High-Performance Gate Sizing with a Signoff Timer

Andrew B. Kahng[‡+], Seokhyeong Kang[‡], Hyein Lee[‡], Igor L. Markov[†] and Pankit Thapar[†]
UC San Diego, [‡]ECE and [+]CSE Depts., La Jolla, CA 92093, {abk, shk046, hyeinlee}@ucsd.edu
[†]University of Michigan, 2260 Hayward St., Ann Arbor, MI 48109, {imarkov, pankit}@eecs.umich.edu

ABSTRACT

Process and device scaling in late-CMOS technologies highlight leakage power as a critical challenge for the semiconductor industry. Careful gate sizing and V_{th}-swapping can reduce leakage, but prior optimizations based on convex or dynamic programming (i) are often based on unrealistic assumptions about circuit delay and slew propagation, (ii) fail to handle practical design rules such as transition time or load upper bounds, and (iii) do not scale well to input complexities when full extracted parasitics are available. Seeing substantial opportunities for improvement, we present a multi-threaded, stochastic optimization (Trident2.0) for gate sizing and V_{th} assignment to minimize leakage power subject to capacitance, slew and timing constraints. Scalability and high performance of Trident2.0 are validated on ISPD-2013 Gate Sizing Contest benchmarks.

1. INTRODUCTION

Clock speed and supply voltage of leading-edge ICs have barely improved in the last few technology nodes due to adverse interconnect and power scaling trends. As pointed out by IBM's James Warnock in his ISPD-2013 invited talk, "the solutions to problems with scaling have been postponed by previous generations... at 14-nm there is no way to get more performance by scaling alone" [26]. The industry anticipates some relief with the introduction of FinFET transistors, but powerful EDA optimizations to control leakage are already becoming critical. One such optimization deals with cell sizes and V_{th} assignments — Intel reported 30% power reduction through the use of recently developed algorithms [16]. Another ISPD-2013 speaker — Mentor's David Chinnery — claimed that "global optimization of cell sizes can on average reduce total power by 16% and leakage power by 29% versus iterative greedy sizing in today's vendor tools." Such optimization can be difficult due to the need for accurate timing analysis on large circuit netlists. Chinnery reported 13-hour runtime when sizing 361K gates [3].

Realizing the importance of gate sizing and V_{th} assignment, academic and industry researchers developed numerous algorithms. It is common to simplify circuit delay models and solve an abstract, continuous version of the problem to facilitate convex optimization — Lagrangian relaxation, linear programming, network flows, etc. [6, 10, 20, 21, 25]. Simultaneous gate sizing and V_{th} assignment has shown promising results [23, 24]. Some industry work emphasizes discrete methods [5], including dynamic programming [16] that assumes structural properties of delay functions that do not quite hold in practice. Device physics imply nonconvex delay functions, causing nonconvexities in SPICE results and nonlinear delay model (NLDM) tables. Capacitance and slew constraints further complicate the problem.

For the ISPD-2012 Gate Sizing Contest [17], Intel researchers prepared several large gate sizing benchmarks to empirically compare competing leakage power optimizations. The contest methodology requires the use of *Synopsys PrimeTime* [29]. While the winners of the contest used Lagrangian optimization, by the end of the year the best (published) performance was attained by a new stochastic optimization [7] without using continuous-valued techniques or convexity assumptions. However, the fact that ISPD-2012 contest omitted interconnect delays led to some doubt that stochastic techniques could remain competitive in realistic gate sizing contexts, where frequent invocations of a signoff timer threatened to make stochastic optimization unacceptably slow.

For the ISPD-2013 Gate Sizing Contest [18], Intel researchers developed a more extensive infrastructure for experimentation that evaluated both gate and interconnect delay, and checked capacitance and slew constraints (in addition to full-netlist SDC timing constraints). Total runtime per benchmark was limited, with tighter limits in a secondary category, where tool runtime could be traded off for leakage power. In this paper, we describe a successful entry from the ISPD-2013 contest that achieves practical large-scale metaheuristic gate sizing and V_{th} optimization with a signoff timer (ST) in the loop. Our enabling innovations include

- mechanisms for closely tracking an external ST without undue loss of efficiency;
- choice of internal delay and slew models that permit sufficient calibration to the external ST;
- metaheuristic innovations that improve the ability to efficiently achieve timing-feasible sizing solutions; and
- implementation and metaheuristic strategies that achieve large speedups without sacrificing solution quality.

The software system we describe, Trident2.0, integrates a number of previously known components and ideas with new ones. Significant effort was spent on identifying *the most pertinent ideas, techniques and components* (the optimization framework, models for interconnect capacitance, delay and slew, etc.) whose runtime-quality tradeoffs are consistent with the desired performance envelope of a high-performance sizer. Another area of major importance is *partitioning the overall optimization into separate stages* that pursue specific goals and call for dedicated components. The *handoff between such stages* has also been critical to the overall performance. Given the large amount of computation involved, *effective use of parallel-computing resources* is required.

The remainder of the paper is structured as follows. In Section 2 we review relevant delay models and discuss tradeoffs between accuracy and speed. Section 3 outlines our framework for high-performance optimization of cell sizes and threshold voltage. Given the complexity of this constrained optimization, a number of practical insights are required for successful implementation. Such insights are offered in Section 4. Section 5 compares our work to prior techniques. Empirical validation on ISPD-2013 benchmarks is given in Section 6, and Section 7 concludes the paper.

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2013, November 18-21, 2013, San Jose, California, USA
Copyright 978-1-4799-1071-7/13/$31.00 ©2013 IEEE.

2. INTERCONNECT MODELING

In this section, we review fast delay models and explain how we selected models appropriate for high-performance optimization. A core insight, well understood in the field, is that *early optimization* does not require signoff timing accuracy and can be performed with simpler, faster delay models. Thus, we perform empirical studies of known models to assess tradeoffs between (i) accuracy relative to signoff-quality analysis tools, (ii) computation complexity and runtime, and (iii) impact on sizing results.

Figure 1 illustrates basic modeling of interconnects. In Figure 1(a), delay from pin X to pin Y is composed of gate ($cell1$, $cell2$) delay and wire (A-B) delay. For the gate/cell delay calculation, lookup table-based nonlinear delay models (NLDMs) are widely used and represent functions of input slew and output capacitance in library (*Synopsys Liberty*) files. With the NLDM, cell delay and slew estimation from a signoff timer (ST) can be reproduced with negligible errors. However, incorrect wire slew estimation (in pin B) can lead to large errors in the slew and delay estimation for $cell2$.

