Optimal Gate Sizing with Multiple Vt Assignment using Generalized Lagrangian Relaxation

***Abstract*— Simultaneous gate sizing with multiple V_t assignment for optimal delay and power is a complicated task in modern custom designs. In this work, we make the key contribution of a novel gate-sizing and multi-V_t assignment technique based on generalized Lagrangian Relaxation. Experimental results show that our technique has linear runtime and memory usage, and can optimally tune circuits with over 15,000 variables and 8,000 constraints in under 8 minutes (250x faster than state-of-the-art optimization solvers).**

I. Introduction

Transistor sizing is a crucial task in modern custom designs for achieving high-performance. From delay optimization [1] [2] [3] to dynamic power reduction [4], sizing plays an important role in shaping a circuit to meet its performance targets. In recent years, due to the exponential surge in leakage power consumption, multi-V_t assignment [5] [6] has also become an essential task in high-end designs. At present, research is ongoing [7] [8] for determining how transistor sizing can be optimally combined with multi-V_t assignment to achieve the best performance.

In this work, we make the key contribution of a novel, optimal timing and power gate-sizing and multi-V_t assignment scheme. Our technique is based on the classical theory of Lagrangian Relaxation [9] and a class of functions known as posynomials [10]. Due to the convexity of our problem as well as the mathematically proven theories behind our formulations, our method is guaranteed to be fast and accurate in finding the globally-optimal solution point. Experimental results confirm the viability of our approach, as our implemented tuning software, 'LARTTE', demonstrates a mere linear runtime and memory usage, and can optimally tune circuits with over 15,000 variables and 8,000 constraints in under 8 minutes. This is over 250x faster than SNOPT [11], a state-of-the-art optimization solver.

The remainder of this paper is organized as follows. Background and posynomial modeling information are detailed in Sections II and III, followed by the main LARTTE algorithm in Section IV. Experimental results and concluding remarks follow in V and VI.

II. Preliminaries

In this section, we provide some background information on posynomial functions and convex optimization problems in general. We will also define several notations for use throughout the rest of this paper.

A. Posynomial Functions and Convex Optimization Problems

A posynomial [10] function has the form

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

where f is a real-valued function whose domain $x \in \Re^n$ is non-negative, $c_j \geq 0$, and $\alpha_{ij} \in \Re$. When k=1, f is called a monomial function. Therefore, a posynomial function is a sum of monomials. Posynomials have the property that they are closed under addition, multiplication, and non-negative scaling.

In general, a convex optimization problem has the form

$$\begin{aligned} \text{minimize} \quad & f_0(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \ i = 1, \dots, m \\ & h_i(x) = 0, \ i = 1, \dots, n \end{aligned} \tag{2}$$

where $x \in \Re^n$ is a n-vector of optimization variables and f_0, g_i, and h_i are convex objective function, convex inequality constraints, and convex equality constraints, respectively. An important property of the convex optimization problem is that any locally optimal solution is also globally optimal. Essentially, this means that if one can find a local solution to the convex problem using any standard numerical optimization technique (as we do in this work), then that solution is guaranteed to be the global solution as well. This is a very powerful property and is what makes posynomials an attractive form for approximating characteristics such as the delay and power of a gate.

B. Notations

The following notations will be used throughout the rest of this paper. Given a combinational circuit shown in Figure 1 with PI primary inputs, NG gates (excluding primary outputs), and PO primary outputs, the transistor widths and V_ts are the optimization variables to tune for minimizing some cost function, i.e. maximal delay and total power subject to area/performance/power constraints.

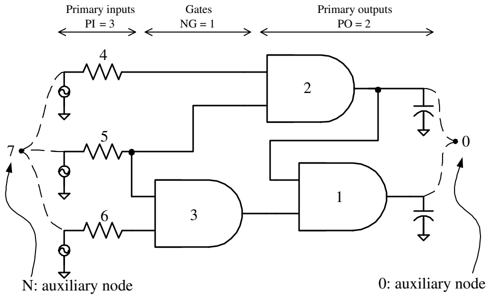


Fig. 1. A combinational circuit.

The primary inputs, gates, and primary outputs are individually referred to as a component. The output of each component is referred to as a node. Two additional auxiliary nodes are introduced in such a way that one has fan-ins from all the PO primary outputs and the other has fan-outs to all the PI primary inputs. Every node is unique.

