Revisiting the Linear Programming Framework for Leakage Power vs. Performance Optimization

Kwangok Jeong[‡], Andrew B. Kahng[†‡], Hailong Yao[†]
[†]CSE and [‡]ECE Departments, University of California at San Diego
kjeong@vlsicad.ucsd.edu, abk@cs.ucsd.edu, hailong@cs.ucsd.edu

Abstract— This paper revisits and extends a general linear programming (LP) formulation to exploit multiple knobs such as multi-L_{gate} footprint-compatible libraries and post-layout L_{gate}-biasing to minimize total leakage power under timing constraints. We minimize positive timing slack for each cell according to its leakage vs. delay sensitivity, so that unnecessary leakage power consumption is saved without degrading circuit performance. A key difference between our work and previous works is that we pre-process timing libraries to estimate the linear relation – in every slew-load condition – between the gate delay and gate length by linear fitting; we then optimize total leakage power by estimating the optimal gate length for each gate using fast linear programming. With a 65GP industry testbed, and directly comparing with commercial tools, we show the QOR and runtime advantages of our method for the multi-L_{gate} and L_{gate}-biasing knobs. We also show a promising application to circuit timing legalization, a problem which frequently arises when implementation and signoff timers differ. Overall, our results show strong viability of LP based estimation and optimization: compared with the commercial tools, we: (1) shift the achievable delay-leakage tradeoff curve in a positive way, and (2) more accurately maintain prescribed timing constraints.

Keywords— Leakage power, timing, linear programming, multi-L_{gate}, L_{gate}-biasing

I. Introduction

The speed, leakage power and dynamic power attributes of a design – and hence the design's parametric yield as well – are ultimately determined by the circuit parameters of supply voltage, threshold voltage, gate length, and gate width. Multi-V_{dd}, multi-V_{th}, multi-L_{gate}, and width sizing are all valuable knobs for dynamic and leakage power reduction. How to use these knobs to optimize the tradeoff of speed, area and power metrics is the *sizing problem*. The sizing problem arises at all stages of the RTL-to-GDS implementation flow, and even beyond. Gupta et al. proposed an effective post-layout, post-signoff gate length biasing technique for parametric yield (leakage and leakage variability) optimization [1].

A rich literature [2-10] addresses the problem of gate sizing with delay, area and power as either objective or constraints. In general, techniques for the sizing problem maximize the conversion of positive timing slack on non-critical timing paths into reductions of area, dynamic power, and/or leakage power. For example, leakage optimization under timing constraints effectively assigns gates on critical paths to operate at high speed (but, with high leakage), while gates on non-critical paths operate at low speed (and, with low leakage).

The works of [11-14] propose heuristic algorithms for simultaneous optimization of V_{dd}, V_{th}, and transistor or gate size with pre-defined sizing options. As with many sizing works, a *sensitivity function*, such as the ratio of incremental power to incremental delay, is typically used, often along with pre-budgeted timing requirements for each cell or path. Chou et al. [15] investigate gate sizing and V_{th} assignment using the Lagrangian relaxation framework previously investigated by [5], [16]. Sarrafzadeh et al. [17] present a convex programming based delay budgeting algorithm and use the budgeting results as net length constraints for placement. Fung et al. [18] present a slack allocation algorithm which computes both lower and upper bounds of the delay budget for each circuit connection, and then applies the resulting delay budget in a router for timing improvement. Luo et al. [19] propose linear programming based placement, geometric programming based gate sizing and multi-V_{th} cell swapping algorithms to achieve power reduction. Chinnery et al. [20] propose a linear programming method for power optimization using sizing, V_{th} and V_{dd} assignment, where a 0-1 cell choice variable determines whether an alternative functionally equivalent cell is chosen. A limitation of [20] is that the LP solver chooses between only one pair of cell alternatives each time, and multiple iterations of the LP program are needed, whereby no claims of solution optimality can be made.

Commercial iterative sizing heuristics (all major implementation platforms such as Synopsys IC Compiler, as well as Prolific, Blaze, etc.) can be viewed simplistically as "swapping methods", whereby variants of a given cell – such as HVT / NVT / LVT – may be substituted for any given cell instance in the netlist. Performing such footprint-compatible swapping after final routing, either before (ICC), in tandem with (Prolific), or after (Blaze) signoff analysis, is an increasingly important aspect of the implementation flow. For this optimization, richer variant libraries give more fine-grain control, and consequently more leakage reduction. In the academic literature, [28] proposes the simultaneous use of L_{gate}-biasing and multi-V_{th} (threshold assignment) techniques to achieve stronger power reduction.

The works of [21-23] present a rough model to decide optimal granularities of multiple voltage supplies (for dynamic power reduction) and multiple threshold voltages (for leakage power reduction). The granularities are determined from an analytical calculation with simplified path timing information and an I_{on}/I_{off} model. The authors of [24] use a similar analytical method, but find the optimal granularity of supply and threshold voltages when considering both dynamic and leakage power simultaneously. While these works ostensibly help decide the granularities of optimization knobs, they are based on very simple assumptions about the design, e.g., a triangular path delay distribution model in [24].