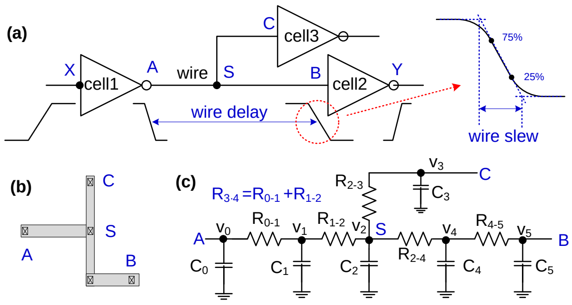


Figure 1: Interconnect modeling; (a) wire between *cell1* and *cell2*, (b) wiring tree with a Steiner point *S*, and (c) RC segment tree with RC nodes (*N*=5).

Wire delay model. We consider Elmore delay (EM) [4], D2M [2], and two 2-pole (DM1, DM2) [8] interconnect models. More complex models, such as PRIMA [15] and RICE [19], are more difficult to implement and too slow for high-performance gate sizing. Furthermore, empirical results in Section 6 show that our modeling is conducive to highly competitive results.

In an RC tree with nodes $v_0, \ldots, v_N$ (v_0 is the source) as shown in Figure 1(c), let C_i be the capacitance at node v_i for $0 < i \leq N$, and let R_{ki} be the total resistance of the intersection (overlap) between the unique path from v_0 to v_i and the unique path from v_0 to v_k. The Elmore delay from node v_0 to node v_i is given by

$$EM_i = \sum_{k=1}^{N} R_{ki} \cdot C_k. \tag{1}$$

Elmore delay is the first moment of the impulse response and can be inaccurate when there is a high degree of resistive shielding. Alpert et al. [2] propose the D2M (delay with 2 moments) metric which is a simple function of the first two circuit moments, m_1 and m_2. Starting with $m_0 = 1$, the j^{th} moment of the impulse response [2] for node v_i is defined as

$$m_j^{(i)} = -\sum_{k=1}^{N} R_{ki} \cdot C_k \cdot m_{j-1}^{(k)} \tag{2}$$

Higher moments for each node in an RC tree can be calculated by traversing the tree recursively. We can express the delay models in terms of the first and second moments as follows.

$$EM = -m_1 \qquad D2M = \ln 2 \frac{m_1^2}{\sqrt{m_2}} \tag{3}$$

$$DM1 = \frac{1}{2}(-m_1 + \sqrt{4m_2 - 3{m_1}^2}) \ln(1 - \frac{m_1}{\sqrt{4m_2 - 3{m_1}^2}}) \tag{4}$$

$$DM2 = \ln 2 \sqrt{2m_2 - {m_1}^2} \tag{5}$$

Wire slew model. We consider the PERI [9] and scaled S2M [1] models, which are calculated as

$$PERI(v_j) = \sqrt{T_{v_i}^2 + (\ln 9 \cdot m_1)^2} \tag{6}$$

where T_{v_i} and $PERI(v_j)$ are the slews at nodes v_i and v_j, respectively, and m_1 is the first moment of node v_j, and

$$S2M(v_j) = \sqrt{T_{v_i}^2 + (\ln 9 \frac{\sqrt{-m_1}}{\sqrt[4]{m_2}} \sqrt{2m_2 - m_1^2})^2} \tag{7}$$

where m_1 and m_2 are the first and second moments of node v_j and T_{v_i} is the slew of node v_i.

Capacitance model. Empirical formulas for delay and transition time of gates depend only on the input slew rate and a single load capacitance, called *effective capacitance*, which represents the cumulative effect of the load. We have implemented McCormick's effective capacitance model [13] based on a normalized 2D lookup table. The method is iterative and converges to an effective capacitance value, but is slower than closed-form models. For ISPD-2013 testcases, *total capacitances* are very close to *effective capacitances* for more than 85% of nets, providing sufficient accuracy for our delay and slew calculation in early optimization stages. Therefore, our calculations of gate delay and transition time use *total capacitance* instead of McCormick's *effective capacitance* model; this improves runtime without discernible loss of accuracy. Repeated calibration (described in Section 3.1) with a signoff timer increases accuracy of modeling at later stages.

Model selection. To select appropriate wire delay and slew model, we evaluate the timing discrepancy between Trident2.0 and the signoff timer. We implement each model for wire delay and slew, then perform static timing analysis (STA) on ISPD-2013 benchmarks [28]. Figure 2 illustrates endpoint slack error distributions for several combinations of delay models (EM, D2M, DM1 and DM2) and slew models (PERI and S2M). Total capacitance is used instead of the effective capacitance model. The plots show that (D2M, PERI) and (DM1, PERI) exhibit negligible error at around 60% of endpoints, while for other models this statistic is <50%. The error distribution for the (D2M, PERI) model combination has smaller mean (-15.9ps) and standard deviation (25.9ps), hence we select it for STA. We validate the overall optimization flow in Section 6 on ISPD-2013 contest benchmarks with a signoff timer.

3. HIGH-PERFORMANCE OPTIMIZATION

Trident2.0 follows the general outline of the Trident methodology [7] that is based on stochastic importance-sampling meta-heuristics and sensitivity-guided optimization. A major improvement upon [7] is accounting for interconnect delay and additional constraints — both extensions require the development of several new algorithmic components and closed-loop control techniques.[1] During early optimization stages, our framework performs similar parameter sweeps (including *power exponent* and *commit ratio*) to those in [7], but with coarser steps, so as to accommodate slower STA and stringent runtime constraints.[2] In response to the inclusion of interconnect delay, sensitivity functions have been revised and additional optimization steps are developed.

[1] Compared to interconnect delay modeling, cell delay modeling is relatively straightforward with lookup table-based NLDMs. At the ISPD-2012 contest, most teams implemented fast internal STA engines that exactly matched *Synopsys PrimeTime* results on ISPD-2012 benchmarks.

[2] Thus, further runtime-quality tradeoffs are possible.

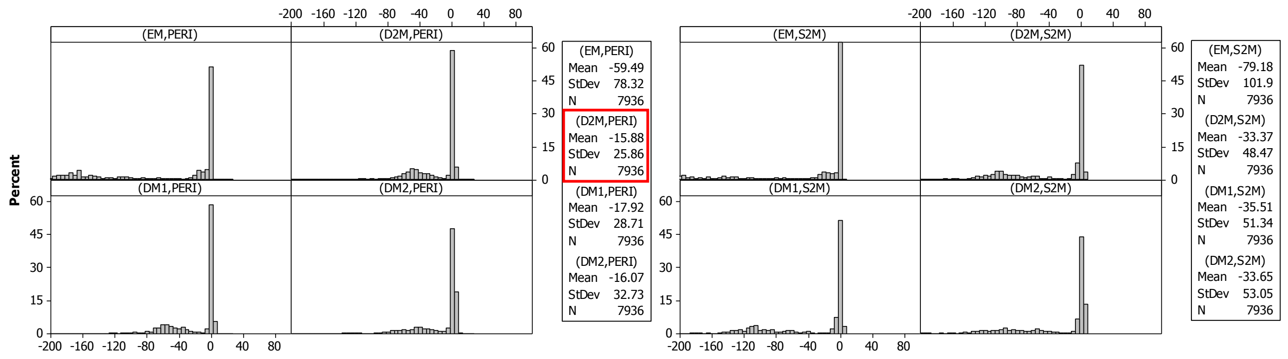


Figure 2: Endpoint slack error distribution reported by the signoff timer for the `fft_fast` testcase (x-axis: slack error (*ps*), y-axis: percentage of endpoints).

