Proc. IEEE 1993 Custom Integrated Circuits Conf.

DELAY AND AREA OPTIMIZATION FOR DISCRETE GATE SIZES UNDER DOUBLE-SIDED TIMING CONSTRAINTS *

Weitong Chuang† Sachin S. Sapatnekar‡ Ibrahim N. Hajj†

†Coordinated Science Laboratory and
Dept. of Electrical & Computer Engineering
University of Illinois at Urbana-Champaign

‡Department of Electrical Engineering
and Computer Engineering
Iowa State University

ABSTRACT

A three-step algorithm is presented for discrete gate sizing problem of delay/area optimization under double-sided timing constraints. The problem is first formulated as a linear program. The solution to the linear program is then mapped onto a permissible set. Using this permissible set, the gate sizes are adjusted to satisfy the delay lower and upper bounds simultaneously.

1. INTRODUCTION

Discrete gate sizing is the problem of selecting a proper size from a standard-cell library for each gate in a circuit design, without changing the logic function of the gate, with the aim of minimizing a certain cost function such as total area or power consumption, and at the same time, meeting the timing constraints imposed on the propagation delay along each path from the primary input (PI) to the primary output (PO). The two-sided timing constraints problem further requires that propagation delay is bounded from below by T_{inf} and bounded from above by T_{sup}. This problem has been shown to be NP-complete [1].

For a combinational circuit, the gate sizing problem is formulated as

$$\begin{aligned} &\text{minimize} && Area \\ &\text{subject to} && T_{inf} \leq Delay \leq T_{sup}. \end{aligned} \qquad (1)$$

Chan [1] proposed an algorithm to obtain a minimum area realization of gates interconnected as a tree structure under double-sided timing constraints. He extended the method for general directed acyclic graphs (DAGs). However, the procedure does not necessarily obtain the optimal solution for a general DAG, and the algorithm is of exponential complexity. Lin *et al.*[2] and Li *et al.*[3] both proposed heuristic algorithms for single-sided timing constraint. The algorithm proposed by Lin *et al.* is based on the TILOS algorithm [4] for continuous transistor sizing, with further generalization. The algorithm is essentially an iterative procedure consisting of two phases: in the first phase, cell sizes are increased from the minimum size until the delay constraint is satisfied; while in the second, the sizes of certain cells are decreased to minimize the total area, while ensuring that the timing specification is not violated. The algorithm proposed by Li *et al.* is exact for serial-parallel graphs. The work is extended to non-serial-parallel circuits by the use of several heuristics. Both of Lin's and Li's approaches are heuristic; hence no conclusive statements can be made on how close their solutions are to the optimal solution. Moreover, neither work shows comparisons with a technique such as simulated annealing that is well-known to give optimal or near-optimal solutions. Although both approaches may be flexible enough to be extended to handle double-sided timing constraints sizing problem, no substantial experiments were conducted to show such flexibility.

In this work, we tackle the discrete gate sizing problem for CMOS standard cells and present an algorithm that works in three phases to find its solution. In the first stage, the gate sizing problem is formulated as a linear program. The solution of this linear program provides us with a set of gate sizes that does not necessarily belong to the set of allowable sizes. Hence, in the second phase, we move from the linear program solution to a set of allowable gate sizes, using heuristic techniques. At the conclusion of the second phase, the set of allowable sizes obtained may not satisfy the delay lower and upper bound simultaneously. Hence in the third stage, we use another set of heuristics to adjust the gate sizes to meet the double-sided delay constraints. Finally, to illustrate the efficacy of our algorithm, we present a comparison of the results of this technique with corresponding solutions obtained by simulated annealing.

2. PHASE I : THE LINEAR PROGRAMMING APPROACH

2.1 Delay Modeling

We assume each gate in a standard cell library can be represented by an equivalent inverter such that the ratio of the p-transistor size to the n-transistor size

*This research was supported by Joint Services Electronics Program under contract N00014-90-J-1270 and by the Semiconductor Research Corp. under contract 92-DP-109.

of that inverter is a constant. Hence, the size of each gate can be parameterized by a single number, which we refer to as the gate size. As in [4], the equivalent inverter is replaced by an RC circuit; the delay of this circuit is taken to be the delay of the inverter. In this case, under the assumption that the capacitance at the output of a gate is constant, the delay of a gate of size x is given by

$$D(x) \;=\; \frac{R_u}{x} \times C_{out}, \tag{2}$$

Here R_u represents the on-resistance of a unit transistor. It can easily be verified that $D(x)$ is a convex function over the range of positive x.

Under this model, we can compute a set of values (x_j, d_j), corresponding to the delay d_j associated with gate size x_j.