In this paper, we present fast and high-quality linear programming estimators and optimizers for (i) leakage power minimization under timing constraints, and (ii) simultaneous circuit timing legalization and leakage power optimization considering multiple knobs of multi-L_{gate} (multiple choices of gate lengths for a given gate) and fine-grain L_{gate}-biasing (gate length biasing from the nominal value).[1] The LP formulation for power-performance optimization is not novel in the literature, but we revisit the problem formulation with new ideas for simultaneous timing and leakage optimization and obtain better QOR and runtime than current commercial tools. A key difference between our work and previous works is that we pre-process timing libraries to estimate the linear relation – in every slew-load condition – between the gate delay and gate length by linear fitting. Also, we optimize total leakage power by estimating the optimal gate length for each gate using fast linear programming (the gate *lengths* are a consequence of maximizing leakage sensitivity-weighted total gate *delay* increase in the circuit). LP can introduce rounding errors when mapping the computed gate length values to available variant cell masters, and hence an integer linear programming (ILP) approach can be more accurate. However, the runtime of ILP is impractical. The main contributions of our work are as follows.

- We present a linear programming (LP) based leakage power minimization method which observes timing constraints and quickly estimates the total leakage power reduction achievable using multi-L_{gate} or other technology options.
- We present an LP based circuit timing *legalization* method which simultaneously minimizes the number of paths violating the given delay constraints and the total leakage power. This is particularly important when 'unrolling' the moves made by a previous optimizer when its internal timer deviates from the golden signoff timer.
- We empirically test the LP formulation for multi-L_{gate} selection and L_{gate}-biasing optimization for circuit timing legalization and leakage power optimization.
- We show strong viability of the LP approach, especially for post-layout 'swapping' optimizations, evidenced by improved QOR, improved accuracy, and comparable runtime versus commercial leakage power optimization tools.

[1] In 65nm and 45nm low-power designs, a multi-L_{gate} methodology is increasingly used – cf. Texas Instruments LLPOLY or STMicroelectronics COR-L libraries – especially as V_{th} variability and mistrack limits the benefits of multi-V_{th}. Although multi-L_{gate} libraries can cause increase in dynamic power [25], the increase is typically much smaller than the leakage power reduction, so that the total power consumption can be improved.

Our paper is organized as follows. Section 2 gives linear programming formulations for leakage power minimization and circuit timing legalization, using multiple knobs of multi-L_{gate} and fine-grain L_{gate}-biasing. Section 3 describes the overall flow of the leakage power optimization and circuit timing legalization process. Section 4 presents and discusses experimental results, and Section 5 presents conclusions and ongoing work.

II. Problem Formulation

A. Preliminaries

Linear approximation of gate delay vs. gate length. Figure 1 shows the fitted curves of cell rise delay vs. gate length from Liberty delay tables of a two-input AND gate in 65GP technology. In the Liberty timing library format, delays for each cell master are given as two-dimensional delay tables indexed by input slew and load capacitance values. In Figure 1, seven different lines are plotted for seven different load capacitance values and one input slew value. Each point in the figure denotes a delay value for the specific timing arc. From Figure 1, we see that the cell delay is approximately linear in gate length, as expected. Our background studies have examined such figures for both cell rise delay and cell fall delay in all timing arcs for more than 50 different types of cell masters, across 26 characterized timing libraries (50nm to 75nm). Additionally, SPICE simulations were performed on 70nm implementations of library cells. These background studies confirm the stability of the linear relationship between cell delay and transistor gate length, as is proposed in [26].

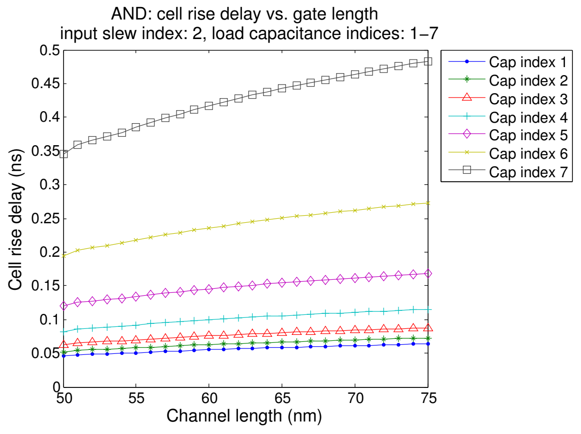

Fig. 1. Rise delay of A-Z arc in an AND2 gate, vs. transistor gate length.

When the gate length changes, the off-path loading, slew propagation, and crosstalk timing windows can all change, and this also affects topologically adjacent gates. However, when the gate length changes within a small range, these effects are typically assumed negligible (e.g., [1]). Based on the above observations, we calibrate coefficients between delay of each timing arc and the transistor gate length, for a given cell master, as

$$d = \alpha \times L_g + \beta \qquad (1)$$

where d is the delay of the timing arc, L_g is the gate length, and α and β are the delay coefficients for the timing arc of the cell. By fitting these coefficients across all Liberty delay tables (total 26 sets of tables corresponding to 50nm, 51nm, ..., 75nm transistor gate lengths), we can obtain an accurate mapping – for every slew-load context – between delay change of the timing arcs and gate length change in the cell master. For example, with 7×7 delay tables for a cell master with two input pins and one output pin, there will be 49×4 (α, β) pairs corresponding to all combinations of input slew and load capacitance values.[2]

Circuit delay calculation. Our circuit delay representation is based on the standard *timing graph*. A given netlist's pin-based timing graph may be modeled as a directed acyclic graph (DAG) with vertex set V and edge set E. Each vertex v corresponds to a cell instance, and each directed source-sink arc within a hyperedge (signal net) is captured as an edge e. In the following, v refers to either a vertex or a cell instance, and e refers to either an edge or a signal net. A timing path is composed of a series of vertices and edges, starting from a primary input or sequential cell output and ending at a primary output or sequential cell input.

