Fast and Efficient Lagrangian Relaxation-Based Discrete Gate Sizing

Vinicius S. Livramento, Chrystian Guth, José Luís Güntzel
Computer Sciences Department - PPGCC
Federal University of Santa Catarina, Brazil
Email:{vini, csguth, guntzel}@inf.ufsc.br

Marcelo O. Johann
Informatics Institute - PGMICRO
Federal University of Rio Grande do Sul, Brazil
Email: johann@inf.ufrgs.br

***Abstract*—Discrete gate sizing has attracted a lot of attention recently as the EDA industry faces the challenge of optimizing large standard cell-based circuits. The discreteness of the problem, along with complex timing models, stringent constraints and ever increasing circuit sizes make the problem very difficult to tackle. Lagrangian Relaxation is an effective technique to handle complex constrained optimization problems and therefore has been used for gate sizing. In this paper, we propose an improved Lagrangian Relaxation formulation for leakage power minimization that accounts for maximum gate input slew and maximum gate output capacitance in addition to the circuit timing constraints. We also present a fast topological greedy heuristic to solve the Lagrangian Relaxation Subproblem and a complementary procedure to fix the few remaining slew and capacitace violations. The experimental results, generated by using the ISPD 2012 Discrete Gate Sizing Contest infrastructure, show that our technique is able to optimize a circuit with up to 959K gates within only 51 minutes. Comparing to the ISPD Contest top three teams, our technique obtained on average 18.9%, 16.7% and 43.8% less leakage power, while being 38, 31 and 39 times faster.**

I. Introduction and Related Work

In the well established standard cell approach, a circuit is physically implemented as an assembly of cells selected from a library. The library contains many implementation options for each gate (function), varying from slow and low power to fast and power-hungry versions. Discrete gate sizing corresponds to the problem of selecting, for each gate in the circuit, a combination of gate width (w) and threshold voltage (V_t) available in the library such that the design goals are met. As leakage power becomes comparable to dynamic power in recent technologies, a common design objective is to minimize leakage while satisfying the timing requirements. The discrete nature of the gate sizing problem applied to standard cell-based circuits together with stringent design constraints and large design sizes, make it very difficult to tackle. It has been recognized by both industry and academia as an important and challenging problem, for which current EDA tools and methods should be improved.

Different optimization techniques are found in the literature within the gate sizing scope. TILOS [1] is a well-known method that uses a sensitivity-based greedy heuristic to iteratively size the transistors in the circuit in order to minimize area subject to timing constraints. Coudert proposed a variation of the greedy method based on randomization [2]. Nguyen et. al. presented a linear programming approach based on a slack allocation method [3]. Tennakoon and Sechen presented an optimization method based on convex delay models [4]. A discrete approach based on snapping the continuous solutions to the sizes available in the library was proposed in [5]. Hu et al. [6] proposed a continuous solution approach that narrows the searching space in which a dynamic programming procedure is used to find a discrete size available in the library.

Recently, several works employed Lagrangian Relaxation (LR) to solve the gate sizing problem. Basically, LR incorporates hard constraints of the original problem into the objective function by multiplying them by Lagrange multipliers (LMs), leading to two problems: Lagrangian Relaxation Subproblem (LRS) and Lagrangian Dual Problem (LDP). A milestone on LR-based gate sizing is [7], which main contribution was the simplification of the LRS by using the so-called Karush-Kuhn-Tucker (KKT) conditions. Chen [7] solved the LRS by using a greedy heuristic and the LDP by the subgradient method. A number of other works on LR-based gate sizing have also simplified the LRS by applying the KKT, e.g. [8], [9], [10], [11] and [12]. Tennakoon [8] proposed a technique to estimate the initial values to obtain a faster convergence for the LDP. Rahman [10] proposed a continuous sizing algorithm along with a branch-and-bound discretization procedure. Zhou [11] improved the delay modeling proposed in [9] and used LR for timing minimization. Huang [12] focused on some convergence issues on the subgradient method to solve the LDP and proposed a projection-based method. The main limitations of the previously mentioned works are: 1) Continuous techniques require the discretization of the continuous solutions to obtain discrete sizes (or V_t) available in the library, in some cases resulting in significant errors [6]. 2) When using simplified delay models, according to [13], even high-order convex formulations are not accurate enough to model the delay in standard cell libraries, perhaps with the exception of [10]. 3) Some techniques require long runtimes.

