Incremental Lagrangian Relaxation based Discrete Gate Sizing and Threshold Voltage Assignment

Dimitrios Mangiras and Giorgos Dimitrakopoulos
Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece

***Abstract*—Timing closure remains one of the most critical challenges of a physical synthesis flow. Even if timing is almost closed at the end of the flow, last-mile placement and routing congestion optimizations may introduce new timing violations. Correcting such violations needs minimally disruptive techniques such as threshold voltage re-assignment and gate sizing that affect only marginally the placement and routing of the almost finalized design. To this end, we transform a powerful Lagrangian-relaxation-based optimizer, used for global timing optimizations at the early stages of the design flow, to a practical incremental timing optimizer that corrects small timing violations with fast runtime and without increasing the area/power of the design. By applying the proposed approach to the optimized designs of the ISPD 2013 gate sizing contest that experience new timing violations due to local wire rerouting, we improve timing by more than 36% on average, using 45% less runtime, when compared to the fully-fledged version of the timing optimizer.**

I. Introduction

Physical synthesis refers to the process of placing and routing the logic netlist of a design, while concurrently optimizing for multiple objectives given a set of area, power, timing, and routability constraints [1]. For achieving those goals the main physical synthesis steps are supported by several intermediate optimizations that focus on logic restructuring (addition, removal or change of logic cells) as well as logic tuning that involves selecting for each gate an appropriate size and threshold voltage from a discrete set of library cells.

At the end of the design flow, the design should satisfy all timing constraints and be free of any design rule violations such as maximum allowed capacitance and transition time. Large timing and design rule violations are analyzed and removed at the first steps of the design flow using efficient global optimization engines [2]. Still, a small set of remaining violations always exist close to the end of the flow. Repairing such violations requires efficient incremental operations that are non-disruptive and execute as fast as possible. For instance, after routing, we don't want cells' placement to change for improving timing since this would cause re-routing a large part of the design thus possibly introducing new violations.

The least disruptive operations for improving design's characteristics during physical synthesis involve threshold voltage (V_T) re-assignment and gate sizing [2], [3] assuming that any new change would not introduce any new design rule violations. V_T re-assignment tradeoffs smaller delay with increased leakage power and does not perturb routing nor it requires a new parasitics extraction after the change. Gate resizing, even if not as simple as V_T re-assignment, is still considered a fairly non-invasive operation. In the worst case, increasing cell's size (possibly avoiding exceedingly large changes) may require an additional local legalization step [4], [5] and local re-routing of certain nets [6].

Inserting buffers is still an option at this step [7], [8]. However, such addition may ruin local placement and routing, which may be hard to fix in highly congested designs. Other highly powerful optimization steps such as useful clock skewing are also considered hard to apply at the end of the flow, unless there is no other obvious or practical way to solve the remaining timing violations [9], [10], [11].

Gate sizing and V_T assignment algorithms have a long history in physical synthesis flows. Among the large set of available solutions [12], [13], those that rely on Lagrangian Relaxation (LR) achieve significantly better results [14], [15], [16]. However, when applied incrementally, i.e., when the LR algorithm is not allowed to initialize all cells to their minimum size [17] thus totally disrupting the existing design, they need many iterations to converge even if the number of timing violators is small.

In this work, we propose a novel initialization strategy for LR-based timing/power optimizers that combines two useful benefits: On one hand we enjoy the optimization efficiency of an LR-based gate sizer and on the other hand we enjoy fast runtimes and true incremental operation, i.e., the optimized design is only marginally different from the original design but with the timing violations repaired.

The proposed approach has been compared to a fully-fledged LR-based gate sizer on optimized versions of the benchmarks of the ISPD2013 contest [18]. The used benchmarks experience small timing violations due local changes of their routed wires. In all cases, the proposed initialization strategy successfully optimizes the timing of each design offering 36% better timing performance on average with reduced leakage power. Besides the improved quality-of-results the introduced initialization strategy for LR-based optimizers reduces the runtime by more than 45% on average, since it simplifies the convergence of the algorithm.

