Fast Comparisons of Circuit Implementations *

Shrirang K. Karandikar and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN, USA
E-mail: {srirang, sachin}@ece.umn.edu

Abstract

Digital designs can be mapped to different implementations using diverse approaches, with varying cost criteria. Post-processing transforms, such as transistor sizing can drastically improve circuit performance, by optimizing critical paths to meet timing specifications. However, most transistor sizing tools have high execution times, and the attainable circuit delay can be determined only after running the tool. In this paper, we present an approach for fast transistor sizing that can enable a designer to choose one among several functionally identical implementations. Our algorithm computes the minimum achievable delay of a circuit with a maximum average error of 5.5% in less than a second for even the largest benchmarks.

1 Introduction

Implementing a design involves synthesis (technology independent optimizations and technology mapping), placement and routing. In a final timing correction step, transistors of logic gates are appropriately sized to speed up critical paths, thus incurring an area overhead for gains in circuit speed. The importance of transistor sizing can be judged by the amount of research carried out both in academia [1–4] and in industry [5, 6]. However, these optimization tools have large running times, and can take up to a few hours to calculate the appropriate solution for an industry-sized circuit. In this scenario, it is difficult for a designer to determine if an implementation will be able to meet performance goals after transistor sizing, or which circuit out of multiple different implementations for the same functionality should be chosen for future detailed optimization.

The *delay* of a circuit is the maximum delay of all PI-to-PO paths of the circuit. Transistor sizing is applied to the circuit to reduce this delay, in order to meet design goals. The smallest value of delay that can be obtained in this manner is referred to as the *minimum achievable delay*. In this paper, we present an approach that can quickly estimate this minimum delay when transistor sizing is applied to a mapped circuit. Gains due to this optimization vary according to the circuit being sized, because of a number of factors, such as which logic gates are used, how these gates are connected, and how much load they drive. The minimum achievable delay captures how amenable a circuit is to transistor sizing. Thus, while circuits are rarely sized to the minimum delay value (due to the associated high area overheads), it is a good measure of circuit quality. Using our tool as a *fast* estimator of the minimum achievable delay, a designer can make early comparisons among different solutions provided by placement and routing for the same functionality, without incurring the cost of an actual sizing step. Once the designer has chosen one of the candidate implementations based on this metric, a more exact optimizer can be used to obtain actual transistor sizes.

Our approach is inspired by logical effort [7,8], a method well suited for estimating the minimum achievable delay of a *single* path in a circuit, with a heuristic branching factor used to account for multiple fanouts. However, the critical path of a circuit changes dynamically according to the choice of distribution of capacitance over multiple fanouts. An important contribution and differentiator of our algorithm is a means of accurately determining the minimum achievable delay of a circuit by simultaneously considering *all* paths of the circuit.

2 Logical Effort

The starting point of our approach is the method of logical effort, which has been widely used in a variety of application domains [9–12] as well as in industry standard EDA synthesis tools [13, 14]. Using logical effort, the delay of a gate with input capacitance c_i is estimated by modeling it as a linear function of the load c_l being driven as:

$$D = g \times \frac{c_l}{c_i} + \text{parasitic delay} \tag{1}$$

where g is the logical effort, $\frac{c_l}{c_i}$ is the electrical effort and the parasitic delay is the intrinsic delay of the gate.

As shown in [8], the above equation can be extended to estimate the minimum delay, $\hat{D}$, of a *path* of logic as

$$\hat{D} = NF^{\frac{1}{N}} + P \tag{2}$$

where $F = GH$ is the path effort, P is the path parasitic delay and N is the number of gates on the path under consideration. The path logical effort, G and path electrical effort, H are obtained as the product of the gate logical and electrical efforts. The minimum delay of Equation (2) is obtained by distributing the path effort F equally to each gate on the path.