It can be shown that the function $D(x)$ above can be approximated by a *convex* function, $PWL(x)$ of the form

$$PWL(x) = \begin{cases} a_1 \cdot x + b_1 & x_1 \le x \le x_2 \\ a_2 \cdot x + b_2 & x_2 \le x \le x_3 \\ \vdots & \\ a_{k-1} \cdot x + b_{k-1} & x_{k-1} \le x \le x_k \end{cases} \tag{3}$$

Since $PWL(x)$ is convex, it can also be written as

$$PWL(x) = \max_{1 \le j < p} \left(a_j \cdot x + b_j\right) \quad \forall\, x \in [x_1, x_k]. \tag{4}$$

2.2 Formulation of the Linear Program

The formal definition of the gate sizing problem is as given in Equation (1). Since the area of the circuit is difficult to estimate accurately, we approximate it as the sum of the gate sizes, as has been done in almost all work on sizing [4, 5, 6, 7].

The delay specification, which states that the circuit delay should be bounded above by T_{sup}, is equivalent to stating that all path delays must be bounded by T_{sup}. Since the number of PI-PO (primary input and primary output) paths could be exponential, the set of constraining delay equations could potentially be exponential in number. Hence we introduce additional variables, m_i, $i = 1 \cdots n$ (where n is the number of gates), correspond to the worst-case delay from the primary inputs up to each gate. Using these variables, for each gate i, we have

$$m_j + d_i \le m_i, \quad \forall\, j \in Fanin(i). \tag{5}$$

This reduces the number of constraining equations to $\sum_{i=1}^{n} Fanin(i)$, which is bounded above by n^2, but is typically $O(n)$ for practical circuits.

Similarly, for the shortest delay, we introduce new variables, p_i, $i = 1 \cdots n$, correspond to the shortest delay from PI's up to the output of G_i.

$$p_j + d_i \ge p_i, \quad \forall\, j \in Fanin(i). \tag{6}$$

We now formulate the linear program as

$$\begin{aligned} &minimize && \sum_{i=1}^{n} x_i \\ &subject\ to \\ &\text{For all gates } i = 1 \cdots n \\ &\quad m_j + d_i \le m_i, \quad p_j + d_i \ge p_i && \forall\, j \in Fanin(i) \\ &\quad m_i \le T_{sup}, \quad\ p_i \ge T_{inf} && \forall\ gates\ i\ at\ PO's \\ &\quad d_i \ge a_{i,j} \cdot x_i + b_{i,j} && for\ 1 \le j < k_i \\ &\quad Minsize_i \ \le\ x_i \ \le\ Maxsize_i \end{aligned} \tag{7}$$

3. PHASE II : THE MAPPING ALGORITHM

3.1 Introduction

The set of permissible sizes for gate i is $\mathcal{S}_i = \{x_{i,1} \cdots x_{i,k_i}\}$. The solution of the linear program would, in general, provide a gate size, x_i, that does not belong to $\mathcal{S}_i$. In this case, we consider the two permissible gate sizes that are closest to x_i; we denote the nearest larger (smaller) size by x_{i+} (x_{i-}). Therefore, we formulate the following problem:

For all $i = 1 \cdots n$: Select $x_i = x_{i+}$ or x_{i-} such that $T_{inf} \le$ Delay $\le T_{sup}$.

Although the solution to this problem is not necessarily the optimal solution, it is very likely that the objective function value at this point is close to optimal. This supposition is borne out by the results presented in Section 4.

In this section, we propose a sensitivity-based heuristic which maps the linear program solution to a permissible set. The complexity of the heuristic is $O(n^2)$ in the most pathological case, and is much less for the average case.

3.2 Generalized Sensitivity Analysis

3.2.1 Path Slack and Path Tension

For every path P in the circuit, we define two quantities known as *path slack*, PS, and *path tension*, PT. The two quantities are defined as follows:

$$\mathrm{PS(P)} = \begin{cases} \min_{j \in FO(i)} \{m_j - (m_i + d_j)\} \\ \qquad if\ gate\ i\ is\ not\ at\ a\ PO. \\ \min\left(\min_{j \in FO(i)} \{m_j - (m_i + d_j)\}, T_{sup} - m_i\right) \\ \qquad if\ gate\ i\ is\ at\ a\ PO. \end{cases} \tag{8}$$

$$\mathrm{PT(P)} = \begin{cases} \min_{j \in FO(i)} \{(p_i + d_j) - p_j\} \\ \qquad if\ gate\ i\ is\ not\ at\ a\ PO. \\ \min\left(\min_{j \in FO(i)} \{(p_i + d_j) - p_j\}, p_i - T_{inf}\right) \\ \qquad if\ gate\ i\ is\ at\ a\ PO. \end{cases} \tag{9}$$

where gate i is the gate that lies at the end of path P, and $FO(i)$ denotes fanout set of gate i. Although the number of paths in a circuit could be exponential, our algorithm needs to compute the path slack for at most n paths in the circuit.

