Fast Estimation of Area-Delay Trade-offs in Circuit Sizing

Abstract

Sizing a circuit can improve performance drastically. However, this is a time consuming transform, and it is therefore difficult to compare different implementations of a circuit in terms of the cost overhead required for a particular delay target. This paper presents a fast estimator of the complete area-delay trade-off curve of a given circuit, allowing a designer to choose the most appropriate implementation for a given delay. We observe excellent fidelity with the actual area-delay curves (98.94% correct comparisons), with an average error of 5.76% in the area differences predicted.

1 Introduction

After a circuit has been placed and routed, it can be sized in order to improve performance, incurring cost overheads, which could be area or power. A number of approaches have been developed for transistor sizing, both in academia [1, 2, 3, 4] and in industry [5, 6]. However, a common drawback of these algorithms is their running time – sizing a reasonably large circuit can take up to a few hours. If a designer is presented with a number of implementations of the same functionality, he would therefore prefer selecting only the best for further sizing. This leads to the question of which is the best implementation, i.e., which implementation will incur the lowest cost, when sized to meet a particular delay. For convenience, we use the *area* of the implementation as a measure of the cost. There is a direct correlation of area with other measures of cost, such as power dissipation, sub-threshold leakage and gate leakage, and a similar approach can be used when the cost function is power, or a weighted combination of area and power. Answering the question of determining the lowest cost (area) implementation requires knowledge of the area-delay curve of each implementation, but these are determined only after sizing has been carried out. In this paper, we present an approach that *estimates* the area-delay curve of a given implementation. We do not use a sizing tool, and therefore, our approach is fast, and as we will show, the estimated curve has high fidelity with the actual area-delay curve.

In [7], the authors presented an approach based on the method of logical effort [8, 9], for determining the minimum achievable delay of an implementation, if transistor sizing were to be applied to it. While this is a useful metric to have, it is not sufficient for comparing circuits that will be sized to arbitrary (non-minimum) delay points. This drawback is illustrated in the following section, where we present the importance of determining the entire area-delay curve of an implementation. We then show how the area-delay curve can be estimated by using the information stored in the minimum achievable delay calculation. Finally, we apply the approach presented in this paper to comparing different implementations of the same benchmark circuits, and show how accurate comparisons can be made quickly.

2 Problem Formulation

Figure 1(a) shows the area-delay curves of multiple implementations of benchmark circuit C7552. Each implementation was obtained by varying parameters given to the optimization and synthesis tool. The area-delay curves were obtained using our implementation of TILOS [1]. In these plots, the area of an implementation is shown on the y-axis, and delay on the x-axis. The extreme right point of each curve corresponds to the unsized circuit; this has maximum delay and the smallest area, and successively smaller delay values require larger areas. Note that the curves have a characteristic point (called the 'knee'), at which the rate of change of area with respect to delay changes drastically.

Each curve is bounded by the maximum delay (i.e., the unsized circuit delay) and the minimum achievable delay. However, as can be seen, the *shape* of each curve can vary significantly. For example, in the curves shown in Figure 1(a), the knee of each curve can either be closer to one of the end points or in the center. This property varies between different circuits, as can be expected, but it also varies between implementations of the same circuit. For implementations I_1 and I_2 of C7552, the knee is closer to the minimum delay point. Hence, we initially observe large improvements in delay for relatively small area cost, for these implementations, but further delay improvement comes at the cost of large increases in area. The situation is reversed for implementations I_5 and I_6, where the knee is closer to the maximum delay point. In this scenario, trying to determine which implementation is the best at some intermediate delay point without having knowledge of the entire area-delay curve is difficult.

Suppose a designer wants to determine the best implementation among those available for some target delay of D_1. Calculating the minimum achievable delay and the unsized circuit delay of all implementations, the designer can determine that implementations I_1, I_2, I_3 and I_4 meet this target delay. At a different target delay of D_2, the implementations that have to be considered are I_3, I_4, I_5 and I_6. Implementations I_1 and I_2 need not be considered, since their minimum achievable delay is larger than this value. However, this information is not sufficient, since *which* of these circuits should be selected is still not known. Ideally, he would like an ordering of these implementations based on the cost, which in this case, is the area. The required ordering for a delay of D_1 is $\{I_3,\ I_4,\ I_2,\ I_1\}$, and for D_2 it is $\{I_4,\ I_3,\ I_5,\ I_6\}$. Simply ranking implementations based on the unsized delays and areas is not enough, e.g., at one delay point, I_4 has lower area, and at the other I_3 is better. This situation, of different implementations being the best at different delay points, is also seen in implementations of other benchmark circuits.