Following classic approaches, and without loss of generality, we introduce a super-source S that is connected to all the initial vertices (primary inputs or sequential cell outputs) of timing paths, and a super-sink T that is connected to terminal vertices (primary outputs or sequential cell inputs) of timing paths, as in [5]. Each edge $e_{u,v}$ connecting u and v in the DAG has a delay value denoted as $w_{u,v}$, representing the delay value of the corresponding signal net. Each vertex v in the DAG has delay values corresponding to different timing arcs of the cell instance. If vertex u connects to the input side of v, then d_v^u represents the delay of the (rise/fall) timing arc on v from the output side of u to the output side of v. When the delay values of all timing arcs of a given cell instance v are equal, d_v can be used to denote the delay value of the timing arcs. Let d_{v0}^u be the original delay of the timing arc before leakage optimization. Let a_v be the arrival time from the super-source S to the 'output side' of v. Then a timing graph may be constructed and all timing constraints can be enforced on the timing graph.

Figure 2 shows an example of a timing path between two flip-flops A and B in the timing graph, where A and B are represented as super-source S and super-sink T, gates C and D are represented as vertices u and v, vertex u connects to the input side of v, the delay of the (rise/fall) timing arc on v from the output side of u to the output side of v is represented as d_v^u, and the arrival times of vertices S, T, u and v are marked as a_S, a_T, a_u and a_v, respectively.

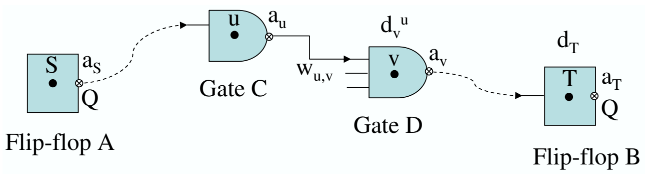

Fig. 2. Example timing path in the timing graph.

In the following subsections of the paper, the linear programming formulations for leakage power optimization and circuit delay legalization are based on the above definitions and assumptions related to the timing graph.

B. LP for Leakage Power Optimization

We first consider the effect of L_{gate}-biasing (gate length biasing [1], [34]) on leakage power optimization. Our objective is to minimize the (leakage-weighted) positive timing slack for non-critical cell instances, which are the cell instances on non-timing critical paths, such that maximum leakage reduction is obtained without degrading overall circuit performance. To avoid getting worse circuit timing after increasing the delays of non-timing critical cell instances, a variable D is introduced to represent the maximum delay bound on all timing paths, which acts as the clock period constraint. The value of D can be obtained from golden timing analysis reports; we use *Synopsys PrimeTime* (version Z-2006.12) [35]. D can be adjusted for different purposes. A smaller D value can result in even better timing of the circuit, as can be observed in the experimental results provided in Section 4. The leakage power optimization problem is formulated as

$$\begin{aligned}
&\textbf{Maximize:} \quad \sum_{v \in V} (\lambda_v \cdot x_v) \\
&\textbf{Subject to:} \\
&\quad a_S = 0 \\
&\quad d_T = 0 \\
&\quad a_T \le D \\
&\quad a_u + w_{u,v} + d_v^u \le a_v \quad (\forall e_{u,v} \in E \text{ and } u, v \in V) \\
&\quad d_v^u = d_{v0}^u + \alpha_v^u \cdot (x_v - L_{v0}) \\
&\quad minL_v \le x_v \le maxL_v \quad (\forall v \in V)
\end{aligned} \qquad (2)$$

Here,
- d_T is the delay of (super-sink) vertex T;
- a_S, a_T, a_u and a_v are respectively the arrival times at the output sides of the corresponding super-source, super-sink, and cell instances;
- D is the maximum delay bound as discussed above;
- $w_{u,v}$ is the delay value of the net connecting u and v;
- d_v^u is the delay of the timing arc of v from u;
- d_{v0}^u is the original delay of the timing arc;
- x_v is the gate length of the cell instance v after leakage optimization;

[2]There are two timing arcs (rise and fall) corresponding to each pair of input and output pins. Thus, for a production library, the total number of (α, β) pairs will be $49 \times$ the total number of timing arcs of all the cells in the library; our C++ program for extracting data from golden Liberty and calibrating the coefficients by linear regression takes around 27 seconds. The (α, β) data occupies a few hundred kB on disk and in runtime memory footprint.

- L_{v0} is the original gate length of v;
- α_v^u is the calibrated delay coefficient (as in Equation (1)) of the timing arc from u to the output side of v;
- $minL_v$ and $maxL_v$ are respectively the minimum and maximum allowable gate lengths for a given cell instance v based on available variant cell masters; and
- λ_v is power vs. delay sensitivity coefficient of v, which can be obtained via characterized timing and leakage libraries. With lookup table-based timing libraries we can compute the average value of delta leakage power per delta cell delay (using the center value in the table) for all the cell variants in the standard cell libraries. Intuitively, more leakage reduction can be expected by increasing the gate lengths (thereby increasing the delay of the timing arcs) of cell instances with larger leakage vs. delay sensitivity values.

