Performance-Constrained Transistor Sizing for Different Cell Count Minimization

Hiroaki Yoshida† and Masahiro Fujita†

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

† VLSI Design and Education Center, the University of Tokyo
Performance-Constrained Transistor Sizing for Different Cell Count Minimization

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 \(p_i^U\):

\[
p_i^L \leq p_i \leq p_i^U.
\]

Figure 2 illustrates our continuously-sized cell model. The model consists of 2 parameters: P-type transistor width \(w\) and beta ratio \(\beta\) 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 \(C_L\):

\[
d = f_d(p_1, \ldots, p_m, s_i, C_L) \tag{2}
\]

\[
s = f_s(p_1, \ldots, p_m, s_i, C_L) \tag{3}
\]

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:

\[
A = f_A(p_1, \ldots, p_m) \tag{4}
\]

\[
C_i = f_{C_i}(p_1, \ldots, p_m) \tag{5}
\]

where \(C_i\) is the capacitance of \(i\)-th input.

A posynomial \(^{12}\) is a function of a positive vector variable \(t = (t_1, \ldots, t_m) \in \mathbb{R}^m\) having the form:

\[
g(t) = \sum_{i=1}^{N} u_i(t) \tag{6}
\]

\[
u_i(t) = b_i t_1^{a_{i1}} t_2^{a_{i2}} \cdots t_m^{a_{im}}, \quad i = 1, 2, \ldots, N \tag{7}
\]

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 \(^{13}\):

\[
t_j = e^{t_j}, \quad j = 1, 2, \ldots, m. \tag{8}
\]

For a convex function, any local minimum is also a global minimum. Therefore, existing nonlinear programming techniques such as the Lagrangian relaxation
method\textsuperscript{13), can solve the minimization problem of a polynomials 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\textsuperscript{14,15). In addition, more accurate cell models based on posynomials have been proposed\textsuperscript{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\textsuperscript{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 capicances 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{align}
\text{minimize} & \quad A(p) = \sum_{i=1}^{n} A_i(p) \\
\text{subject to} & \quad AT_{\text{worst}} \leq AT_{\text{max}} \\
& \quad p_i^L \leq p_{ij} \leq p_i^U \quad (i = 1, \ldots, n, j = 1, \ldots, m) \\
& \quad AT_i \leq AT_{\text{worst}} \quad (i = 1, \ldots, o) \\
& \quad AT_1 + d_1(p) \leq AT_5, \quad AT_2 + d_2(p) \leq AT_3 \\
& \quad AT_3 + d_3(p) \leq AT_6, \quad AT_4 + d_4(p) \leq AT_6 \\
& \quad AT_5 + d_5(p) \leq AT_7, \quad AT_6 + d_6(p) \leq AT_7 \\
& \quad AT_7 + d_7(p) \leq AT_8
\end{align}

Common constraints

where $p = (p_{i1}, p_{i2}, \ldots, p_{im})$ is the set of all parameters, $A_i(p)$ is the area of $c_i$, and $AT_{\text{max}}$ 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{align}
\text{minimize} & \quad AT_{\text{worst}} \\
\text{subject to} & \quad A(p) \leq A_{\text{max}}
\end{align}

where $A_{\text{max}}$ 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\textsuperscript{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\textsuperscript{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 $g_i = (c_i, p_{i1}, \ldots, p_{im})$ and $g_j = (c_j, p_{j1}, \ldots, p_{jm})$ 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 $G$ is defined as an equivalence class on the set of gates $G$, i.e., $g_i, g_j \in G \iff g_i \sim g_j$. A different cell count $N(p)$ is defined as $|G/\sim|$, the size of the quotient set of $G$ (the number of all equivalence classes on $G$). Two gate groups $G_1 = (c_{11}, p_{11}, \ldots, p_{1m})$ and $G_2 = (c_{21}, p_{21}, \ldots, p_{2m})$ are said to be compatible if and only if $c_{11} = c_{21}$. 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: $G_1 = \{g_1\}$, $G_2 = \{g_2\}$ and $G_3 = \{g_3, g_4\}$. Therefore, $N(p) = 3$ and hence three cells are required to implement the circuit. Also, $G_1$ and $G_2$ are compatible, and $G_1$ and $G_3$ are not. Using these definitions, the problem addressed in this paper is formulated as follows:

\[
\begin{align*}
\text{minimize} & \quad N(p) \\
\text{subject to} & \quad A(p) \leq A_{\text{max}}, \quad AT_{\text{wost}} \leq AT_{\text{max}}. \quad (11)
\end{align*}
\]

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 $G_i = (c_{i1}, p_{i1}, \ldots, p_{im})$ and $G_j = (c_{j1}, p_{j1}, \ldots, p_{jm})$ is the Euclidean distance between two vectors of parameters:

\[
D(G_i, G_j) = \sqrt{\sum_{k=1}^{m} (K_i(p_{ik} - p_{jk}))^2}
\]

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 $\mu$m 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 $G_i$ and $G_j$, and (c) merging them into a gate group $G_k$, 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...
Performance-Constrained Transistor Sizing for Different Cell Count Minimization

(a) Maximize the slacks of the gate groups
(b) Find two close compatible gate groups
(c) Merge them into a new gate group

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

Fig. 5 Pseudocode for the iterative heuristic MinimizeCellCount.

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 90nm 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.

<table>
<thead>
<tr>
<th>Logic Type</th>
<th>Function</th>
<th>Drive Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV</td>
<td>A + B</td>
<td>1x, 2x, 4x, 8x, 16x</td>
</tr>
<tr>
<td>NAND2</td>
<td>A - B</td>
<td>1x, 2x, 4x, 8x</td>
</tr>
<tr>
<td>NAND3</td>
<td>A - B + C - D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>NAND4</td>
<td>A + B - C - D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>NOR2</td>
<td>A + B</td>
<td>1x, 2x, 4x, 8x</td>
</tr>
<tr>
<td>NOR3</td>
<td>A + B + C</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>NOR4</td>
<td>A + B + C + D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI21</td>
<td>A - B + C</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI22</td>
<td>A - B + C - D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI211</td>
<td>A - B + C + D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI221</td>
<td>A - B + C - D - E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI31</td>
<td>A - B - C + D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI311</td>
<td>A - B - C + D + E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI32</td>
<td>A - B + C - D + E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI321</td>
<td>A - B + C - D + E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI312</td>
<td>A - B - C + D - E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI3121</td>
<td>A - B - C + D - E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI41</td>
<td>(A + B) - C</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI411</td>
<td>(A + B) - C - E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI412</td>
<td>(A + B) - C + D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI4121</td>
<td>(A + B) - C + D - E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI413</td>
<td>(A + B + C) - D</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI4131</td>
<td>(A + B + C) - D + E</td>
<td>1x, 2x, 4x</td>
</tr>
<tr>
<td>AOI414</td>
<td>(A + B + C + D) - E</td>
<td>1x, 2x, 4x</td>
</tr>
</tbody>
</table>

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 μm to 8 μm 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 with HSPICE 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.