Let N=PI+NG+PO+1. The nodes are labeled by indices $0, \dots,$ N in a reverse topological ordering of the circuit viewed as a weighted directed acyclic graph (DAG). See Fig. 1 for illustration. For $0 \leq i \leq$ N-1, let a_i be the arrival time at node i, and 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 respectively. For example, $input(0)$=$\{1, 2\}$ and $output(3)$=$\{1\}$ for the circuit shown in Fig. 1. Let $\mathcal{D}$ and $\mathcal{G}$ be the set of primary input and gate(including primary outputs) component indices in the circuit, respectively. For example, $\mathcal{D}$=$\{4, 5, 6\}$ and $\mathcal{G}$=$\{1, 2, 3\}$ for the circuit shown in Fig. 1. For $i \in \mathcal{G}$, let W_{g_i} be the parameter controlling the widths of all the NMOSs and PMOSs(adjusted by a γ ratio), V_{tn_i} and V_{tp_i} be the NMOS and PMOS threshold voltages respectively, C_{L_i} be the load capacitance of i, and s_i be the output slew of i. For simplicity of presentation, a_i and s_i can be either the rising or the falling version. Let T_i, D_i, $P_{dynamic_i}$, and $P_{leakage_i}$ denote the slew, propagation delay, dynamic power, and leakage power functions of i respectively. Finally, Let L_{w_i} and U_{w_i} be the lower and upper bound of W_{g_i} respectively, L_{tn_i} and U_{tn_i} be the lower and upper bound of V_{tn_i} respectively, and L_{tp_i} and U_{tp_i} be the lower and upper bound of V_{tp_i} respectively.

III. Posynomial Delay and Power Approximations

The benefits of using posynomials as a form of approximation was described earlier in Section II-A. In this section, we detail the process by which we generated accurate, posynomial characterizations of the delay and power(both dynamic and leakage) behavior of all simple CMOS gates. These forms will be used in the core LARTTE algorithm, as we will show in Section IV.

A. Posynomial Parametric Regression

Regression analysis was performed to generate the posynomial approximations. In other words, we tried to best fit a set of SPICE-simulated data points to the general posynomial equation. A posynomial parametric regression problem has the following form:

$$\begin{array}{lll} \textbf{Posyfit:} & \text{minimize} & \left\| \sum_{j=1}^{k} c_j x_1^{\alpha_{1j}} x_2^{\alpha_{2j}} \ldots x_n^{\alpha_{nj}} - b_j \right\|_2^2 \\ & \text{subject to} & c_j \geq 0,\ 1 \leq j \leq k \end{array} \qquad (3)$$

where $x \in \Re^n$ is a n-vector of tunable parameters (i.e. W$_g$s, V$_t$s, etc.), $c \in \Re^k$ and $\alpha \in \Re^{k \times n}$ are the unknown characterization coefficients to be determined, and $b \in \Re^k$ is a k-vector of SPICE-simulated sample data values corresponding to a particular metric which we are trying to approximate (i.e. delay, power, etc.).

B. Sample Data Point Generation

To generate the necessary data for curve-fitting, we first designed a series of experiments such that the worst-case delay, leakage power, and dynamic power of all the various gates can be captured. This was done with slew effects taken into account for the highest accuracy. Then, for each gate, we exhaustively ran tens of thousands of SPICE simulations(in 0.1μm technology) to obtain a meaningful sample of data points for use in regression analysis in III-C.

C. Posynomial Characterizations

After enough sample data points have been collected, we then used a general SQP package, CFSQP [12], to solve the parametric regression problem for the needed coefficients 'c' and 'α' in equation 3. This was performed as follows. First, we guess a value for the vector 'α' and its dimension 'k'. Then, using that 'α', 'k', and the available SPICE-simulated sample data point vector 'b', we solve the corresponding least-squares problem in equation 3 for the coefficient vector 'c' (using CFSQP). We iteratively and exhaustively repeat this procedure for different guesses of 'α' and 'k' until we obtain a least-square error that is below a certain threshold level, at which point we will have found an accurate posynomial approximation for the particular metric involved(i.e. delay, power, etc.). This posynomial approximation process was performed for every relevant metric of every simple CMOS gate (i.e. NAND, NOR, etc.) until the resulting fitting errors for all the gates came out to have at least 90% of their errors contained within ±10%. For illustration purpose, the posynomial approximation we found for the propagation delay of a CMOS inverter is shown below in equation 4. All other forms for all other gates are omitted due to space limitation.

