IPSJ Journal Vol. 51 No. 12 2195–2205 (Dec. 2010)

Regular Paper

Performance-Constrained Transistor Sizing for Different Cell Count Minimization

Hiroaki Yoshida[†1] and Masahiro Fujita[†1]

A continuously-sized circuit resulting from transistor sizing consists of gates with a large variety of sizes. In the standard cell based design flow where every gate is implemented by a cell, a large number of different cells need to be prepared to implement an entire circuit. In this paper, we first provide a formal formulation of the performance-constrained different cell count minimization problem, and then propose an effective heuristic which iteratively minimizes the number of cells under performance constraints such as area, delay and power. Experimental results on the ISCAS 85 benchmark circuits implemented in a 90 nm fabrication technology demonstrate that different cell counts are reduced by 74.3% on average while accepting a 1% delay degradation. Compared to circuits using a typical discretely-sized cell library, we also demonstrate that the proposed method can generate better circuits using the same number of cells.

1. Introduction

Design optimization at the transistor-level has been successfully used to achieve significant performance benefits above and beyond gate-level design optimization. The approaches range from transformations such as sizing [1)–4)], all the way to macro-cell based design methodologies. More recently, transistor-level optimization techniques targeting standard cell based design flow have also been proposed [5),6)]. These optimization techniques take advantage of the recent progress in automated cell-layout solutions. In particular, continuous transistor sizing is known to have a significant impact on circuit performance and hence has been extensively studied. Although early work does not guarantee the optimality [1)], Sapatnekar, et al. first provided an exact sizing method based on an interior-point algorithm [2)]. More recently, Chen, et al. showed an elegant formulation of the sizing problem [3)] which can be optimally and efficiently solved by Lagrangian relaxation method. The IBM EinsTuner [4)] circuit tool which is a state-of-the-art transistor sizer software has been successfully applied to block designs of IBM zSeries and IBM/Sony/Toshiba Cell processors [7)].

A continuously-sized circuit resulting from transistor sizing consists of gates with a large variety of sizes. **Figure 1** shows a cell size distribution of 2-input NOR gates after delay-optimal sizing in an ISCAS 85 benchmark circuit C499 implemented in a 90 nm fabrication technology. In the figure, a circle indicates the number of instances of the cell is 1 and a triangle indicates between 2 and 10. The cells are parameterized with two parameters. One is the P-type transistor width. The other is the beta ratio which is the ratio of N-type transistor width to P-type transistor width. In the standard cell based design flow where every gate is implemented by a cell, a large number of cells need to be prepared to implement an entire circuit. As technology advances, the number of effects which need to be taken into account, e.g., performance variability and manufacturability, is increasing. Reflecting this situation, the design and characterization of cells are also becoming more complex [8)]. Also, different cell counts can directly impact the production throughput in the character projection based electron beam direct writing (CP-EBDW) method [9)] in which each gate is masklessly projected onto a wafer at a time. Thus, minimizing the different cell counts is becoming increasingly important.

This paper addresses a performance-constrained different cell count minimization problem. Unlike the gate selection problem [10),11)] whose objective is to build a general-purpose cell library, the proposed method minimizes the number of cells of a circuit under performance constraints such as area, delay and power. Our primary objective is to apply the proposed method to high-performance block design where the state-of-the-art transistor sizers such as EinsTuner are used. We demonstrate that the proposed method can yield benefit from continuous sizing with as few cells as a typical standard cell library. Thus, the proposed method can also be applied to the CP-EBDW method to improve the circuit area and/or performance without sacrificing production throughput. The rest of the paper is organized as follows. Section 2 describes a posynomial cell model which we use to model cell characteristics, and provides a quick overview of a geometric

[†1] VLSI Design and Education Center, the University of Tokyo

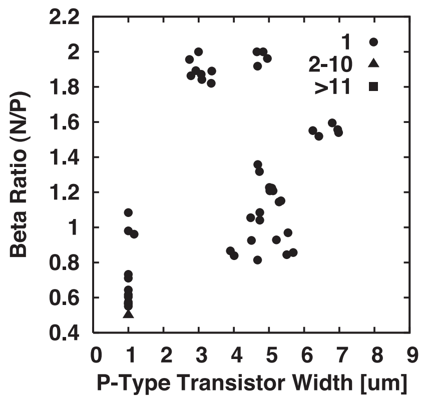


Fig. 1 Cell size distribution of 2-input NOR gates after delay-optimal sizing in an ISCAS 85 benchmark circuit C499 implemented in a 90 nm fabrication technology. A circle indicates the number of instances of the cell is 1 and a triangle indicates between 2 and 10.

programming based transistor sizing algorithm[3]. In Section 3, we first formulate the performance-constrained different cell count minimization problem formally, and then propose an effective heuristic for the problem. Section 4 presents the experimental results on a benchmark suite to demonstrate the effectiveness of the proposed method. We also provide a discussion on the runtime complexity of the proposed method.