For realistic circuits, with gates having multiple fanouts, the path effort is modified to $F = GBH$, where B, the path branching effort, is the product of the gate branching efforts of all gates on the path being analyzed. The branching effort of a gate G, b_{G}, is calculated as $\frac{C_{total}}{C_{useful}}$, where

*This research was supported in part by the SRC under contract 2001-TJ-884

1530-1591/04 $20.00 (c) 2004 IEEE

C_{total} is the total load being driven by G, and C_{useful} is the load contributed by the fanout on the path of interest. Thus, the branching factor tries to capture the effect of off-path fanouts. However, this definition compels non-critical fanouts to contribute a load that scales in proportion to the load of the fanout of interest, which is determined *only* by gates on the path being analyzed. This has two disruptive effects. First, the load on the path under consideration, and hence its delay estimate, is larger than necessary, due to the contribution of the non-critical fanouts. Second, assumptions of gate sizes on the path being analyzed affect the loads on the other gates in the circuit. This may lead to the delay of a different path becoming dominant. The branching factor does not capture this interaction among different paths in a circuit. Analyzing every path, and the interactions among all paths is not feasible, because of the exponential number of such paths in a circuit. Thus, while the method of logical effort is well suited to analyze single path delays, it cannot be used directly when critical paths are not well defined, or can change. In the following section, we present an approach that can handle such scenarios.

3 Algorithm for Minimum Delay Estimation

Every gate has multiple sizes available, each of which corresponds to an input capacitance, C_{in}. We define $\mathcal{C}_{in_{\mathrm{G}}}$ to be the set of all possible values of C_{in} of a gate G. A gate may drive multiple fanouts, and the load capacitance being driven, c_l, is the input capacitances of these fanouts, combined with routing capacitances, c_r. For a gate G driving n fanouts, $\mathrm{F}_1, \mathrm{F}_2, \cdots \mathrm{F}_n$, the possible values of the load capacitance is described by the sum of c_r and the elements of the set $\mathcal{C}_{L_{\mathrm{G}}}$, defined as

$$\mathcal{C}_{L_{\mathrm{G}}} = \left\{ \sum_{j=1}^{n} c_j : \forall c_j \in \mathcal{C}_{in_{\mathrm{F}_j}}, j = 1 \cdots n \right\} \qquad (3)$$

With this terminology, we now present a dynamic programming based approach for calculating the minimum achievable delay of a circuit. The basic approach is to traverse the circuit from primary outputs to primary inputs. For each size of a gate, we are interested in minimizing the maximum delay from the input of the gate to any primary output PO. This is achieved by computing a delay-C_{in} curve, each point of which is represented by $D_{\mathrm{G}\rightarrow PO}[c_i], \forall c_i \in \mathcal{C}_{in_{\mathrm{G}}}$, defined as the maximum delay from G, with size c_i, to any PO. This value is determined by first calculating the delay of the gate (represented by $D_G[c_{in_i}][c_l]$) for each load value obtained by $c_l = c_j + c_r, c_j \in \mathcal{C}_{L_{\mathrm{G}}}$, and then adding it to the delay (corresponding to the load) from the fanout of the gate to the primary outputs.

The details of our approach, which is provably optimal for trees, are presented in the following subsections. We first present the calculation for gates with single fanouts, and then show how this calculation can be extended to gates with multiple fanouts. We use the circuit in figure 1 to illustrate the discussion.

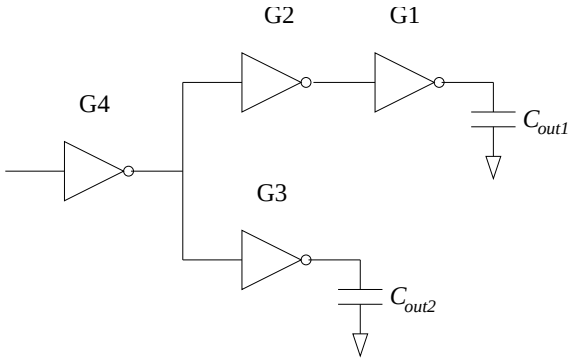

Figure 1. Delay-C_{in} Calculation and Propagation Across Multiple Fanouts

Single Fanouts

The output of a gate having a single fanout can be connected to a fixed load, such as at the primary output, or to another gate, in which case the load being driven depends on the size of the gate at the output. We present these two scenarios individually for ease of explanation; the fixed load is actually a special case of the variable load with only one load value.

