Interleaving Buffer Insertion and Transistor Sizing into a Single Optimization *

Yanbin Jiang, Sachin S. Sapatnekar, Cyrus Bamji and Juho Kim

August 12, 1997

Abstract

Buffer insertion is a technique that is used either to increase the driving power of a path in a circuit, or to isolate large capacitive loads that lie on noncritical or less critical paths. Gate sizing sets the sizes of gates within a circuit to achieve a given timing specification. Traditional design techniques perform gate sizing and buffer insertion as two separate and independent steps during synthesis. However, until sizing is performed, any information on capacitive loads is incomplete and therefore a buffer insertion algorithm must operate with incomplete information, leading to suboptimal results. Moreover, the insertion of buffers can change the structure of the circuit sufficiently so that it may lead to a different sizing solution from the unbuffered circuit. Therefore, these techniques of buffer insertion and sizing are intimately linked and it makes a lot of sense to integrate them into a single optimization.

This work presents strategies to insert buffers in a circuit, combined with gate sizing, to achieve better power-delay and area-delay tradeoffs. The purpose of this work is to examine how combining sizing algorithm with buffer insertion will help us achieve better area-delay or power-delay tradeoffs, and to determine where and when to insert buffers in a circuit. The delay model incorporates placement-based information and the effect of input slew rates on gate delays. The results obtained by using the new method are significantly better than the results

*This work was supported in part by a grant from Cadence Design Systems, Inc.

Yanbin Jiang is with the Department of Electrical and Computer Engineering at Iowa State University, Ames, IA 50011.

Sachin S. Sapatnekar was with the Department of Electrical and Computer Engineering at Iowa State University, Ames, IA 50011. He is now with the Department of Electrical and Computer Engineering at the University of Minnesota, Minneapolis, MN 55455.

Cyrus Bamji is with Cadence Design Systems, San Jose, CA 95134.

Juho Kim was with Cadence Design Systems, San Jose, CA 95134. He is now with Sogang University, Seoul, South Korea.

given by merely using a TILOS-like gate sizing algorithm alone, as is illustrated by several area-delay tradeoff curves shown in this paper.

1 Introduction

While a combinational CMOS circuit with minimum-sized transistors has a small area, its delay may not be acceptable. It is often possible to reduce the delay of such a circuit at the expense of increased area by increasing the sizes of certain transistors in the circuit. The well-studied optimization problem that deals with this area-delay tradeoff is known as the sizing problem [1, 2, 3] and it is often formulated as

$$\begin{aligned} \text{minimize} \quad & Area \\ \text{subject to} \quad & Delay \leq T_{spec} \end{aligned} \tag{1}$$

In some formulations, it is the power that is minimized instead of the area. For edge-triggered circuits, we need consider only one combinational subcircuit at a time, minimizing its area while meeting the timing requirements that state that the delay of each combinational segment should satisfy the clock period. Therefore, the problem of sizing even a very large circuit can be decomposed into individual problems of sizing individual combinational blocks to meet the clock period, and the problem complexity is considerably reduced. For the remainder of this paper, we will therefore assume that the circuit is purely combinational.

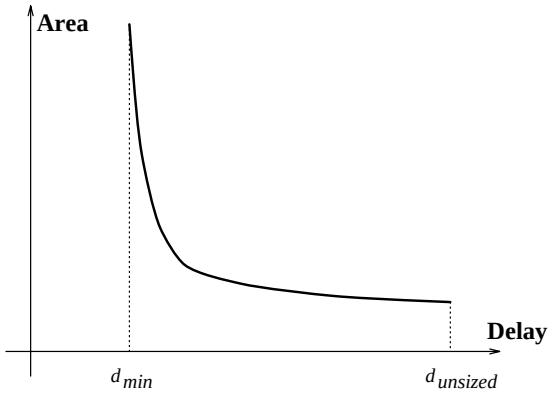

Figure 1: Area-delay curve for sizing.

For a given combinational circuit, the nature of the area-delay tradeoff curve for gate sizing

is as shown in Figure 1. Typically, a small amount of sizing is adequate to reduce the delay corresponding to the unsized circuit, $d_{unsized}$, in order to meet a loose delay specification. However, as the specification is tightened, the circuit has to be sized by greater degrees, until we reach the knee of the curve where it must be sized tremendously to achieve further delay reduction. Further, it is impossible to reduce the delay of a circuit indefinitely through sizing, and there is a minimum achievable delay, d_{min}, that cannot be bettered through sizing.