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

$$\begin{aligned} D_i(W_{g_i}, C_{L_i}, V_{tn_i}, V_{tp_i}, s_j) = {} & (3.92e^{-1}) V_{tn_i} V_{tp_i}^{-1} \\ & + (1.80e^{-4}) W_{g_i}^{-1} C_{L_i}^{0.5} \\ & + (2.14) W_{g_i}^{-1} C_{L_i} V_{tp_i} \\ & + (6.23e^{2}) V_{tp_i}^{0.5} W_{g_i}^{0.5} \\ & + (1.22e^{1}) V_{tn_i}^{3} \\ & + (2.90e^{1}) W_{g_i}^{0.5} V_{tn_i}^{-1} V_{tp_i}^{0.5} \\ & + (3.61e^{-5}) W_{g_i}^{-0.5} V_{tn_i}^{-1} V_{tp_i}^{0.5} \\ & + (1.42e^{-1}) {s_j}^{0.5} \\ & + (1.07) W_{g_i}^{-1} C_{L_i} V_{tn_i}^{2} V_{tp_i}^{-1} \qquad (4) \end{aligned}$$

For the slew-related term in equation 4, $j \in input(i)$ where $i \in (\mathcal{D} \cup \mathcal{G})$. Note that each individual term in the posynomial approximation may not have any direct physical meaning due to the nature of the multi-dimensional curve-fitting and guessing procedure.

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. Suffixes TP, PL, and PD denote delay, leakage power, and dynamic power respectively.

IV. The LARTTE Algorithm

We now present the main LARTTE algorithm. Problem formulations and theories involving optimality conditions are detailed to give insights to the superior runtime and performance of LARTTE.

A. Delay and Total Power Optimization: Problem Formulation

The problem of minimizing the maximum delay and total power subject to arrival time and slew constraints can be formulated as a general, large-scale nonlinear constrained optimization problem as follows:

$$\begin{array}{ll} \text{minimize} & \alpha_1 \bar{a}_0 + \alpha_2 \bar{P}_{leakage}(Wg, Vtn, Vtp, s) \\ & \quad + \alpha_3 \bar{P}_{dynamic}(Wg, C_L, Vtn, Vtp, s) \\ \text{subject to} & a_j \leq a_0,\ j \in input(0) \\ & a_j + D_i \leq a_i,\ i \in \mathcal{G} \cap \forall j \in input(i) \\ & D_i \leq a_i,\ i \in \mathcal{D} \\ & T_i \leq s_i,\ i \in (\mathcal{D} \cup \mathcal{G}) \\ & L_{w_i} \leq W_{g_i} \leq U_{w_i},\ i \in \mathcal{G} \\ & L_{tn_i} \leq V_{tn_i} \leq U_{tn_i},\ i \in \mathcal{G} \\ & L_{tp_i} \leq V_{tp_i} \leq U_{tp_i},\ i \in \mathcal{G} \end{array} \qquad (5)$$

where α_1, α_2 and α_3 are user-specified weighting factors to the normalized maximum delay $\bar{a}_0$, normalized total leakage power $\bar{P}_{leakage}$, and normalized total dynamic power $\bar{P}_{dynamic}$ functions respectively. α_1+α_2+α_3=1. The weights are there to allow the overall importance to be divided amongst the various terms based on application-specific conditions, i.e. the percentage of time the circuit spends in idling mode, etc. The weighting factors also enable tradeoff analysis between delay, leakage, and dynamic power to be performed easily. W_g, V_{tn} and V_{tp} are vectors of tunable parameters consisting of the parameters controlling the widths of all transistors and transistor V_ts respectively. C_L and s are vectors of load capacitance and slews.

