Performance-Constrained Different Cell Count Minimization for Continuously-Sized Circuits

Hiroaki Yoshida[†,‡] Masahiro Fujita[†,‡]

†VLSI Design and Education Center (VDEC), University of Tokyo
‡CREST, Japan Science and Technology Agency

Abstract

A continuously-sized circuit resulting from transistor sizing consists of gates with large variety of sizes. In this paper, we first provide a formal formulation of performance-constrained different cell count minimization problem, and then propose an effective hill-climbing heuristic which iteratively minimizes the number of cells under performance constraints such as area, delay and power. To the best of our knowledge, this is the first attempt to address the different cell count minimization problem.

1 Introduction

Design optimization at the transistor-level has been successfully used to achieve significant performance benefits above and beyond gate-level design optimization. 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.

A continuously-sized circuit resulting from transistor sizing consists of gates with 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 an industrial 90nm technology. In the figure, 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. 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 a whole circuit. This drawback renders any continuous sizing solution in the standard cell based design flow to be impractical. This paper addresses a performance-constrained cell count minimization problem. Unlike the gate selection problem [4, 5] 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.

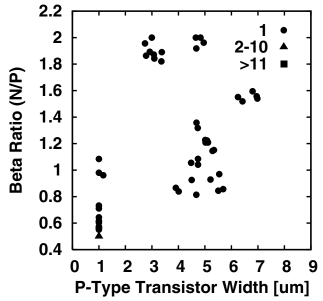


Figure 1. Cell size distribution of 2-input NOR gates after delay-optimal sizing

2 Preliminaries

2.1 Posynomial Cell Model

Our cell model is the posynomial cell model [1] which is most-commonly used by convex optimization based transistor sizing. Each cell is a *parameterized cell* where the sizes of the transistors in the cell are specified by a set of parameters $(p_1, \ldots, p_m)$, *e.g.* beta ratio and taper factor. Each parameter p_i has its lower bound p_i^L and upper bound p_i^U. In particular, the model used in our experiments 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. Note that other parameters such as taper factor can be incorporated to increase the degree of freedom and/or to improve the accuracy. 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 arc of the cell for a given input slew s_i and an output load C_L. Likewise, a cell power is typically modeled in the same way. Also, an area A and an input capacitance C_i of a cell are given as the functions of the parameters.

A *posynomial* [6] is a function g of a positive vector variable $t \in R^m$ having the form: $g(t) = \sum_{i=1}^{N} b_i t_1^{a_{i1}} t_2^{a_{i2}} \cdots t_m^{a_{im}}$ where the exponents a_{ij} are arbitrary real numbers and the coefficients b_i are positive. For a convex function, any local minimum is also a global minimum. The characteristics of a cell 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.

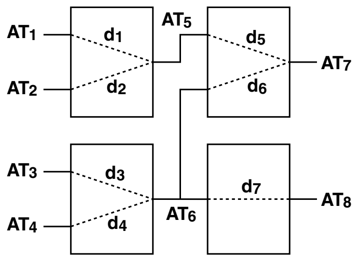


Figure 2. An example circuit model for continuous transistor sizing

2.2 Optimal Continuous Transistor Sizing

This section overviews an optimal continuous transistor sizing algorithm [3] which we use in the proposed method. Figure 2 shows an example circuit model for continuous transistor sizing. *For ease of explanation*, the following formulation does not take account of slews and load capacitances, 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 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. Then, area minimization problem under delay constraints can be formulated as follows:

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

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. Similarly, delay minimization problem under an area constraint can be formulated as follows (throughout the remainder of this paper, constraints for parameters and arrival times at internal wires are omitted for ease of explanation):

$$\begin{array}{l}\textbf{minimize } AT_{worst} \\ \textbf{subject to } A(p) \le A_{max}, \ AT_i \le AT_{worst} \ (i = 1, \ldots, o)\end{array} \qquad (2)$$

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.

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.