Two state-of-the-art works on discrete gate sizing are [13] and [14]. Ozdal [13] uses LR along with dynamic programming to simultaneously optimize leakage power and timing,

This work was partially supported by the Brazilian Council for Scientific and Technological Development (CNPq) through INCT-Namitec and CTINFO projects, and PNM (133999/2010-6) and PQ (313724/2009-1) grants

978-3-9815370-0-0/DATE13/©2013 EDAA

whereas Rahman [14] presents a sensitivity-based technique for V_t selection. Both works report significantly better results than those obtained with commercial EDA tools for industrial-size circuits. Since discrete gate sizing is NP-hard [15], efficient heuristics must be used to find good solutions within reasonable runtime. However, so far there is no consensus about a definitive solution and hence, there is still room for reasearch. This motivated Intel researchers to organize the ISPD 2012 Discrete Gate Sizing Contest [16], which included an up-to-date infrastructure to serve as reference for the forthcoming research in this topic.

In this paper we present a new integrated discrete gate sizing technique for leakage power minimization under timing constraints, relying on contemporary standard cell libraries. Comparing to other recent works, our method provides better results for realistic industrial-size circuits in only a fraction of their runtime. Our main contributions are:

- A Lagrangian Relaxation formulation that incorporates into the objective function maximum gate input slew and maximum gate output capacitance constraints in addition to the usual timing constraints. While conforming with contemporary cell library restrictions, this avoids overloading circuit gates and possibly contributes to eliminate the negative slacks.
- A fast topological greedy heuristic for solving the proposed LR formulation relying only on local information to guide the algorithm's decisions. We claim that a local selection of each gate implementation is enough to optimize the LRS, since the gate delay depends only on its quite narrow neighborhood and the circuit arrival times (which have global dependency) are eliminated from LRS by using the KKT conditions. We also propose a complementary LR-based heuristic to treat the few violations resulted from the greedy optimization.
- The adoption of the modified subgradient method from [8] together with the power weighting factor scheduling from [17] in a technique that tackles the problem directly in the discrete space.
- Rigorous experimental validation using the infrastructure provided by the ISPD Contest demonstrated that our technique is able to provide significant leakage power reduction when compared to the Contest top three teams, while being more than 30 times faster.

II. Problem Formulation

A circuit can be modeled as a directed acyclic graph (DAG) $G(V,E)$, where V is the set of nodes and E is the set of edges. Each $v_i \in V$ represents either a combinational gate ($v_i \in X$), a primary input ($v_i \in PI$) or a primary output ($v_i \in PO$), and thus $V = X \cup PI \cup PO$. In addition $e_{j,i} \in E$ represents a wire connection between v_j and v_i. Each gate $v_i \in X$ has a width $w_i \in W_i$, where W_i is a set of discrete widths, and a threshold voltage $u_i \in U_i$, where U_i is a set of discrete threshold voltages. A particular combination of w_i and u_i is referred to as an *implementation* option of v_i. Therefore, for each gate $v_i \in X$ there is a total of $|W_i|x|U_i|$ implementation options.

The sizing problem to minimize leakage power for standard cell-based circuits can be stated as follows: given a circuit, find an implementation (w_i,u_i) for each $v_i \in X$ such that the total leakage power is minimized and the circuit timing constraints are satisfied. Hence, the constrained optimization problem, usually known as primal problem (*PP*), is expressed by (1).

$$\begin{aligned} \textbf{Primal Problem } (PP)\textbf{: Minimize } & \sum_{v_i \in X} \alpha p_i(w_i,u_i) \\ \textbf{Subject to}: & \; a_j \leq A_o, \; \forall v_j \in fanin(v_i), \; \forall v_i \in PO \\ & a_j + D_{ji}(w_i,u_i) \leq a_i, \; \forall v_j \in fanin(v_i), \; \forall v_i \in X \\ & D_{ji}(w_i,u_i) \leq a_i, \; \forall v_j \in fanin(v_i), \; \forall v_i \in PI \\ & w \in W_i \text{ and } u \in U_i, \; \forall v_i \in X \end{aligned} \quad (1)$$

In the previous formulation $p_i(w_i,u_i)$ refers to the leakage power of a given gate implementation v_i and α is the global power weighting factor. $D_{ji}(w_i,u_i)$ corresponds to the delay of timing arc $j \rightarrow i$ of a given implementation of gate v_i and a_i corresponds to the arrival time at its output. Finally, A_o is the required arrival time constraint of a given circuit. It is worth noting that separate fall/rise delays are ommited in (1) due to space limitation.

In contemporary standard cell libraries, the delay and output slew of a given implementation of gate v_i are obtained by mapping v_i's output capacitance and input slew to a bi-dimensional lookup table containing previously characterized values. In addition, cell libraries limit both the maximum slew for internal pins of the circuit and the maximum capacitance that can be driven by a given gate implementation option. Such limitations are captured by the following additional constraints:

$$\begin{aligned} out\_slew_i &\leq max\_slew, \; \forall v_i \in (X \cup PI) \\ out\_cap_i &\leq max\_cap_i(w_i,u_i), \; \forall v_i \in (X \cup PI) \end{aligned} \quad (2)$$

III. Lagrangian Relaxation

Lagrangian Relaxation is a well-known technique to handle problems with hard constraints and conflicting objectives. For this reason LR has been recently used for gate sizing to optimize power/area subject to timing constraints. The commonly used LR-based sizing formulation for power/area minimization comprises to remove the timing constraints of the problem (presented in (1)) and to incorporate them into the objective function, multiplying by a Lagrange multipliers vector (λ). By doing so, each arrival time constraint has a non-negative weight referred to as Lagrange multiplier [7].