1. **Fixed Load:** Consider gate[1] G1 driving a load of C_{out1} at a primary output as an example of this case, as shown in figure 1. Calculating the delay-C_{in} curve is straightforward; each possible gate size corresponds to a different value of $c_i \in \mathcal{C}_{in_{\mathrm{G1}}}$ for the gate, and the delay can be calculated using Equation (1). The delay to the primary output is the same as the gate delay in this case. Therefore,

$$\begin{aligned} D_{\mathrm{G1}\rightarrow PO}[c_i] &= D_{\mathrm{G1}}[c_i][c_l] \qquad (4) \\ &= [g \times \frac{c_l}{c_i} + \text{parasitic delay}]_{\mathrm{G1}} \\ \text{where } c_l &= C_{out1} \end{aligned}$$

   For different sizes of G1 in Figure 1, Plot I of Figure 2 presents the delay from the input of G1 to the primary output. Since the load is fixed, we obtain monotonically decreasing values of delay for increasing gate sizes.

2. **Variable Load:** Next, consider gate G2 driving gate G1. The load seen by G2 is not fixed, as in the previous case, but varies according to the size of G1. The

[1] In this discussion, all gates are shown as inverters for illustration purposes only. The method applies directly to more complex gates, with appropriate values for logical effort and parasitic delay.

delay-C_{in} curve for gate G1 has already been calculated. The delay-C_{in} curve for gate G2 is calculated in a two-step procedure. First, for a particular gate size of G2, corresponding to an input capacitance of $c_i \in \mathcal{C}_{in_{\mathrm{G2}}}$, we examine all sizes of G1, and calculate the delay as shown in equation (5). Here, the load c_l driven by gate G2 is the input capacitance of gate G1, and the parasitics of the wire connecting the output of G2 to the input of G1. Since there is a single gate connected to the output, $\mathcal{C}_{L_{\mathrm{G2}}} \equiv \mathcal{C}_{in_{\mathrm{G1}}}$.

$$D_{\mathrm{G2}}[c_i][c_l] = [g \times \frac{c_l}{c_i} + \text{parasitic delay}]_{\mathrm{G2}} \quad (5)$$

$$D_{\mathrm{G2}\rightarrow PO}[c_i] = \min_{c_j \in \mathcal{C}_{L_{\mathrm{G2}}}} \{D_{\mathrm{G2}}[c_i][c_l] + D_{\mathrm{G1}\rightarrow PO}[c_j]\} \quad (6)$$

$$\text{where } c_l = c_j + c_r \quad \forall c_j \in \mathcal{C}_{L_{\mathrm{G2}}}$$

Next, the delay from the input of G2 to the primary output is obtained by combining $D_{\mathrm{G2}}[c_i][c_l]$ with $D_{\mathrm{G1}\rightarrow PO}[c_j]$, which is the delay corresponding to the size of G1 under consideration. Thus, the minimum delay that we can obtain for the selected size of G2 is determined using Equation (6). Note that this size of G1 that minimizes $D_{\mathrm{G2}\rightarrow PO}[c_i]$ may be suboptimal when G1 is considered in isolation, i.e., it may not be the size that minimizes the delay from G1 to the primary output.

