Power Reduction via Near-Optimal Library-Based Cell-Size Selection

Mohammad Rahman, Hiran Tennakoon and Carl Sechen
Department of Electrical Engineering, University of Texas at Dallas, United States
rahman@student.utdallas.edu, hiran.tennakoon@utdallas.edu, carl.sechen@utdallas.edu

***Abstract*— Assuming continuous cell sizes we have robustly achieved global minimization of the total transistor sizes needed to achieve a delay goal, thus minimizing dynamic power (and reducing leakage power). We then developed a feasible branch-and-bound algorithm that maps the continuous sizes to the discrete sizes available in the standard cell library. Results show that a typical library gives results close to the optimal continuous size results. After using state-of-the-art commercial synthesis, the application of our discrete size selection tool results in a dynamic power reduction of 40% (on average) for large industrial designs.**

Keywords- power-delay optimization, discrete cell-size selection, delay modelling, parallelism

I. INTRODUCTION

Automatic synthesis tools have not been effective at global power minimization for a target frequency. We need an efficient library-based cell-size selection technique to achieve near optimal power efficiency for any desired frequency target for a synthesized design. The cell selection problem is to find the optimum mapping of the library cells to a design, without changing the logic composition of the netlist, such that all design constraints are met. This is inherently a discrete problem and has been shown to be NP-Hard [1]. Two solution approaches are: 1) solve the problem within the discrete domain [2][3], or 2) relax the discrete solution space to be continuous, *i.e.,* any size for a cell can be realized between minimum and maximum size constraints. Then mathematical optimization is used to solve the continuous problem with models for delay, dynamic power, and leakage power. Then there is an additional mapping step that converts the continuous sizes to the discrete sizes available in the library [4][5]. The advantages of the second approach are: 1) the continuous solution provides a guide for selecting candidate discrete sizes to be investigated per cell from a larger set available in the library, and 2) if the continuous solution is optimal, it can be used as a reference point to analyze the quality of the discrete selection algorithm. We adopt this approach here, and developed an optimal cell size selection algorithm yielding the minimum total active area (sum of transistor sizes) for any feasible delay target. We also developed a new feasible branch-and-bound discrete cell size selection algorithm that maps an optimal continuous result to the library cells, and which achieves for the first time active area versus delay results close to the continuous results.

II. OPTIMAL CONTINUOUS CELL SIZING

A. Problem Formulation

Let C be the set of all cells in a design and $w_{n,i}$, $w_{p,i}$ represent nMOS and pMOS sizes of the transistors used in cell i, respectively. D_i is the delay of cell i. Vector $W_n = \{w_{n,i}\}$ ($W_p = \{w_{p,i}\}$) represents the set of all nMOS (pMOS) sizes. Each cell has lower and upper bounds on its size, for both nMOS and pMOS transistors, leading to vectors L_n, U_n, L_p and U_p. $f(W)$ in (1) represents the total active area.

$$\begin{aligned} &\text{Minimize } f(W) \\ &\underset{w}{\text{Subject to :}} \\ &a_i \le \text{Target Delay} \qquad i \in \{\, primary\ outputs \,\} \\ &a_j + D_i \le a_i \qquad i \in C,\ j \in input\{\, i \,\} \\ &L_n \le W_n \le U_n,\ \ L_p \le W_p \le U_p \end{aligned} \tag{1}$$

B. Delay Model

Industry sign-off static timing analyzers use pre-characterized delay data present in the library database (*.lib*) file. We propose a new table-lookup delay model derived from existing *.lib* files by extending the logical effort model [6]. The key idea is to define separate rise/fall values for electrical efforts (h_r/h_f), logical efforts (g_r/g_f), and parasitic delays (p_r/p_f) enabling modeling of separate rise/fall delays for individual cells as well as circuit paths. The rise electrical effort (h_r) is defined as the output capacitive load (C_{out}) divided by the size of a pMOS transistor (w_p) in the pull-up network. Similarly, the fall electrical effort (h_f) is defined as C_{out} divided by the size of an nMOS transistor (w_n) in the pull-down network. The rise and fall delays (D_r/D_f) for a gate are shown in (2). C_{wire} is the wire load driven by the gate and c is the capacitance per unit transistor width derived from the *.lib* file. Distinct rise and fall electrical efforts enable separate optimization of nMOS and pMOS devices, thus optimizing the beta ratio of each gate.