As one of the contributions of this paper, we propose to relax max slew and max cap constraints imposed by standard cell libraries, shown in (2), in addition to the timing constraints. The LMs vectors $\vec{\gamma}$ and $\vec{\beta}$ are associated with the max slew and max cap constraints, respectively. Although the work in [18] has relaxed additional constraints such as crosstalk, to the best of our knowledge this is the first work to incorporate max slew and max cap constraints within the LR-based sizing formulation. Let $\vec{a}$ be the vector of arrival times $\forall v_i \in V$. Let

$\vec{w}$ be the vector of widths $\forall v_i \in X$ and $\vec{u}$ be the vector of threshold voltages $\forall v_i \in X$. The resulting objective function $L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a})$, called Lagrangian function, is shown in (3).

$$\begin{aligned} L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a}) : & \sum_{v_i \in X} \alpha p_i(w_i,u_i) \; + \sum_{v_j \in fanin(vi), v_i \in PO} \lambda_{jpo}(a_j - A_o) \\ & + \sum_{v_i \in X} (\sum_{v_j \in fanin(v_i)} \lambda_{ji}(a_j + D_{ji}(w_i,u_i) - a_i)) \\ & + \sum_{v_j \in fanin(v_i), v_i \in PI} \lambda_{jpi}(D_{ji}(w_i,u_i) - a_i) \\ & + \sum_{v_i \in (X \cup PI)} \gamma_i(out\_slew_i - max\_slew) \\ & + \sum_{v_i \in (X \cup PI)} \beta_i(out\_cap_i - max\_cap_i(w_i,u_i)) \end{aligned} \quad (3)$$

The Lagrangian Relaxation Subproblem (LRS), shown in (4), has to minimize $L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a})$ for a set of multipliers $\vec{\lambda}$, $\vec{\gamma}$ and $\vec{\beta}$ while satisfying the discrete constraints on width and threshold voltage.

$$\begin{aligned} & LRS1: \; Minimize \;\; L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a}) \\ & Subject\ to: \; w_i \in W_i \; and \; u_i \in U_i, \forall v_i \in V \end{aligned} \quad (4)$$

Let the function $Q(\vec{\lambda},\vec{\gamma},\vec{\beta})$ be the optimal value of *LRS*1, the Lagrangian Dual Problem (*LDP*), shown in (5), has to update the LMs vectors to maximize the function $Q(\vec{\lambda},\vec{\gamma},\vec{\beta})$.

$$\begin{aligned} & LDP: \; Maximize \;\; Q(\vec{\lambda},\vec{\gamma},\vec{\beta}) \\ & Subject\ to: \; \lambda \geq 0, \; \gamma \geq 0, \; \beta \geq 0 \end{aligned} \quad (5)$$

To simplify *LRS*1, we use the Karush-Kuhn-Tucker conditions (KKT), as proposed in [7]. The KKT conditions imply that at the optimal solution $(\vec{w},\vec{u},\vec{a})^*$, the *PP* must satisfy the following conditions: $\frac{\partial L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a})^*}{\partial a_i} = 0, \forall v_i \in (X \cup PI)$. By considering such optimality conditions, Lagrange multipliers (λ) associated with each arrival time constraints must satisfy:

$$\sum_{v_j \in fanin(v_i)} \lambda_{ji} \; = \sum_{v_k \in fanout(v_i)} \lambda_{ik}, \; \forall v_i \in (X \cup PI) \quad (6)$$

Applying the KKT optimality conditions to (4) as proposed in [7], replacing $\sum_{v_j \in fanin(v_i)} \lambda_{ji}$ by μ_i, $\forall v_i \in (X \cup PI)$ and ignoring the fixed term $\lambda_{jpo}A_o$ the Lagrangian function $L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a})$ is reduced to:

$$\begin{aligned} L_{\mu,\gamma,\beta}(\vec{w},\vec{u}) : & \sum_{v_i \in X} \alpha p_i(w_i,u_i) \; + \sum_{v_i \in (X \cup PI)} \mu_i D_{ji}(w_i,u_i) \\ & + \sum_{v_i \in (X \cup PI)} \gamma_i(out\_slew_i - max\_slew) \\ & + \sum_{v_i \in (X \cup PI)} \beta_i(out\_cap_i - max\_cap_i(w_i,u_i)) \end{aligned} \quad (7)$$