Recall that using a sizing tool to obtain the area-delay curves of *one* implementation of a circuit is time-consuming. Obtaining the area-delay curves of multiple implementations is prohibitively

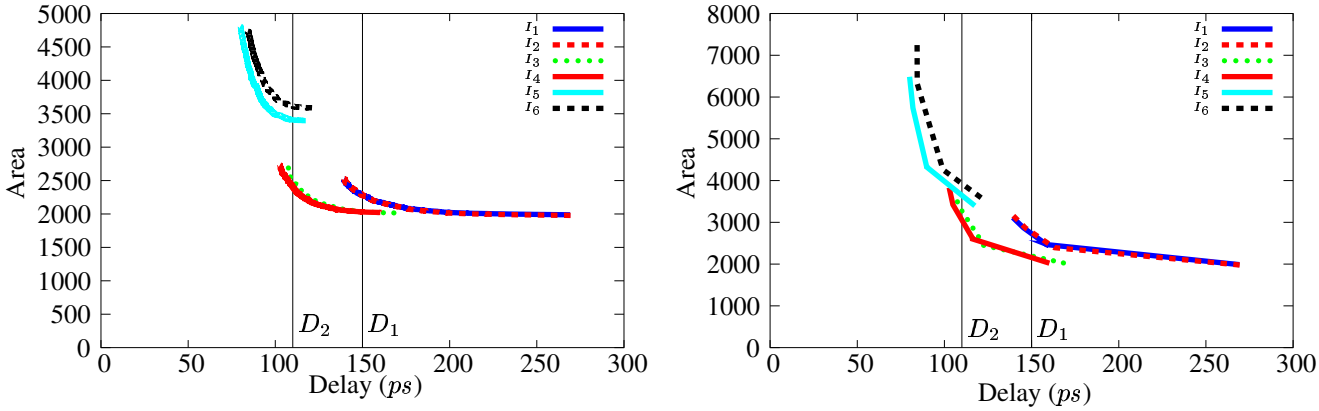

Figure 1: Area-Delay Curves of 6 Implementations of Benchmark Circuit C7552 (a) Generated using TILOS and (b) Estimated

expensive. Our heuristic, presented in the following section, addresses this issue by *estimating* the area-delay curve of a given implementation. These curves can be used to compare different implementations in two ways. First, given a target delay, we can generate a cost-based ordering based on the estimated area-delay curve of each implementation. Second, instead of calculating the actual areas at the delay value of interest, we measure the relative area difference between the implementations. The relative area difference has a good correlation with the actual area difference, and can be estimated quickly.

The area-delay curves obtained using our approach are as shown in Figure 1(b). A rough comparison with the plots of Figure 1(a) shows that this heuristic captures the behavior of the area-delay curves well. In particular, the shape of the estimated curve, with respect to the position of the knee matches that seen in the actual area-delay curves. A complete comparison is made in section 4.

3 Area-Delay Curve Estimation

Algorithm 1 is a listing of Algorithm `MDE`, presented in [7]. It estimates the minimum achievable delay of a circuit, by calculating the Delay-C_{in} curve for each gate of the circuit. This curve stores the best possible delay from the input of a gate to any primary output, for each input capacitance value, corresponding to the gate size. It also implicitly stores the sizes of gates on the path to the primary output, which achieve this delay. Determining the Delay-C_{in} curve of a gate that has a single fanout is relatively straightforward, since using dynamic programming, the Delay-C_{in} curve of the immediate output is the only one that has to be considered. For gates with multiple fanouts, all points on the Delay-C_{in} curves of each fanout have to be considered, and the the number of possible combinations of these points can be extremely large. However, Delay-C_{in} curves of multiple fanouts can be combined in a manner that leads to tractable runtimes without any loss of information. The Delay-C_{in} curve of a gate thus captures the delay characteristics of the entire transitive fanout cone of the gate in a compact and elegant formulation. Once the Delay-C_{in} curves at the primary inputs have been calculated, the minimum achievable delay of the circuit can be determined by selecting the minimum delay point from these curves. The gate sizes associated with the selected point can be propagated to the primary outputs, and adding these sizes gives us an estimate of the area required to meet the selected (minimum) delay.