In the above problem formulation, original delay values of all timing arcs (d_{v0}^u) and the signal nets ($w_{u,v}$) are obtained beforehand from golden timing analysis reports. To reduce errors arising from coefficient calibration, we avoid coefficient β in Equation (1) for computing the delay of a given timing arc as follows. Given the new gate length, the original gate length and the original delay of the timing arc, the new delay value of the timing arc is computed as in Equation (2). In our implementation, we use the maximum delay coefficient over both rise and fall arcs. For better accuracy, the delay coefficient can be selected according to the status of the original timing arc (that is, either rise or fall), which can be obtained from golden timing analysis reports.

Note that the computed gate length values in the above problem formulation are real values, which need to be rounded to match the available variant cell masters in the characterized standard-cell timing libraries. In spite of the rounding errors, our LP based leakage optimizer is still very effective, as confirmed by the experimental results in Section 4. The above formulation differs from previous work in that we directly maximize the amount of delay related to gate lengths (as a proxy for leakage reduction) that is added into the timing graph without violating delay upper bounds.[3]

C. LP for Circuit Timing Legalization

In this subsection, we investigate *circuit timing legalization* as another application of our LP based framework. Given a design with timing violations, we seek to improve the worst negative slacks of the design with minimum leakage increase. This is a difficult problem that has not been well-studied in the literature; yet, it arises in almost every production power minimization flow today due to mismatch between internal and golden STA engines or limitations in optimization methods. In other words, due to timer mismatches, leakage optimization tools either guardband the optimization (leading to poor QOR), or deliver netlists that are not timing-clean according to the golden timer. Starting from a leakage-optimized design that has timing violations, we want to push the worst path delays down, and variable D is now treated as a *lower bound*. We formulate the problem of legalizing all violating paths as

$$\begin{aligned}
&\textbf{Minimize: } \gamma \times a_T - \sum_{v\in V} (\lambda_v \cdot x_v)\\
&\textbf{Subject to:}\\
&\quad a_S = 0\\
&\quad d_T = 0\\
&\quad a_T \geq D\\
&\quad a_u + w_{u,v} + d_v^u \leq a_v \ (\forall e_{u,v} \in E \text{ and } u,v \in V)\\
&\quad d_v^u = d_{v0}^u + \alpha_v^u \cdot (x_v - L_{v0})\\
&\quad minL_v \leq x_v \leq maxL_v \quad (\forall v \in V)
\end{aligned} \tag{3}$$

where

- d_T, a_S, a_T, a_u and a_v are as defined above;
- $w_{u,v}$ is the delay of the net connecting u and v;
- d_v^u is the delay of the timing arc of v from u;
- d_{v0}^u is the original delay of the timing arc;
- x_v is the gate length of v after leakage optimization;
- L_{v0} is the original gate length of v;
- α_v^u is the calibrated delay coefficient of the timing arc related to u and v;
- $minL_v$ and $maxL_v$ are respectively the minimum and maximum allowable gate lengths for v;
- λ_v is the power vs. delay sensitivity coefficient defined as above;
- γ is a user-specified scaling parameter such that the first item ($\gamma \times a_T$) dominates the second one ($\sum_{v\in V}(\lambda_v \cdot x_v)$); and
- D is the variable representing the *lower bound* on the circuit delay. Again, the path delay bound D can be obtained from golden STA reports. Note that the smaller the value of D, the better the resulting circuit delay and the worse the total leakage power. A useful methodology degree of freedom that we have not fully explored is to allow tuning of the value of D (within a 5% range of the actual circuit delay value reported by golden STA) to balance between circuit delay and total leakage power.

There are two objectives in the above problem formulation: (1) minimize the delay of all paths according to the given delay lower bound D, and (2) maximize the total weighted gate lengths of the cell instances (thereby maximizing the total leakage weighted delay to minimize total leakage power). On the one hand, as mentioned above, the first objective dominates the second one so that the delay values of timing critical cells will not be increased for leakage power optimization if it results in worse total circuit delay. Again, delay values of timing critical

[3]Chen et al. [27] propose a power reduction method based on gate sizing, where the objective is to select a set of gates in a circuit that can be slowed down, by replacing the gates with cells in the cell library having smaller area and thus smaller capacitance load, without violating the timing constraints, and in the meantime minimizing the power consumption. However, since Chen et al. [27] use slack values of cells that are affected by other cells' sizing results, they must resize a cell iteratively and update timing at every iteration. Chuang et al. [29] also use a similar LP formulation, where the objective is to minimize the total area of the circuit.

cells will be decreased to minimize the total circuit delay, at the cost of leakage power increase. On the other hand, the delay values of non-timing critical cells will still be increased to save leakage power without degrading the circuit performance. Experimental results in Section 4 confirm our intuition: we obtain total leakage power after legalization that is smaller than that of the original design, along with better circuit performance after the legalization process.

The above problem formulations are linear, and efficient linear programming methods can be used to obtain solutions. Specifically, we use CPLEX [30] in our experimental platform described below.

III. Timing and Leakage Optimization Flow

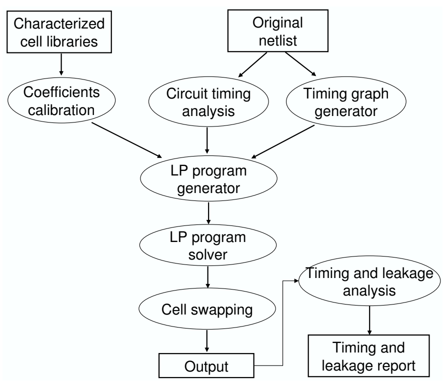

Fig. 3. The overall optimization flow.