II. Basics of LR-based gate sizing

A timing-driven optimizer tries to minimize the power (or area) of the design given a set of timing constraints.

$$\begin{aligned} \text{minimize} \quad & \sum_i leakage_i \qquad (1) \\ \text{subject to} \quad & a_i + d_{ij} \le a_j \quad \forall\, i \rightarrow j \\ & a_k \le r_k \quad \forall \text{ endpoints } k \end{aligned}$$

For cell i, a_i denotes the arrival time at its output, d_{ij} is the delay of the timing arc $i \rightarrow j$ which is defined from the output of the gate i to the output of the gate j including any intermediate wires and r_k is the required arrival time at timing endpoint k [19]. The set of timing endpoints include the primary output pins of the design as well as the input pins of all flip-flops.

Associating the constraint for each timing arc with a non-negative Lagrange multiplier λ_{ij}, that acts as a penalty factor when the respective constraint gets violated, and computing the KKT optimality conditions [15], [20], allows us to simplify the constrained minimization problem (1) to the equivalent unconstrained minimization problem (2).

$$\text{minimize} \quad \sum_{i} leakage_i + \sum_{i \to j} \lambda_{ij} d_{ij} \tag{2}$$

State-of-the-art LR-based optimizers [14], [15], [16] try to minimize the global cost function (2) using many iterations of local gate resizing and V_T re-assignment steps. Both steps would be referred as gate sizing for brevity for the rest of the paper.

Initially, all gates are replaced with their least leakage power option (lowest size and highest V_T) [17] and all LMs are set to 1. Then, each iteration of LR-based gate sizing evolves in two phases. In the first phase, for each gate, examined in topological order, all possible discrete cell sizes and threshold voltages are tried, assuming constant LMs. The new version selected for the resized gate is the one that minimizes the cost function (2) computed on the local neighborhood of gates around the gate under examination. In the second phase, the LMs are updated to reflect the new criticality of the corresponding timing arcs. LM update may take different forms and can be either additive ($\lambda_{\text{new}} = \gamma + \delta\lambda_{\text{old}}$) or multiplicative ($\lambda_{\text{new}} = \gamma\lambda_{\text{old}}$)[21]. On every LM update, a full incremental timing update takes place. The LR gate sizer converges after many iterations to an optimized solution, where the gate sizes are updated one after the other until no TNS or no power gains are observed.

The value of each LM reflects the timing criticality of each timing arc. LMs increase fast for critical timing arcs and reduce for non-critical timing arcs to favor power reduction. Implicitly, LMs keep also historic information (for the lifetime of an optimization run) with respect to the criticality of each timing arc. If a timing arc remained critical for multiple iterations it is still assumed critical by keeping a high value of LM, even if the slack at its output becomes positive in a certain iteration. In this way, drastic oscillation between critical and non-critical timing arcs are avoided and the optimization evolves smoothly reducing power while satisfying timing constraints.

III. Incremental LR-based gate sizing

The overall effectiveness of an LM-based gate sizer is the combined result of the initialization of gate sizes, the strength of the local optimization and the appropriate update of LMs.

Initializing all cells to their minimum size simplifies the removal of any design rule violations and also allows the fast optimization of the timing of the design. After initialization, the total leakage power in cost function (2) assumes its minimum value. Thus, the sum of $\lambda_{ij}\, d_{ij}$ products determine which cell should be selected for each gate. This conclusion holds even if leakage and delay participate normalized to the cost function. Increasing fast the LMs of critical timing arcs guides the optimization to reduce their corresponding delay in order to minimize their $\lambda_{ij}\, d_{ij}$ product. As long as timing constraints are not satisfied, LMs keep increasing thus leading to cells with improved delay.