$$\begin{aligned} &D_r = g_r h_r + p_r \ \text{ and } \ D_f = g_f h_f + p_f \\ &h_r = \frac{C_{out}}{w_p} = \frac{\sum_{fanout} c(w_n + w_p) + C_{wire}}{w_p} \\ &h_f = \frac{C_{out}}{w_n} = \frac{\sum_{fanout} c(w_n + w_p) + C_{wire}}{w_n} \end{aligned} \tag{2}$$

We characterized the extended logical effort model parameters ($g_{r,f}$, $p_{r,f}$) in the format of lookup tables as functions of transistor sizes (w_n,w_p), input slew-rate, and threshold voltage from the pre-characterized standard *.lib* files. We found that the $g_{r,f}$, $p_{r,f}$ values vary smoothly (relatively linearly), enabling very accurate table look-up for multi-level interpolation. In this way, we maintained high accuracy (within 1-2% for the worst case and far less for the average case) with respect to the leading commercial static timing analyzers.

C. Optimal Sizing Formulation

We applied the Lagrangian relaxation technique to optimally solve (1) (Primal Problem). The Lagrangian L is formed by

978-3-9810801-7-9/DATE11/©2011 EDAA

introducing a vector of Lagrange multipliers λ for each delay constraint which leads to the Lagrangian dual problem:

$$\begin{aligned}&\text{Maximize } L(W,\lambda) = f(W) + \lambda^T\left(g(W) - A\right)\\ &\underset{\lambda}{\text{Subject to}}:\\ &L_n \le W_n \le U_n,\ L_p \le W_p \le U_p \text{ and } \lambda \ge 0\end{aligned} \tag{3}$$

λ is a column vector of Lagrange multipliers associated with each delay constraint for a cell and ($g(W)$-A) is a column vector corresponding to the difference in the delay and arrival time requirements for each cell. Thus, the primal problem is to optimize the sizes for a given set of Lagrange multipliers and the dual problem is to optimize the multipliers to get the best lower bound of the primal problem. If the solutions to both problems are equal then a global optimum has been achieved [7]. For the circuit fragment shown in Figure 1, following the general Lagrangian formulation of [8] that we extended to handle separate rise and fall delays, we can formulate the problem of active area minimization for a given target delay a_o as in (4), where each D is given by an expression of the type in (2).

$$\begin{aligned}&\text{Minimize } \sum_i \alpha\left(w_{n,i} + w_{p,i}\right)\\ &\text{Subject to}:\\ &\quad a_{6r} \le a_o, a_{6f} \le a_o, a_{7r} \le a_o, a_{7f} \le a_o\\ &\quad a_{5r} + D_{56f} \le a_{6f}, a_{5f} + D_{56r} \le a_{6r}\\ &\quad a_{5r} + D_{57f} \le a_{7f}, a_{5f} + D_{57r} \le a_{7r}\\ &\quad a_{4r} + D_{45f} \le a_{5f}, a_{4f} + D_{45r} \le a_{5r}\\ &\quad a_{3r} + D_{35f} \le a_{5f}, a_{3f} + D_{35r} \le a_{5r}\\ &\quad a_{2r} + D_{24f} \le a_{4f}, a_{2f} + D_{24r} \le a_{4r}\\ &\quad a_{1r} + D_{13f} \le a_{3f}, a_{1f} + D_{13r} \le a_{3r}\\ &\quad L_n \le X_n \le U_n,\ L_p \le X_p \le U_p\end{aligned} \tag{4}$$

The constraints in (4) can be relaxed by introducing a vector of Lagrange multipliers (λ). Kuhn-Tucker (KT) optimality conditions for the arrival times, which dictate that the sum of the multipliers for a gate's rising (falling) inputs must equal the sum of multipliers at the falling (rising) inputs of the fan-out gates [8], reduces primal problem (PP) (4) to (5). The (positive) factor α is dynamically adjusted to facilitate convergence. The Lagrangian dual problem (LDP) is given by (6).