Figure 3 summarizes the flow of our LP based timing and leakage optimization. Input consists of characterized standard-cell timing libraries, along with original netlist timing constraints and extracted wire parasitics. We assume table-based nonlinear delay modeling in the standard-cell timing libraries. As described above, we fit the coefficients of the gate delay function for each entry of the lookup table, which is a combination of input slews and load capacitances. Then, we use *Synopsys PrimeTime* (version Z-2006.12) [35] to obtain input slews, load capacitances and original delay values for each cell instance in the post-routed design. Wire delay values are also obtained from the timing analysis report. According to the input slew and load capacitance values obtained in the timing analysis step, the delay coefficients for timing arcs of each cell instance, α and β in Equation (1), can be calculated based on the nearest entry in the delay table. With the coefficients obtained, the delay values of the timing arcs can be estimated as in Equations (2) and (3). When the new gate length values are computed, a different cell master, which is the same type of cell with different nominal gate length, can then be instantiated to replace the cell instance.

In the timing graph generation, sequential circuits are processed as combinational circuits using standard techniques that traverse from primary inputs and sequential cell outputs, to sequential cell inputs and primary outputs. Then, the timing graph is constructed (with fictitious source node connecting to all primary inputs, and fictitious sink node connecting from all primary outputs) as described in Section 2.A.

Golden STA reports yield the delay bound D that we use in our LP formulations. Note that different values of D can be used for different optimization objectives. For leakage power optimization, D can be the same as the actual circuit delay value. However, for circuit timing legalization, D can be less than the actual circuit delay value, so that total circuit delay can be reduced (or legalized). Given the timing graph and the delay bound value D, the LP program is generated and solved using CPLEX [30]. Then the LP-computed gate length values for each cell instance are rounded to the nearest matching cell master, and the netlist is updated. Finally, the timing report including worst negative slack (WNS), total negative slack (TNS) and the leakage information is obtained using signoff timing and leakage analysis (*Synopsys PrimeTime* (version Z-2006.12) [35] and Cadence *SOC Encounter* (v05.20) [31], respectively).

IV. Experimental Results and Discussion

A. Experimental Setup

TABLE I
CHARACTERISTICS OF ARTISAN TSMC 65NM DESIGNS.

Design	Block Size (mm^2)	#Cell Instances	#Nets
AES	0.08	16989	17651
JPEG	0.39	86782	91479
JPEG3	1.16	255247	269330

Library preparation. We first prepare libraries to verify our LP based optimization techniques. For the initial library, we use 65GP libraries from TSMC. We make different L_{gate} variant libraries from the initial library to test our multi-L_{gate} and L_{gate}-biasing techniques. For multi-L_{gate} experiments, 50nm and 70nm L_{gate} libraries are prepared. For L_{gate}-biasing, 5 negatively biased libraries (55nm, ..., 59nm libraries) and 5 positively biased libraries (61nm, ..., 65nm libraries) are generated around the 60nm L_{gate} library. In total, we have 13 different L_{gate} libraries for evaluating our LP based timing and leakage optimization. The other L_{gate} variant libraries from 50nm to 75nm are prepared similarly around the 50nm and 70nm L_{gate} libraries for the study of the linear relationship of gate delay vs. gate length as discussed in Section 2.A. We use "L{gate length}" as a synonym for the specific library. For example, "L60" refers to the library with nominal L_{gate} of 60nm.

L_{gate} variant libraries are obtained by using scaling factors for each rise delay, fall delay, rise transition, fall transition and state-dependent leakage power. These are calculated from SPICE simulation on a representative inverter circuit. The library preparation process takes substantial time, but this is a one-time cost. Since the prepared libraries are used by all the different tools including our LP based methods, the runtime comparison between the tools is not affected by the library preparation process, and hence the runtime of library preparation is not included in the experimental results.

Design of experiments. For testcases, we use *AES* and *JPEG* from *opencores.org*, and to assess the scalability of our LP based optimization methods, we design an artificial core *JPEG3* which instantiates the *JPEG* core three times. The target clock period value is 2.0ns for all three testcases. Table I shows the parameters of the testcases, including the block size, the number of cell instances and the number of nets.

To evaluate our LP optimizer on various knobs of leakage optimization, we study the following scenarios that can arise in practice.

- **L_{gate}-biasing.** Optimize leakage power of an original design implemented in L60, using L_{gate}-biasing libraries.
- **Multi-L_{gate}.** Optimize leakage power of an original design implemented in L60, with L50 and L70 libraries.

To evaluate the quality of timing legalization, we use the testcases after leakage optimization by a commercial tool, Cadence *SOC Encounter* (v05.20) [31], where much more timing violations are introduced during the leakage optimization process. We cure the timing slack by our LP based timing legalizer, maintaining leakage power as small as possible.

We have implemented our leakage optimization and circuit timing legalization methods in C++. Each testcase is synthesized from RTL using Cadence *RTL Compiler* (v05.10) [33], scan-inserted using Synopsys *Design Compiler* (vY-2006.06-SP5) [32], and placed and routed using *SOC Encounter* [31]. In the experimental results, we optimize leakage power using *SOC Encounter* and Synopsys *Astro* (vY-2006.06-SP5) [36] in each scenario, and compare the results with our LP optimizer.