3.1 Key ideas and techniques in Trident2.0

Calibration-free early optimization. Given that calibration with the signoff timer is time-consuming, we first "warm-up" metaheuristics with a low-accuracy internal timer in order to optimize parameters of individual search heuristics. When timing constraints are loose, this stage may be sufficient to produce feasible or near-feasible solutions quickly. In general, it also enables more effective use of parallel computing resources.

Offset-based timing calibration. Moon et al. [14] introduce the idea of improving the accuracy of a given STA engine by periodically invoking a signoff timer and storing slack differences (offsets) at every timing endpoint. When the STA engine produces new estimates (e.g., during optimization), they are adjusted by slack offsets. Following up on this idea, we perform calibration with the signoff timer at every iteration of heuristic search. We use slack offsets both in full and incremental STA. As a result, there is a perfect agreement with signoff timing immediately after calibration, but the discrepancy slowly increases as cells are changed during gate sizing optimization. The frequency of calibration is determined by the maximal fraction of cells that are allowed to change. We evaluated possible thresholds of 5%, 10%, 15% and 30% cells in terms of average slack errors. Figure 3 shows the results, e.g., with the 5% threshold slack errors average $<10ps$. Based on these observations, we have chosen 5% and 10% thresholds for our flow.

Dedicated critical path optimization. We optimize critical paths by (i) downsizing non-critical fanout cells, and (ii) peephole optimization using a Gray code.
(i) *Downsizing fanout cells*. Large and low-V_{th} cells can be faster and can help reducing path delays, but their larger input capacitances degrade upstream slews and delays. The presence of interconnects aggravates this effect by increasing the overall capacitance. Given both cell and wire delay increase, upsizing alone is insufficient for reliable timing recovery. Our insight is that downsizing certain non-critical cells can reduce critical path delay. In particular, we focus on fanout cells of the cells lying on critical paths — downsizing these fanout cells reduces capacitance driven by critical cells. As a side effect, the delay of those fanout cells increases, thus they should not themselves lie on critical paths. We select downsizing candidates as fanout cells of critical path cells c based on the *sensitivity function*, $SF_{down} = C_{out}(c)/size(c)$, where $C_{out}(c)$ is the capacitance driven by cell c.[3] If downsizing a candidate cell decreases negative slack, we restore previous size and continue to the next candidate.
(ii) *Peephole optimization using a Gray code*. We consider several cells at a time and exhaustively evaluate size combinations within a given radius of the cells' current sizes. For example, if three choices are considered for three cells — one size up, one size down and no change, — then 27 combinations would be evaluated. The use of a Gray code, i.e., traversing all size combinations by modifying one cell at a time, accelerates incremental timing analysis. In particular, our incremental STA engine performs timing updates faster when the amount of change is smaller.

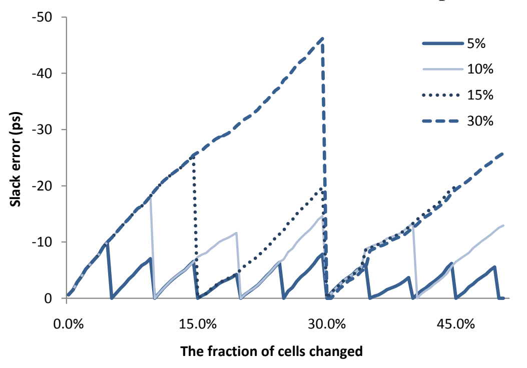


Figure 3: The impact of *calibration frequency* on *slack error* while sizing the `fft_fast` benchmark. Each line corresponds to a frequency and shows the average slack error at timing (end)points against the fraction of cells changed.

Sensitivity functions. To identify the most promising cells to size, several stages of our optimization take into account (i) the direct impact of sizing a given cell on its slack, (ii) the required increase in leakage power, and (iii) the number of critical paths whose slack is improved. These parameters are combined into a *sensitivity score*, by which candidate cells are ranked. Thus, non-critical cells are not considered for upsizing during sensitivity-guided optimization, but small cells lying on numerous critical paths (bottleneck cells) are given higher priority. In practice, no single sensitivity function dominates other functions and the most accurate computations are prohibitively expensive. Trident [7] approximates the impact of single-cell changes on total negative slack. Significant efficiency is achieved by only propagating cell delay, and this abstraction works well for the ISPD-2012 contest infrastructure where only *gate* delays are computed. In contrast, the ISPD-2013 contest adds interconnect considerations and requires more comprehensive delay modeling. In particular, one must model slew degradation in wires on a timing path and the impact of slew on delay.

To track the impact of single-cell changes more accurately, we calculate slack updates considering both delay and slew. In our two global timing recovery (GTR) stages, we account for slack change ($\Delta slack$), the number of paths passing through the cell ($\#paths$), and the change in leakage ($\Delta leakage\_power$). The latter is raised to power ($power\_exponent$) — a configurable parameter for the sensitivity function (SF).

$$SF_{GTR} = \frac{\Delta slack \cdot \#paths}{\Delta leakage\_power^{power\_exponent}} \tag{8}$$

The power reduction with feasible timing (PRFT) stage uses sensitivity functions from Trident [7].

[3] Cells that have higher SF_{down} values are optimized first.

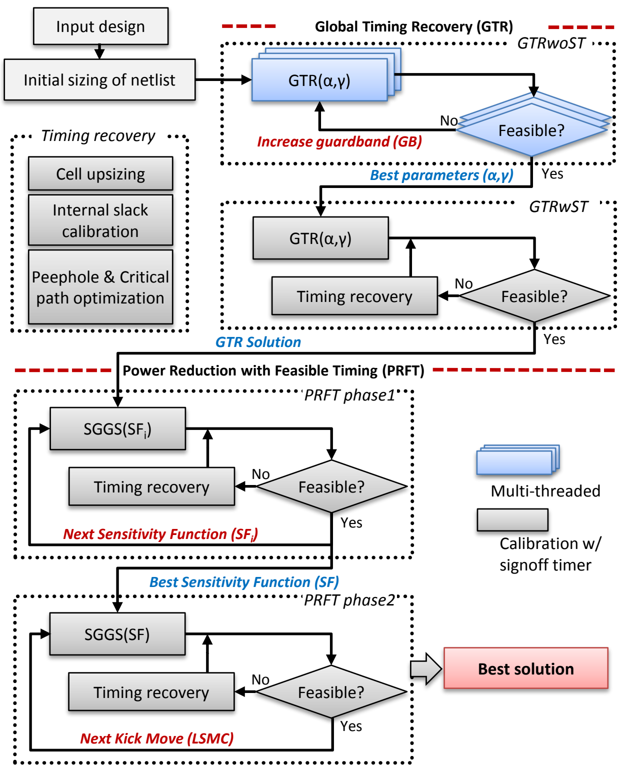


Figure 4: Overall optimization flow.