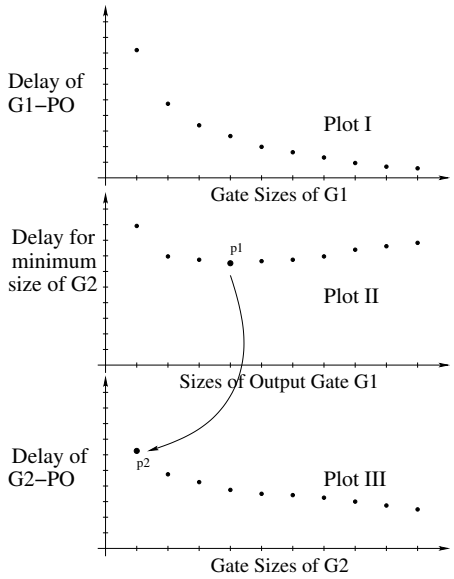

Figure 2. Delay-C_{in} Curve Propagation Across Gates G1 and G2 from figure 1

For a particular size of gate G2, the delay from G2 to the primary output varies due to two factors changing simultaneously, the load being driven by G2 (which corresponds to different sizes of gate G1) and the corresponding delays of G1. Thus, for the selected size of G2, there is a tradeoff between selecting larger sizes of G1 (which reduce the delay of G1 but slow G2 down), and smaller sizes of G1 (which have higher delays for G1 but decrease the delay of G2).

Equations (5) and (6) calculate the best solution for the selected size of gate G2. This calculation is repeated for different sizes of G2, to obtain its complete delay-C_{in} curve.

To illustrate this calculation, consider the smallest gate size of G2. For this size, Plot II in figure 2 is the delay from G2 to the output, for different sizes of G1, corresponding to different values of load capacitance for gate G2. The minimum delay value of this set is at the point labeled p1, and all other points that have higher delay can be discarded, as they are suboptimal.

For different sizes of G2, the delay to the primary output is as shown in Plot III of figure 2. Point p2 is the delay for the smallest size of G2, and is obtained from Plot II as described above. The remaining points in Plot III are obtained by repeating this calculation for all other sizes of G2.

Multiple Fanouts

The scenarios presented above are the most basic cases, with a single fanout on a gate. We now consider the case when a gate drives multiple fanouts, such as gate G4 in Figure 1. It is in our approach here that we differ the most with respect to the method of logical effort, since we take into account different values of delays and gate sizes on *each* fanout simultaneously. Depending on the sizes of the gates, either of the paths through gate G2 and gate G3 may have larger delays, and hence could be critical. Our formulation explicitly accounts for this changing dynamic.

In Figure 1, assume the delay-C_{in} curves of gates G2 and G3 have been calculated. In this case, $\mathcal{C}_{L_{\mathrm{G4}}} \equiv \mathcal{C}_{in_{\mathcal{G}\in}} \times \mathcal{C}_{in_{\mathcal{G}\ni}}$. The load being driven by gate G4, c_l, is the sum of the input capacitances of gates G2 and G3, and the routing capacitance c_r. For a particular size of G4 (and a corresponding value of input capacitance $c_i \in \mathcal{C}_{in_{\mathrm{G4}}}$), we calculate the delay of gate G4 for each value of this load as shown in Equation (7).

$$D_{\mathrm{G4}}[c_i][c_l] = \{g \times \frac{c_l}{c_i} + \text{parasitic delay}\}_{\mathrm{G4}} \quad (7)$$

$$D_{\mathrm{G4}\rightarrow PO}[c_i] = \min_{c_j, c_k} \{D_{\mathrm{G4}}[c_i] + D_{OP\rightarrow PO}\} \quad (8)$$

$$\text{where } D_{OP\rightarrow PO} = \max\{D_{\mathrm{G2}\rightarrow PO}[c_j], D_{\mathrm{G3}\rightarrow PO}[c_k]\}$$
$$\text{and } c_l = c_j + c_k + c_r \quad \forall c_j \in \mathcal{C}_{in_{\mathrm{G2}}}, c_k \in \mathcal{C}_{in_{\mathrm{G3}}}$$