B. Results and Discussion

Leakage power optimization by L_{gate}-biasing. Our first experiment obtains leakage optimization results from our LP based optimizer and commercial tools, *SOC Encounter* and *Astro*, with L_{gate} as the sole degree of freedom. Thus, our context for optimization is similar to that of the Blaze MO leakage optimization tool [34]. In the experiments, original designs ("original") are synthesized, placed and routed with nominal L_{gate} (60nm) library. The leakage optimization uses the nominal library as well as libraries with positive and negative L_{gate} (CD) biases, which are layout-swappable cell variants that are slower but less leaky, or faster but more leaky, than the nominal library masters. The maximum available L_{gate} biases are $+5$nm (corresponding to 65nm) and -5nm (corresponding to 55nm). This model is of interest when exploring limits of the tradeoff between frequency and (leakage) power. Intuitively, the use of negatively-biased cells on the critical (max) path will increase frequency, and at the same time create slack on other paths that can be exploited with use of positively-biased cells. It may be possible in some cases to find a solution that has both higher frequency and lower leakage power than the original netlist.

In Table II, our results from LP based optimization are better in terms of solution quality than those from the commercial tools, but are achieved with substantially smaller runtime especially when compared with *SOC Encounter*. Library preparation time is not included in the runtime comparison because all tools use the same set of prepared libraries, as discussed in Section 4.A. From the results, both *SOC Encounter* and *Astro* obtain leakage power reduction at the cost of timing degradation. By contrast, our LP based leakage power optimizer can always obtain leakage power reduction with better timing. For example, for *AES*, the worst negative slack (WNS) and the total leakage power in the original design are -0.065ns and 360.5uW, respectively. *SOC Encounter* reduces the leakage power to 293.5uW with degraded worst negative slack -0.135ns. *Astro* reduces the leakage power to 304.2uW with degraded worst negative slack -0.079ns. Our LP based optimization effectively reduces the leakage power to 294.6uW with improved worst negative slack -0.062ns. The runtime of LP based optimization is 25.9s, approximately $16\times$ faster than *SOC Encounter*'s runtime of 425.7s.

Leakage power optimization by multi-L_{gate}. Our second experiment studies leakage power optimization in a multi-L_{gate} regime. Table III shows that our LP based optimization again obtains improved leakage power and better timing slack values versus the original design. Although the commercial tools can obtain more leakage power reduction, much worse timing slacks are again observed. For example, for *JPEG*, the worst negative slack and the total leakage power in the original design are -0.109ns and 2576.2uW, respectively. *SOC Encounter* reduces the leakage power to 2304.2uW with degraded worst negative slack -0.164ns. *Astro* reduces the leakage power to 2253.6uW with degraded worst negative slack -0.166ns. Our LP based optimization effectively reduces the leakage power to 2365.4uW with improved worst negative slack -0.093ns. The runtime of LP based optimization is 312.5s, approximately $5\times$ faster than *SOC Encounter*'s runtime of 1701.2s. Comparing with L_{gate}-biasing results, we observe that multi-L_{gate} optimization is less effective (e.g., for *JPEG*, L_{gate}-biasing improves the leakage power to 2180.0uW vs. 2365.4uW using multi-L_{gate}), perhaps due to larger rounding errors when mapping computed gate length values to a smaller number of available cell masters.

Timing legalization by L_{gate}-biasing. Our third experiment studies timing legalization using L_{gate}-biasing. Though there have been many works on cell-swapping based leakage optimization, with standard flows implemented with commercial tools, leakage optimization frequently hurts the timing of original design. This is likely due to the suboptimality of the tools and the discrepancy of extraction/timing between optimization tools and signoff analysis tools. From the results in Table II and Table III, we again see such a problem.

Our LP based timing legalization efficiently cures timing errors to recover the original signoff timing. Table IV shows the timing and leakage power of (i) the original design before the

TABLE II

LP BASED LEAKAGE POWER OPTIMIZATION USING L_{gate}-BIASING.

Design	WNS (ns)				TNS (ns)				Leakage (uW)				Runtime (s)		
	original	SOCE	ASTRO	LP	original	SOCE	ASTRO	LP	original	SOCE	ASTRO	LP	SOCE	ASTRO	LP
AES	-0.065	-0.135	-0.079	-0.062	-0.690	-5.721	-1.184	-1.216	360.491	293.542	304.212	294.597	425.7	316.7	25.9
JPEG	-0.109	-0.170	-0.110	-0.103	-8.650	-20.586	-10.066	-14.977	2576.213	2271.080	2466.459	2179.962	3744.9	427.0	323.4
JPEG3	-0.155	-0.329	-0.166	-0.151	-38.473	-133.168	-43.170	-77.118	7444.126	6283.386	7166.493	6272.001	14568.6	1473.2	1753.7

TABLE III

LP BASED LEAKAGE POWER OPTIMIZATION USING MULTI-L_{gate}.

Design	WNS (ns)				TNS (ns)				Leakage (uW)				Runtime (s)		
	original	SOCE	ASTRO	LP	original	SOCE	ASTRO	LP	original	SOCE	ASTRO	LP	SOCE	ASTRO	LP
AES	-0.065	-0.129	-0.098	-0.051	-0.690	-10.084	-1.725	-2.353	360.491	290.316	274.884	304.634	284.5	62.2	19.9
JPEG	-0.109	-0.164	-0.166	-0.093	-8.650	-16.635	-16.460	-1.267	2576.213	2304.193	2253.639	2365.425	1701.2	200.9	312.5
JPEG3	-0.155	-0.333	-0.215	-0.154	-38.473	-151.103	-64.238	-49.815	7444.126	6347.532	6599.442	6535.158	16104.1	735.0	1449.1

TABLE IV

LP BASED TIMING LEGALIZATION USING L_{gate}-BIASING.