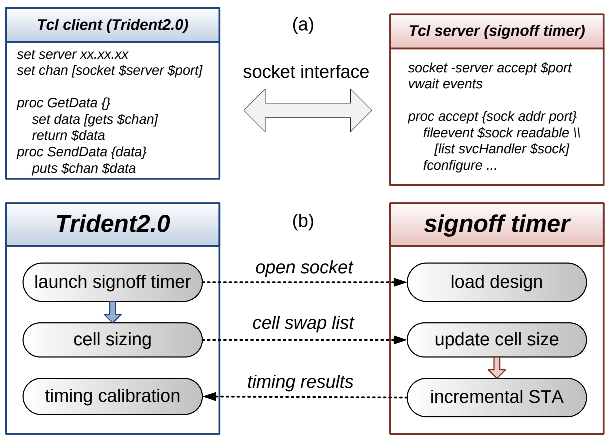
3.2 The overall optimization flow

Figure 4 shows the overall optimization flow in Trident2.0. We initially rely on an internal timer. A multi-threaded metaheuristic optimizes individual parameters of lower-level search heuristics, similar to the GTR stage in Trident [7], but with interconnect delay models and constraints. The second stage performs timing calibration with the signoff timer and seeks to produce a feasible solution with respect to signoff timing. This stage also uses techniques from GTR [7], but is more accurate and more constrained in terms of runtime. The third stage performs power reduction with feasible timing and roughly corresponds to PRFT in [7].

Stage 1. GTR without a signoff timer (*GTRwoST*). This stage seeks to satisfy timing constraints with respect to our internal STA engine. Due to discrepancies with the signoff timer, this solution may not be signoff-feasible. Further improvements will be performed by slower yet more accurate optimization stages. However, much of the work in exploring the overall solution space is performed at this early stage with a fast timer. Moreover, this stage optimizes the configurations of sensitivity functions (e.g., power exponent α and commit ratio γ in Trident [7]) by metaheuristic search based on importance sampling. If a timing-feasible solution cannot be found, *guardband* (GB)[4] is applied until a feasible solution is found with a loose timing constraint.

Stage 2. GTR with a signoff timer (*GTRwST*). With the best parameters from first phase of GTR, this phase finds a feasible solution based on the signoff timer. To remove the timing discrepancy versus the signoff timer, we calibrate internal slack values as described in Section 3.1.After each iteration of global cell upsizing, we calibrate internal slack values, and check the number of violations. If the number of violations does not decrease after each global upsizing, Trident2.0 optimizes the worst negative-slack paths as well. For the critical path optimization, we downsize

[4]A positive (negative) GB means tighter (looser) timing constraint than the original clock period.



```
set server xx.xx.xx
set chan [socket $server $port]

proc GetData {}
    set data [gets $chan]
    return $data
proc SendData {data}
    puts $chan $data
```

```
socket -server accept $port
vwait events

proc accept {sock addr port}
    fileevent $sock readable \\
      [list svcHandler $sock]
    fconfigure ...
```

Figure 5: Socket interface between Trident2.0 and the signoff timer: (a) *Tcl* socket code and (b) timing calibration with the socket interface.

fanout cells which are not in the critical path. Peephole optimization using a Gray code is applied on worst paths if there are still violation after the previous optimizations. These two optimizations were discussed in detail in Section 3.1. The *GTRwST* phase uses one thread with one signoff timer invocation.

Stage 3. Power Reduction with Feasible Timing (*PRFT*) is performed in two phases. Starting with the best feasible solution from GTRwST, Trident2.0 performs a sensitivity-guided greedy downsizing to reduce leakage power subject to timing constraints. The greedy optimization is attempted with different sensitivity functions, and violations that occur as a result of downsizing are fixed with timing recovery iterations (upsizing). We try different sensitivity functions and carry over the best solution among trials to the second phase (or, otherwise, the configuration with the smallest amount of violations). Downsizing occasionally introduces constraint violations, which we fix on-demand with iterative upsizing, peephole optimization and critical path optimizations (similar to GTRwST). The latter two optimizations are also capable of improving leakage. The *second phase of PRFT* uses the most successful sensitivity function from the first phase and performs additional greedy (down) sizing with kick-moves. Kick-moves described in [7] can be viewed as implementing large-step Markov chain (LSMC) optimization.

4. PRACTICAL INSIGHTS FOR SUCCESSFUL IMPLEMENTATION

In this section, we present several key implementation issues and insights that were not stressed in the above discussions.

4.1 Signoff-timer interface

Given frequent timing calibration, the interface between the internal timer and the signoff timer must be efficient. ISPD-2013 contest infrastructure [28] includes a file-based interface that requires saving and reading large files, and starting a new instance of the signoff timer on each invocation. This interface is particularly inefficient for incremental STA and when performing targeted optimization of critical paths. Instead, we interface with the signoff tool using a *Tcl*-socket interface, similar to the one in UCSD SensOpt [22].[5] Our interface is illustrated in Figure 5, including client-server *Tcl* socket code [31].

When timing calibration is initiated, we launch the signoff timer and open a Unix socket. An open socket allows a program to send

[5]Unix sockets are a standard means for interprocess communication (IPC). Commercial signoff timing tools, such as ExtremeDA GoldTime [30] and Cadence Encounter Timing System [27], provide engine access through sockets.

Benchmarks	GTR		PRFT		Runtime		1st place team		2nd place team		3rd place team	
	leakage (mW)	runtime (min)	leakage (mW)	runtime (min)	total (min)	limit (min)	leakage (mW)	runtime (min)	leakage (mW)	runtime (min)	leakage (mW)	runtime (min)
`usb_phy_fast`	1.78	0.12	**1.60**	0.01	**0.14**	48	1.64	0.25	2.63	2.71	1.61	0.58
`usb_phy_slow`	1.12	0.11	**1.08**	0.01	**0.13**	48	1.08	0.25	1.09	0.67	1.08	0.40
`pci_b32_fast`	155.39	1.40	105.30	0.60	**2.40**	48	112.64	2.91	504.98	48.03	**96.11**	23.61
`pci_b32_slow`	67.60	0.90	60.20	0.50	**1.80**	48	60.17	2.25	136.95	48.02	**57.89**	9.58
`fft_fast`	678.90	3.40	342.10	1.69	**5.50**	48	361.30	6.93	974.60	48.03	**224.53**	30.54
`fft_slow`	144.40	1.90	96.25	1.11	**3.48**	48	98.15	4.1	418.99	48.03	**90.32**	22.78
`cordic_slow`	1546.30	20.00	394.70	5.20	**25.70**	50	563.09	26.16	1961.09	60.06	**323.71**	49.65
`des_perf_slow`	616.40	12.00	391.90	5.90	**19.23**	72	395.86	20.48	2823.88	72.26	**353.80**	67.85
`edit_dist_fast`	1260.40	30.04	**689.90**	10.60	**42.20**	84	704.82	44.25	9769.38	84.18	—	—
`edit_dist_slow`	721.30	20.18	**487.90**	9.80	**31.60**	84	489.47	33.21	7485.66	84.18	90.31	22.78
`matrix_m_slow`	1118.48	37.70	**562.40**	37.50	**77.30**	84	570.74	65.17	7540.34	84.16	—	—
`netcard_fast`	5764.40	100.98	**5277.40**	71.80	**190.77**	310	—	—	—	—	—	—
`netcard_slow`	5371.04	71.20	**5184.20**	59.50	**148.60**	310	5371.10	336.30	—	—	—	—