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 $c_i \in \mathcal{C}_{in_{\mathrm{G}}}$ **do**
    $D_{\mathrm{G}\rightarrow PO}[c_i] = \infty$
    for every $c_j \in \mathcal{C}_{L_{\mathrm{G}}}$ that is not redundant **do**
      // G has n fanouts $\mathrm{F}_1, \mathrm{F}_2, \cdots \mathrm{F}_n$
      $c_l = \sum_{j=1}^{n} c_j + c_r$
      // determine the delay of gate G
      $D_{\mathrm{G}}[c_i][c_l] = [g \times \frac{c_l}{c_i} + \text{parasitic delay}]_{\mathrm{G}}$
      // determine maximum delay from any fanout F
      // to any PO, using the delay-C_{in} curves of F
      $temp = D_{\mathrm{G}}[c_i][c_l] + \max_{j=1\cdots n}(D_{\mathrm{F}_j\rightarrow PO}[c_j])$
      $D_{\mathrm{G}\rightarrow PO}[c_i] = \min(temp, D_{\mathrm{G}\rightarrow PO}[c_i])$
    end for
  end for
end for
Minimum Delay $= \max\{\min_{\text{all PI's}}\{\text{delay to PO}\}\}$

Since we are interested in determining the area-delay curve of the implementation, the obvious approach is to calculate the area with the delays during the Delay-C_{in} calculation. However, there are a few problems with this approach. There are multiple configurations of gate sizes that can achieve the same delay value, and hence multiple solutions for each delay value have to be stored. These enhanced Delay-C_{in} curves do not have the optimal substructure property, and hence we can no longer use dynamic programming. Finally, every combination of points in the enhanced Delay-C_{in} curves of multiple fanouts has to be considered, which further increases the complexity.

We therefore need another approach to estimating the area-delay curve. Recall that the Delay-C_{in} curves calculated in Algorithm `MDE` implicitly store sizes of gates in the transitive fanout cone required for for achieving the minimum delay for each value of C_{in}. Hence, we can size the circuit using points on the Delay-

C_{in} curves of the primary inputs, and calculate the corresponding area. However, these points may not be optimal i.e., the area calculated using the above approach may not be the smallest area for a particular delay. For example, say we have a minimum delay of d_1 for C_{in_1} and d_2 for C_{in_2}, with corresponding circuit areas of a_1 and a_2, and $d_1 > d_2$. It is possible that there was a non-minimum delay $d_2' = d_1$ for an input capacitance of C_{in_2} that had a corresponding circuit area a_2', that is less than a_1. The solution (a_2', d_2') is clearly better than the (a_1, d_1) solution, but since only minimum delay points are considered, the superior solution is hidden.

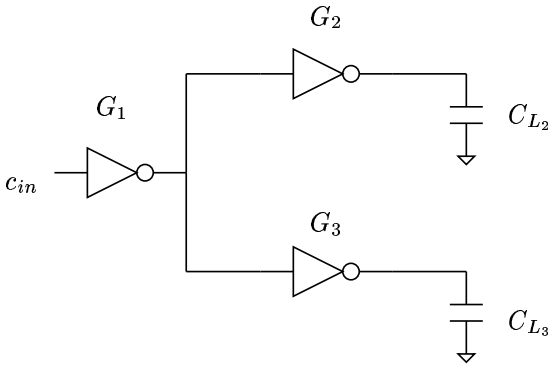

Figure 2: Example Circuit

Consider the circuit shown in Figure 2, with two branches of the circuit driving different loads [1]. For some input capacitance of c_{in}, we obtain a number of delay values, the minimum of which is stored in the Delay-C_{in} curve, and the other delay values are discarded. However, we can size the circuit using the minimum as well as the discarded delay values (this is for the same input capacitance of c_{in}), and calculate the corresponding areas. These points are shown in Figure 3, and the best points for an area-delay curve perspective are the ones marked by a line. This procedure can be repeated for other values of C_{in}, and the union of the solutions obtained gives us the area-delay curve desired. This is shown in Figure 4 for three values of C_{in}. Note the intersection in the curves corresponding to $c_{in} = 4$ and $c_{in} = 5$, this is an example of sub-optimality if only the minimum delay points were to be considered.