3.2.2 Incremental and Decremental Sensitivity

For each gate i, we define the *incremental sensitivity*, s_+, and the *decremental sensitivity*, s_-, as :

$$s_+ = d(x_{i-}) - d(x_i). \quad (10)$$

$$s_- = d(x_i) - d(x_{i+}). \quad (11)$$

Our heuristic rank-orders the gates on a path in increasing order of their incremental (decremental) sensitivities, and places them in this order on an *incremental (decremental) sensitivity list* or *ISL (DSL)*. This list is then used to decide the sequence in which gate sizes are to be changed, as is explained in Section 3.3.

3.2.3 The Cone Criterion

In additional to s_+ and s_-, another factor that is taken into consideration while deciding the order in which gates are to be processed is what we call the *cone criterion*. It is based on the fact that since the objective is to minimize the sum of the gate sizes, we would like to reduce the sizes of as many gates as possible, and increase the sizes of as few gates as we can.

Cone criterion : If two gates have the same value of s_+, then the gate which is farthest from a primary input is given precedence in the ISL. Likewise, if two gates have the same value of s_-, then the one which is farthest from a primary output is given precedence in the DSL.

It can be observed that gates towards the PO's have fewer gates within their fanout cones than gates towards the PI's. In particular, when we consider gates that lie on a common path, the fanout cone of a gate towards the output is a subset of that of a gate that is closer to the input.

Hence, if the delay along some path is to be increased, it would be more advantageous to decrease the size of a gate closer to PO, since decreasing the size of a gate near to PI could possibly have negative effects on the delays of other PI-PO paths. On the other hand, if the delay along some path is to be decreased, it would be more advantageous to increase the size of a gate closer to PI, since this increase would decrease the delays of a larger number of paths. As a result, it is likely that a larger number of gates could change their sizes from x_i to x_{i-}, which is in line with the task of minimizing the objective function.

In practice, we exercise the cone criterion to determine the order of the ISL (DSL) when the values of s_+ (s_-) for two gates are within a constant factor (such as 5%) of each other.

3.3 The Heuristic Algorithm

The algorithm first places all gates in a queue in decreasing order of their worst case signal arrival time, m_i. The longest path P_l to the gate on the head of the queue is detected. Unprocessed gates along P_l are placed on the ISL and DSL, using the procedure described in Section 3.2. The philosophy behind the algorithm is to fill up the path slack of P_l by decreasing the sizes of gates at the head of the ISL. When the path slack can no longer accommodate any more decrements in the size of a gate, we take the gate at the head of the DSL and increase its size; this reduces the delay and generates some additional slack for path P_l.

The idea behind processing a gate with a small incremental sensitivity earlier is that it would be expected that such a gate would have a large decremental sensitivity and would be near the tail of the DSL. After we set the size of this gate to the next allowable higher size, its decremental sensitivity is withdrawn from the DSL since the gate has been processed. As a result, processing the ISL from the top also achieves the effect of pruning the DSL from below. Hence, it is expected that only small perturbations about the LP solution are required.

After each size change, the arrival times of the affected gates are updated; it is adequate to use incremental techniques, since only one gate size is changed.

After the longest path to the gate on the head of the queue is processed, the shortest path P_s is detected. Similarly, unprocessed gates along P_s are placed on the ISL and DSL. The path tension of P_s is consumed by increasing the sizes of gates at the head of the DSL. When the path tension can no longer afford any more increments in the gate size, we take the gate at the head of the ISL and decrease its size; this increases the delay and produces some additional tension for path P_s.

The procedure continues until each gate has been processed.

4. PHASE III : THE ADJUSTING ALGORITHM

At the conclusion of the second phase, the set of allowable sizes obtained may not satisfy the delay lower and upper bounds simultaneously. Hence we have to adjust the sizes of some gates to meet the double-sided delay constraints.

For every gate i in the circuit, we define two quantities known as *gate slack, gs,* and *gate tension, gt.* The two quantities are defined as follows:

$$gs_i = \begin{cases} \min_{j \in FO(i)} \{(m_j + gs_j) - (d_j + m_i)\} \\ \qquad\qquad \textit{if gate } i \textit{ is not at a PO.} \\ \min\{\min_{j \in FO(i)}((m_j + gs_j) - (d_j + m_i)), T_{sup} - m_i\} \\ \qquad\qquad \textit{if gate } i \textit{ is at a PO.} \end{cases} \quad (12)$$

$$gt_i = \begin{cases} \min_{j \in FO(i)} \{(p_i + d_j + gt_j) - p_j\} \\ \qquad\qquad \textit{if gate } i \textit{ is not at a PO.} \\ \min\{\min_{j \in FO(i)}((p_i + d_j + gt_j) - p_j), p_i - T_{inf}\} \\ \qquad\qquad \textit{if gate } i \textit{ is at a PO.} \end{cases} \quad (13)$$

Table 1 Performance comparison of our algorithm with simulated annealing.