2. Preliminaries

2.1 Posynomial Cell Model

Our cell model is the posynomial cell model[1] which is the model most-commonly used in convex optimization based transistor sizing. Each cell is a *parameterized cell* where the sizes of the transistors in a cell are specified by a set of parameters $(p_1, \ldots, p_m)$, e.g., beta ratio and drive strength. Each parameter p_i has its lower bound p_i^L and upper bound p_i^U:

$$p_i^L \le p_i \le p_i^U. \qquad (1)$$

Figure 2 illustrates our continuously-sized cell model. The model consists of 2 parameters: P-type transistor width w and beta ratio β which is the ratio of N-type transistor width to P-type transistor width. A cell is characterized with respect to the following characteristics: *timing*, *power*, *area* and *input capacitances*. A *timing* of a cell can be defined as a delay d or slew s of an input-to-output arc of the cell for a given input slew s_i and an output load C_L:

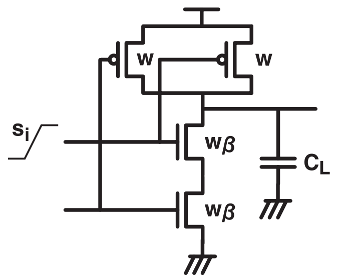


Fig. 2 Our continuously-sized cell model where s_i is the input slew and C_L is the output load capacitance. A cell has 2 parameters: P-type transistor width w and beta ratio β which is the ratio of N-type transistor width to P-type transistor width.

$$d = f_d(p_1, \ldots, p_m, s_i, C_L) \qquad (2)$$
$$s = f_s(p_1, \ldots, p_m, s_i, C_L). \qquad (3)$$

Likewise, a cell power is typically modeled in the same way. Next, an area A and an input capacitance C_i of a cell are given as functions of the parameters:

$$A = f_A(p_1, \ldots, p_m) \qquad (4)$$
$$C_i = f_{C_i}(p_1, \ldots, p_m) \qquad (5)$$

where C_i is the capacitance of i-th input.

A *posynomial*[12] is a function g of a positive vector variable $t = (t_1, \ldots, t_m) \in R^m$ having the form:

$$g(t) = \sum_{i=1}^{N} u_i(t) \qquad (6)$$
$$u_i(t) = b_i t_1^{a_{i1}} t_2^{a_{i2}} \cdots t_m^{a_{im}}, \quad i = 1, 2, \ldots, N \qquad (7)$$

where the exponents a_{ij} are arbitrary real numbers and the coefficients b_i are positive. An important property of a posynomial is that a posynomial is convex under a variable transformation[12]:

$$t_j = e^{z_j}, \quad j = 1, 2, \ldots, m. \qquad (8)$$

For a convex function, any local minimum is also a global minimum. Therefore, existing nonlinear programming techniques such as the Lagrangian relaxation

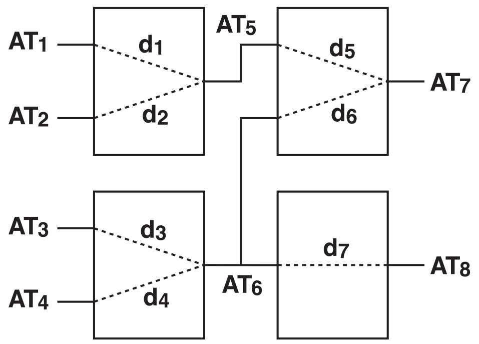


Fig. 3 An example circuit model for continuous transistor sizing.

method[13], can solve the minimization problem of a posynomial while guaranteeing optimality. The characteristics of a cell given by Eqs. (2)–(5) are modeled by posynomials. A posynomial for a cell characteristic can be obtained by fitting a number of data points which are obtained by a circuit simulation. There are several fitting techniques proposed for posynomials[14),15)]. In addition, more accurate cell models based on posynomials have been proposed[16),17)]. In Section 4, we will demonstrate the accuracy of the posynomial cell model in a 90 nm fabrication technology.

2.2 Optimal Continuous Transistor Sizing

This section overviews an optimal continuous transistor sizing algorithm[3] which is the basis for our proposed method. **Figure 3** shows an example of a circuit model used for continuous transistor sizing. *For ease of explanation*, the following formulation does not take slews and load capacitances into account, nor does it distinguish rise and fall delays.

A *gate* $g_i = (c_{g_i}, p_{i1}, \ldots, p_{im})$ is an instance of a cell $c_{g_i} \in \{c_1, c_2, \ldots\}$ with an associated set of parameters $(p_{i1}, \ldots, p_{im})$. Note that a cell c_i represents its functional characteristics, i.e., transistor-level topologies and logic functions, which are independent of the cell parameters. A circuit consists of a set of gates $G = \{g_1, \ldots, g_n\}$ and a set of wires $W = \{w_1, \ldots, w_o\}$. Each wire w_i has its associated arrival time AT_i and each input-to-output arc in a gate has its associated delay d. An area minimization problem under delay constraints can then be formulated as follows:

$$\begin{array}{l} \textbf{minimize } A(p) = \sum_{i=1}^{n} A_i(p) \\ \textbf{subject to} \\ \quad AT_{\text{worst}} \le AT_{\max} \\ \quad \left.\begin{array}{l} p_j^L \le p_{ij} \le p_j^U \ (i = 1, \ldots, n, j = 1, \ldots, m) \\ AT_i \le AT_{\text{worst}} \ (i = 1, \ldots, o) \\ AT_1 + d_1(p) \le AT_5,\ AT_2 + d_2(p) \le AT_5 \\ AT_3 + d_3(p) \le AT_6,\ AT_4 + d_4(p) \le AT_6 \\ AT_5 + d_5(p) \le AT_7,\ AT_6 + d_6(p) \le AT_7 \\ AT_6 + d_7(p) \le AT_8 \end{array}\right\} \textit{Common constraints} \end{array} \qquad (9)$$

where $p = (p_{i1}, p_{i2}, \ldots, p_{nm})$ is the set of all parameters, $A_i(p)$ is the area of g_i and $AT_{\max}$ is the maximum arrival time at any output. Since the constraints for the cell parameters and the arrival times at internal wires are common among the following minimization problems, they are omitted in the remainder of this paper for ease of explanation. Similarly, delay minimization problem under an area constraint can be formulated as follows:

$$\begin{array}{l} \textbf{minimize } AT_{\text{worst}} \\ \textbf{subject to } A(p) \le A_{\max} \end{array} \qquad (10)$$

where $A_{\max}$ is the maximum area. Since the convexity is preserved under sums and maxima, a local optimum of these problems is the global optimum. Therefore, any nonlinear solver which finds a local minimum can find the global optimum solution. Chen, et al. showed that these constrained problems are efficiently and optimally solved by Lagrangian relaxation method[3].