Finally, note that $L_{\mu,\gamma,\beta}(\vec{w},\vec{u})$, shown in (7), no longer depends on the arrival times. Let $\Omega_\lambda = \{\lambda \geq 0, \lambda$ satisfies the KKT optimality conditions$\}$. For any $\lambda \in \Omega_\lambda$, solving *LRS*1 is equivalent to solving:

$$\begin{aligned} & LRS2: \; Minimize \;\; L_{\mu,\gamma,\beta}(\vec{w},\vec{u}) \\ & Subject\ to: \; w_i \in W_i \; and \; u_i \in U_i, \forall v_i \in V \end{aligned} \quad (8)$$

The proposed formulation corresponds to minimizing $L_{\mu,\gamma,\beta}(\vec{w},\vec{u})$ in which the variables are the gate widths and the threshold voltages $\forall v_i \in X$.

IV. Discrete Gate Sizing

In this section, we propose a technique for discrete gate sizing which comprises a topological greedy heuristic to solve the formulation presented in *LRS*2 and a complementary procedure to fix the few capacitance and slew violations. In addition, the technique also combines for the first time the scheduling of power weighting factor conceived by [17] and the modified subgradient method (to solve the LDP) proposed in [8], assuming only discrete implementation options.

The top-level function of our technique is described in Algorithm 1, in which the main loop (lines 7-18) is executed for a fixed number of iterations. It performs Static Timing Analysis (STA) to update the arrival times, schedules the power weighting factor according to the current critical path, invokes the procedure to solve *LRS*2 and updates all LMs. Unlike the commonly used subgradient method to update the timing LMs, which relies on a unique step size ρ_k value for the whole circuit, the method in [8] is sensitive to local delay information, shown in Eq. (9). To update the max cap LMs (line 15), we used the step size that is scaled by the max cap allowed for the current implementation of v_i, as shown in (10).

$$\rho_k = \frac{\lambda_{ji}}{a_i}, \; \forall v_j \in fanin(v_i), \forall v_i \in (X \cup PI) \quad (9)$$

$$\theta_k = \frac{\beta_i}{max\_cap(w_i,u_i)}, \forall v_i \in PO \quad (10)$$

The distribution of $\vec{\lambda}$ to satisfy KKT (line 17) is perfomed similarly to [8]. Such distribution traverses the circuit in reverse topological order and visits every v_i to distribute the sum of its output LMs proportionally to each input LM. Basically, the λs at the primary outputs correlate with the circuit timing constraint (A_o). On the other hand, the λs at the gates represent their relative criticality and thus are used to distribute the primary outputs' λs along the paths, while satisfying the KKT conditions. We have observed through experimental results (Section V) that by controlling only the max cap constraints within Algorithms 2 and 3 is also sufficient to control the max slew constraints. Therefore, the LMs (γ) related to max slew constraints are not used in our Algorithms.

In LRS2 (8), the global delay information (arrival times) was eliminated as a result of applying the KKT conditions. Therefore, the goal becomes to minimize the weighted sum of delays and leakage power for each gate. However, these individual delays are not completely independent since the delay of a gate is affected by the implementation option of its neighboring cells (i.e. input slew and output load). A few recent works [13] [9] employ dynamic programming to take most of those dependencies into consideration. They spend a

lot of effort in trying to select the best global set of choices but do not take advantage of the fact that the dependencies are most very localized, affecting the delay of immediate fanins and generally fading away at three levels of fanout. Dynamic programming also requires the circuit graph to be decomposed into trees, what introduces errors at arbitrary positions and makes it suboptimal. In our experiments, we found that a greedy selection that takes into account the effects on the immediate neighbors of each gate is much more efficient for solving the LRS2 formulation. Obviously, the choice of faster or slower gate implementation is guided by the criticality of its path, being such information already incorporated in the LMs (λ).

Algorithm 2 shows our proposed topologic greedy heuristic. Even though a greedy heuristic was proposed in [7] to solve the LRS, it was applied for continuous sizing in which each gate optimal width was chosen by considering all other gates fixed. Differently, our heuristic operates in topological order, by choosing a gate's implementation only when all of its predecessors' implementation have been chosen. As another feature, our algorithm captures within the gate cost any possible capacitance violations, resulting from an alternative implementation, on the gate itself and on its fanins. We also include in the gate cost the impact of its output slew on the delay of a fanout node. This is relevant since a slow implementation option may greatly increase the fanout delay. Although we consider separate rise and fall delays, they are ommited in the algorithms due to space limitation. After all gates implementations have been chosen in a given iteration, the possibly remaining capacitance and slew violations are tightly controlled (or eliminated, in later iterations) by Algorithm 3.