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 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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>16</td>
<td>2955.4</td>
<td>1.3508</td>
<td>99</td>
<td>2955.4</td>
<td>1.3643</td>
<td>66</td>
<td>31.9</td>
</tr>
<tr>
<td>C499</td>
<td>9</td>
<td>7374.8</td>
<td>0.8545</td>
<td>250</td>
<td>7173.7</td>
<td>0.8632</td>
<td>32</td>
<td>71.6</td>
</tr>
<tr>
<td>C880</td>
<td>20</td>
<td>2884.2</td>
<td>1.1643</td>
<td>213</td>
<td>2884.2</td>
<td>1.1759</td>
<td>79</td>
<td>35.5</td>
</tr>
<tr>
<td>C1355</td>
<td>9</td>
<td>7443.3</td>
<td>0.8507</td>
<td>259</td>
<td>7443.3</td>
<td>0.8643</td>
<td>45</td>
<td>74.0</td>
</tr>
<tr>
<td>C1908</td>
<td>21</td>
<td>6145.6</td>
<td>1.2704</td>
<td>274</td>
<td>5813.0</td>
<td>1.2833</td>
<td>116</td>
<td>103.9</td>
</tr>
<tr>
<td>C2670</td>
<td>20</td>
<td>3712.8</td>
<td>1.0555</td>
<td>554</td>
<td>3708.9</td>
<td>1.0660</td>
<td>58</td>
<td>401.3</td>
</tr>
<tr>
<td>C5354</td>
<td>24</td>
<td>9512.5</td>
<td>1.7405</td>
<td>628</td>
<td>9512.5</td>
<td>1.7479</td>
<td>148</td>
<td>1293.2</td>
</tr>
<tr>
<td>C5315</td>
<td>20</td>
<td>8427.9</td>
<td>1.2830</td>
<td>941</td>
<td>8427.0</td>
<td>1.2959</td>
<td>153</td>
<td>1297.4</td>
</tr>
<tr>
<td>C6288</td>
<td>17</td>
<td>14636.1</td>
<td>4.8085</td>
<td>1587</td>
<td>14636.1</td>
<td>4.8567</td>
<td>265</td>
<td>5101.5</td>
</tr>
<tr>
<td>C7552</td>
<td>23</td>
<td>11268.6</td>
<td>1.4335</td>
<td>1341</td>
<td>11268.5</td>
<td>1.4678</td>
<td>172</td>
<td>12705.6</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figures 6, 7 and 8 present tradeoff curves between the maximum path delay and 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 \(D_{\text{opt}} \times 1.01\) and the maximum area of \(A_{\text{opt}}\) where \(D_{\text{opt}}\) and \(A_{\text{opt}}\) 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 \((N_{\text{opt}} - N_{1\%}) / N_{\text{opt}} \times 100\) where \(N_{\text{opt}}\) 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.
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
Performance-Constrained Transistor Sizing for Different Cell Count Minimization

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.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Discretely-sized Library</th>
<th>Continuously-sized Library (Maximum delay = $D_{ds}$)</th>
<th>Continuously-sized Library (Maximum area = $A_{ds}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area [$\mu$m²]</td>
<td>Delay [ns]</td>
<td>Area [$\mu$m²]</td>
</tr>
<tr>
<td>C432</td>
<td>1804.2</td>
<td>1.5183</td>
<td>1163.4</td>
</tr>
<tr>
<td>C499</td>
<td>4121.8</td>
<td>1.0062</td>
<td>2436.2</td>
</tr>
<tr>
<td>C880</td>
<td>2471.0</td>
<td>1.3705</td>
<td>1805.5</td>
</tr>
<tr>
<td>C1355</td>
<td>3790.1</td>
<td>1.0137</td>
<td>2349.6</td>
</tr>
<tr>
<td>C1908</td>
<td>3725.4</td>
<td>1.4467</td>
<td>2925.8</td>
</tr>
<tr>
<td>C2670</td>
<td>4395.2</td>
<td>1.2076</td>
<td>3541.9</td>
</tr>
<tr>
<td>C3540</td>
<td>7358.2</td>
<td>2.0254</td>
<td>6315.6</td>
</tr>
<tr>
<td>C5315</td>
<td>8720.1</td>
<td>1.4418</td>
<td>6986.5</td>
</tr>
<tr>
<td>C6288</td>
<td>24601.4</td>
<td>5.4583</td>
<td>13981.7</td>
</tr>
<tr>
<td>C7552</td>
<td>12857.8</td>
<td>1.5732</td>
<td>9369.2</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 bench-
mark 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


(Received March 1, 2010) (Accepted September 17, 2010) (Released December 8, 2010)
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.