Next, the delay to the primary output is calculated by combining $D_{\mathrm{G4}}[c_i]$ with the maximum of the delays of each

fanout. In this scenario, the identity of the branch with the maximum delay to a primary output can change according to which branch has higher delay. For a particular size of G4, one branch may determine the critical path, while the other may be critical for another size of G4. The formulation of Equation (8) automatically accounts for this. Thus, for the selected size of G4, we can determine the optimal size of each of its outputs, in order to obtain the minimum delay. This procedure is repeated for all sizes of G4, to compute the entire delay-C_{in} curve for G4.

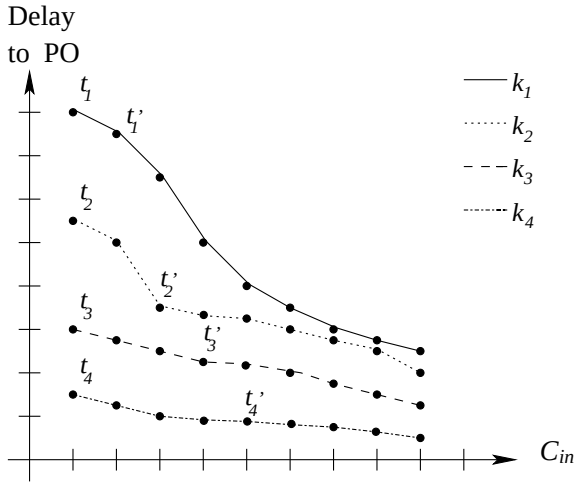

Figure 3. Combining Delay-C_{in} Curves at Multiple Fanouts

It may seem that the size of the set $\mathcal{C}_{L_G}$ for a gate G with multiple fanouts is proportional to the product of the number of sizes of the fanout gates. Assume gate G drives four outputs, whose delay-C_{in} curves are represented by k_1, k_2, k_3 and k_4, shown in Figure 3. If each fanout has m sizes, each curve has m points, and the size of $\mathcal{C}_{L_G}$ is m^4. However, we can show that most of the values in $\mathcal{C}_{L_G}$ are redundant. For example, consider the tuple $\mathcal{T}$ of the first points t_1, t_2, t_3 and t_4 from each of the curves in figure 3. A tuple $\mathcal{T}'$ of the point t_1 from curve k_1 and any other point from k_2, k_3 and k_4 (say t_2', t_3' and t_4'), is *inferior* to $\mathcal{T}$ for the following reason. There are two values that are extracted from $\mathcal{T}$ and $\mathcal{T}'$, the maximum delay to a primary output, and the sum of the input capacitances represented by these combinations, which is used as the load in the delay calculation of gate G. The maximum delay is the same in tuples $\mathcal{T}$ and $\mathcal{T}'$, but the load presented by $\mathcal{T}'$ is greater than that of $\mathcal{T}$. Hence, the delay of G (calculated using Equation (7)), and therefore the delay to a primary output (calculated using Equation (8)) is larger in this case. Since we are interested in minimizing the delay to a primary output, the solution offered by tuple $\mathcal{T}'$ will never replace that calculated using $\mathcal{T}$.

The above discussion directly leads to a strategy for efficiently selecting useful values of c_l from the delay-C_{in} curves of outputs. First, these curves are stored in order of non-increasing delay (and hence increasing sizes). The first c_l is the routing capacitance c_r plus the capacitance corresponding to the maximum-delay points from each curve, as in tuple $\mathcal{T}$. The next value is obtained by replacing the point with maximum delay (e.g., t_1 of curve k_1 in $\mathcal{T}$), with the next point from the same curve (t_1'). This effectively ignores the combination of t_1 with remaining points from the other curves. This process is continued till the maximum delay point is the last point on its curve. Thus, the total number of combinations is of the order of the *sum* of number of points on each curve, rather than the product.