Circuit	T_{spec}	Our Algorithm				Simulated Annealing		$\frac{A_{HS}}{A_{SA}}$
	$[T_{inf}, T_{sup}]$	T_{min}	T_{max}	Runtime	Area (A_{HS})	Runtime	Area (A_{SA})	
c17	[1.0, 2.0]	1.03	1.71	0.09s	312	1m 53s	312	1.000
c432	[2.0, 15.0]	2.00	14.93	8.21s	6766	32m 9s	5986	1.130
c499	[1.0, 5.0]	1.19	4.96	20.55s	15756	1h 4m	15132	1.041
c880	[1.2, 10.0]	1.22	9.99	1m 1s	15034	1h 50m	14293	1.052
c1355	[2.0, 16.0]	2.15	15.94	2m 13s	24206	5h 14m	23036	1.051
c1908	[1.8, 35.0]	1.88	34.96	3m 48s	22964	6h 33m	20780	1.105
c3540	[1.4, 25.0]	1.46	24.99	31m 44s	50511	10h 11m	48795	1.035
c6288	[1.1, 135.0]	1.19	134.38	33m 21s	66748	17h 21m	65325	1.022

Starting from PO's, we back-propagate gate slack and gate tension. For each PO which violates timing constraints, we find the longest path and/or shortest path to that PO. For example, if gate i at the PO violates T_{sup}, we first find the longest path to i. For each gate along that longest path, we check if it is possible to further increase its size. If so, we calculate the reduction of delay due to the increase of the gate size. We ignore the gates for which the reduction in the gate delay is greater than the gate tension, to ensure that the delay lower bound would not be violated. The cone criterion and generalized sensitivity can be applied to determine which gate size is to be bumped up first. A similar procedure is applied for all PO's violating delay lower bound. The procedure continues until the delay constraints are satisfied at all PO's.

5. EXPERIMENTAL RESULTS AND CONCLUSIONS

The algorithm above was implemented in C on a Sun Sparc 10 station. To prove the efficacy of the approach, a simulated annealing algorithm was implemented for the purpose. The cell library contains five different sizes for each of the logic gates.

The results, in comparison with simulated annealing, are shown in Table 1. The test circuits include most of the ISCAS85 benchmarks. T_{inf} and T_{sup} in the second column indicate the delay lower bound and delay upper bound, respectively. The third and the fourth columns of the table are the minimum and the maximum delays in the optimized circuit. The runtimes are moderate even for large circuits, and are considerably smaller than those for simulated annealing. We also would like to point out that the chief component (over 95%) of the runtime was solving the linear program; the heuristic was extremely fast in comparison. A_{HS} is the total area of the circuit using our algorithm, while A_{SA} is the total area using simulated annealing. The total area is measured as the sum of the gate sizes. It can be seen that the accuracy of the results of our approach ranges from being as good as simulated annealing for c17 to an discrepancy of 13.0% for c432 in comparison with simulated annealing. The average discrepancy is 4.5%.

In conclusion, in this paper, a three-step algorithm is presented for discrete gate sizing problem under double-sided timing constraints. The results show that the algorithm is able to find a near optimal solution in a reasonable amount of time. Although the algorithm is for delay/area optimization problem, it can be extended to handle delay/power optimization problem.

REFERENCES

[1] P. K. Chan, "Algorithms for library-specific sizing of combinational logic," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 353–356, 1990.

[2] S. Lin, M. Marek-Sadowska, and E. S. Kuh, "Delay and area optimization in standard-cell design," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 349–352, 1990.

[3] W. Li, A. Lim, P. Agrawal, and S. Sahni, "On the circuit implementation problem," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 478–483, 1992.

[4] J. Fishburn and A. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 326–328, 1985.

[5] J.-M. Shyu, A. Sangiovanni-Vincentelli, J. Fishburn, and A. Dunlop, "Optimization-based transistor sizing," *IEEE J. Solid-State Circuits*, vol. 23, pp. 400–409, Apr. 1988.

[6] S. S. Sapatnekar, V. B. Rao, and P. M. Vaidya, "A convex optimization approach to transistor sizing for CMOS circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, pp. 482–485, 1991.

[7] M. R. Berkelaar and J. A. Jess, "Gate sizing in MOS digital circuits with linear programming," in *Proc. European Design Automation Conf.*, pp. 217–221, 1990.