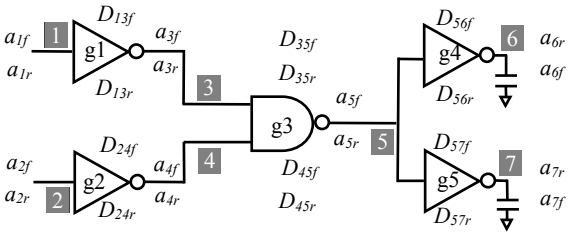

Figure 1. Circuit fragment in which D_{ijr} and D_{ijf} are the rise and fall stage delay from net i to j. a_{ir} and a_{if} are the rise and fall arrival times at net i.

$$\begin{aligned}&\text{Minimize } L(W,\lambda) = \Sigma\alpha\left(w_{n,i} + w_{p,i}\right)\\ &\underset{w}{} + \lambda_{56f}D_{56f} + \lambda_{56r}D_{56r} + \lambda_{57f}D_{57f} + \lambda_{57r}D_{57r}\\ &\quad + \lambda_{45f}D_{45f} + \lambda_{45r}D_{45r} + \lambda_{35f}D_{35f} + \lambda_{35r}D_{35r}\\ &\quad + \lambda_{24f}D_{24f} + \lambda_{24r}D_{24r} + \lambda_{13f}D_{13f} + \lambda_{13r}D_{13r}\\ &\text{Subject to}: L_n \le X_n \le U_n,\ L_p \le X_p \le U_p\end{aligned} \tag{5}$$

$$\begin{aligned}&\text{Maximize } L(W,\lambda) = \Sigma\alpha\left(w_{n,i} + w_{p,i}\right)\\ &\underset{\lambda}{} + \lambda_{56f}D_{56f} + \lambda_{56r}D_{56r} + \lambda_{57f}D_{57f} + \lambda_{57r}D_{57r}\\ &\quad + \lambda_{45f}D_{45f} + \lambda_{45r}D_{45r} + \lambda_{35f}D_{35f} + \lambda_{35r}D_{35r}\\ &\quad + \lambda_{24f}D_{24f} + \lambda_{24r}D_{24r} + \lambda_{13f}D_{13f} + \lambda_{13r}D_{13r}\\ &\text{Subject to}: \lambda \ge 0\end{aligned} \tag{6}$$

The gradient of the Lagrangian function (5) with respect to the transistor sizes generates the sizing equation for w_n shown in (7) (similarly for w_p, replacing f with r in the numerator).

$$w_n = \sqrt{\frac{C_{out} \sum_{j \in gate\ fanin} \sum_{k \in gate\ fanout} \lambda_{jkf}\, g_{jkf}}{\sum_{j \in g.\,fanin}\left(\alpha + \sum_{i \in dr.\,fanin}\left(\frac{c\lambda_{ijf}\, g_{ijf}}{w_{n,driver}} + \frac{c\lambda_{ijr}\, g_{ijr}}{w_{p,driver}}\right)\right)}} \tag{7}$$

Our continuous cell-sizing algorithm is in Figure 2. Step 5, using (7) with respect to W (λ's fixed) provides a feasible solution to the PP. Then, the LDP provides a lower bound to the PP. The maximization in step 6 increases the lower bound making it closer to the optimum solution. The algorithm proceeds till the PP value is arbitrarily close to the LDP value.

***Algorithm 1*: Continuous Cell Sizing**

1. pick initial transistor sizes and Lagrange multipliers
2. **while** (Primal > Dual) **do**
3. **until** no change in solution **do**
4. **for** each cell in reverse topological order **do**
5. solve for continuous w_n and w_p
6. update Lagrange multipliers

Figure 2. Continuous cell sizing algorithm

III. Discrete Cell Size Selection

Rounding the continuous sizes to the nearest discrete sizes available in the library destroys much of the benefit of continuous optimization. In many cases a driver cell is downsized and its fanouts are upsized, pushing the delay out. Our new approach utilizes a *branch-and-bound* methodology guided by the continuous solution. We examine each cell in turn, in topological order (from primary inputs to primary outputs). Each cell is assigned a candidate set of discrete sizes that are "close" in some sense to the continuous size, and which preserve the continuous beta ratio as much as possible (Figure 3).

A *partial solution* has some cells discretized, and all the others with a continuous size. For a selected cell (step 4), we add each of the k (typically 6) candidate discrete sizes to the partial solutions retained so far. This creates k times $|M|$ new candidate partial solutions. To minimize the number of retained partial solutions, we set upper bounds (bounding box) in both active area and delay based on the continuous solution. Candidate partial solution points are discarded if they fall outside these bounds. Active-area/delay values significantly away from the starting continuous point are better derived from a closer continuous point. Our active area boundary is +5% and the delay boundary is +½ fanout-of-four (FO4) inverter delay above the continuous solution point. Remaining partial solutions are then evaluated for delay and active area. Dominated partial solutions are discarded. Thus, for our new discrete cell

size selection algorithm, the decision on the discretization of a particular cell is delayed until it is clearer how it will globally impact active area and delay for the whole circuit.