Table 1: Results for Trident2.0 (GTR+PRFT) on ISPD-2013 benchmarks *in fast mode*. Benchmarks where no team satisfied timing constraints are not shown because detailed results were not reported at the contest. All solutions reported exhibit zero violations. Long dashes (—) indicate solutions with violations, crashes and time-outs. Contest scores are computed based on runtime and leakage power. Note that Trident2.0 outperforms the winning team in both runtime and leakage power on every benchmark except for `pci_b32_slow`, where leakage is very slightly higher, but runtime is significantly smaller. Benchmark statistics are listed in Table 2.

Benchmarks	Clock period (ps)	#Cells	GTR		PRFT		Runtime		Leakage achieved by top-ranked teams		
			leakage (mW)	runtime (min)	leakage (mW)	runtime (min)	total (min)	limit (min)	1st place (mW)	2nd place (mW)	3rd place (mW)
`usb_phy_fast`	300	608	1.59	0.13	**1.59**	0.08	0.21	240	1.61	1.68	6.55
`usb_phy_slow`	450	608	1.11	0.12	**1.07**	0.05	0.17	240	1.08	1.07	1.12
`pci_b32_fast`	750	30603	147.30	3.19	101.90	8.79	12.00	240	**96.51**	106.93	—
`pci_b32_slow`	1000	30603	65.12	1.93	58.83	3.40	5.39	240	**57.89**	59.26	77.18
`fft_fast`	1400	32766	609.20	13.80	305.29	18.80	32.58	240	**226.20**	321.45	637.81
`fft_slow`	1800	32766	128.0	6.70	93.10	10.30	17.40	240	**90.34**	97.71	106.68
`cordic_slow`	3000	42903	1546.30	64.80	511.91	33.10	98.39	300	**323.79**	443.61	1077.73
`des_perf_slow`	1300	113112	601.28	27.40	375.80	33.50	62.30	360	**353.00**	380.44	2391.83
`edit_dist_fast`	3000	126665	1173.40	94.50	619.30	74.50	170.60	420	**596.32**	639.01	—
`edit_dist_slow`	3600	126665	632.50	64.79	465.60	42.80	107.20	420	**447.40**	468.45	—
`matrix_m_slow`	2800	156440	957.30	117.20	499.90	93.40	212.60	420	**469.73**	512.85	1381.37
`netcard_fast`	2000	982258	5764.40	269.40	**5271.80**	429.35	716.80	1650	5317.84	—	19152.00
`netcard_slow`	2400	982258	5371.04	184.60	**5183.89**	237.07	439.60	1650	5302.27	5371.10	5245.66

Table 2: Results for Trident2.0 (GTR+PRFT) on ISPD-2013 benchmarks *in normal mode*. Benchmarks where no team satisfied timing constraints are not shown because detailed results were not reported at the contest. All solutions reported exhibit zero violations.

commands (e.g., cell sizing and timing query commands) to the signoff timer and receive data (e.g., updated transition time and slack). Changes made to gate sizes during optimization (GTR and PRFT) are communicated to the signoff timer, which returns results of incremental STA used to re-calibrate our internal timer. In Trident2.0, communications with the signoff timer are always performed in conjunction with the internal timer.

4.2 Handoff between optimization stages

As noted earlier, each optimization stage in Trident2.0 pursues its own goals, often using parallel search, and hands off the best solutions found to the next stage. This modularity is not only convenient for software development, maintenance and testing, but also allows us to combine optimizations with very different runtime-quality tradeoffs as well as carefully tune each stage for reliability and performance. Two GTR stages seek violation-free (timing-feasible) solutions; GTRwoST is more computationally efficient than GTRwST, but also less accurate. The PRFT stage iteratively reduces total leakage power while respecting timing constraints. Combining such diverse optimization stages requires particular attention to the handoff between them. Approximately matching the strength of optimization before and after the handoff helps improve the stability of the overall flow. Note that out of 13 benchmarks where timing constraints were satisfied, GTRwoST finds timing-feasible solutions in eight, and the signoff timer takes <50% runtime in those cases. Thus, using our internal timer without calibration during early search is compatible with later optimization stages, helps reduce runtime and allows Trident2.0 to explore a larger solution space.

4.3 Scalability

Support for parallelism. The initial search for a timing-feasible configuration (GTRwoST) is performed in parallel. Our implementation uses up to 16 threads. As these threads are essentially independent, further scalability is mostly limited by memory usage and diminishing returns in terms of solution quality from randomized multi-starts. While searching for feasible cell sizes/configurations, GTRwoST identifies best parameters for search heuristics in GTR, as explained in [7]. To exploit parallelism in further stages, it is important to parallelize invocations of the signoff timer and have multiple licenses available. The file-based interface provided with the ISPD-2013 contest infrastructure does not support parallel invocation of the signoff timer. We found such extensions challenging with our socket-based interface as well, mostly due to undeterministic race conditions. Assuming a reliable infrastructure for parallelism, last-stage local optimizations appear *a priori* amenable to parallel execution. However, one must first study runtime breakdown and identify bottlenecks.