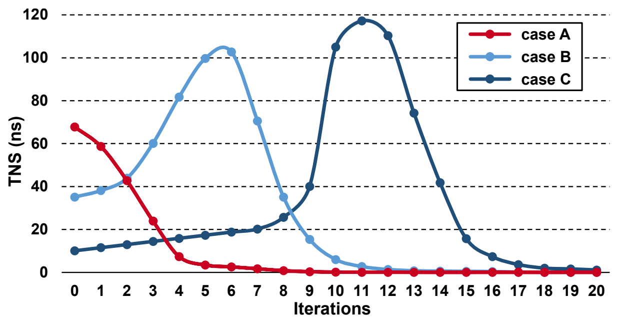


Fig. 1. The evolution of TNS on each iteration of a LR-based gate sizer for three cases of the same design. In case A the design is not optimized. In cases B and C it is partially optimized thus exhibiting initially less TNS.

A. *What is the problem?*

In an incremental optimization scenario, which is the focus of this work, one piece of the state-of-the-art LR-based gate sizing cannot be applied. Since the design is almost finalized, the gate sizer is not allowed to “reset” the state of the design and initialize every gate to its minimum size. Therefore, since all gates keep their already decided size the sum of leakage power in (2) may possibly dominate the cost function. The LMs that fit to this occasion are unknown and initializing them to 1 may not be the best choice.

Inevitably, at the first iterations of LR-based gate sizing, lower power cells would be preferred for each gate since they would minimize local cost at the expense of timing. Once timing would starting getting much worse and the corresponding LMs start to take higher values, only then the $\lambda_{ij} d_{ij}$ products would favor the selection of the delay-optimal cells. Due to improper initialization, state-of-the-art LR-based gate sizers exhibit a counterproductive behavior. The less timing critical is the initial state of the design, the more time an LR-based gate sizer would need to optimize it, when resetting the state of the design is not allowed.

To highlight this behavior we performed an experiment using the pci_bridge32_fast design of the ISPD13 benchmark set. Fig. 1 depicts the evolution of the design’s Total Negative Slack (TNS) during LR-based gate sizing for three different cases. When the design suffers from many timing violations (case A), the LR-based gate sizer is able to find fast the way to improve timing, leading to almost closed timing after the first six iterations. The rest iterations are used to improve leakage power without degrading timing in the meantime.

On the other hand, if the design had initially less TNS (case B), LR-based gate sizer prefers to improve power by degrading timing in the first six iterations before it starts solving timing violations and achieving timing closure in iteration eleven. Similarly, if LR-based gate sizing is applied on an already optimized design with only few timing violations (case C), it will first convert many non timing critical paths to critical before actually reducing TNS to almost zero.

It is clear that regardless of the initial TNS, LR-based gate sizing is powerful enough to solve all timing violations. However, due to improper initialization, it fails to do this fast in cases that it should have. Therefore, for the case of partially optimized designs with a small set of timing violations, like case C of Fig. 1, we need to derive an incremental version of the LR-based gate sizer that would achieve high quality-of-results and fast convergence.

B. What can we do about it?

To improve the applicability of the LR-based discrete gate sizer in an incremental optimization context, we propose an efficient method for initializing the values of the LMs. In this way, the values of the LMs would reflect the proper timing criticality of each gate relative to its already selected size, as seen near the end of the physical synthesis flow. The proposed approach is non intrusive, since it deals only with the initialization of the LMs, and can be used with any LR-based gate sizer [14], [15], [16].

Determining the initial values of the LMs should not be based solely on the criticality of the corresponding timing arcs. Assume, for instance, that the design contains a very large gate that contributes a lot to its leakage power and currently has zero timing violations. In fact, we may assume that its output pin has a small positive slack. If we assign to this gate a small initial LM due to its positive slack, we would lead the optimizer to downsize it in the first iterations to save power. This choice may seem reasonable but it fails to answer one critical question: why this gate has not been downsized earlier by the multiple optimization steps that preceded? The most probable answer is that this gate originally belonged to a set of critical timing paths. Optimizing those paths in the first steps of the flow, resulted in selecting for this gate a fast (with small delay) but large cell. Thus, any trial to reduce its size at the end of the flow would directly translate to new timing violations.