***Algorithm 2*: Discrete Sizing**

1. **Select_Candidate_Discrete_sizes()** // say k sizes per cell//
2. Set active-area / delay bounding box
3. Let M be the set of partial solutions // initially just one //
4. **for** each cell in topologically sorted order **do**
5. Create $k*|M|$ partial solution points
   //combine the k candidate discrete sizes of the cell to each partial solution //
6. Prune partial solution points lying outside bounding box
7. Prune dominated partial solutions

Select_Candidate_Discrete_sizes()

1. **for** each continuous sized cell ($w_{n\text{-}cont}$ and $w_{p\text{-}cont}$) **do**
2. Choose library cells having w_p close to $w_{p\text{-}cont}$
3. From chosen cells at step 2:
4. Select 2 cells A, B such that $\beta_A < \beta_{cont} < \beta_B$ and such that β_A and β_B are as close as possible to β_{cont}
5. Select 2 cells C, D such that $w_{nC} < w_{n\text{-}cont} < w_{nD}$ and with w_{nC} and w_{nD} as close as possible to $w_{n\text{-}cont}$
6. Choose library cells having w_n close to $w_{n\text{-}cont}$
7. From chosen cells at step 6:
8. Select 2 cells E, F such that $\beta_E < \beta_{cont} < \beta_F$ and such that β_E and β_F are as close as possible to β_{cont}
9. Select 2 cells G, H such that $w_{pG} < w_{p\text{-}cont} < w_{pH}$ and with w_{pG} and w_{pH} as close as possible to $w_{p\text{-}cont}$
10. Remove duplicate library cells among A-H
11. Retain best k candidate cells according to eqn. (8)
12. If no candidate has $w_p > w_{p\text{-}cont}$ and $w_n > w_{n\text{-}cont}$, add such a candidate having minimum rounding cost (8)
    // ensures the possibility of no loss of drivability //

Figure 3. Discrete cell sizing algorithm

$$Cost = \frac{abs(w_{n.cont} - w_{n.disc})}{w_{n.cont.}} + \frac{abs(w_{p.cont} - w_{p.disc})}{w_{p.cont.}} \quad (8)$$

Each generated partial solution, with one additional discretized gate, requires a call to STA to evaluate its delay. This is the most CPU intensive part of the algorithm. We exploited parallelism to reduce the total number of required STA runs as well as the CPU time per each STA run. The number of STA runs can be reduced by noting that changing the size of a cell impacts: 1) the delay of the driver(s) of the cell due to the change in their load capacitance, and 2) delays downstream from the cell, possibly all the way to the primary outputs. The aggregate fan-out cones of all of the drivers of the cell are termed the *effective fan-out cone* of the cell. If a cell is not in the effective fan-out cone of another cell, discrete sizes for these two cells can be evaluated using a single STA run. We also reduced the run-time of each STA run by in parallel performing the timing analysis of all cells in a logic level (these can be computed independently), using multi-core processing.

IV. Results

The algorithms were implemented in C and tested on a 2.9 GHz Intel Xeon CPU running Linux. We used 40nm and 65nm industrial standard cell libraries where each cell type included 8-10 drive strengths and one or two beta ratio options. We accounted for multiple PVT corners, as guided by the supplier of the industrial circuits, using weak, 0.9V, 125°C for delay, and strong, 1.05V, 105°C for power (and leakage). In Figure 4, the active area was minimized for delay targets set between the minimum and maximum delay. For all designs, the differences between primal and dual function values were less than 1%.

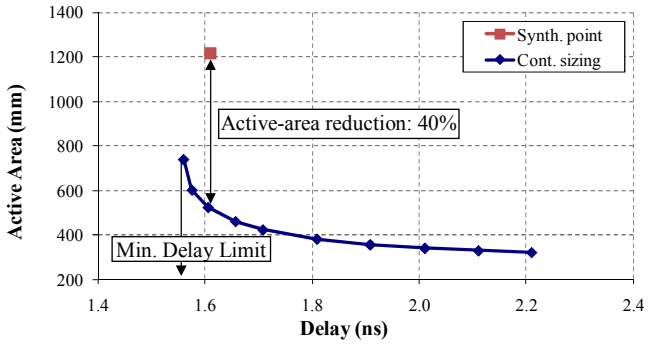

Figure 4. Active-area vs. delay for a 356k sizeable cell industrial design.