Algorithm

Algorithm 1 `MDE: Minimum_Delay_Estimation`

for each gate G whose outputs have been processed **do**
  // calculate the delay-C_{in} curves for G
  for all $c_i \in \mathcal{C}_{in_G}$ **do**
    $D_{G \rightarrow PO}[c_i] = \infty$
    for every $c_j \in \mathcal{C}_{L_G}$ that is not redundant **do**
      // G has n fanouts $F_1, F_2, \cdots F_n$
      $c_l = \sum_{j=1}^{n} c_j + c_r$
      // determine the delay of gate G
      $D_G[c_i][c_l] = [g \times \frac{c_l}{c_i} + \text{parasitic delay}]_G$
      // determine maximum delay from any fanout F
      // to any PO, using the delay-C_{in} curves of F
      $temp = D_G[c_i][c_l] + \max_{j=1 \cdots n}(D_{F_j \rightarrow PO}[c_j])$
      $D_{G \rightarrow PO}[c_i] = \min(temp, D_{G \rightarrow PO}[c_i])$
    end for
  end for
end for
Minimum Delay $= \max\{\min_{\text{all PI's}}\{\text{delay to PO}\}\}$

Algorithm 1, `Minimum_Delay_Estimation (MDE)`, presents our algorithm for estimating the minimum achievable delay of a circuit. The calculation is based on the delay-C_{in} curve computation presented in the previous subsection. All gates are processed in topological order, from POs to PIs. At each primary input, the data point corresponding to the minimum delay is selected, the maximum over all PIs is the desired minimum achievable delay.

Assume that there are m sizes for each gate in a circuit with N gates, and the maximum fanout on any gate is $|FO|$. The innermost `for` loop is executed $O(m \times |FO|)$ times, as shown previously, and the cost of determining the maximum delay point is $O(|FO|)$. The second `for` loop is executed m times, since we assume m sizes for each gate. Finally, since there are N gates in the circuit, the outermost `for` loop is executed N times. Thus, the running time of algorithm `MDE` is $O(N \cdot m \cdot m \cdot |FO| \cdot |FO|)$. However, note that this is a very loose upper bound, since very few gates actually have $|FO|$ fanouts.

Algorithm MDE is optimal for trees. However, most circuits are DAGs, with reconvergent fanouts. The main problem with DAGs is that there are multiple paths from a particular gate to primary outputs, or between two gates. An implicit assumption of our algorithm is that the delay-C_{in} curves at multiple fanout points are independent, and that we are free to choose the combination of output delays and capacitances that best suit the current gate. However, with reconvergent fanouts, these choices are not independent of each other. Selecting a data point on one output restricts the choices on the other, and determining the relation between different outputs is intractable for general circuits. However, assuming independence is not unreasonable. If the reconvergent paths are completely unbalanced, i.e., their structure and logic is such that one always has smaller delay than the other, no errors are introduced due to the manner in which their delay-C_{in} curves are combined. The smallest C_{in} value will consistently be selected for the path with smaller delay. An example of this situation is if the paths correspond to curves k_1 and k_4 in figure 3. On the other hand, if the delays of the two paths are roughly of the same order (e.g., if they correspond to curves k_1 and k_2), our approach selects approximately similar values of input capacitances. This may lead to small inaccuracies, since the actual values of input capacitance may be slightly different. However, the error in delay estimation is limited, as shown by the results.

Our approach can also be used to obtain actual sizes of all gates in the circuit. In Algorithm MDE, we can store the value of the load of each output that induces the minimum delay. This information can be used in a forward traversal of the circuit, in order to generate sizes for every gate. A gate with multiple fanins has multiple choices for its size, which can be resolved by selecting the size imposed by the critical input. The effect on the non-critical inputs is that they now have a load different from what was initially assumed. However, the difference in the delays from the primary inputs to the critical and non-critical inputs can be used to compensate for this. In fact, this difference can be usually be used to *reduce* the sizes of the transitive fanin cone of the non-critical inputs, as long as their delay does not become larger than that of the critical input. Gate sizes determined in this manner correspond to a circuit sized for minimum delay. However, these sizes can also be used as an initial feasible solution for an exact sizing tool, instead of using the original unsized circuit. This can lead to a large improvement in running times of the transistor sizing tool, since a circuit sized using our approach is closer to the final solution than the initial, unsized circuit.