Note that gate sizing does not change the topology of the circuit, but merely changes the sizes of individual transistors within gates. We note that some gates in a circuit can be sized excessively because of the large loads that they drive. The appropriate insertion of buffers in a circuit can be used to prevent excessive sizing while meeting delay specifications. In fact, as we will see, buffer insertion in conjunction with sizing often permits greater circuit delay reductions than sizing alone.

Traditionally, gate sizing and buffer insertion (the "fanout problem") [4, 5] have been carried out separately and at different stages of the design process[1]. However, as sizing changes the capacitances driven by various gates, the locations of high-capacitance nodes are accurately established only during sizing, and any optimizations performed before sizing are necessarily based only on educated guesses. Therefore, it is useful to combine the two optimizations into a single step, and this is the objective of this research.

In this paper, we first present the delay model used here, and then list the situations in which it is advantageous to insert buffers. Next, we present an algorithm to combine sizing with buffer insertion, and show that the application of these two transformations in unison can provide significant benefits.

2 Delay and area modeling

2.1 Transistor level modeling

We first show how an n-transistor of width $w_{n,i}$ is modeled by a set of capacitances and resistors. A p-transistor of width $w_{p,i}$ is similarly modeled. Since all the transistors are set to minimum length, the capacitances can be modeled in terms of only the transistor widths. For an n-transistor, we can write the source/drain capacitance $C_{sdn_i} = C_{d,n_1} \cdot w_{n,i} + C_{d,n_2}$, and the gate capacitance as $C_{gn_i} = C_{g,n_1} \cdot w_{n,i} + C_{g,n_2}$, where $C_{d,n_1}, C_{d,n_2}, C_{g,n_1}$ and C_{g,n_2} are constants. The on-resistance, $R_{i,n}$, of an n transistor is given by $R_{i_n} = \frac{R_n}{w_{n,i}}$. As in previous work (for example, [1, 2]), the circuit

[1] The fanout problem, however, only tackles what we will later refer to as Type B buffer insertion.

area is modeled as the sum of all transistor sizes.

At the gate level, each gate G_i is modeled by an equivalent inverter, parameterized with all n-($p-$) transistor sizes set to $w_{n,i}$ ($w_{p,i}$). In this implementation, only static CMOS gates are considered. All transistors of the same type in a gate are assumed to have a uniform size. The ideas presented in this work are also applicable to the case where every transistor is allowed to have a different size. The pull-up (pull-down) structure is represented by an equivalent inverter with a p-transistor (n-transistor) size of $S_{p,i}$ ($S_{n,i}$) that corresponds to the worst-case situation; this number is referred to as the gate size. The relation between the gate sizes in the equivalent inverter and transistor widths in the gate can easily be computed for various type of gates. For example, for a k-input NAND gate, $S_{n,i} = w_{n,i}/k$, $S_{p,i} = w_{p,i}$ [2].

The capacitance loading, C_L, of gate G_i can be calculated from the transistor sizes of its fanouts as follows:

$$C_L = \sum_{j \in fanout_i} (C_{gn_j} + C_{gp_j}) + C_{intrinsic} + C_{wire} \tag{2}$$

where $C_{intrinsic}$ corresponds to the source and drain capacitance connected to the output node of G_i. The wire capacitance values are based on the placement.

2.2 Delay computations

We first demonstrate the calculation of the step delay, i.e., the delay under the assumption that the input to each gate is a step transition with zero transition time. Next, we will show how this assumption is relaxed to allow for the realistic case where nonzero transition times are possible.

The Elmore fall step delay, t_{f_i}, of gate G_i can then be obtained from C_L and $S_{n,i}$ as [6, 7]

$$t_{f_i,\text{step}} = \frac{R_n \cdot C_L}{S_{n,i}}. \tag{3}$$

The rise delay is similarly obtained as $t_{r_i,\text{step}} = \frac{R_p \cdot C_L}{S_{p,i}}$.

To allow for the effect of nonstep input transitions, we use the inverter delay model presented in [8]. The effect of the input-to-output coupling capacitance and input slope effects are considered in this model. Consider the CMOS inverter structure driving a load C_L, as shown in Figure 2, where C_M is the coupling capacitance between the input and the output nodes. When the applied input

[2] Notice that $S_{p,i}$ is $w_{p,i}$ (and not $k \cdot w_{p,i}$) since in the worst case, only one of the k transistors in parallel will be on.