Based on this intuition, we choose to initialize the LMs following a balanced approach. We assign increased LMs to timing arcs that are either critical at the moment or belong to high-power cells assuming that those cells may have been timing critical in the past. This approach may lead to a temporary power overhead to cells that are indeed not critical but remained large for the wrong reasons (e.g., a previously applied optimization skipped them to save runtime). However, the first iterations of LR-based gate sizer would identify this by gradually reducing their corresponding LMs thus turning them to good candidates for power reduction.

The initial value for the LM of timing arc $i \to j$ is set to:

$$\lambda ij = \left(\frac{a_i + d_{ij}}{a_j} \frac{P(g)}{\min P(g)} \right)^K \; \forall \text{ arc } i \to j \text{ of gate } g \quad (3)$$

Gate g refers to the gate where the timing arc $i \to j$ belongs. The starting value for each LM is the product of two ratios. The first ratio reveals the timing criticality of the arc $i \to j$. If the corresponding timing arc is responsible for the late arrival time at the output pin of gate j, the sum of a_i and the delay $d_{i,j}$ will be equal to a_j thus setting the ratio to 1. In any other case, a_j will be greater than the numerator and thus the ratio will result to a value less than 1 signifying the non criticality of the arc. The second ratio describes how much more power the current version of the cell spends $P(g)$ relative to the minimum possible leakage power that it can spend using any compatible library cell for gate g. In overall, when timing critical arcs are coupled with high power cells will get much greater LM values. The exponent K helps to increase faster the assigned LMs values and we empirically set it to $K = 2$.

Similarly, for the LMs that correspond to the timing arcs $i \to k$, where k is a timing endpoint:

$$\lambda_{ik} = \left(\frac{a_k}{r_k} \frac{\sum_{gates} P(g)}{\sum_{gates} \min P(g)} \right)^K \; \forall \text{timing endpoint } k \quad (4)$$

If the signal arrives at the timing endpoint k earlier than its required time r_k, i.e., $a_k < r_k$, signaling that there is no timing violation, the first ratio will result in a value less than one. On the contrary, in cases that late timing is violated, with $a_k > r_k$, the first ratio will be as big as the actual violation. For the power ratio in the case of timing endpoints, we suggest that it should consider the design as a whole. For this reason, the power ratio that is multiplied to the the timing ratio, divides the current total leakage power of the design relative to the minimum leakage power that the design can achieve after replacing each gate with a minimum leakage power cell. This ratio actually quantifies how far the design is from its virtually minimum leakage power.

IV. Experimental Results

The proposed method was implemented in C++ inside the open-source RSyn physical design framework [22]. The evaluation involves already optimized benchmarks with only few timing violations. For this purpose, we used the fully optimized versions of the benchmarks of the ISPD 2013 gate sizing contest [18]. Those designs exhibit closed timing and minimized leakage power. To introduce additional timing violations, we randomly changed the resistance and capacitance of each net by $\pm 10\%$ thus mimicking local re-routing operation at the end of the physical synthesis flow.

Initially, we report the quality-of-results achieved for the proposed method (New) relative a state-of-the-art LR-gate sizer [14] (called Base) without allowing it to reset the state of the design. Both cases actually utilize the same LR-based gate sizer. They only difference is on how the initialize the value of the LMs. The obtained results are shown in Table I. Columns Init correspond to the design produced after randomly perturbing the resistance and capacitance of the wires. In all cases, the optimization stops if the improvement in terms of timing and leakage power across two iterations is less than 1%. Table I reports the late Worst Negative Slack (WNS), the late TNS and the total leakage power of each design. The final reported timing results are validated by OpenTimer [23]. Please note that ISPD2013 benchmarks do not exhibit early timing violations and thus early timing information is omitted.