Note that the above formulation does not take interconnect loads and delays into account which have a real and considerable impact on circuit delay. In general, interconnect lengths depend upon the circuit size. Moreover, estimating them at the prelayout level is not a trivial task. State-of-the-art transistor sizers, such as EinsTuner[4], can incorporate the effect of interconnects by performing a simulation instead of using an analytical model. By applying the same technique to the proposed method, interconnect effects can be taken into account.

3. Different Cell Count Minimization

3.1 Problem Formulation

Informally speaking, the objective of the problem addressed in this paper is to minimize the number of cells required to implement a circuit under performance constraints such as area, delay and power. Note that only the cell parameters are subject to this optimization problem, i.e., neither the topology of a circuit nor any cell logic type is changed.

Before formulating the problem formally, we start with the following series of definitions. Two gates $g_i = (c_{g_i}, p_{i1}, \ldots, p_{im})$ and $g_j = (c_{g_j}, p_{j1}, \ldots, p_{jm})$ are said to be *equivalent* if and only if $c_i = c_j$ and $p_{ik} = p_{jk}$ for all k, and are denoted by $g_i \sim g_j$. For example, consider a circuit consisting of the following gates: $g_1 = (c_1, 1, 1)$, $g_2 = (c_1, 1, 2)$, $g_3 = (c_2, 2, 3)$ and $g_4 = (c_2, 2, 3)$. Here, g_1 and g_2 are not equivalent because the second parameters are different. Also, g_2 and g_3 are not equivalent because the cells are different. Since all parameters are same, g_3 and g_4 are equivalent. A *gate group* Γ is defined as an equivalence class on the set of gates G, i.e., $g_i, g_j \in \Gamma \Longleftrightarrow g_i \sim g_j$. A *different cell count* $N(p)$ is defined as $\left|G/{\sim}\right|$, the size of the quotient set of G (the number of all equivalence classes on G). Two gate groups $\Gamma_i = (c_{\Gamma_i}, p_{i1}, \ldots, p_{im})$ and $\Gamma_j = (c_{\Gamma_j}, p_{j1}, \ldots, p_{jm})$ are said to be *compatible* if and only if $c_{\Gamma_i} = c_{\Gamma_j}$. Here, $N(p)$ can also be viewed as the number of cells which are required to implement the circuit. In the previous example, there are three gate groups: $\Gamma_1 = \{g_1\}$, $\Gamma_2 = \{g_2\}$ and $\Gamma_3 = \{g_3, g_4\}$. Therefore, $N(p) = 3$ and hence three cells are required to implement the circuit. Also, Γ_1 and Γ_2 are compatible, and Γ_1 and Γ_3 are not. Using these definitions, the problem addressed in this paper is formulated as follows:

$$\begin{aligned} &\textbf{minimize } N(p) \\ &\textbf{subject to } A(p) \leq A_{\max},\ AT_{\text{worst}} \leq AT_{\max}. \end{aligned} \tag{11}$$

Other performance constraints such as maximum power can be incorporated in a straightforward manner. Obviously, $N(p)$ is a non-smooth and non-convex function. Since conventional nonlinear programming techniques do not solve this problem properly, we propose an effective heuristic to solve this problem.

3.2 Iterative Heuristic

In this section, we present a procedure which solves the problem in Eq. (11). Starting from an optimally-sized circuit which satisfies the constraints, it iteratively reduces $N(p)$ by one at a time while satisfying the constraints, and this is repeated until no further change can be made. First, we explain the notions of slack and distance. The *slack of a wire* is defined as the difference between the required time and the arrival time at the wire. The *slack of a gate* is the worst (smallest) slack of the wires connected to the gate. The *slack of a gate group* is the worst slack of the gates in the gate group. The slack of a gate group is used as an estimate of its freedom. A gate group without slack cannot move since changing its parameters may violate the performance constraints. The *distance* between two compatible gate groups $\Gamma_i = (c_{\Gamma_i}, p_{i1}, \ldots, p_{im})$ and $\Gamma_j = (c_{\Gamma_j}, p_{j1}, \ldots, p_{jm})$ is the Euclidean distance between two vectors of parameters:

$$D(\Gamma_i, \Gamma_j) = \sqrt{\sum_{k=1}^{m} \bigl(K_i(p_{ik} - p_{jk})\bigr)^2} \tag{12}$$

where K_i is the weight factor for i-th parameter. The weight factor K_i is intended to equalize the impact of the i-th parameter p_i to the transistor sizes. For instance, when p_1 is a transistor width in μm and p_2 is a beta ratio, K_1 and K_2 can be set to 10^6 and 1. In our experience, weight factors tend to have little impact on results as far as the weight factors are set within the proper range. The distance between two gate groups can be viewed as an estimate of the impact on the circuit area and performance when the gate groups are merged.