The complementary LR-based procedure to fix capacitance and slew violations traverses the circuit graph in reverse topological order trying to increase the width of gates that violate max cap. Priority is given to gate widths that fix current violation with the lowest cost. By including in the gate cost its delay and power it is possible to fix violations and also to tradeoff between power and delay based on its λ and α. The inclusion of a max cap violation in the cost (line 7) accounts for the cases when a gate violation cannot be fixed in the current iteration.

V. Experimental Results

The proposed algorithms were implemented in C++ and compiled with g++ 4.4.5. To provide the timing information needed in Algorithm 1 (line 8), we used a built-in STA tool that complies with realistic standard cell libraries. To validate our discrete gate sizing technique, we relied on the infrastructure provided by the ISPD 2012 Discrete Gate Sizing Contest [16]. It comprises a circuit benchmark suite, a realistic standard cell library and scripts to check the final timing results by using Synopsys PrimeTime [19], and to check max slew and max cap violations. The ISPD Contest library contains 11 different combinational gates each one in 3 different Vt and 10 different widths, totalizing 30 implementation options per

Algorithm 1 Discrete Gate Sizing

1: **function** DISCRETE_SIZING(circuit graph V, cell library L)
2: Assign all $v_i \in V$ to minimum leakage implementation
3: $iteration \leftarrow 0$
4: $\vec{\lambda} \leftarrow$ initial vector $\in \Omega_\lambda$
5: $\vec{\beta} \leftarrow$ initial positive vector
6: $\alpha \leftarrow$ initial positive value
7: **repeat**
8: Static Timing Analysis to update timing information
9: $\alpha \leftarrow \alpha * \frac{A_o}{max(a_j)}, \forall v_j \in fanin(v_i), \forall v_i \in PO$
10: $V' \leftarrow$ **Call** SOLVE_LRS(V, L, α)
11: **for all** $v_i \in (V \cup PI \cup PO)$ **do** ▷ Update Lagrange multipliers
12: $\lambda_{jpo} \leftarrow \lambda_{jpo} * (\frac{a_j}{A_o}), \ \forall v_j \in fanin(v_i), \forall v_i \in PO$
13: $\lambda_{ji} \leftarrow \lambda_{ji} * (\frac{a_j + D_{ji}(w_i,u_i)}{a_i}), \ \forall v_j \in fanin(v_i), \ \forall v_i \in X$
14: $\lambda_{jpi} \leftarrow \lambda_{jpi} * (\frac{D_{ji}(w_i,u_i)}{a_i}), \ \forall v_j \in fanin(v_i), \forall v_i \in PI$
15: $\beta_i \leftarrow \beta_i * (\frac{out\_cap_i}{max\_cap_i(w_i,u_i)}), \forall v_i \in (X \cup PI)$
16: **end for**
17: Distribute $\vec{\lambda}$ to satisty KKT conditions, as shown in (6)
18: $iteration \leftarrow iteration + 1$
19: **until** $iteration = it_{max}$
20: **return** Best V' without violations over all iterations
21: **end function**

Algorithm 2 Forward Topological Greedy Heuristic

1: **function** SOLVE_LRS(circuit graph V, library L, power weighting factor α)
2: **for all** $v_i \in X$ in topological order **do**
3: $bestCost \leftarrow \infty$
4: **for all** $w_i \in W_i$ and $u_i \in U_i$ **do**
5: **for all** $v_j \in fanin(v_i)$ **do**
6: update out_cap_j w.r.t. current w_i, u_i
7: $cost \leftarrow \sum_{v_g \in fanin(v_j)} D_{gj}(w_j,u_j) * \lambda_{gj}$
8: $cost \leftarrow cost + \beta_j * (outCap_j - max\_cap_j)$
9: **end for**
10: $cost \leftarrow cost + \alpha p_i(w_i,u_i) + \sum_{v_j \in fanin(v_i)} D_{ji}(w_i,u_i) * \lambda_{ji}$
11: $cost \leftarrow cost + \beta_i * (out\_cap_i - max\_cap_i)$
12: **for all** $v_k \in fanout(v_i)$ **do**
13: $cost \leftarrow cost + D_{ik}(w_k,u_k) * \lambda_{ik}$
14: **end for**
15: **if** $cost < bestCost$ **then**
16: $bestCost \leftarrow cost$
17: $best_w \leftarrow w_i$
18: $best_u \leftarrow u_i$
19: **end if**
20: **end for**
21: implement v_i by $best_w$ and $best_u$
22: **end for**
23: **Call** FIX_VIOLATIONS(V, L, α)
24: **return** V
25: **end function**