Runtime breakdown. Figure 6 shows the runtime breakdown of Trident2.0 for individual ISPD-2013 benchmarks and the fraction of runtime taken by the signoff timer. In both fast and normal modes, difficult netlists such as `cordic` and `edit_dist` take

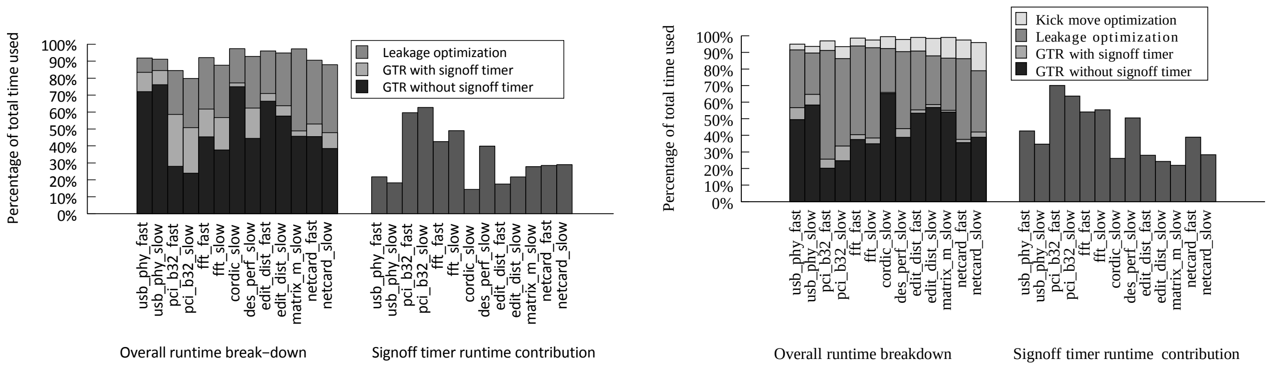


Figure 6: Runtime breakdown for Trident2.0 on ISPD-2013 benchmarks in fast mode (left) and normal mode (right). Multiple optimization stages use the signoff timer.

longer in GTRwoST because it is harder to satisfy timing constraints, even with respect to our internal timer. Once a timing-feasible solution is found, GTRwST runtime is less sensitive to the difficulty of the benchmark, even though GTRwST is much slower due to calibration with the signoff timer. This trend is also apparent in the percentage runtime contribution of the signoff timer being comparatively small for these benchmarks in both modes. When GTRwoST does not find a timing-feasible solution (`pci_b32_fast`, `fft_*`), relatively more time is spent in signoff-timer calls.

5. COMPARISONS TO PRIOR RESEARCH

ISPD-2012 and 2013 Gate Sizing Contests [17,18] have dramatically changed the landscape of research in the field. To this end, the benchmarking infrastructure developed by Intel researchers does not have academic precedents in terms of

- using discrete gate sizes and V_{th} assignment,
- relying on an industry-standard signoff timer,
- increasing the scale of optimization to netlists with hundreds of thousands of cells,
- using realistic technology models (cell timing, drive, power) and timing constraints, and
- imposing capacitance and slew constraints.

The two contests attracted several dozen research teams from all over the world, but few teams produced competitive solutions. Consequently, few publications describe relevant algorithms. Among pre-contest tools, UCSD SensOpt [22] uses sensitivity-guided optimization for post-layout discrete gate sizing. It communicates with an industry signoff timer through a *Tcl* interface. UCLA OA Sizer [22] implements greedy optimization, linear programming, Lagrangian relaxation and dynamic programming, while relying on the OAGear-Static-Timer.

Core optimization techniques used in our work — metaheuristic optimization with importance sampling and sensitivity-guided search — have been developed in [7], and we extend them. Direct comparisons with the tool from [7] on ISPD-2012 benchmarks show that our implementation runs approximately 10× faster, primarily due to streamlined data structures, but results in slightly higher leakage power. Unlike [7], we keep track of interconnect delay and slew, and per-pin timing slack (plus, offsets with respect to a signoff timer). Given the relatively simple timing models used at the ISPD-2012 contest, [7] did not need to directly invoke a signoff timer. Our optimization must invoke a signoff timer, creating a number of complications that lead to major structural changes and different parameter settings. Several new techniques for modeling and optimization are required to make metaheuristics from [7] successful when interconnects are considered.

The performance of Lagrangian relaxation techniques on ISPD-2012 benchmarks was described in [11,12]. Empirically, runtimes show significant improvement over [7], but at the cost of increased leakage. Interconnect delay modeling and optimization are not discussed in [11,12]. These considerations completely change the nature of the overall optimization, making it impossible to reliably extrapolate the performance of Lagrangian relaxation to the ISPD-2013 benchmark suite, as several algorithmic components must be developed to enable a full comparison.

6. EMPIRICAL VALIDATION

Optimization trajectories pursued by Trident2.0. Figure 7 illustrates the reduction of normalized worst negative slack (WNS) with GTR iterations during GTRwoST and GTRwST on multiple ISPD-2013 benchmarks. GTRwoST reduces WNS quickly because many cells can be upsized to improve circuit delay. Figure 8 illustrates the progress of normalized leakage power with GTR iterations during GTRwoST and GTRwST. When Trident2.0 does not find a feasible solution in GTRwoST in the first 8-10 iterations, leakage power quickly increases with many changes made to cells, but then saturates because few possible cell moves are available.

The impact of timing calibration. We evaluate leakage-power optimization with PRFT in five cases: (*i*) frequent calibration (after every 5% of cells change), (*ii*) one-time calibration before PRFT, (*iii*) no calibration, (*iv*) using a 5*ps* guardband (GB) without calibration and (*v*) using a 10*ps* GB without calibration. For each case, Figure 9 shows leakage (normalized to case *i* after timing recovery), total negative slack (TNS) and WNS after PRFT and timing recovery. Frequent calibration achieves smallest leakage power without timing violations.

Without calibration (case *iii*), solutions may be infeasible with respect to signoff timing. Feasible solutions can sometimes be produced without any calibration by using a GB (cases *iv* and *v*), but this pessimism will limit leakage reduction. For example, violation-free solutions are produced with a 10*ps* GB for `pci_b32_fast` and 5*ps* for `fft_fast`, but leakage power increases by 6% versus frequent calibration, due to the excessive cell upsizing. At the PRFT stage, WNS and TNS are larger (better) with calibration due to pessimism in our internal timer (Figure 2). One-time calibration exhibits the largest errors in WNS and TNS, suggesting that the timing discrepancy with the signoff timer increases as more cells undergo size changes.

Comparisons to ISPD-2013 contest results. Tables 1 and 2 report our results on ISPD-2013 benchmarks and official ISPD-2013 contest results. Trident2.0 finds violation-free solutions whenever any contestants found them. For other benchmarks, detailed data are unavailable and we omit them from the tables. By the primary metric (leakage power), Trident2.0 places between the first- and