The basic idea of reducing $N(p)$ by one is: (a) maximizing the slacks of the gate groups, (b) finding two close compatible gate groups Γ_i and Γ_j, and (c) merging them into a gate group Γ_k, as shown in **Fig. 4**. If an initial circuit is generated by minimizing the area under performance constraints, the slack of each gate is also minimized under the constraints. In such a case, changing any parameters of the gates may violate the performance constraints. Step (a) maximizes the slack of each gate groups to increase the chance of merging gate groups. Then, Step (b) selects two compatible gate groups close to each other so that merging the gate groups has a minimum impact on the performance. After merging two gate groups, the parameters of the new merged gate group are determined by

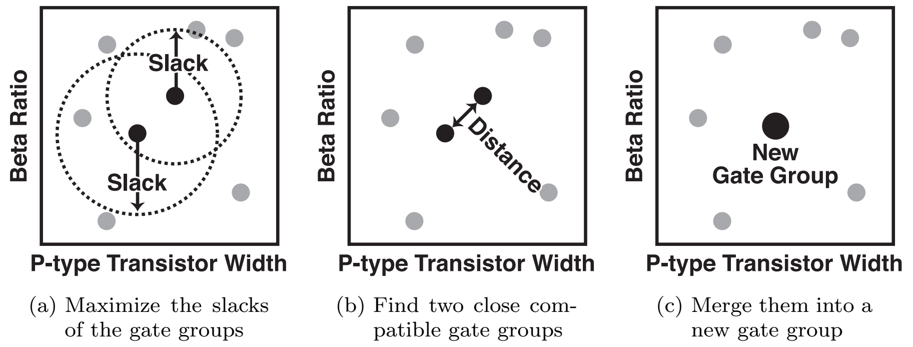


Fig. 4 A conceptual illustration of the iterative heuristic. Three steps (a)(b)(c) are iteratively performed until no further changes can be made.

```
procedure MinimizeCellCount
 1: Maximize total slack under performance constraints
 2: repeat
 3:   for all gate group $\Gamma_i$ in descending order of slack do
 4:     $H \leftarrow$ k-nearest neighbor compatible gate groups of $\Gamma_i$
 5:     for all $\Gamma_j \in H$ in ascending order of the distance do
 6:       Merge two gate groups $\Gamma_i$ and $\Gamma_j$
 7:       Maximize total slack under performance constraints
 8:       if constraints are satisfied then
 9:         break
10:       else
11:         Undo Step 6 and 7
12:       end if
13:     end for
14:   end for
15: until no further change can be made
```

Fig. 5 Pseudocode for the iterative heuristic MinimizeCellCount.

performing Step (a) again.

A pseudocode for the iterative heuristic is presented in **Fig. 5**. As mentioned above, the proposed method first performs total slack maximization under the given performance constraints (Step 1) formulated as follows:

$$\begin{aligned}&\textbf{maximize } S(p)\\&\textbf{subject to}\\&\quad A(p) \le A_{\max}, AT_{\text{worst}} \le AT_{\max}\\&\quad p_{ik} = p_{jk} \ \ (k = 1, \ldots, m,\ g_i \in \Gamma_l,\ g_j \in \Gamma_l,\ \Gamma_l \in G/\sim)\end{aligned} \tag{13}$$

where $S(p)$ is the sum of the slacks of the wires. The computation of $S(p)$ requires the required times at all wires. Since the computation of required times includes subtraction and minimum operations, nonlinear solvers may not guarantee its optimality. Instead, we use the sum of slacks at all endpoints (i.e., primary outputs) as an estimate of total slack. Since required times at endpoints are all fixed, this problem is simply equivalent to the minimization of the sum of arrival times at all endpoints. After total slack maximization, the slack of each internal wire can be computed independently. The last constraint in Eq. (13) forces the gates in each gate group to have the same set of parameters. Thus, the different cell count remains the same during the total slack maximization. After the total slack maximization, all gate groups are sorted in descending order of slack and the gate group Γ_i with the largest slack is chosen (Step 3). Next, the k-nearest neighbor (i.e., k closest neighbors) compatible gate groups of Γ_i are computed where k is a user-defined parameter which controls the runtime and quality (Step 4), and the nearest gate group Γ_j is chosen (Step 5). After merging the two gate groups into a gate group, the optimal parameter set is determined by maximizing the total slack under the performance constraints (Steps 6 & 7). If the constraints cannot be satisfied after this slack maximization, the gate group pair is discarded and the next nearest gate group of Γ_i is chosen as Γ_j (Step 11). If there is no more k-nearest neighbor, the gate group with the next largest slack is chosen as Γ_i (Step 3). The process is repeated until no more groups can be merged.

4. Experimental Results

4.1 Experimental Setup

First, as a reference, we prepared a discretely-sized cell library consisting of 24 typical logic types in a 90 nm fabrication technology shown in **Table 1**. Each logic type has several drive strengths and the total number of the cells is 77.

Table 1 Statistics of a typical discretely-sized cell library in a 90 nm fabrication technology. The number of logic types is 24 and the total number of cells is 77.