Algorithm 3 Reverse Topological Heuristic to Fix Violations

1: **function** FIX_VIOLATIONS(circuit graph V, library L, power weighting factor α)
2: **for all** $v_i \in X$ in reverse topological order **do**
3: **if** $out\_cap_i > max\_cap_i$ **then**
4: $bestCost \leftarrow \infty$
5: **for all** $w_i \in W_i$ **do**
6: $cost \leftarrow \alpha p_i(w_i,u_i) + \sum_{v_j \in fanin(v_i)} D_{ji}(w_i,u_i) * \lambda_{ji}$
7: $cost \leftarrow cost + \beta_i * (out\_cap_i - max\_cap_i)$
8: **if** $out\_cap_i \leq max\_cap_i$ w.r.t current w_i **then**
9: **if** $cost < bestCost$ **then**
10: $bestCost = cost$
11: $best_w = w_i$
12: **end if**
13: **end if**
14: **end for**
15: implement v_i by $best_w$
16: **end if**
17: **end for**
18: **return** V
19: **end function**

gate. The ISPD Contest benchmark suite comprises 7 different industrial-size circuits (from 25K to 959K gates), each one subject to 2 different timing constraints (slow and fast).

The experiments were performed on a machine with two CPUs Intel(R) Xeon(R) E5620 @ 2.4GHz with 12GB RAM. Table I shows the following experimental results for our technique: leakage power, remaining critical path slack and runtime after running 60 iterations on each circuit. Our reported results were obtained by using the ISPD Contest scripts. Table I also brings the leakage power obtained by the ISPD 2012 Contest top three teams (1^{st} NTUgs, 2^{nd} UFRGS-Brazil and 3^{rd} PowerValve) as well as the lowest leakage obtained over all teams for each circuit, labeled as "Lowest" [16].

Concerning the indivual leakage power results, our technique outperforms all teams for circuits with more than 100K gates except for DES_PERF_SLOW. Unlike NTUgs and UFRGS-Brazil teams, our technique does not violate any constraint (neither timing nor max slew and max capacitance constraints). Table I also brings the average leakage power excluding the infeasible results. Compared to the top three teams, our technique obtained on average 18.9%, 16.7% and 43.8% less leakage power, respectively. The resulted critical path slacks indicate that there is still room for further optimization.

The impact of scheduling the power weighting factor (Algorithm 1, line 9) on the convergence of the proposed technique when optimizing LEON3MP_FAST (which resulted in violation for the top two teams of the Contest) is illustrated by Fig. 1 and Fig. 2. On one hand, when power factor is scheduled, the importance of power is quickly reduced in the initial iterations, allowing for the timing violations to be solved. In the following iterations the TNS (total negative slacks at the PO) violations are kept under control and the importance of power is slightly increased to recover the power spent to solve the timing violations. On the other hand, when power factor is not scheduled, some paths (Fig. 2, iterations 13-21) do not meet the timing constraints even though their LMs (λ) are increased. As the step size (9) of subgradient method is reduced each iteration, it is difficult to control the large timing violations in the later iterations. Moreover, despite the LMs (λ) of gates with small timing violations are increased, the high power weighting factor makes such gates to trade delay for power. Those results put in evidence the importance of such scheduling during a LR-based power optimization.

Another important feature of our technique is the relaxation of capacitance violations within the objective. Fig. 3 shows the sum of max cap and max slew violations on LEON3MP_FAST for three different scenarios: 1) max cap constraints are not relaxed and hence not accounted for in the cost (Algorithm 2, lines 8 and 11). 2) max cap constraints are relaxed but not fixed (i.e. Algorithm 3 is not applied). 3) max cap constraints are relaxed and fixed. The exhibited behavior shows that both the relaxation of max cap and the fix procedure are essential to guarantee the convergence. Furthermore, we have also observed through experiments that it is important to keep out_cap and out_slew within the range specified in the

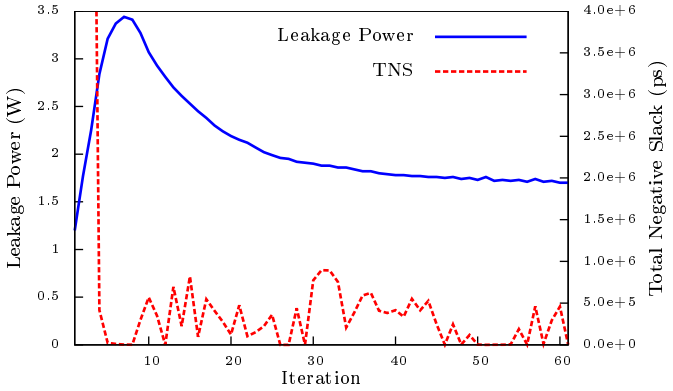

Fig. 1: Optimization of circuit LEON3MP_FAST (649K gates) by scheduling the power weighting factor.