The continuous and discrete curves for a large industrial design are presented in Figure 5. Note that the two curves are fairly close. For comparison we also included the curve generated by the nearest rounding method, which used (8). As a figure of merit, we compute the average delay increment (push-out) in moving from a continuous solution to the corresponding discrete solution. Average delay increment results for 20 points per area-delay curve per benchmark, for both our new branch-and-bound discrete mapping algorithm and for the nearest-rounding method, are shown in Table I. For the first time, we have shown that it is possible to obtain discrete cell sizing results that are very close (within a few percent) to the optimal continuous sizing results. With respect to the nearest rounding technique, we achieved ~3X smaller delay increment.

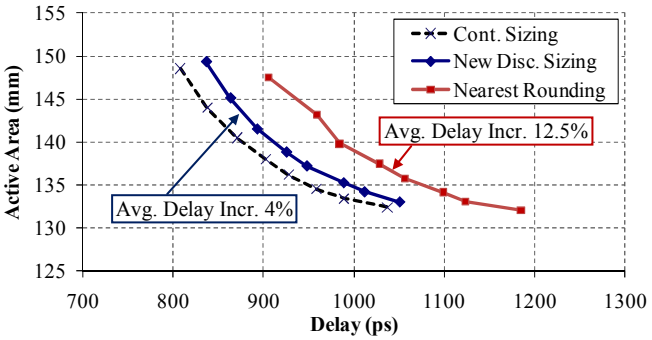

Figure 5. Continuous vs. discrete cell sizing for ~100k sizable cell design

TABLE I. Quality of Discrete Cell Sizing Using Ind. Lib.

Benchmark	Cell Count	Avg. Delay Increment (%)		Ratio
		Nearest Rounding	*Discrete Sizing*	
b20	13700	10.7	3.2	3.3
b22	20517	11.1	4.2	2.6
b17	21590	10.2	3.9	2.6
zm	98800	12.5	4.0	3.1
pl	116038	8.1	1.5	5.4
dw	356564	8.2	3.6	2.3

Near-optimal cell sizing can result in substantial reductions in both active area (corresponding directly to cell dynamic power, DP) as well as leakage power (LP). The latter is directly proportional to active area. Typical power reduction results (both DP and LP) for a large commercial design are shown in Figure 6 and Table II, where the result from a leading commercial synthesis tool (our starting point) is shown. Overall post-layout optimization results (delay, active area and leakage) are shown in Table III for three industrial circuits. "Baseline" was obtained by leading commercial tools as executed by a leading semiconductor company. Our post-layout results after discrete sizing are in column "Sized". The active area was reduced by an average of 40% and leakage power by an average of 40%. For the 356k sizeable cell industrial design, as illustrated in Figure 4, the active area (for the same wire load) was reduced by 40% compared to the leading commercial synthesis tool.

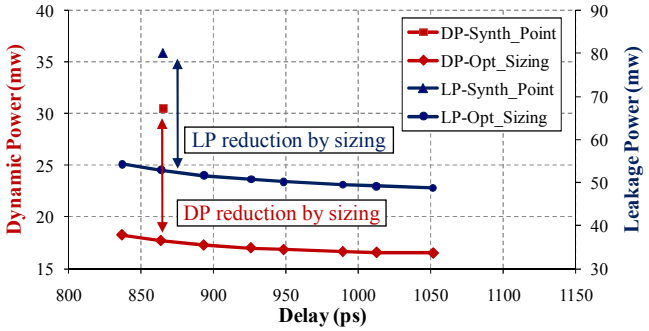

Figure 6. DP and LP for a ~100k sizeable cell commercial design.

TABLE II. POWER REDUCTION FOR 100K SIZEABLE-CELL DESIGN

Parameter	Synth. Point	Optimized Point	Reduction (%)
Delay (ps)	865	843	2.5
Active Area (µm)	274281	149355	45.5
Dynamic Power (mW)	30.5	17.9	41.3
Leakage Power (mW)	80.1	52.8	34.1
Total Power (mW)	110.6	70.7	36.1