Logic Type	*Function*	*Drive Strengths*
INV	$\overline{A+B}$	1x, 2x, 4x, 8x, 16x
NAND2	$\overline{A \cdot B}$	1x, 2x, 4x, 8x
NAND3	$\overline{A \cdot B \cdot C}$	1x, 2x, 4x, 8x
NAND4	$\overline{A \cdot B \cdot C \cdot D}$	1x, 2x, 4x
NOR2	$\overline{A+B}$	1x, 2x, 4x, 8x
NOR3	$\overline{A+B+C}$	1x, 2x, 4x
NOR4	$\overline{A+B+C+D}$	1x, 2x, 4x
AOI21	$\overline{A \cdot B+C}$	1x, 2x, 4x
AOI22	$\overline{A \cdot B+C \cdot D}$	1x, 2x, 4x
AOI211	$\overline{A \cdot B+C+D}$	1x, 2x, 4x
AOI221	$\overline{A \cdot B+C \cdot D+E}$	1x, 2x, 4x
AOI31	$\overline{A \cdot B \cdot C+D}$	1x, 2x, 4x
AOI311	$\overline{A \cdot B \cdot C+D+E}$	1x, 2x, 4x
AOI32	$\overline{A \cdot B \cdot C+D \cdot E}$	1x, 2x, 4x
AOI41	$\overline{A \cdot B \cdot C \cdot D+E}$	1x, 2x, 4x
OAI21	$\overline{(A+B) \cdot C}$	1x, 2x, 4x
OAI22	$\overline{(A+B) \cdot (C+D)}$	1x, 2x, 4x
OAI211	$\overline{(A+B) \cdot C \cdot D}$	1x, 2x, 4x
OAI2111	$\overline{(A+B) \cdot C \cdot D \cdot E}$	1x, 2x, 4x
OAI221	$\overline{(A+B) \cdot (C+D) \cdot E}$	1x, 2x, 4x
OAI31	$\overline{(A+B+C) \cdot D}$	1x, 2x, 4x
OAI32	$\overline{(A+B+C) \cdot (D+E)}$	1x, 2x, 4x
OAI311	$\overline{(A+B+C) \cdot D \cdot E}$	1x, 2x, 4x
OAI41	$\overline{(A+B+C+D) \cdot E}$	1x, 2x, 4x

Also, we constructed a continuously-sized cell library consisting of the same set of logic types. The cells were characterized for the posynomial cell model described in Section 2.1. For each logic type, P-type transistor widths were varied from $1\,\mu$m to $8\,\mu$m and beta ratios (the ratio of N-type transistor width to P-type transistor width) were varied from 0.5 to 2. Input slews were varied from 10 ps to 1,000 ps, output loads were varied from 1 fF to 100 fF. Cell delays and slews were simulated using a prelayout cell characteristic estimator [18] with HSPICE [19] for 256 combinations of the parameters. We then fitted the data to a posynomial function and obtained the coefficients and exponents. **Table 2** presents the average fitting error and the standard deviation of cell delays and slews of each logic type. Overall, the average fitting error was about 1.06% and the standard deviation was 1.19%. For cell areas, the average fitting error and the standard deviation were both less than 0.01%. For input loads, the average fitting error and the standard deviation were 0.23% and 0.19%, respectively.

Table 2 Fitting errors of the posynomial gate delay models in a 90 nm fabrication technology.

Logic Type	Average Error [%]	Standard Deviation [%]
INV	0.92	0.85
NAND2	1.55	2.12
NAND3	1.02	1.19
NAND4	0.75	0.82
NOR2	2.33	1.70
NOR3	1.55	1.88
NOR4	1.25	1.30
AOI21	1.56	2.25
AOI22	0.99	1.08
AOI211	1.17	1.33
AOI221	0.91	0.97
AOI31	1.11	1.53
AOI311	0.91	1.05
AOI32	0.79	0.85
AOI41	0.86	1.16
OAI21	1.23	1.41
OAI22	0.92	0.99
OAI211	0.82	0.90
OAI2111	0.62	0.65
OAI221	0.66	0.70
OAI31	1.02	1.13
OAI32	0.77	0.80
OAI311	0.65	0.71
OAI41	1.13	1.27
Overall	1.06	1.19

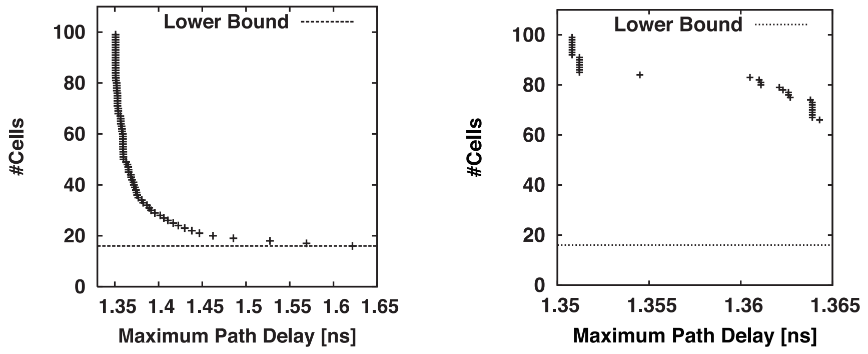
Next, we implemented the optimal continuous transistor sizing algorithm explained in Section 2.2 and the performance-constrained cell minimization algorithm proposed in Section 3.2. To solve the nonlinear problems, a state-of-the-art nonlinear optimizer IPOPT [20] is used. The weight factor K_1 for P-type transistor widths was set to 10^6, K_2 for beta ratios was set to 1 considering the range of transistor widths. As circuit examples, we prepared 10 circuits from the ISCAS 85 benchmark circuits as follows. The benchmark circuits were first synthesized for optimal delay using the reference discretely-sized cell library. After replacing the cells with the continuously-sized cells, delay-optimal circuits were obtained by performing an unconstrained optimal-delay sizing followed by an optimal-area sizing under the optimal delay constraint.

4.2 Cell Count Minimization Results

Then, we applied the proposed different cell count minimization method to

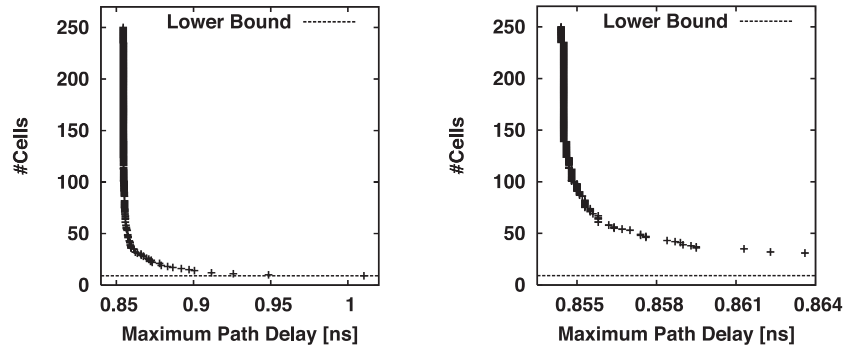
Table 3 Different cell count minimization results in a 90 nm fabrication technology.