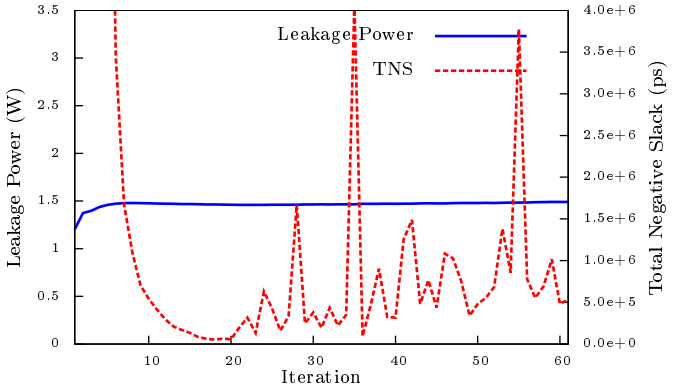

Fig. 2: Optimization of circuit LEON3MP_FAST (649K gates) without scheduling the power weighting factor.

standard cell library. This avoids overloading some gates in the circuit what increases too much the delay of some paths and hampers the convergence.

The short execution runtime of our technique makes possible to optimizate circuits with up to 959K gates within only 51 minutes. Comparing to the top three teams of the ISPD Contest, our technique is on average 38, 31 and 39 times faster, respectively. The time complexity of a single iteration of SOLVE_LRS algorithm is linear to the number of gates (n) in the circuit. However, the overall algorithm (Algorithm 1) runtime complexity can vary with the number of iterations used. Fig. 4 plots the runtime behavior of Algorithm 1 over all ISPD circuits for a fixed number of 60 iterations. The linear regression reveals a runtime complexity of $O(n^{1.0254})$ which is very close to linear runtime. Such runtime complexity allows for optimizing circuits with up to 10M gates within 10 hours.

VI. Conclusion

Based on the observation that in the commonly used LR formulation the circuit arrival times are eliminated from LRS by the KKT conditions, we have devised a fast LR-based topological greedy heuristic to solve LRS. Our technique relies on local information to guide the algorithm's decision, since the paths criticalities are already embedded in the LMs. This is in opposition to costly dynamic programming-based techniques that poorly scale to very-large circuits. Our formulation also incorporates max capacitance and max slew restrictions within the objective function, which greatly contributes to reduce

TABLE I: Leakage power (W) and runtime (hours) comparison with the ISPD 2012 Discrete Gate Sizing Contest top 3 teams [16]. "X" corresponds to results with violations. [a] Ignoring results with violations.

		ISPD 2012 Contest Results (Leakage Power (*W*))				Proposed Technique		
BENCHMARKS	**# of GATES**	**NTUgs**	**UFRGS-Brazil**	**PowerValve**	**Lowest**	**Leakage Power [W]**	**Crit. Path Slack [ps] (% of A_o)**	**Runtime [hrs]**
DMA_SLOW	25K	0.205	0.158	0.147	0.147	0.165	7.91 (0.88)	0.02
DMA_FAST	25K	0.511	0.323	0.312	0.312	0.380	11.23 (1.46)	0.02
PCI_BRIDGE32_SLOW	33K	0.203	0.115	0.116	0.115	0.124	3.03 (0.42)	0.03
PCI_BRIDGE32_FAST	33K	0.512	0.168	0.226	0.168	0.225	2.5 (0.38)	0.03
DES_PERF_SLOW	111K	0.674	0.884	0.697	0.674	0.807	6.96 (0.77)	0.10
DES_PERF_FAST	111K	2.390	3.520	2.320	2.320	1.970	1.24 (0.17)	0.10
VGA_LCD_SLOW	165K	0.415	0.378	0.391	0.378	0.359	14.9 (2.13)	0.13
VGA_LCD_FAST	165K	0.758	0.580	0.773	0.580	0.534	1.85 (0.30)	0.13
B19_SLOW	219K	0.627	0.614	0.736	0.614	0.607	16.25 (0.65)	0.26
B19_FAST	219K	2.710	X	4.490	1.040	0.973	5.95 (0.28)	0.26
LEON3MP_SLOW	649K	1.420	1.790	2.960	1.420	1.363	8.91 (0.50)	0.58
LEON3MP_FAST	649K	X	X	4.940	2.020	1.757	1.67 (0.11)	0.6
NETCARD_SLOW	959K	1.770	1.970	1.940	1.770	1.763	14.69 (0.77)	0.85
NETCARD_FAST	959K	2.010	2.300	2.970	2.010	1.909	11.73 (0.98)	0.85
AVERAGE LEAKAGE[a]		1.140	1.109	1.644	0.969	0.924		
AVERAGE RUNTIME[a]		10.764	8.868	11.027	8.170			0.283

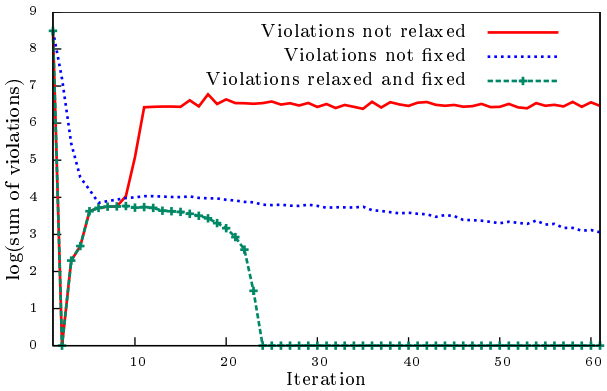

Fig. 3: Effect of controlling max cap and max slew violations on circuit LEON3MP_FAST (649K gates).