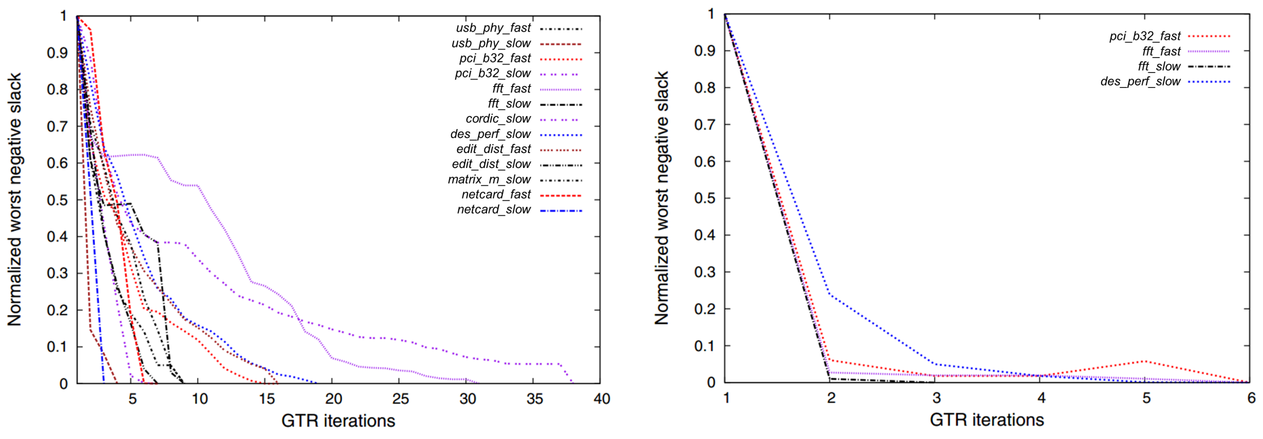


Figure 7: Normalized worst negative slack during GTR without (left) and with (right) signoff timer *in fast mode*. The corresponding plot for normal mode is very similar.

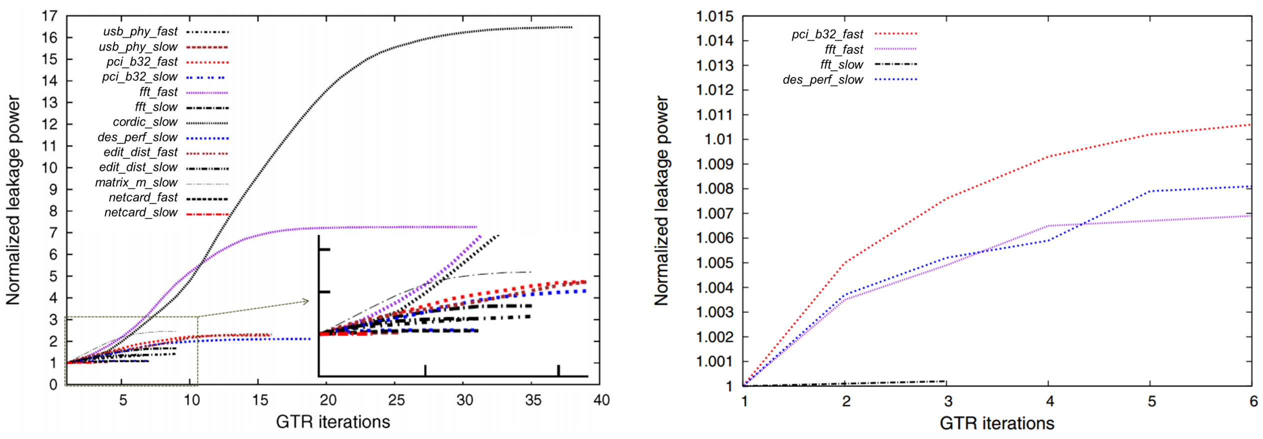


Figure 8: Normalized leakage during GTR without (left) and with (right) signoff timer *in fast mode*. The corresponding plot for normal mode is very similar.

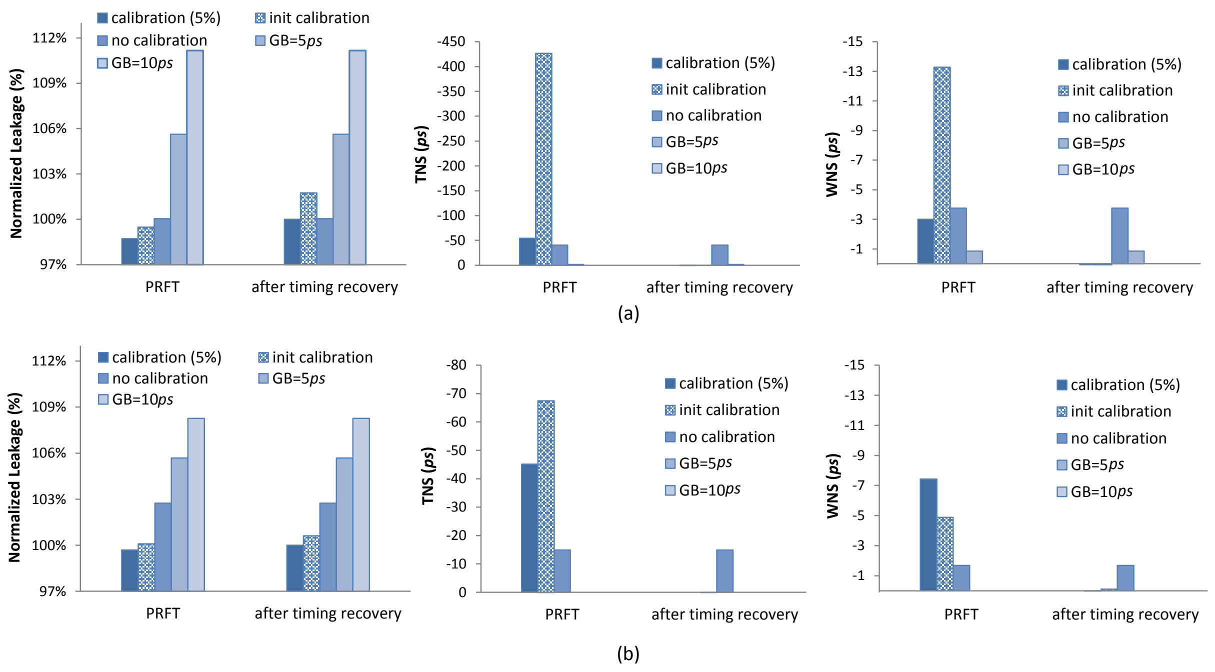


Figure 9: Impact of calibration on leakage reduction and timing recovery with various guardband (GB) values: (a) `pci_b32_fast`, (b) `fft_fast`.

second-place teams. Our results with the secondary metric (based on leakage power and runtime) are ahead of the first-place team — we achieve smaller leakage, spend less runtime, and find feasible solutions on more benchmarks. In the normal (not fast) mode, Trident2.0 finishes ahead of the runtime limit and could produce better results by using this runtime wisely.

7. CONCLUSIONS

We have studied power minimization during discrete gate-sizing and V_{th} assignment in large netlists with respect to a signoff timer. The significance of this optimization is growing because semiconductor scaling is not providing the same reductions in circuit power and delay that it provided in previous technology nodes. As a result of scaling trends, leakage power reduction has become one of the main challenges to the semiconductor and EDA industries, and will likely remain critical in future technology nodes. Prior literature in the field leaves obvious room for improvement and obvious gaps, which our research seeks to address. For example, published methods reliant on continuous-valued and convex optimization can be improved upon by embracing the combinatorial nature of the problem, which we accomplish through metaheuristics for discrete stochastic optimization based on [7]. In contrast to simplified timing evaluation in prior academic research, our empirical validation uses a signoff timer and the infrastructure of the ISPD-2013 Gate Sizing Contest.