The first noticeable result is that "New" offers better timing results than "Base" in the majority of the designs. With the proposed LM initialization, WNS is further decreased by 24% on average, while TNS is improved by more 36% on average compared to the corresponding results of "Base". "New" also achieves slightly better leakage power than "Base". For fair comparison, we take into account only the leakage power savings from designs where both the "Base" and the "New" flow succeeded to resolve all timing violations. In those cases,

TABLE I

THE TIMING AND THE LEAKAGE POWER OF ALL DESIGNS INITIALLY (INIT) AND AT THE END OF INCREMENTAL LR-BASED SIZER WITHOUT (BASE) AND WITH (NEW) THE PROPOSED LM INITIALIZATION.

Design	#Cells	Late WNS (ps)			Late TNS (ps)			Leakage (mW)		
		Init	Base	New	Init	Base	New	Init	Base	New
usb_phy_slow	623	-1.53	0.00	0.00	-1.53	0.00	0.00	1	1	1
usb_phy_fast		-0.61	0.00	0.00	-0.61	0.00	0.00	2	2	2
pci_bridge32_slow	30763	-11.21	0.00	0.00	-333.10	0.00	0.00	58	58	58
pci_bridge32_fast		-16.66	-0.44	0.00	-614.66	-0.96	0.00	98	97	100
fft_slow	33792	-16.35	0.00	0.00	-320.92	0.00	0.00	88	88	87
fft_fast		-18.18	-6.58	-1.88	-234.28	-63.37	-4.25	217	228	228
cordic_slow	42937	-13.99	-14.43	-1.24	-801.84	-116.70	-2.11	306	349	309
cordic_fast		-13.26	-4.26	-6.94	-752.72	-30.00	-31.40	1139	1142	933
des_perf_slow	113346	-30.40	-1.88	0.00	-11920.00	-5.26	0.00	449	410	420
des_perf_fast		-25.80	-3.51	-4.10	-11412.20	-49.94	-8.69	609	522	556
edit_dist_slow	129227	-54.44	0.00	0.00	-21881.50	0.00	0.00	452	447	445
edit_dist_fast		-63.59	-3.34	0.00	-36639.50	-15.16	0.00	624	630	610
matrix_mult_slow	159642	-44.00	0.00	0.00	-3292.93	0.00	0.00	481	487	476
matrix_mult_fast		-33.07	0.00	0.00	-2694.75	0.00	0.00	1056	1230	1020
netcard_slow	984094	-30.19	0.00	0.00	-1477.58	0.00	0.00	5160	5101	5102
netcard_fast		-28.97	0.00	0.00	-6394.27	0.00	0.00	5203	5144	5141
Average		-25.14	-2.15	-0.89	-6173.27	-17.59	-2.90	996	996	968

"New" is 2% better on average. The reason is that whenever there are timing violations, the design's power is lower than the power of the design with closed timing.

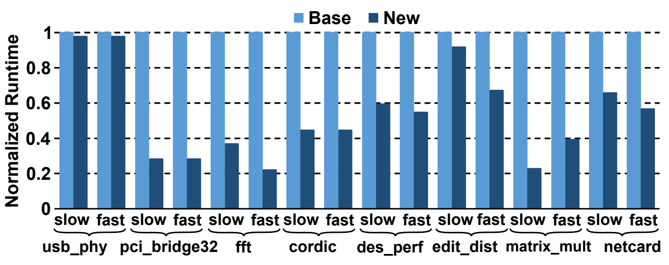
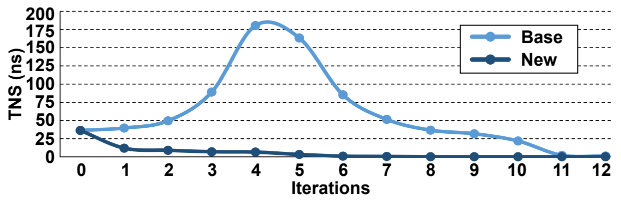