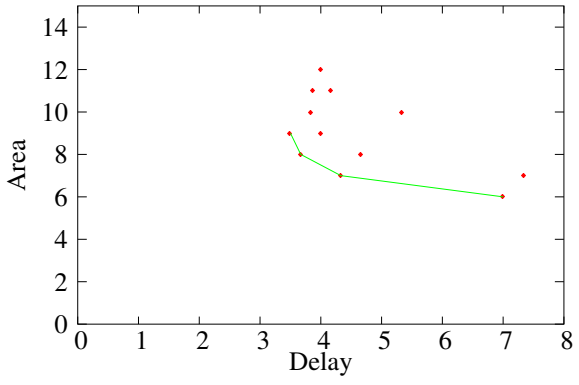

Figure 3: Calculating the Area-Delay Curve for one value of c_{in}

Thus, we estimate the area-delay curve of a circuit by sizing it for different values of delay, for every value of C_{in} and measuring the

[1] We use inverters for simplicity of presentation; this discussion extends to other gates as well.

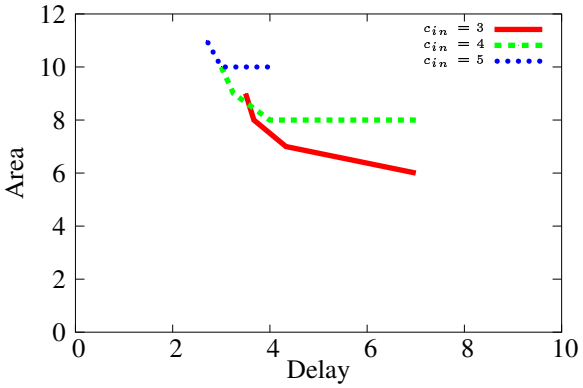

Figure 4: Area-Delay Curve of the Circuit in Figure 2

area. In order to keep the run time low, rather than sizing for all delay values, we size the circuit for a limited number of values (in our experiments, we found that selecting 10 sub-optimal delay points was sufficient). This has an impact on the accuracy of our results, but the effect is limited, especially since our focus is on comparing implementations, rather than on determining actual areas.

Our heuristic, called Algorithm `ADC` (shown in Algorithm 2) is obtained by modifying Algorithm `MDE` as follows. At the primary inputs, we store *sets* of Delay-C_{in} curves. Each time $D_{G \to PO}[c_i]$ is updated to a new value, we store the replaced value as an entry in secondary curves. The minimum delay value from these secondary curves are then used to size the circuit, and obtain other points on the delay-area curve. Circuits sized in this manner have greater delay than the minimum achievable delay, and after area recovery, they have smaller area as well.

The solution obtained using this approach is naturally not exact. However, as discussed above, since the auxiliary data of points on the secondary curve encode sizes of the outputs (and particularly, of sizes of multiple fanouts), these solutions still provide a good representation of the area behavior of the circuit at different delay points. i.e., though we cannot use the area-delay curves to make absolute judgments, we can still make comparative judgments between different circuits.

Once the circuit has been sized, we determine the arrival and required times at each gate, and use the slack to reduce the sizes of the gates. This step can drastically reduce the area of a circuit, since the non-critical parts of the circuit are usually sized to be unnecessarily fast.

Algorithm 2 is almost as fast as Algorithm 1. Once the Delay-C_{in} curves have been calculated, the actual calculation of arrival and required times only needs two traversals of the circuit, and sizing each gate requires a maximum of $O(m)$ operations, if there are m gate sizes available. This is done a fixed number of times, for each set of Delay-C_{in} curves that have been calculated. Thus, the running time is dominated by that of running Algorithm 1.

4 Results

In order to validate our algorithm, we generated multiple implementations of the ISCAS combinational benchmark circuits using SIS [10], and a technology library consisting of minimum sized inverter and two-input NAND, NOR and XOR gates. These gates are calibrated to obtain accurate values of logical effort and parasitic

Algorithm 2 `ADC: Area-Delay Curve Calculation`

for each gate G whose outputs have been processed **do**
  if G is not a PI **then**
    Calculate Delay-C_{in} curve of G as in Algorithm 1
  else
    Calculate Delay-C_{in} curve as before, but for each c_i store *all* solutions
  end if
end for

for each set of Delay-C_{in} curves of the PIs **do**
  Minimum Delay $= \max\{\min_{\text{all PI's}}\{\text{delay to PO}\}\}$
  // forward traversal
  Size the circuit based on the selected point. Also determine the arrival time at each gate
  // reverse traversal
  Determine the required time at each gate
  // area recovery
  for each gate G in reverse topological order **do**
    $slack =$ arrival time $-$ required time
    while $slack > 0$ **do**
      reduce the size of G
      update the arrival and required times of G and its inputs
    end while
  end for
  Determine area and delay of the sized circuit
end for

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 actual area-delay curves, against which the estimated curves obtained by Algorithm `ADC` can be benchmarked.

