|---|---|---|---|---|
| | Critical Delay (ps) | Nearest Delay (ps) | Critical Delay (ps) | Nearest Delay (ps) |
| C432 | 1230 | 1165 | 1279 | 1165 |
| C499 | 895 | 889 | 968 | 890 |
| C880 | 872 | 847 | 924 | 847 |
| C1355 | 914 | 914 | 994 | 914 |
| C1908 | 1220 | 1207 | 1312 | 1204 |
| C2670 | 1520 | 1519 | 1593 | 1520 |
| C3540 | 1800 | 1784 | 1921 | 1786 |
| C5315 | 1590 | 1561 | 1703 | 1558 |
| C6288 | 5170 | 5170 | 5582 | 5170 |
| C7552 | 1250 | 1248 | 1362 | 1249 |

## REFERENCES


[1] C. P. Chen, C. C. N. Chu, and D. F. Wong, "Fast and exact simultaneous gate and wire sizing by lagrangian relaxation," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 18, no. 7, pp. 1014 –1025, July 1999.

[2] H. Tennakoon and C. Sechen, "Gate sizing using lagrangian relaxation combined with a fast gradient-based pre-processing step," in *ICCAD*, 2002, pp. 395–402.

[3] A. Grove, *Intl. Electron Devices Meeting*, Dec. 2002.

[4] L. Wei, Z. Chen, M. Johnson, K. Roy, and V. De, "Design and optimization of low voltage high performance dual threshold cmos circuits," in *Proc. of the Design Automation Conference*, 1998, pp. 489–494.

[5] M. Hirabayashi, K. Nose, and T. Sakurai, "Design methodology and optimization strategy for dual-vth scheme using commercially available tools," in *Proc. of the International symposium on Low power electronics and design*, 2001, pp. 283 – 286.

[6] S. R. Nassif, "Modeling and analysis of manufacturing variations," in *IEEE 2001 Custom Integrated Circuits Conference*, 2001, pp. 223–228.

[7] X. Bai, C. Visweswariah, P. Strenski, and D. Hathaway, "Uncerty-aware circuit optimization," in *Design Automation Conference*, 2002, pp. 58–63.

[8] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impacts on circuits and microarchitecture," in *Design Automation Conference*, 2003, pp. 338–342.

[9] M. Hashimoto and H. Onodera, "Increase in delay uncertainty by delay optimization," in *IEEE International Symposium on Circuits and Systems*, 2001, pp. 379–382.

[10] K. Kasamsetty, M. Ketkar, and S. S. Sapatnekar, "A new class of convex functions for delay modeling and their application to the transistor sizing problem," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 19, no. 7, pp. 779–788, July 2000.

[11] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," Department of Mathematics, University of California, San Diego, La Jolla, CA, Numerical Analysis Report 97-2, 1997.

[12] J. Nocedal and S. J. Wright, *Numerical Optimization*. Heidelberg, Berlin, New York: Springer Verlag, 1999.

[13] Lawrence, C., Zhou, J. L., Tits, and A. L., "User's guide for cfsqp version 2.4: A c code for solving (large scale) constrained nonlinear (minmax) optimization problems, generating iterates satisfying all inequality constraints," Institute for Systems Research, University of Maryland, College Park, MD, Tech. Rep. TR-94-16r1, 1996.

[14] A. R. Conn, N. Gould, and P. L. Toint, "Global convergence of a class of trust region algorithms for optimization with simple bounds," *SIAM J. Numerical Analysis*, vol. 25, pp. 433–460, 1988.

[15] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," Northwestern University EECS," Technical Report NAM-08, 1994.