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 $g_i \sim g_j$. A *gate group* Γ is defined as an equivalence class on the set of gates G, *i.e.*, $g_i, g_j \in \Gamma \iff g_i \sim g_j$. 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 $c_{\Gamma_i} = c_{\Gamma_j}$. $N(p)$ can also be viewed as the number of cells which are required to implement the circuit. Using these definitions, the problem addressed in this paper is formulated as follows:

$$\begin{array}{l}\textbf{minimize } N(p) \\ \textbf{subject to } A(p) \le A_{max}, \ AT_i \le AT_{max} \ (i = 1, \ldots, o).\end{array} \qquad (3)$$

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 the conventional nonlinear programming techniques do not solve this problem properly, we propose an effective heuristic to solve this problem.

3.2 Hill-Climbing Heuristic

The proposed heuristic is based on hill-climbing method [7]. Starting from an optimally-sized circuit which satisfies the constraints, it reduces $N(p)$ by one at a time while satisfying the constraints, and it is repeated until no further change can be made. The basic idea of reducing $N(p)$ by one is (a) finding two compatible gate groups Γ_i and Γ_j, (b) merging them into a gate group Γ_k, and (c) finding the optimal set of parameters of Γ_k, as shown in Figure 3. Step (c) is the key to a successful different cell count minimization. A simple approach consists only of steps (a) and (b) does not work for the following reason. Each gate group has the freedom of its movement according to the performance constraints. Merging gate groups decreases the freedom because the freedom of the merged gate group is the intersection of the freedoms of the original gate groups. Step (c) increases the freedom by enabling the move of the other gate groups, and hence more gate groups can be merged. In addition, step (c) is formulated as a general transistor sizing problem, so any existing transistor sizing methods and cell models can be used.

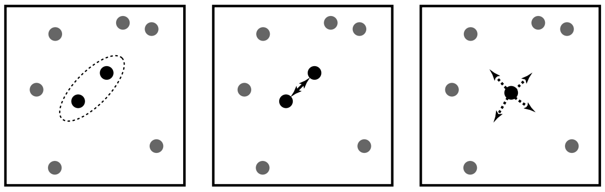

(a) Finding two compatible gate groups (b) Merging them into a gate group (c) Finding the optimal set of parameters of the gate group

Figure 3. Illustration of hill-climbing heuristic

The quality of this heuristic also depends upon the choice of two gate groups to be merged. Basically, merging two gate groups reduces the degrees of freedom of sizing and hence the resulting performance can also degrades. Therefore, two gate groups need to be chosen such that merging them has the least impact on the circuit performance. The proposed method uses the notions of slack and distance which are defined as follows. 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. 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. Since two gate groups to be merged should have a large freedom and a small impact on the area and performance, the basic criteria of choosing them is that (1) the distance between the gate groups is small and (2) at least one of the slacks of the gate groups is large.

As mentioned above, step (c) is the most important step in the proposed procedure. To accurately analyze the slacks of gate groups, the proposed method performs total slack maximization under the given performance constraints:

$$\begin{aligned}
&\textbf{maximize } S(p)\\
&\textbf{subject to}\\
&\quad A(p) \le A_{max}\\
&\quad AT_i \le AT_{max}\ (i = 1, \ldots, o)\\
&\quad p_{ik} = p_{jk}\ \ (k = 1, \ldots, m,\ g_i \in \Gamma_l,\ g_j \in \Gamma_l,\ \Gamma_l \in G/\sim)
\end{aligned} \tag{4}$$

where $S(p)$ is the sum of the slacks of the wires. In the last constraint in (4), the gates in each gate group are forced to have the same set of parameters. Thus, the different cell count remains the same during the total slack maximization. A pseudocode for the hill-climbing heuristic is presented in Procedure 1.