From simple rearrangement, equation 5 can be transformed into the following geometric program, which we will denoted as the primal problem ($\mathcal{PP}$).

$$\begin{aligned}
\mathcal{PP}: \quad & \text{minimize} && \alpha_1 \bar{a}_0 + \alpha_2 \bar{P}_{leakage}(Wg, Vtn, Vtp, s) \\
& && + \alpha_3 \bar{P}_{dynamic}(Wg, C_L, Vtn, Vtp, s) \\
& \text{subject to} && \frac{a_j}{a_0} \leq 1,\ j \in input(0) \\
& && \frac{a_j + D_i}{a_i} \leq 1,\ i \in \mathcal{G} \cap \forall j \in input(i) \\
& && \frac{D_i}{a_i} \leq 1,\ i \in \mathcal{D} \\
& && \frac{T_i}{s_i} \leq 1,\ i \in (\mathcal{D} \cup \mathcal{G}) \\
& && L_{w_i} W_{g_i}^{-1} \leq 1,\ W_{g_i} U_{w_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& && L_{tn_i} V_{tn_i}^{-1} \leq 1,\ V_{tn_i} U_{tn_i}^{-1} \leq 1,\ i \in \mathcal{G} \\
& && L_{tp_i} V_{tp_i}^{-1} \leq 1,\ V_{tp_i} U_{tp_i}^{-1} \leq 1,\ i \in \mathcal{G} \qquad (6)
\end{aligned}$$

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}. After that, y is used to represent the vector of all new tunable parameters and is thus used in the tuner. After tuning is complete, the original targets, x_i's, can be easily recovered from the optimal y_i's via exponentiation.

B. Generalized Lagrangian Relaxation with Logarithmic Constraint Transformations

From PP, after making the necessary exponential variable transformations, the next step is to make a Logarithmic transformation 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:

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

where parameters with a $*$ superscript represent those after an exponential change of variables. The reason why this logarithmic-transformation was done was because empirically, we found that this formulation resulted in greater stability in our tuning process than the original formulation, PP. The log function also couples nicely with the exponential function to reduce the complexity of the optimality conditions(to be shown later).

From 7, we can form the general Lagrangian function [13] 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_ts are not relaxed. For example, for $j \in input(0)$, let λ_{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 λ_{ji}^A denote the multipliers for the constraints $\ln(\frac{e^{a_j^*} + D_i^*}{e^{a_i^*}}) \leq 0$, and for $i \in (\mathcal{D} \cup \mathcal{G}) \cap \forall j \in input(i)$, let λ_{ji}^S denote the multipliers for the constraints $\ln(\frac{T_i^*}{e^{s_i^*}}) \leq 0$. For $i \in \mathcal{D}$, let λ_{mi}^A denote the multipliers for the constraints $\ln(\frac{D_i^*}{e^{a_i^*}}) \leq 0$. Finally, let λ be the vector of all the multipliers introduced. Then, the general Lagrangian function can be written as:

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

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

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

From basic theory on the Lagrangian function [13], it is known that there exists a vector value of λ for which the optimal solution of $\mathcal{LRS}/\lambda$ is actually equal to the optimal solution of the original problem, $\mathcal{PP}$. Hence, if we can find this λ value, then we can find the desired optimal solution of the original problem, PP (through solving $\mathcal{LRS}/\lambda$).

Before we discuss our strategy for finding the correct λ value, we shall 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, proven mathematical theories [13] tell us that for a particular vector value λ 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 λ 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:

$$
\begin{aligned}
&\sum_{j \in input(0)} \lambda^A_{j0} = \alpha_1 e^{a_0^*} \\
&\sum_{j \in input(i)} \lambda^A_{ji} = \sum_{k \neq 0 \in output(i)} \frac{\lambda^A_{ik} \cdot e^{a_i^*}}{e^{a_i^*} + D_k^*},\ i \in (\mathcal{D} \cup \mathcal{G}) \\
&\sum_{j \in input(i)} \lambda^S_{ji} = \sum_{k \neq 0 \in output(i)} \frac{\lambda^A_{ik}}{e^{a_i^*} + D_k^*} \frac{\partial D_k^*}{\partial s_i^*} + \frac{\lambda^S_{ik}}{T_k^*} \frac{\partial T_k^*}{\partial s_i^*} \Big) \\
&\qquad + \alpha_2 \frac{\partial P^*_{leakage}}{\partial s_i^*} + \alpha_3 \frac{\partial P^*_{dynamic}}{\partial s_i^*},\ i \in (\mathcal{D} \cup \mathcal{G}) \qquad (10)
\end{aligned}
$$

Note that each line in 10 applies to an individual set of components of λ and is independent to the other lines. For example, if a particular vector value λ^* is to be deemed a candidate for the correct, optimal multiplier λ, then all of its outgoing PO multiplier components (from a PO gate to the sink node 0) 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 λ^* which satisfy equation 10 as solution candidates for the correct, optimal multiplier λ, our tuning process can significantly cut down on runtime by avoiding unnecessary computation involving impossible λ candidates.