Circuit	*#Used Logic Types*	*Delay Optimal*			*Accepting 1% Degradation of Optimal Delay*				*Diff. Cell Count Reduction*
		Area [μm^2]	*Delay* [ns]	*Diff.* Cell Count	*Area* [μm^2]	*Delay* [ns]	*Diff.* Cell Count	*CPU time* [sec.]	[%]
C432	16	2955.4	1.3508	99	2955.4	1.3643	66	31.9	33.3
C499	9	7374.8	0.8545	250	7173.7	0.8632	32	71.6	87.2
C880	20	2884.2	1.1643	213	2884.2	1.1759	79	35.5	62.9
C1355	9	7343.3	0.8557	250	7343.3	0.8643	45	73.0	82.0
C1908	18	6145.6	1.2704	274	5813.0	1.2833	116	103.9	57.7
C2670	21	3712.8	1.0555	554	3708.9	1.0660	58	401.3	89.5
C3540	24	9512.5	1.7305	628	9512.5	1.7479	148	1203.2	76.4
C5315	20	8427.0	1.2830	941	8427.0	1.2959	153	1207.4	83.7
C6288	17	44636.1	4.8085	1587	44636.1	4.8567	265	9101.0	83.3
C7552	23	14268.6	1.4533	1341	14268.6	1.4678	172	12758.6	87.2
Average									74.3



(a) Full range (b) Range within 1% of optimal delay

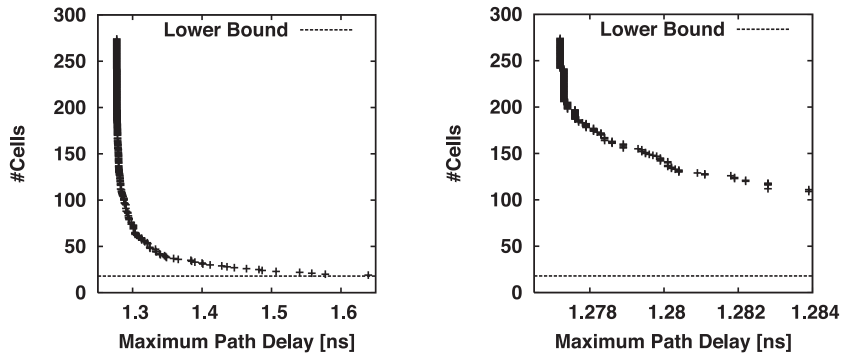
Fig. 6 Delay vs. different cell count tradeoff curve on C432.



(a) Full range (b) Range within 1% of optimal delay

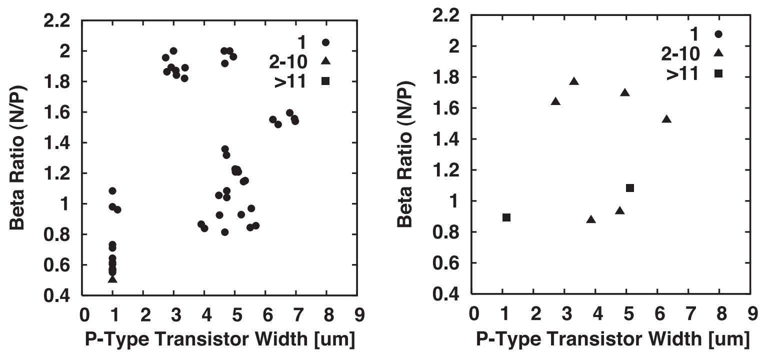
Fig. 7 Delay vs. different cell count tradeoff curve on C499.

the delay-optimal circuits as follows. The different cell count of each circuit is minimized while accepting 1% degradation of optimal delay and keeping the area, i.e., under the constraints of the maximum path delay of $(D_{opt} * 1.01)$ and the maximum area of A_{opt} where D_{opt} and A_{opt} are the maximum path delay and the area of a delay-optimal circuit, respectively. **Table 3** compares the different cell counts of the delay-optimal circuits and the circuits after the different cell count minimization. In the table, the second column shows the number of logic types used in the circuit. Note that the number of logic types is the lower bound on the different cell count. The last column shows the different cell count reduction rate calculated by $(N_{opt} - N_{1\%})/N_{opt} * 100$ where N_{opt} and $N_{1\%}$ are the different cell counts of the delay-optimal circuit and the circuit after the different cell count minimization, respectively. The results demonstrate that the different cell counts could be reduced by 74.3% on average while accepting 1% degradation. **Figures 6**, **7** and **8** present tradeoff curves between the maximum path delay and



(a) Full range (b) Range within 1% of optimal delay

Fig. 8 Delay vs. different cell count tradeoff curve on C1908.