TABLE III. ACTUAL AND RELATIVE POWER MINIMIZATION FOR A SET OF INDUSTRIAL DESIGNS

Bench-mark	Tech.	Cells Count	Delay (ps)	Active Area (um)		Leak. Power (mW)	
				Baseline	*Sized*	*Baseline*	*Sized*
B1	65nm 1.1V	103212	2450	308277	138662 (-55%)	54	30 (-45%)
B2	65nm 1.1V	85756	860	210049	130386 (-38%)	59	29 (-52%)
B3	40nm 1V	101113	1925	284110	183486 (-35%)	81	59 (-28%)

Table IV shows average run times for the continuous cell-sizing algorithm for a set of designs (run times are averaged for 20 delay target runs per benchmark). In Figure 7, run time as a function of design size (number of cells, n) is shown. Linear regression shows that the run-time complexity of the algorithm is $O(n^{1.1})$. Nearly optimal solutions (primal-dual tolerance less than 1%) are obtained with almost linear run time complexity.

Average run times for the branch-and-bound discrete cell-size selection algorithm for the same of benchmarks are in Table V. Column 2 has the serial single core results. Column 3 shows the reduced run times due to STA sharing. Columns 4 and 5 have the run times for four cores and eight cores (4/8 parallel processes) for the parallel STA scheme (including the STA sharing technique). Column 6 (ratio of column 2 to 5), has the speed-up of our run-time enhancement methods, which was as high as 13X for the largest design (356k sizeable cells).

TABLE IV. AVG. CPU TIME FOR CONTINUOUS SIZING

Bench-mark	Cell Count (n)	Time (s)
Des	2945	29
C6288	3210	43
b20	13700	136
b22	20517	211
b17	21590	297
zm	98800	1310
Pl	116038	2471
dw	356564	3884

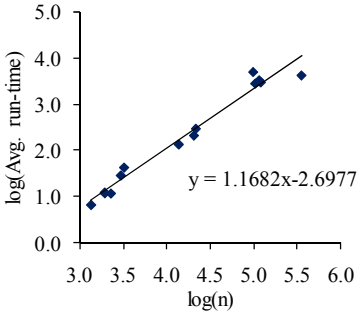

Figure 7. Run-time Complexity

TABLE V. AVERAGE CPU TIME FOR DISCRETE CELL SIZING

Bench-mark	Avg. CPU time per point (s)				Overall Speed up
	Single Core (SC)	*SC + Shared STA*	*Four Core + Shared STA*	*Eight Core + Shared STA*	
b20	1522	940	330	213	7.1x
b22	2570	1504	541	414	6.2x
b17	2505	1544	582	500	5.0x
zm	26387	9912	4457	3575	7.4x
pl	140997	70822	36123	24089	5.8x
dw	535874	138743	66320	40001	13.4x

V. CONCLUSION

We demonstrated an accurate table look-up model based optimal continuous cell-size selection algorithm along with a new feasible branch-and-bound method that maps the continuous sizes to the discrete sizes available in the standard cell library. We achieved a 40% reduction (on average) in active area and a 40% reduction (on average) in leakage power for a set of industrial designs via cell sizing. The size selection algorithms are efficient enough to handle contemporary ASIC partitions, which typically have upwards of 400k sizeable cells.

REFERENCES

[1] W. N. Li, "Strongly NP-hard Discrete Gate-sizing Problems", *IEEE Trans. on Comp.-Aid. Des.*, vol. 13, no. 8, pp. 1045-1051, August 1994.

[2] S. Lin, M. Marek-Sadowska, and E. S. Kuh, "Delay and Area Optimization in Standard-Cell Design," *Proc. DAC,* p. 349, June 1990.

[3] O. Coudert, "Gate Sizing for Constrained Delay/Power/Area Optimization," *IEEE Trans. VLSI*, vol. 5, no. 4, pp. 465-472, Dec. 1997.

[4] W. Chuang, S. S. Sapatnekar, and I. N. Hajj, "Timing and area optimization for standard-cell VLSI circuit design." *IEEE Trans. on Comp.-Aided Design*, vol. 14, no. 3, pp. 308-320, March 1995.

[5] S. Hu, M. Ketkar, and J. Hu, "Gate Sizing for Cell Library-Based Designs," *IEEE Trans. CAD,* vol. 28, no. 6, pp. 818-825, June 2009.

[6] E. Sutherland, R. F. Sproull, D. Harris, Logical Effort: Designing Fast CMOS Circuits, Morgan Kaufmann, 1999.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004, pp. 215-241.

[8] C. P. Chen, C. C. N. Chu, and D.F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation," *IEEE Trans. CAD,* vol. 18, no. 7, p. 1014, 1999.