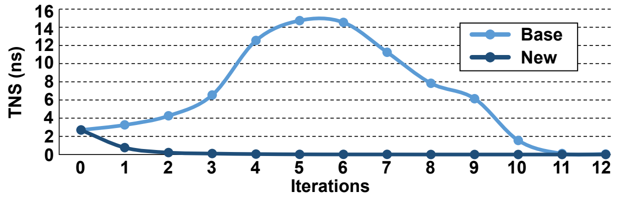
Fig. 2. The runtime of both methods under comparison for all benchmarks normalized to the runtime of the "Base". In all cases, "New" allows for faster convergence saving up to 45% execution time on average.

Fig. 2 compares the two approaches in terms of runtime. All experiments were performed on the same Linux-based workstation using a 3.6 GHz Intel Core i7-4790 with four cores and 32 GB of RAM. "New" is able to save up to 45% of runtime on average achieving also better quality-of-results. In terms of absolute runtime, "Base" finishes optimizing all designs in 9hrs, while the proposed flow needs 5hrs for the same task. The runtime of both methods for designs usb_phy (slow and fast) is similar due to their small size of the designs.

To observe more clearly how the proposed LM initialization helps the convergence of an LR-based gate sizer, we monitor the evolution of TNS across consecutive iterations for two representative designs. In the case of edit_dist_fast, shown in Fig. 3(a), "Base" achieves timing closure in iteration 11, while "New" converged faster closing timing five iterations earlier. Similarly, in Fig.3(b) for matrix_mult_fast, "New" was able to solve all timing violations in the first 2 iterations, while "Base" needed 9 more iterations to converge. Similar results are obtained for all other designs. The proposed LM initialization successfully "predicts" the value of the LM that fits better to the status of the design thus avoiding un-necessary



(a): edit_dist_fast



(b): matrix_mult_fast

Fig. 3. The progression of late TNS without (Base) and with (New) the proposed LM initialization on designs (a) edit_dist_fast and (b) matrix_mult_fast.

power reductions at the first iterations that would hurt timing initially and delay convergence later on.

To be certain for the quality-of-results of the proposed approach, we repeated the same experiment for each benchmark 100 times. Each time, the methods under comparison were applied on designs produced after perturbing randomly the wire parasitics of the already optimized version of each benchmark. The histogram of TNS for the initial design, and the ones produced after applying "Base" and "New" methods are depicted in Fig. 4 for benchmark fft_fast, while similar results are obtained for all other benchmarks.

TNS histograms reveal that both approaches successfully decreased the original TNS. "Base" decreased the mean of initial TNS from 225ps to 65ps, while "New" managed to compress the TNS histogram to the left side of the diagram, with the majority of samples gathered close to 5 ps.

For completeness, we evaluated both methods under comparison in a more restrictive scenario. In this case, gate sizing

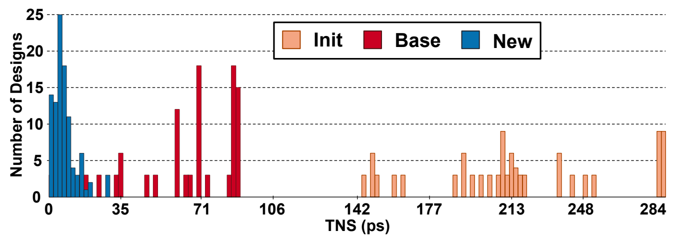


Fig. 4. The histogram of late TNS initially (Init) and at the end of LR-based gate sizing without (Base) and with (New) the proposed LM initialization. Histograms correspond to 100 versions of fft_fast with randomly perturbed RC characteristics.