4 Results

In order to validate our algorithm, we generated multiple implementations of the ISCAS combinational benchmark circuits using SIS [15], and a technology library consisting of minimum sized inverter and two-input NAND, NOR and XOR gates. This choice of gates was selected simply because they have been calibrated in order to obtain accurate values of logical effort and parasitic delay with respect to the models used in our implementation of TILOS[2]. Each benchmark circuit was mapped using different scripts and options, and randomly generated wire parasitics were added to each mapped circuit, in order to simulate the effect of placement and routing considerations. Finally, our implementation of TILOS was used to determine the minimum delay that sizing could realize. This minimum delay was compared with the estimates calculated by Algorithm MDE.

Figure 4 presents the comparison of Algorithm MDE with TILOS. For each implementation, the first bar represents the delay of the unsized circuit. The second bar is the minimum delay obtained when the mapped circuit is sized using TILOS, and the last bar is the minimum achievable delay estimated using Algorithm MDE. As can be seen by the correspondence between the last two bars for each implementation, our results agree with those obtained via TILOS. In every case, the execution time for our algorithm was less than a second, while our implementation of TILOS took from a few seconds for C17 up to more than 1500 seconds for C6288[3]. The average error for each circuit over all implementations is presented in table 1.

Another interesting point to note is that comparisons based on unsized circuit delays can be misleading. Consider implementations 9 and 14 of C5315. The unsized circuit delay of implementation 9 is larger than that of implementation 14, whereas the sized circuit delay of implementation 14 is greater than that of 9. A naive approach to evaluating implementations would have chosen implementation 14, and would have foregone the superior solution.

5 Conclusion and Future Directions

In this paper, we have presented an algorithm that quickly and accurately estimates the performance improvement that can be obtained in a circuit via transistor sizing. Current placement tools try to provide a solution that is delay-optimal, among other objectives. However, they ignore the gains that may be obtained via sizing. Our approach can be used to guide the placement tool, in effect making it "transistor-sizing aware," so that the final solution is globally optimal.

References

[1] J. P. Fishburn and A. E. Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pages 326–328, November 1985.

[2] S. S. Sapatnekar *et al*. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimiza-

[2] [8] describes how these values can be obtained from the reference model

[3] This version does not use incremental timing analysis

Table 1. Percentage Error of `MDE` w.r.t TILOS

Circuit	C1908	C2670	C3540	C5315	C6288	C7552
Error	5.50%	4.53%	4.79%	3.76%	3.42%	4.45%