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

D. Iterative Multiplier Adjustment for Determining Optimal λ

We employ an iterative, modified sub-gradient method [14] for finding the desired λ vector. First, we arbitrarily pick a starting lambda value which satisfies equation (10). For example, we started by assigning each of the λ^A_{j0} to be $\frac{\alpha_1 e^{a_0^*}}{N}$, where N is the number of inputs to sink node 0(the number of actual primary outputs). All other multiplier components were assigned in a similar way via reverse topological order. After an initial λ^* guess was formed, we then iteratively update λ^* using a modified sub-gradient approach shown in Table II, line 3, to form a new guess at every iteration. θ_k is a step size value which was initialized to be 1 and gradually modified over iterations using a Trust-Region approach [15]. We continue to iterate and make new guesses for the correct, optimal value of λ until our $\mathcal{LRS}/\lambda^*$ value converges to that of the PP value, at which point we will have found our desired multiplier λ, which is just equal to the λ^* at the stopped iteration.

E. Solving $\mathcal{LRS}/\lambda$

Our LARTTE algorithm terminates when the solution of $\mathcal{LRS}/\lambda$ converges to that of PP. In order to do this, we must present a method for solving the unconstrained optimization problem in $\mathcal{LRS}/\lambda$ (neglecting simple bound constraints). Since the field of unconstrained optimization is mature [13], we resort to using an off-the-shelf unconstrained solver in L-BFGS-B [16] to do this. L-BFGS-B implements the well-known BFGS-method [13], which has been proven to be exceptional for handling large-scale unconstrained problems with limited memory usage. The efficiency provided by L-BFGS-B contributes largely to the fast runtime of LARTTE.

```
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
         $\lambda^{NEW}_{ji} := \begin{cases} \lambda^A_{ji} * \quad \frac{e^{a_j^*}}{e^{a_0^*}}^{\theta_k} & \text{if } i = 0 \\ \lambda^A_{ji} * \quad \frac{e^{a_j^*} + D_i^*}{e^{a_i^*}}\Big)^{\theta_k} & \text{if } i \in \mathcal{G} \\ \lambda^A_{ji} * \quad \frac{D_i^*}{e^{a_i^*}}^{\theta_k} & \text{if } i \in \mathcal{D} \\ \lambda^S_{ji} * \quad \frac{T_i^*}{e^{s_i^*}}^{\theta_k} & \text{if } i \in (\mathcal{D} \cup \mathcal{G}) \end{cases}$
         Project $\lambda^{NEW}_{ji}$ 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.

F. Vt Discretization and LARTTE Summary

Up to now, we have treated the parameter V_t as a continuously tunable parameter. This was done because the Lagrangian Relaxation technique is a technique for continuously differentiable optimization problems. Obviously, this is a problem because in practice, there are usually only a fixed and limited number of varying V_t devices to choose from(due to fabrication issues). 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 0.17V, 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 continuous V_t solutions.

One may question the validity of this 'solve-continuous-then-discretize' heuristic, since the solution after discretization may no longer correspond to the optimal solution in the original problem. However, as will be shown in our experimental results (Section V), the solution after discretization is actually always very close to the ideal, optimal solution in the original problem. This will be demonstrated to hold even when the number of V_ts to discretize from is small (i.e 4, which was the value used in this work). Hence, our strategy is justifiable and sound.

LARTTE has now been fully presented and is summarized in Table II for clarity.

V. Experimental Results

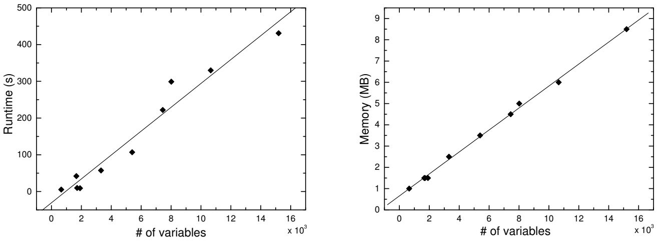
We implemented LARTTE in C/C++ and ran all our experiments on a 1.0GHz P4 machine with 1.0Gb of RAM. The stopping criterion of LARTTE was set to when $\mathcal{PP}$ and $\mathcal{LRS}/\lambda$ agree to within 1.0%. Lower and upper bounds of transistor widths were 0.2 μm and 1.1 μm respectively. For V_t, the lower and upper bounds were 0.14V and 0.26V. V_{DD} was 1.0V and a 0.1 activity factor was used. Input slew ranged from 30 to 150 ps. For multi-V_t selection, (Table III), the four V_t values were made to be available for discretization: 0.14V, 0.18V, 0.22V, and 0.26V. All SPICE simulations were done in 0.1 μm technology with multiple Vt transistor models. We conducted our experiments on the ISCAS85 benchmarks, 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.