Design	Before leakage optimization			After leakage optimization by SOCE			After timing legalization by LP			
	WNS (ns)	TNS (ns)	Leakage (uW)	WNS (ns)	TNS (ns)	Leakage (uW)	WNS (ns)	TNS (ns)	Leakage (uW)	Runtime (s)
AES	-0.065	-0.690	360.491	-0.135	-5.720	293.542	-0.064	-1.165	298.281	17.4
JPEG	-0.109	-8.650	2576.213	-0.170	-20.586	2271.080	-0.105	-7.856	2293.995	327.5
JPEG3	-0.155	-38.473	7444.126	-0.329	-133.168	6283.386	-0.149	-23.976	6554.970	1275.1

TABLE V

LP BASED TIMING LEGALIZATION USING MULTI-L_{gate}.

Design	Before leakage optimization			After leakage optimization by SOCE			After timing legalization by LP			
	WNS (ns)	TNS (ns)	Leakage (uW)	WNS (ns)	TNS (ns)	Leakage (uW)	WNS (ns)	TNS (ns)	Leakage (uW)	Runtime (s)
AES	-0.065	-0.690	360.491	-0.129	-10.084	290.316	-0.051	-2.333	346.364	17.8
JPEG	-0.109	-8.650	2576.213	-0.164	-16.635	2304.193	-0.101	-11.211	2338.479	312.6
JPEG3	-0.155	-38.473	7444.126	-0.333	-151.103	6347.532	-0.151	-86.648	6506.326	1266.3

leakage power optimization process, (ii) the design after leakage optimization by *SOC Encounter*, and (iii) after our LP based timing legalization. We observe that the SOCE leakage optimization process worsens timing slack, but our LP based timing legalization returns all timing slacks to original or even better values; at the same time, the total leakage power is still smaller than the original. For example, for *AES*, the worst negative slack -0.135ns from *SOC Encounter* is returned to -0.064ns with leakage power 298.3uW, which is better than the original design with worst negative slack -0.065ns and leakage power 360.5uW.

Timing legalization by multi-L_{gate}. Our final experiment studies timing legalization using multi-L_{gate}. Table V shows that our LP based timing legalization again successfully turns back the worst timing slacks to be even better than the original design. At the same time, total leakage power is also improved in all testcases. For example, for *JPEG3*, the worst negative slack -0.333ns from *SOC Encounter* is returned to -0.151ns with leakage power 6506.3uW, which is better than the original design with worst negative slack -0.155ns and leakage power 7444.1uW. The fact that the circuit performance for all the testcases can be recovered to the original frequency confirms that our LP based circuit timing legalization is an effective complement to current leakage optimization tools.

V. Conclusions and Ongoing Work

In this paper, we have revisited the LP based circuit delay legalization and leakage power optimization framework, proposed the relevant problem formulations, and compared our methods with commercial tools, for both L_{gate}-biasing and multi-L_{gate} optimization scenarios. Linear modeling and LP based optimization have not gained traction in commercial sizing/swapping tools, mostly because of modeling inaccuracy and concerns about slew change effects, etc. However, we show that for the applications we study, linear modeling (pre-fitted from Liberty) is effective and practical. Experimental results show that our methods enable very fast, high-quality power-delay tradeoff estimation and optimization: we improve QOR, accuracy, and runtime significantly over the commercial tools. Most current commercial leakage power optimization tools suffer QOR and timing violation challenges due to miscorrelation between internal and golden timing engines; our circuit timing legalization circumvents this issue. Our ongoing work includes the testing of the proposed circuit delay and leakage power optimization methods on larger industrial testcases. We are also improving our flow to handle SI signoff via appropriate margining and don't-touch methodologies. Finally, we are growing the scope of the optimization to combine multiple knobs of multi-V_{th}, multi-L_{gate} and L_{gate}-biasing, as well as to address hold time, multi-mode/multi-corner, and dynamic/total power constraints.

REFERENCES

[1] P. Gupta, A. B. Kahng, P. Sharma and D. Sylvester, "Gate-Length Biasing for Runtime-Leakage Control", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 25(8) (2006), pp. 1475-1485.

[2] J. Fishburn and A. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor-Sizing", *Proc. IEEE/ACM International Conf. on Computer-Aided Design*, 1985, pp. 326-328.

[3] S. S. Sapatnekar, V. Rao, P. Vaidya and S. Kang, "An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(11) (1993), pp. 1621-1634.

[4] W. Chuang, S. Sapatnekar and I. Hajj, "Delay and Area Optimization for Discrete Gate Sizes Under Double-Sided Timing Constraints", *Proc. IEEE Custom Integrated Circuits Conf.*, 1993, pp. 9.4.1-9.4.4.

[5] C.-P. Chen, C. C. N. Chu and D. F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 18(7) (1999), pp. 1014-1025.

[6] S. S. Sapatnekar and W. Chuang, "Power-Delay Optimizations in Gate Sizing", *ACM Trans. on Design Automation of Electronic Systems*, 5(1) (2000), pp. 98-114.

[7] S. P. Boyd, S. J. Kim, D. D. Patil and M. A. Horowitz, "Digital Circuit Optimization via Geometric Programming", *Operations Research*, 53(6) (2005), pp. 899-932.

[8] A. Srivastava and D. Sylvester, "Minimizing Total Power by Simultaneous V_{dd}/V_{th} Assignment", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23(5) (2004), pp. 665-677.