(a) After delay-optimal sizing (*equivalent to Figure 1*) (b) After different cell count minimization while accepting 1% degradation

Fig. 9 Cell size distributions of 2-input NOR gates in C499.

the different cell count on C432, C499 and C1908, respectively. The curves were obtained by increasing the maximum path delay constraint from D_{opt} and keeping the area constraint the same. In the figures, (a) presents the curve in the full range from the optimal delay to the minimum different cell count, and (b) presents the same curve in a range within 1% of the optimal delay. An important observation from these results is that different cell counts can be reduced dramatically while accepting very little delay degradation. Another important observation is that the cell count reduction rate varies considerably, from 33.3% to 89.5%. This variation can also be observed by the fact that the curve on C499 decreases more rapidly than the curve on C432. The cell count reduction on C432 terminated after reducing 33 cells because it reached the maximum delay and the maximum area. No two cells can be merged because merging cells increases either the area or delay. By relaxing the maximum delay, the cells along the critical path can become smaller so that more cells can be merged further. **Figure 9** (a) and (b) show the cell size distributions of 2-input NOR gates in a circuit C499 after delay-optimal sizing and after cell count minimization while accepting 1% degradation of optimal delay, respectively. In the figures, a circle indicates the number of instances of the cell is 1, a triangle indicates between 2 and 10, and a square indicates more than 10.

Next, we compared the circuits using the discretely-sized library and continuously-sized libraries as follows. To make a fair comparison, the number of cells in continuously-sized circuits was reduced to 77 which is equivalent to the number of the cells in the discretely-sized library. The comparisons are given in **Table 4**. First, the discretely-sized circuits were synthesized for optimal delay using the discretely-sized library. In the table, the second and third columns show the area A_{ds} and the delay D_{ds} of the discretely-sized circuits. The delay-constrained area-optimal continuous sizing was then performed and the cell count was reduced to 77 under the maximum delay constraint of D_{ds}. The sixth column shows the area improvement against the discretely-sized circuit and the average area improvement was 28.9%. Likewise, area-constrained delay-optimal continuous sizing was performed and the cell count was reduced to 77 under the maximum area constraint of A_{ds}. The ninth column shows the delay improvement against the discretely-sized circuit and the average delay improvement was 8.0%. The results clearly demonstrate the effectiveness of our approach.

4.3 Runtime Complexity

Figure 10 which plots the runtime with respect to the circuit size from Table 3 shows an exponential runtime complexity. This fact is not surprising because geometric programming problems such as the transistor sizing/optimization problems in this paper are generally known as *NP-hard* problems. In the proposed

Table 4 Comparisons between circuits using the discretely-sized library and circuits using continuously-sized libraries where the number of the cells in every library is limited to 77.

Circuit	*Discretely-sized Library*		*Continuously-sized Library (Maximum delay = D_{ds})*			*Continuously-sized Library (Maximum area = A_{ds})*		
	Area A_{ds} [μm^2]	*Delay* D_{ds} [ns]	*Area* [μm^2]	*Delay* [ns]	*Area Improv.* [%]	*Area* [μm^2]	*Delay* [ns]	*Delay Improv.* [%]
C432	1804.2	1.5183	1163.4	1.5183	35.5	1804.2	1.4175	6.6
C499	4121.8	1.0062	2436.2	1.0062	40.9	4121.8	0.9092	9.6
C880	2471.0	1.3705	1805.5	1.3705	26.9	2471.0	1.2339	10.0
C1355	3790.1	1.0137	2349.6	1.0135	38.0	3790.1	0.9218	9.1
C1908	3725.4	1.4467	2925.8	1.4467	21.5	3725.4	1.3913	3.8
C2670	4395.2	1.2076	3541.9	1.2077	19.5	4395.2	1.1252	6.8
C3540	7358.2	2.0254	6315.6	2.0254	14.2	7358.2	1.9664	2.9
C5315	8720.1	1.4418	6986.5	1.4418	19.9	8720.1	1.3070	9.4
C6288	24601.4	5.4583	13300.7	5.4583	45.9	24601.4	4.8679	10.8
C7552	12857.8	1.5732	9369.2	1.5732	27.1	12857.8	1.4035	10.8
Average					28.9			8.0

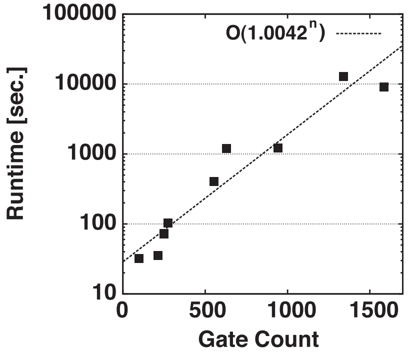


Fig. 10 Gate count vs. runtime plot from Table 3.

procedure, the total slack maximization step (Step 7) is enclosed by three levels of loops and the upper bound for the number of iterations is $O(n^2)$ where n is the number of gates. In our experiments, we observed that the number of iterations was typically $O(n)$ because the undoing of gate group merge (Step 11) does not frequently take place. Therefore, the total slack maximization step dominantly determines the overall runtime. The runtime of the total slack maximization increases particularly when the constraints are either very tight or non-satisfiable. At the sacrifice of the optimization quality, the runtime complexity can be mitigated to some extent by the following techniques:

- forcing the maximum iteration limit (i.e., time-out) during a geometric programming
- reducing the number of total slack maximization by merging many gate groups at a time instead of merging two groups at a time.

As mentioned earlier, the primary objective of this paper is to provide an effective solution to the addressed problem formulated as a non-smooth and non-convex nonlinear programming problem. Our future work includes the application of these techniques for improving the runtime on large-scale circuits.

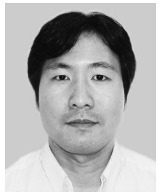
5. Conclusions

This paper addressed a performance-constrained different cell count minimization problem for continuously-sized circuits. After providing a formal formulation of the problem, we proposed an effective heuristic for the problem. The proposed heuristic iteratively minimizes the number of cells under performance constraints such as area, delay and power. The experimental results on the ISCAS 85 benchmark circuits implemented in a 90 nm fabrication technology demonstrated that different cell counts were reduced by 74.3% on average while accepting 1% delay degradation. Compared to circuits using a typical discretely-sized cell library, we also demonstrated that the proposed method could generate better circuits using the same number of cells. We also provided a discussion on the runtime complexity of the proposed method.