A. Optimal Timing and Power Gate-Sizing and V_t Assignment

In Table III, the ‘optimize delay’ columns show the maximum delay before and after tuning, with only timing involved in the objective function (α_1=1, α_2=α_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 optimal-delay value. Hence, the solution obtained from tuning the power consumption will be guaranteed to have a critical path delay not exceeding the optimal delay value shown in the ‘optimize delay’ column. For power tuning, the dynamic and leakage power terms were arbitrarily assigned equal weights (In practice, these weights should be assigned based on application-specific conditions, such as the percentage of time the target circuit spends in idle mode). The resulting optimized power solution from tuning both the transistor widths and V_ts are shown in the ‘optimize total power’ columns. This is compared to the power consumption of the circuit after tuning for delay only (with nominal V_ts). The table shows an average of over 58% total power reduction can be achieved with the same delay target using simultaneous gate-sizing and multi-V_t assignment. The table also shows that LARTTE has a mere linear runtime and memory usage requirement (see Fig. 2 as well). Lastly, in order to justify our strategy of first treating V_t as a continuous variable, then discretizing in the end, we show the leakage power consumption of the various tested circuits before and after discretization in Table III. As expected, the discretized solution is always inferior to the continuous solution. However, it can be seen that the difference in leakage power consumption before and after discretization is relatively trivial in all of the tested circuits. This suggests that our heuristic works fairly well in practice and can result in a solution point which is not too far from the globally optimal solution.

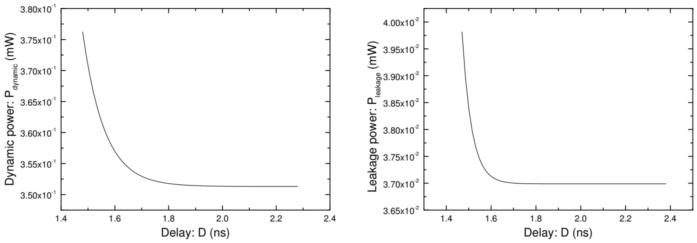
To gauge the effectiveness and runtime of LARTTE, we employ a state-of-the-art general-purpose large-scale convex optimization solver in SNOPT [11] to solve the same primal problem. The runtime results are tabulated in Table III, where it can be seen that our method is over 250x faster. Furthermore, we verified that our LARTTE solution agreed with that from SNOPT to within 1% in all cases. Surveying the literature, we find that another previously-propose sizing-with-Vt-assignment technique [17] took over 1.5 hours to tune a circuit with only 5318 transistors on a Sparc 60. This is obviously much slower than LARTTE, as c7552 has many more components and takes only 7.2 minutes to finish with LARTTE. In [8], it was reported that their concurrent sizing-with-V_t scheme achieves on average 37% total power reduction, which is again inferior to LARTTE. Similarly, in [18], their dual V_t with sizing method can reduce total power by 50% without any timing optimization. As we have shown, LARTTE can achieve a higher power savings on top of delay optimization. Many other works [19] [20] exhibit similar inferiority to LARTTE.

By simultaneously optimizing for delay, dynamic power, and leakage power using varying α weights, LARTTE can also be used to explore several tradeoff relationships between delay, leakage and dynamic power. Fig. 3 shows the dynamic power versus delay and leakage power versus delay optimal tradeoff curves for a 12-bit ALU, and Fig. 4(a) shows the dynamic power versus leakage power optimal tradeoff curve for the same 12-bit ALU. In Fig. 4(b), we show the effects of varying the number of V_ts available for discretization. The circuit used was c432. It can be seen that any more than 4 available V_ts results in minor savings.



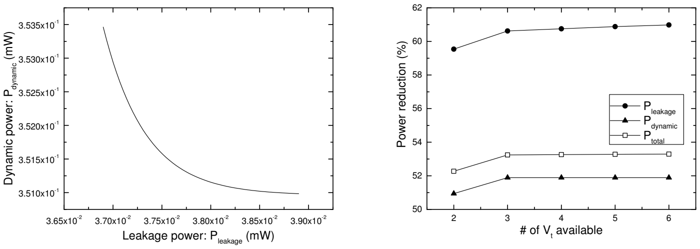
(a) (b)

Fig. 2. The (a) runtime and (b) storage requirements of LARTTE vs. number of variables.



(a) (b)

Fig. 3. Dynamic power vs. delay (a) and Leakage power vs. delay (b) trade-off curves for c2670.