The first goal of our approach is to correctly predict which implementation is the best for different delay points. Our methodology for measuring the effectiveness of Algorithm `ADC` is as follows. For the entire range of possible delay values, we select ten equally spaced delay points. Note that the number of implementations that can be sized to meet a particular delay value varies by circuit. We make pairwise comparisons between all implementations available at the selected delay point, and determine which implementation is better. In Table 1, for each benchmark circuit, the number of comparisons made are shown in the second column. Next, we make the same comparison using the delay curves obtained from our implementation of TILOS. An incorrect comparison is when the ranking according to Algorithm `ADC` is different from that obtained from TILOS. As shown in the next column, incorrect comparisons occur only 1.86% of the time.

Next, we measure the error in the predicted area difference. Let implementations I_1 and I_2 have estimated areas of $A_{I_1}\text{est}$ and $A_{I_2}\text{est}$, and assume $A_{I_1}\text{est} < A_{I_2}\text{est}$, so that I_1 is the better implementation. The difference between the estimated areas of I_1 and I_2, is calculated as $\Delta A_{\text{est}} = 100(1-\frac{A_{I_1}\text{est}}{A_{I_2}\text{est}})$. Similarly, the difference between the areas from the actual area-delay curves, $A_{I_1}\text{act}$ and $A_{I_2}\text{act}$ is calculated as $\Delta A_{\text{act}} = 100(1-\frac{A_{I_1}\text{act}}{A_{I_2}\text{act}})$. The absolute error of our approach is $E = |\Delta A_{\text{est}} - \Delta A_{\text{act}}|$, and the maximum average value of this error over all comparisons are presented in columns 4 and 5 of Table 1. The maximum error is high, but it does not happen often, and over all circuits, the average error is 5.76%. The last two columns present the maximum and average errors in area estimation for comparisons that were mis-predicted. Once again, while the maximum is large, it is rare, and the average error in this case is 5.75%.

[2] [9] describes how these values can be obtained from the reference model.

Table 1: Full ADC comparison

Circuit	Comparisons		E_{Total}		E_{False}	
	Total	False	max.(%)	avg.(%)	max.(%)	avg.(%)
C432	102	3	21.85	6.42	11.33	9.51
C499	158	1	21.52	6.38	16.08	7.83
C880	41	3	13.95	4.11	9.89	6.93
C1355	136	1	28.61	8.17	18.11	8.02
C1908	113	4	25.72	5.62	5.22	4.00
C2670	121	1	18.87	3.49	3.28	3.28
C3540	101	5	18.24	4.49	14.51	8.12
C5315	163	8	27.51	7.24	5.09	2.34
C6288	57	1	22.50	4.65	11.62	4.85
C7552	30	2	25.08	7.02	4.70	2.68
Total	1022	29(1.86%)				
Max.			28.61		18.11	
Avg.				5.76		5.75

References

[1] J. P. Fishburn and A. E. Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing. In *Proc. ICCAD*, pages 326–328, 1985.

[2] S. S. Sapatnekar *et al*. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization. *IEEE Trans. on CAD*, 12(11):1621–1634, Nov 1993.

[3] C.-P. Chen *et al*. Fast and Exact Simultaneous Gate and Wire Sizing by Lagragian Relaxation. In *Proc. ICCAD*, pages 617–624, 1998.

[4] V. Sundararajan *et al*. Fast and Exact Transistor Sizing Based on Iterative Relaxation. *IEEE Trans. on CAD*, 21(5):568–581, May 2002.

[5] A. R. Conn *et al*. JiffyTune: Circuit Optimization Using Time-Domain Sensitivities. *IEEE Trans. on CAD*, 17(12):1292–1309, Dec 1998.

[6] X. Bai *et al*. Uncertainty-aware circuit optimization. In *Proc. DAC*, pages 58–63, 2002.

[7] S. K. Karandikar and S. S. Sapatnekar. Fast Comparisons of Circuit Implementations. In *Proc. DATE*, pages 910–915, 2004.

[8] 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.

[9] I. Sutherland *et al*. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann, San Fransisco, CA, 1999.

[10] E. M. Sentovich *et al*. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, Electronics Research Laboratory, Department of EECS, University of California, Berkeley, May 1992.