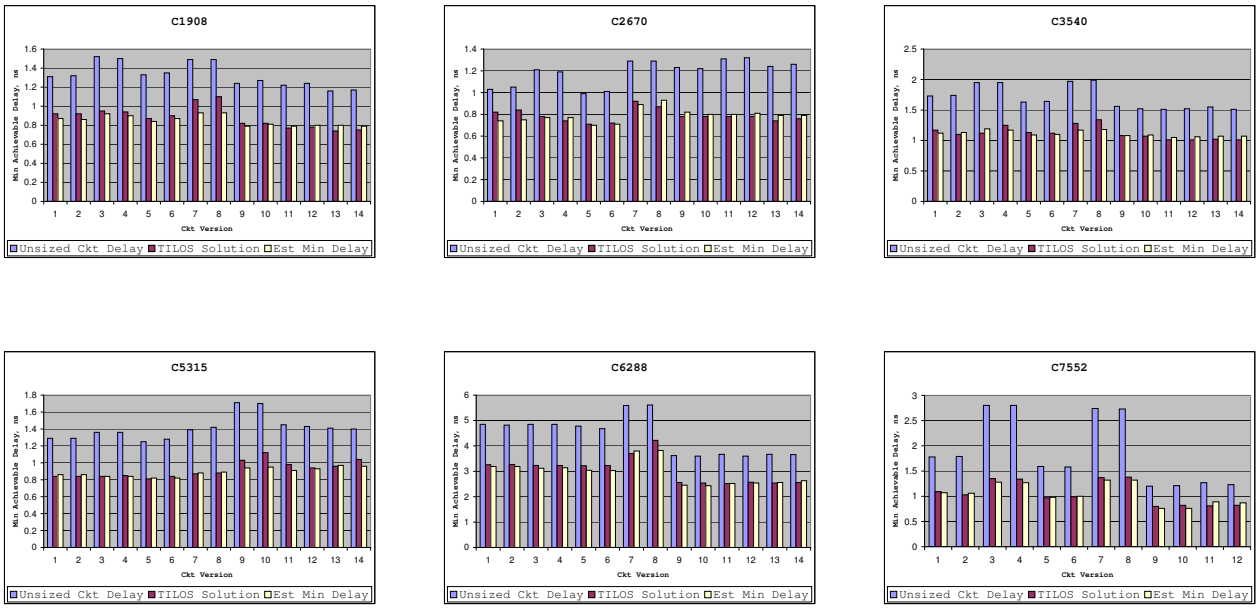

Figure 4. Results for Selected ISCAS Benchmark Circuits

tion. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(11):1621–1634, November 1993.

[3] C.-P. Chen, C. C. N. Chu, and M. D. F. Wong. Fast and Exact Simultaneous Gate and Wire Sizing by Lagragian Relaxation. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pages 617–624, November 1998.

[4] V. Sundararajan, S. S. Sapatnekar, and K. K. Parhi. Fast and Exact Transistor Sizing Based on Iterative Relaxation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(5):568–581, May 2002.

[5] A. R. Conn *et al*. JiffyTune: Circuit Optimization Using Time-Domain Sensitivities. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1292–1309, December 1998.

[6] X. Bai *et al*. Uncertainty-aware circuit optimization. In *Proceedings of the IEEE/ACM Design Automation Conference*, pages 58–63, June 2002.

[7] R. F. Sproull and I. E. Sutherland. Theory of Logical Effort: Designing for Speed on the Back of an Envelope. In *IEEE Advanced Research in VLSI*, 1991.

[8] I. Sutherland, B. Sproull, and D. Harris. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann, San Fransisco, CA, 1999.

[9] F. Beeftink *et al*. Gate-Size Selection for Standard Cell Libraries. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pages 545–550, November 1998.

[10] L. Stok, M. A. Iyer, and A. J. Sullivan. Wavefront Technology Mapping. In *Proceedings of the Design, Automation and Test in Europe Conference*, pages 531–536, March 1999.

[11] W. Donath *et al*. Transformational Placement and Synthesis. In *Proceedings of the Design, Automation and Test in Europe Conference*, pages 194–201, March 2000.

[12] K. Sulimma *et al*. Improving Placement Under the Constant Delay Model. In *Proceedings of the Design, Automation and Test in Europe Conference*, pages 677–682, March 2002.

[13] L. Stok *et al*. BooleDozer: Logic Synthesis for ASICs. *IBM Journal of Research and Development*, 40(4):407–430, July 1996.

[14] Gain Based Synthesis: Speeding RTL to Silicon. `http://www.magma-da.com/articles/Magma_GBS_White_Paper.pdf`. White Paper.

[15] Ellen M. Sentovich *et al*. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, Electronics Research Laboratory, Department of Electrical Engineering and Computer Science, University of California, Berkeley, May 1992.