TABLE II
THE TIMING AND THE LEAKAGE POWER OF ALL DESIGNS WITH GATE SIZE SELECTION RESTRICTION WITHOUT (BASE) AND WITH (NEW) THE PROPOSED LM INITIALIZATION.

Design	Late				Leakage (mW)	
	WNS (ps)		TNS (ps)			
	Base	New	Base	New	Base	New
usb_phy_slow	0.0	0.0	0.0	0.0	1	1
usb_phy_fast	0.0	0.0	0.0	0.0	2	2
pci_bridge32_slow	0.0	0.0	0.0	0.0	58	58
pci_bridge32_fast	-1.7	0.0	-6.1	0.0	98	98
fft_slow	0.0	0.0	0.0	0.0	88	87
fft_fast	-6.9	-1.0	-20.2	-2.2	224	221
cordic_slow	-8.8	-3.0	-67.2	-3.0	378	310
cordic_fast	-17.1	-2.7	-133.1	-4.8	1209	942
des_perf_slow	-27.5	-1.4	-67.5	-4.5	480	464
des_perf_fast	-14.4	-7.6	-47.4	-23.3	637	611
edit_dist_slow	0.0	0.0	0.0	0.0	450	449
edit_dist_fast	-20.8	-2.0	-698.8	-2.2	623	619
matrix_mult_slow	0.0	0.0	0.0	0.0	478	479
matrix_mult_fast	0.0	0.0	0.0	0.0	1174	1020
netcard_slow	0.0	0.0	0.0	0.0	5152	5153
netcard_fast	0.0	0.0	0.0	0.0	5197	5194
Average	-6.1	-1.1	-65.0	-2.5	1016	982

is only allowed to resize cells only to their next bigger or smaller size without limiting V_T swapping options, since they do not alter the physical layout. This restriction makes sense at the final steps of physical design flow to preserve as much as possible the already defined detailed wire routes. The obtained results are depicted in Table II. Besides the restricted availability of gate sizes, "New" achieves considerable improvements. Late WNS is improved by 36% on average while the savings in TNS reach 39% on average, when compared to the baseline LR-based gate sizer. In terms of leakage power, the restricted "New" method achieves less leakage power by 2% on average, when considering only the designs without negative slack at both methods under comparison.

V. CONCLUSIONS

The incremental application of LR-based gate sizer at the end of the physical synthesis flow, where resetting the size of each gate is not allowed, needs special treatment in order to achieve good quality-of-results with reasonable runtime. To this end, the proposed approach offers a viable solution by initializing appropriately the value of the Lagrange multipliers after taking into account both their timing criticality as well as the current size of the gates. In this way, we expedite successfully the convergence of the LR-based gate sizer, when applied in an incremental optimization context, without affecting any part of its internal functions and without reducing the achieved quality-of-results.

ACKNOWLEDGMENTS

Dimitrios Mangiras is supported by the Onassis Foundation - Scholarship ID: G ZO 014-1/2018-2019.