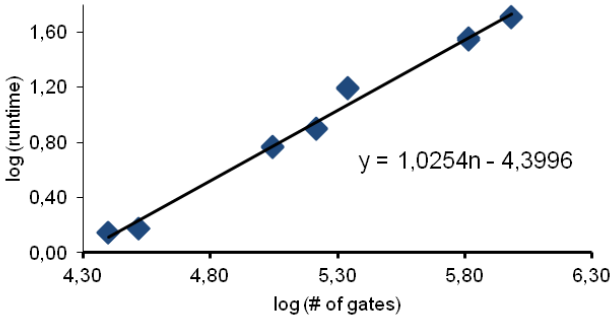

Fig. 4: Runtime complexity of the proposed technique (considering 60 iterations).

the timing violations of a circuit. We also showed, through experimental results, the importance of scheduling the power weighting factor throughout the optimization process. The efficiency of our technique became evident when evaluated under the realistic ISPD 2012 Contest infrastructure. Our technique can optimize large circuits with hundreds of thousands of gates within only one hour. Furthermore, when compared to state-of-the-art techniques our method could obtain at least 16.7% less leakage power, while being more than an order of magnitude faster.

REFERENCES

[1] J. P. Fishburn and A. E. Dunlop, "Tilos: A posynomial programming approach to transistor sizing," in *Proc. ICCAD*, 1985, pp. 326–328.

[2] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *IEEE Trans. on VLSI Systems*, vol. 5, pp. 465–472, Dec 1997.

[3] 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 *Proc. ISLPED*, 2003, pp. 158–163.

[4] H. Tennakoon and C. Sechen, "Efficient and accurate gate sizing with piecewise convex delay models," in *Proc. DAC*, 2005, pp. 807–812.

[5] S. S. Shah, A. Srivastava, V. Zolotov, D. Sharma, D. Sylvester, and D. Blaauw, "Discrete vt assignment and gate sizing using a self-snapping continuous formulation," in *Proc. ICCAD*, 2005, pp. 705–711.

[6] J. H. S. Hu, M. Ketkary, "Gate sizing for cell library-based designs," in *Proc. DAC*, 2007, pp. 847–852.

[7] 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 CAD*, vol. 18, pp. 1014–1025, July 1999.

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

[9] Y. Liu and J. Hu, "A new algorithm for simultaneous gate sizing and threshold voltage assignment," *IEEE Trans. on CAD*, vol. 2, pp. 223–234, Feb 2010.

[10] M. Rahman, H. Tennakoon, and C. Sechen, "Power reduction via near-optimal library-based cell-size selection," in *Proc. DATE*, 2011, pp. 1–4.

[11] S. Zhou, H. Yao, Q. Zhou, and Y. Cai, "Minimization of circuit delay and power through gate sizing and threshold voltage assignment," in *Proc. ISVLSI*, 2011, pp. 212–217.

[12] Y.-L. Huang, J. Hu, and W. Shi, "Lagrangian relaxation for gate implementation selection," in *Proc. ISPD*, 2011, pp. 167–174.

[13] M. M. Ozdal, S. Burns, and J. Hu, "Algorithms for gate sizing and device parameter selection for high-performance designs," *IEEE Trans. on CAD*, vol. 31, pp. 1558–1571, Oct 2012.

[14] M. Rahman and C. Sechen, "Post-synthesis leakage power minimization," in *Proc. DATE*, 2012, pp. 99–104.

[15] W. N. Li, "Strongly np-hard discrete gate sizing problems," in *Proc.ICCD*, 1993, pp. 468–471.

[16] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo, "The ispd-2012 discrete cell sizing contest and benchmark suite," in *Proc. ISPD*, 2012, pp. 161–164.

[17] H. Tennakoon and C. Sechen, "Nonconvex gate delay modeling and delay optimization," *IEEE Trans. on CAD*, vol. 27, no. 9, pp. 1583–1594, September 2008.

[18] I. H.-R. Jiang, Y.-W. Chang, and J.-Y. Jou, "Crosstalk-driven interconnect optimization by simultaneous gate and wire sizing," *IEEE Trans. on CAD*, vol. 19, no. 9, pp. 999–1010, September 2000.

[19] Synopsys primetime user's manual. [Online]. Available: http://www.synopsys.com