References

1) Fishburn, J.P. and Dunlop, A.E.: TILOS: A posynomial programming approach to transistor sizing, *Proc. IEEE Int. Conf. Computer-Aided Design*, pp.326–328 (1985).
2) Sapatnekar, S.S., et al.: An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization, *IEEE Trans. Computer-Aided Design*, Vol.12, No.11, pp.1621–1634 (1993).
3) Chen, C.P., et al.: Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation, *IEEE Trans. Computer-Aided Design*, Vol.18, No.7, pp.1014–1025 (1999).
4) Conn, A.R., Elfadel, I.M., Molzen, W.W. Jr., O'Brien, P.R., Strenski, P.N., Visweswariah, C. and Whan, C.B.: Gradient-Based Optimization of Custom Circuits Using a Static-Timing Formulation, *Proc. ACM/IEEE Design Automation Conf.*, pp.452–459 (1999).
5) Panda, R., Dharchoudhury, A., Edwards, T., Norton, J. and Blaauw, D.: Migration: A New Technique to Improve Synthesized Designs through Incremental Customization, *Proc. ACM/IEEE Design Automation Conf.*, pp.388–391 (1998).
6) Bhattacharya, D. and Boppana, V.: Design Optimization with Automated Flex-Cell Creation, *Closing the Gap Between ASIC & Custom*, pp.14–23 (2002).
7) Wolpin, S.: Chip Checker: EinsTuner automates the transistor-design process, *IBM Think Research* (2002). http://www.research.ibm.com/thinkresearch
8) Bittlestone, C., Hill, A.M., Singhal, V. and Arvind, N.: Architecting ASIC Libraries and Flows in Nanometer Era, *Proc. ACM/IEEE Design Automation Conf.*, pp.776–781 (2003).
9) Inanami, R., Magoshi, S., Kousai, S., Hamada, M., Takayanagi, T., Sugihara, K., Okumura, K. and Kuroda, T.: Throughput enhancement strategy of maskless electron beam direct writing for logic device, *Int. Electron Devices Meeting Technical Digest*, pp.833–836 (2000).
10) Beeftink, F., et al.: Gate Size Selection for Standard Cell Libraries, *Proc. IEEE Int. Conf. Computer-Aided Design*, pp.545–550 (1998).
11) Kung, D.S. and Puri, R.: Optimal P/N Width Ratio Selection for Standard Cell Libraries, *Proc. IEEE Int. Conf. Computer-Aided Design*, pp.178–184 (1999).
12) Ecker, J.G.: Geometric Programming: Methods, Computations and Applications, *SIAM Review*, Vol.22, No.3, pp.338–362 (1980).
13) Bertsekas, D.P.: *Nonlinear Programming 2nd Edition*, Athena Scientific (2000).
14) Roy, S. and Chen, W.: ConvexFit: An optimal minimum-error convex fitting and smoothing algorithm with application to gate-sizing, *Proc. IEEE Int. Conf. Computer-Aided Design*, pp.196–203 (2005).
15) Roy, S. and Chen, C.C.-P.: ConvexSmooth: A simultaneous convex fitting and smoothing algorithm for convex optimization problems, *Proc. IEEE Int. Symp. Quality Electronic Design*, pp.665–670 (2006).
16) Ketkar, M., Kasmasetty, K. and Sapatnekar, S.S.: Convex delay models for transistor sizing, *Proc. ACM/IEEE Design Automation Conf.*, pp.655–660 (2000).
17) Tennakoon, H. and Sechen, C.: Efficient and accurate gate sizing with piecewise convex delay models, *Proc. ACM/IEEE Design Automation Conf.*, pp.807–812 (2005).
18) Yoshida, H., De, K. and Boppana, V.: Accurate Pre-layout Estimation of Standard Cell Characteristics, *Proc. ACM/IEEE Design Automation Conf.*, pp.208–211 (2004).
19) Synopsys, Inc.: *HSPICE Data Sheet* (2003).
20) Wachter, A. and Biegler, L.T.: On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, *Mathematical Programming*, Vol.106, No.1, pp.25–57 (2006).

(Received March 1, 2010)
(Accepted September 17, 2010)
(Original version of this article can be found in the Journal of Information Processing Vol.18, pp.252–262.)



Hiroaki Yoshida received his B.S., M.S. and Ph.D. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 2000, 2002 and 2007, respectively. From 2002 to 2006, he was a Senior Software Engineer at Zenasis Technologies, Inc., in San Jose, CA., where he was working on the development of a leading-edge logic/physical/transistor-level timing optimization tool. He is currently a Project Assistant Professor with VLSI Design and Education Center (VDEC), the University of Tokyo. His research interests include high-level, logic-level and transistor-level optimization of high-performance digital circuits.

Masahiro Fujita received his B.S. degree in electrical engineering in 1980, and the M.S. and Ph.D. degrees in information engineering from the University of Tokyo, Tokyo, Japan in 1982 and 1985, respectively. From 1985 to 1993, he was a Research Scientist with Fujitsu Laboratories, Kawasaki, Japan. From 1994 to 1999, he was the Director of the Advanced Computer-Aided Design Research Group, Fujitsu Laboratories of America, Sunnyvale, CA. He is currently a Professor in the Department of Electrical Engineering, the University of Tokyo, Tokyo, Japan. He has been on program committees for many conferences dealing with digital design and is an Associate Editor of Formal Methods on Systems Design. His primary research interest is in the computer-aided design of digital systems. Dr. Fujita received the Sakai Award from the Information Processing Society of Japan in 1984.