Procedure 1 Different Cell Count Minimization

```
1: repeat
2:   Maximize total slack under performance constraints
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:       (Locally) 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
```

4 Experimental Results

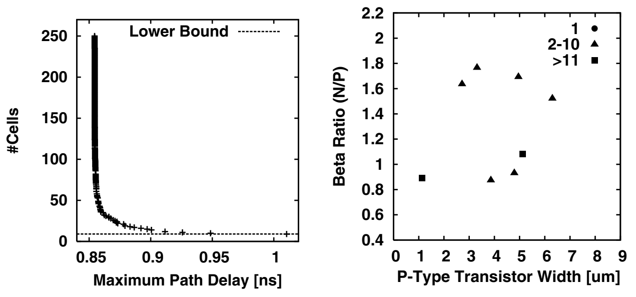
First, we constructed a continuously-sized cell library consisting of 24 typical logic types. The cells were characterized for the posynomial cell model described in Section 2.1 using an industrial 90nm technology. Cell delays and slews were simulated using a prelayout cell characteristic estimator [8]. Then, we fitted the data to a posynomial function and obtained the coefficients and exponents. Overall, the average fitting error was about 1.06% and the standard deviation was 1.19%. For cell areas, the average fitting and the standard deviation were both less than 0.01%. For input loads, the average fitting 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 [9] is used. We then applied them to 10 circuits from the ISCAS 85 benchmark circuits. The benchmark circuits were first synthesized for optimal delay using a discretely-sized cell library in the same 90nm technology. 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. Then, we applied the proposed different cell count minimization method to the delay-optimal circuits as follows. The different cell count of each circuit is minimized with 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 1 compares the different cell counts of the delay-optimal circuits and the circuits after the different cell count

Table 1. Different cell count minimization results in an industrial 90nm technology.

Circuit	*#Used Logic Types*	*Delay Optimal*			*Accepting 1% Degradation of Optimal Delay*				*Different Cell Count Reduction [%]*
		Area [μm^2]	*Delay [ns]*	*Different* Cell Count	*Area [μm^2]*	*Delay [ns]*	*Different* 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) Delay vs. different cell count tradeoff curve

(b) Cell size distributions after cell count minimization with accepting 1% degradation

Figure 4. Different cell count minimization results on C499.

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 with accepting 1% degradation which is almost equivalent to the delay model error. Figure 4 (a) presents a tradeoff curve between the maximum path delay and the different cell count on C499. The curve was obtained by increasing the maximum path delay constraint from D_{opt} and keeping the area constraint the same. An important observation from these results is that the different cell count can be reduced dramatically with accepting very little delay degradation. Figure 4 (b) show the cell size distributions of 2-input NOR gates in a circuit C499 after delay-optimal sizing and after different cell count minimization with 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.

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. As far as the authors know, this is the first attempt to address the different cell count minimization problem. The experimental results on a benchmark suite demonstrated its effectiveness. We expect that the presented solution will be successfully used to achieve a practical realization of continuous sizing in the standard cell based design flow.

References

[1] J. P. Fishburn and A. E. Dunlop. Tilos: A posynomial programming approach to transistor sizing. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 326–328, Nov. 1985.

[2] S. S. Sapatnekar et al. An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. *IEEE Trans. Computer-Aided Design*, 12(11):1621–1634, Nov. 1993.

[3] C. P. Chen et al. Fast and exact simultaneous gate and wire sizing by lagrangian relaxation. *IEEE Trans. Computer-Aided Design*, 18(7):1014–1025, July 1999.

[4] F. Beeftink et al. Gate size selection for standard cell libraries. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 545–550, Nov. 1998.

[5] D. S. Kung and R. Puri. Optimal P/N width ratio selection for standard cell libraries. In *Proc. IEEE Int. Conf. on Computer-Aided Design*, pages 178–184, Nov. 1999.

[6] J. G. Ecker. Geometric programming: Methods, computations and applications. *SIAM Review*, 22(3):338–362, July 1980.

[7] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag, 2002.

[8] H. Yoshida, K. De, and V. Boppana. Accurate pre-layout estimation of standard cell characteristics. In *Proc. ACM/IEEE Design Automation Conf.*, pages 208–211, June 2004.

[9] A. Wachter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.