[9] A. Srivastava, D. Sylvester and D. Blaauw, "Power Minimization Using Simultaneous Gate Sizing, Dual-Vdd and Dual-Vth Assignment", *Proc. IEEE/ACM Design Automation Conf.*, 2004, pp. 783-787.

[10] S. Joshi and S. Boyd, "An Efficient Method for Large-Scale Gate Sizing", *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, 55(9) (2008), pp. 2760C2773.

[11] P. Pant, R. K. Roy and A. Chatterjee, "Dual-Threshold Voltage Assignment with Transistor Sizing for Low Power CMOS Circuits", *IEEE Trans. on VLSI Systems*, 9(2) (2001), pp. 390-394.

[12] P. Pant, V. K. De and A. Chatterjee, "Simultaneous Power Supply, Threshold Voltage, and Transistor Size Optimization for Low-Power Operation of CMOS Circuits", *IEEE Trans. on VLSI Systems*, 6(4) (1998), pp. 538-545.

[13] 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", *Proc. International Symp. on Low Power Electronics and Design*, 2003, pp. 158-163.

[14] S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhury, R. Panda and D. Blaauw, "Stand-by Power Minimization through Simultaneous Threshold Voltage Selection and Circuit Sizing", *Proc. IEEE/ACM Design Automation Conf.*, 1999, pp. 436-441.

[15] H. Chou, Y.-H. Wang and C.-P. Chen, "LARTTE: A Posynomial-Based Lagrangian Relaxation Tuning Tool for Fast and Effective Gate-Sizing and Multiple Vt Assignment", *Workshop on Synthesis And System Integration of Mixed Information technologies*, 2004.

[16] H. Tennakoon and C. Sechen, "Gate Sizing Using Lagrangian Relaxation Combined with a Fast Gradient-Based Pre-processing Step", *Proc. IEEE/ACM International Conf. on Computer Aided Design*, 2002, pp. 395-402.

[17] M. Sarrafzadeh, D.A. Knol and G.E. Tellez, "A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16(11) (1997), pp. 1332-1341.

[18] R. Fung, V. Betz and W. Chow, "Slack Allocation and Routing to Improve FPGA Timing While Repairing Short-Path Violations", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(4) (2008), pp. 686-697.

[19] T. Luo, D. Newmark and D. Z. Pan, "Total Power Optimization Combining Placement, Sizing and Multi-V_{th} Through Slack Distribution Management", *Proc. IEEE/ACM Asia and South Pacific Design Automation Conf.*, 2008, pp. 352-357.

[20] D. Chinnery and K. Keutzer, "Linear Programming for Sizing, Vth and Vdd Assignment", *Proc. International Symp. on Low Power Electronics and Design*, 2005, pp. 149-154.

[21] T. Kuroda and M. Hamada, "Low-Power CMOS Digital Design with Dual Embedded Adaptive Power Supplies", *IEEE J. Solid-State Circuits*, 35(4) (2000), pp. 652-655.

[22] M. Hamada, Y. Ootaguro and T. Kuroda, "Utilizing Surplus Timing for Power Reduction", *Proc. IEEE/ACM Custom Integrated Circuits Conf.*, 2001, pp. 89-92.

[23] T. Kuroda, "Low-Power, High-Speed CMOS VLSI Design", *Proc. IEEE/ACM International Conf. on Computer Design*, 2002, pp. 310-315.

[24] A. Srivastava and D. Sylvester, "Minimizing Total Power by Simultaneous Vdd/Vth Assignment", *Proc. IEEE/ACM Asia and South Pacific Design Automation Conference*, 2003, pp. 400-403.

[25] N. Sirisantana, L. Wei and K. Roy, "High-performance Low-power CMOS Circuits Using Multiple Channellength and Multiple Oxide Thickness", *Proc. IEEE International Conference on Computer Design*, 2000, pp. 227-232.

[26] K. Jeong, A. B. Kahng, C.-H. Park and H. Yao, "Dose Map and Placement Co-Optimization for Timing Yield Enhancement and Leakage Power Reduction", *Proc. ACM/IEEE Design Automation Conference*, 2008, pp. 516-521.

[27] D.-S. Chen and M. Sarrafzadeh, "An Exact Algorithm for Low Power Library-Specific Gate Re-Sizing", *Proc. IEEE/ACM Design Automation Conf.*, 1996, pp. 786-788.

[28] A. B. Kahng, S. Muddu and P. Sharma, "Impact of Gate-Length Biasing on Threshold-Voltage Selection", *Proc. IEEE/ACM International Symp. on Quality Electronic Design*, 2006, pp. 747-754.

[29] W. Chuang, S. S. Sapatnekar and I. N. Hajj, "Timing and Area Optimization for Standard-Cell VLSI Circuit Design", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(3) (1995), pp. 308-320.

[30] ILOG CPLEX, http://www.ilog.com/products/cplex/.

[31] Cadence SOC Encounter, http://www.cadence.com/products/digital_ic/soc_encounter/index.aspx.

[32] Synopsys Design Compiler, http://www.synopsys.com/products/logic/design_compiler.html.

[33] Cadence RTL Compiler, http://www.cadence.com/products/digital_ic/rtl_compiler/index.aspx.

[34] Blaze MO, http://www.blaze-dfm.com/products/products1.html.

[35] Synopsys PrimeTime. http://www.synopsys.com/products/analysis/prime time_ds.html.

[36] Synopsys Astro. http://www.synopsys.com/products/products.html.