REFERENCES

[1] L. Lavagno, G. Martin, I. L. Markov, and L. K. Scheffer, *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*. Taylor and Francis group, 2016.
[2] N. D. MacDonald, "Timing closure in deep submicron designs," in *Design Automation Conference (DAC)*, 2010.
[3] D. G. Chinnery and K. Keutzer, "Linear programming for sizing, vth and vdd assignment," in *Proc. of the Intern. Symp. on Low Power Electronics and Design (ISLPED)*, 2005, pp. 149–154.
[4] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: Fast legalization of standard cell circuits with minimal movement," in *Intern. Symp. on Physical Design (ISPD)*, 2008, pp. 47–53.
[5] J. C. Puget, G. Flach, R. Reis, and M. Johann, "Jezz: An effective legalization algorithm for minimum displacement," in *Symp. on Integrated Circuits and Systems Design (SBCCI)*, Aug 2015, pp. 1–5.
[6] A. Chowdhary, K. Rajagopal, S. Venkatesan, T. Cao, V. Tiourin, Y. Parasuram, and B. Halpin, "How accurately can we model timing in a placement engine?" in *ACM/IEEE Design Automation Conference (DAC)*, 2005, pp. 801–806.
[7] C. Alpert, C. Chu, G. Gandham, M. Hrkić, J. Hu, C. Kashyap, and S. Quay, "Simultaneous driver sizing and buffer insertion using a delay penalty estimation technique," in *Intern. Symp. on Physical Design (ISPD)*, 2002, pp. 104 – 109.
[8] A. Stefanidis, D. Mangiras, C. Nicopoulos, D. Chinnery, and G. Dimitrakopoulos, "Design Optimization by Fine-Grained Interleaving of Local Netlist Transformations in Lagrangian Relaxation," in *Intern. Symp. on Physical Design (ISPD)*, 2020, pp. 87–94.
[9] J. P. Fishburn, "Clock Skew Optimization," *IEEE Trans. on Computers*, vol. 39, no. 7, pp. 945–951, 1990.
[10] S. Kim, S. Do, and S. Kang, "Fast predictive useful skew methodology for timing-driven placement optimization," in *ACM/IEEE Design Automation Conference (DAC)*, 2017, pp. 55:1–55:6.
[11] A. Stefanidis, D. Mangiras, C. Nicopoulos, D. Chinnery, and G. Dimitrakopoulos, "Autonomous application of netlist transformations inside lagrangian relaxation-based optimization," *in IEEE Trans. on CAD*. [Online]. Available: https://ieeexplore.ieee.org/document/9201479
[12] O. Coudert, "Gate Sizing for Constrained Delay/Power/Area Optimization," *IEEE Trans. on VLSI Systems*, vol. 5, no. 4, pp. 465–472, 1997.
[13] J. Hu and et al., "Sensitivity-guided metaheuristics for accurate discrete gate sizing," in *IEEE Intern. Conf. CAD*, 2012, p. 233239.
[14] G. Flach and et al., "Effective method for simultaneous gate sizing and vth assignment using lagrangian relaxation," *IEEE Trans. on CAD*, vol. 33, no. 4, pp. 546–557, April 2014.
[15] 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, no. 10, pp. 1558–1571, October 2012.
[16] A. Sharma, D. Chinnery, S. Bhardwaj, and C. Chu, "Fast lagrangian relaxation based gate sizing using multi-threading," in *IEEE Inter. Conf. on Computer-Aided Design*, 2015, pp. 426–433.
[17] L. Li, P. Kang, Y. Lu, and H. Zhou, "An efficient algorithm for library-based cell-type selection in high-performance," in *2012 IEEE/ACM Intern. Conf. on Computer-Aided Design (ICCAD)*, pp. 226–232.
[18] M. Ozdal, C. Amin, A. Ayupov, S. M. Burns, G. R. Wilke, and C. Zhuo, "An improved benchmark suite for the ISPD-2013 discrete cell sizing contest," in *Int. Symp. on Physical Design*, 2013, p. 168170.
[19] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer, 2009.
[20] D. Mangiras, A. Stefanidis, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "Timing-Driven Placement Optimization Facilitated by Timing-Compatibility Flip-Flop Clustering," *in IEEE Trans. on CAD*, vol. 39, no. 10, pp. 2835 – 2848, Oct. 2020.
[21] H. Tennakoon and C. Sechen, "Nonconvex gate delay modeling and delay optimization," *IEEE Trans. on CAD*, vol. 27, pp. 1583–1594, 2008.
[22] G. Flach, M. Fogaça, J. Monteiro, M. Johann, and R. Reis, "Rsyn: An extensible physical synthesis framework," in *Int. Symp. on Physical Design*, 2017, pp. 33–40.
[23] T. W. Huang and M. D. F. Wong, "Opentimer: A high-performance timing analysis tool," in *Intern. Conf. CAD*, 2015, pp. 895–902.