To achieve a high-performance gate-sizing optimization, Trident2.0 has addressed several major challenges, namely, to

- identify interconnect models whose accuracy-vs.-complexity tradeoffs are compatible with large-scale optimization;
- develop an internal timer fast enough for move-based optimization, yet accurate enough to track a signoff timer;
- obtain sharp tradeoffs between timing and power reduction;
- satisfy timing, slew and capacitance constraints; and
- use parallel computing resources effectively.

In solving these challenges, we put significant effort into not only individual techniques and components, but also into the entire system, paying attention to the stability and scalability of optimization. The software we developed achieves highly competitive results compared to the ISPD-2013 contest winners. In particular, Trident2.0 outperforms the winners in the secondary-metric competition and places between first and second by the primary metric. Given how recently the ISPD-2013 contest concluded, We anticipate that our ongoing research will yield further improvements.

8. REFERENCES

[1] K. Agarwal, D. Sylvester and D. Blaauw, "A Simple Metric for Slew Rate of RC Circuits Based on Two Circuit Moments", *IEEE Trans. on CAD* 23(9) (2004), pp. 1346–1354.

[2] C. J. Alpert, A. Devonna and C. Kashyap, "A Two Moment RC Delay Metric for Performance Optimization", *Proc. ISPD*, 2000, pp. 73–78.

[3] D. Chinnery, "High Performance and Low Power Design Techniques for ASIC and Custom in Nanometer Technologies", *Proc. ISPD*, invited talk, 2013.

[4] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard To Wideband Amplifiers", *Journal of Applied Physics* 19(1) (1948), pp. 55–63.

[5] S. Held, "Gate Sizing for Large Cell-Based Designs", *Proc. DATE*, 2009, pp. 827–832.

[6] S. Hu, M. Ketkar and J. Hu, "Gate Sizing for Cell-Library-Based Designs", *IEEE Trans. on CAD* 28(6) (2009), pp. 818–825.

[7] J. Hu, A. B. Kahng, S. Kang, M.-C. Kim and I. L. Markov, "Sensitivity-guided Metaheuristics for Accurate Discrete Gate Sizing", *Proc. ICCAD*, 2012, pp. 233–239.

[8] A. B. Kahng and S. Muddu, "An Analytical Delay Model for RLC Interconnects", *IEEE Trans. on CAD* 16(12) (1997), pp. 1507–1514.

[9] C. V. Kashyap, C. J. Alpert, F. Liu and A. Devgan, "PERI: A Technique for Extending Delay and Slew Metrics to Ramp Inputs", *Proc. TAU*, 2002, pp. 57–62.

[10] Y. Liu and J. Hu, "A New Algorithm for Simultaneous Gate Sizing and Threshold Voltage Assignment", *IEEE Trans. on CAD* 29(2) (2010), pp. 223–234.

[11] V. S. Livramento, C. Guth, J. L. Güntzel and M. O. Johann, "Fast and Efficient Lagrangian Relaxation-Based Discrete Gate Sizing", *Proc. DATE*, 2013, pp. 1855–1860.

[12] Y. Lu, H. Zhou, Li Li, P. Kang, Y. Lu and H. Zhou, "An Efficient Algorithm for Library-based Cell-type Selection in High-Performance Low-Power Designs", *Proc. ICCAD*, 2012, pp. 226-232.

[13] S. P. McCormick, "Modeling and Simulation of VLSI Interconnects with Moments", *PhD Thesis*, MIT, June 1989.

[14] C. Moon, P. Gupta, P. J. Donehue and A. B. Kahng, "Designing a Digital Circuit by Correlating Different Static Timing Analyzers", *U.S. Patent* No. 7,823,098, 2010.

[15] A. Odabasioglu, M. Celik and L.T. Pileggi, "PRIMA: Passive Reduced-Order Interconnect Macromodeling Algorithm", *Proc. ICCAD*, 1997, pp. 58–65.

[16] M. M. Ozdal, S. Burns and J. Hu, "Gate Sizing and Device Technology Selection Algorithms for High-Performance Industrial Designs", *Proc. ICCAD*, 2011, pp. 724–731.

[17] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, "ISPD-2012 Discrete Cell Sizing Contest and Benchmark Suite", *Proc. ISPD*, 2012, pp. 161–164, `http://archive.sigda.org/ispd/contests/12/ispd2012_contest.html`.

[18] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, "An Improved Benchmark Suite for the ISPD-2012 Discrete Cell Sizing Contest", *Proc. ISPD*, 2013, pp. 168–170, `http://archive.sigda.org/ispd/contests/13/ispd2013_contest.html`.

[19] C.L. Ratzlaff, N. Gopal and L.T. Pillage, "RICE: Rapid Interconnect Circuit Evaluator", *Proc. DAC*, 1991, pp. 555–560.

[20] S. Roy, W. Chen, C. C.-P. Chen and Y. H. Hu, "Numerically Convex Forms and Their Application in Gate Sizing", *IEEE Trans. on CAD* 26(9) (2007), pp. 1637–1647.

[21] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya and S.-M. Kang, "An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization", *IEEE Trans. on CAD* 12(11) (1993), pp. 1621–1634.

[22] UCSD SensOpt Leakage Optimizer (A. B. Kahng and S. Kang, 2010-2011), `http://vlsicad.ucsd.edu/SIZING/optimizer.html`.

[23] S. Shah, A. Srivastava, D. Sharma, D. Sylvester, D. Blaauw and V. Zolotov, "Discrete V_t Assignment and Gate Sizing Using a Self-Snapping Continuous Formulation", *Proc. ICCAD*, 2005, pp. 704–711.

[24] A. Srivastava, D. Sylvester and D. Blaauw, "Power Minimization Using Simultaneous Gate Sizing, Dual-V_{dd} and Dual-V_{th} Assignment", *Proc. DAC*, 2004, pp. 783–787.

[25] H. Tennakoon and C. Sechen, "Gate Sizing Using Lagrangian Relaxation Combined with a Fast Gradient-Based Pre-Processing Step", *Proc. ICCAD*, 2002, pp. 395–402.

[26] J. Warnock, "Circuit and PD Design Challenges at the 14nm Technology Node", *Proc. ISPD*, invited talk, 2013.

[27] *Cadence Encounter Timing System User's Manual*, `http://www.cadence.com`.

[28] ISPD 2013 Discrete Gate Sizing Contest and Benchmark Suite, `http://ispd.cc/contests/13/ispd2013_contest.html`.

[29] *Synopsys PrimeTime User's Manual*, `http://www.synopsys.com`.

[30] *ExtremeDA GoldTime User's Manual*, Extreme DA Corp., 2010.

[31] *Tcl/Tk Built-in Socket Commands Manual*, `http://www.tcl.tk/man/tcl8.4/TclCmd`.