(a) (b)

Fig. 4. (a) Dynamic vs. leakage power trade-off curve for c2670 (b) Effects of variable V_t on power reduction.

VI. Conclusion, Shortcoming, and Future Work

In this work, we made the key contribution of a novel gate-sizing and multi-V_t assignment technique using Lagrangian Relaxation. Our solution is mathematically guaranteed to find the most timing and power-optimal solution point due to the use of accurate, convex posynomial approximations.

Although our experimental results validate the effectiveness of LARTTE, there is currently one shortcoming with our approach that we would like to acknowledge. That is, in the tuning process, the pmos-to-nmos ratio γ was not tunable. We actually statically assigned this ratio for each gate based on sound heuristics involving fan-in count and gate type information. Obviously, not being able to tune γ can non-trivially reduce the optimization space. The reason why this problem exists was because of the way we simulated our SPICE

TABLE III

RESULTS OF OPTIMIZATION ON ISCAS'85 BENCHMARK CIRCUITS

Circuit	# of	# of	# of	Optimize Delay (ps)			Optimize Total Power (0.1mW)						Leakage Power		Memory
Name	Gates	Var.	Constr.	Min. size	Sizing	%	Sizing	Sizing	%	Runtime (s)		Speed	Before	After	(MB)
				nom.-V_t	nom.-V_t		nom.-V_t	multi-V_t		SNOPT	LARTTE	up	Discretize	Discretize	
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

sample data points (vector b in Section III-B) in the posynomial characterization process. Due to time limitation, we had to carry out the thousands of SPICE simulations in such a way that the statically assigned ratio was always inherently enforced. Hence, because our posynomial approximations were generated based on a fixed γ, the tuning process had to also abide by this γ value to preserve accuracy. We intend to correct this issue in a future work by spending more time on the posynomial characterization process and adding in a new constraint $L_{\gamma_i} \leq \gamma_i \leq U_{\gamma_i}$ for each gate i.

REFERENCES

[1] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah, "Jiffytune: circuit optimization using time-domain sensitivities," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 17, no. 12, pp. 1292–1309, December 1998.

[2] 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.

[3] Y. Jiang, S. Sepatnekar, C. Bamji, and J. Kim, "Combined transistor sizing with buffer insertion for timing optimization."

[4] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor sizing for low power cmos circuits," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 15, no. 6, pp. 665–671, June 1996.

[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] N. Tripathi, A. Bhosle, D. Samanta, and A. Pal, "Optimal assignment of high threshold voltage for synthesizing dual threshold cmos circuits," in *VLSI Design, India*, 2001, pp. 227–232.

[7] T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, V. De, and S. Borkar, "Total power optimization by simultaneous dual-vt allocation and device sizing in high performance microprocessors," in *IEEE/ACM DAC*, 2002, pp. 486–491.

[8] A. Srivastava, D. Silvester, and D. Blaauw, "Concurrent sizing, vdd, and v/sub th/ assignment for low power design," in *Design, Automation, and Test in Europe*, 2004, pp. 718–719.

[9] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed. New York: Wiley, 1997.

[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] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhary, R. Panda, and D. Blauuw, "Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing," in *IEEE/ACM DAC*, 1999, pp. 436–441.

[18] P. Pant, R. K. Roy, and A. Chatterjee, "Dual-threshold voltage assignment with transistor sizing for low power cmos circuits," *IEEE Transactions on VLSI Systems*, vol. 9, no. 2, pp. 390–394, April 2001.

[19] M. Ketkar and S. Sapatnekar, "Parameter variations and impacts on circuits and microarchitecture," in *IEEE Conference on Computer-Aided Design*, 2002, pp. 375–378.

[20] D. Nguyen, A. Davare, M. Orshansky, D. Chinnery, B. Thompson, and K. Keutzer, "Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization," in *International Symposium on Low-Power Electronics and Design*, 2003, pp. 158–163.