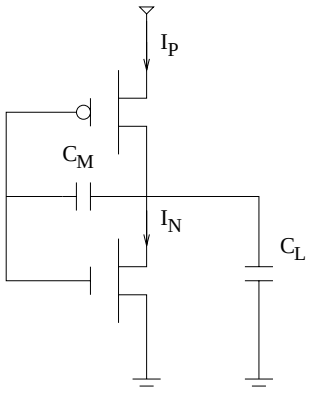

Figure 2: CMOS inverter structure

is the ramp

$$V_{in} = \begin{cases} 0 & t \leq 0 \\ \frac{V_{DD}}{\tau} t & 0 \leq t \leq \tau \\ V_{DD} & t \geq \tau \end{cases} , \tag{4}$$

where τ is the slope of the input ramp, the delay is given by

$$t_{f_i,\text{ramp}} = v_{TN} \cdot \frac{\tau}{2} + \left(1 + 2\frac{C_M}{C_L}\right) t_{f_i,\text{step}} \tag{5}$$

Here, v_{TN} is $\frac{V_{TN}}{V_{DD}}$ where V_{TN} is the threshold voltage of the n-transistor and V_{DD} is the supply voltage. Typical values of v_{TN} and $\frac{C_M}{C_L}$, which we use in this work, are 0.2 and 0.1, respectively [8]. The term $t_{f_i,\text{step}}$ corresponds to the step input response corresponding to the fall transition. A similar expression is used for the rise transition. The value of τ is taken to be twice the Elmore delay of the preceding gate, as in [2].

Using this method to calculate the delays of individual gates, the PERT procedure is used to find the critical path in the circuit as in [1].

The proposed algorithm also requires the computation of the sensitivity of the gate delay with respect to a gate size. It is well-known [1] that the step delay sensitivity to a gate size can be computed by considering only that gate (whose resistance is affected by the gate size) and its fanin gates (whose load capacitances are affected by the size of that gate). Therefore, the delay sensitivity computation under step inputs is a very local computation.

Under the improved delay model above that considers input transition times, the size of a gate

affects not only the delay of that gate and its fanin gates, but also the delay of all gates in the transitive fanout. The delays of the gates in the transitive fanout depend on the value of their input slew rates (τ values), which in turn, are dependent on the delay of the current gate. However, it can easily be shown from the application of Equation (5) that for real parameter values, the effect of changing a gate size is vastly diluted as one moves further and further away from the gate along its transitive fanout. For real circuits, we found that the size of a gate affects only the current gate, its fanin gates, its immediate fanout gates, and their fanouts. Therefore, for all practical purposes, the sensitivity computation remains an inexpensive local computation, even under the improved delay model that considers input transition times.

3 Buffer insertion

The essential idea of buffer insertion is to reduce the delay at high capacitance nodes by reducing the load on the driving gate. To maintain signal polarities, we assume that each buffer consists of a pair of inverters that may be sized appropriately. Thus, the addition of each buffer implies the addition of four new transistors to the circuit.

3.1 Notions of criticality

As a preliminary step, we define a *critical* path as any path that violates the timing specification. We also explain a nonquantitative and somewhat fuzzy term that we term as the criticality of a path. Roughly speaking, the criticality of a path is dependent on the magnitude of the violation, so that paths with large violations are identified as being highly critical, and those with small violations are only mildly critical. This notion is important since we observe that the greater the criticality of the path, the larger the amount of sizing required for the path to meet specifications. Later in this paper, we will work towards developing measures to quantify the criticality of a path.

Generally speaking, it has been our experience that buffer insertion is useful only for highly critical paths. This experience is based on our experimental results which use a measure of criticality, developed later in this paper, to quantify the criticality of a path. For mildly critical paths, it may be more advantageous to use sizing than buffer insertion. The intuition behind this is that mildly critical paths can be made to meet timing specifications through a small amount of sizing; inserting a buffer implies an increase in area corresponding to the four new transistors that constitute the buffer, which is likely to be larger. Moreover, the addition of an excessive number of buffers can

actually increase the delay of some paths of the circuit, and therefore we add them only where we must, namely, to reduce the delays on the highly critical paths.

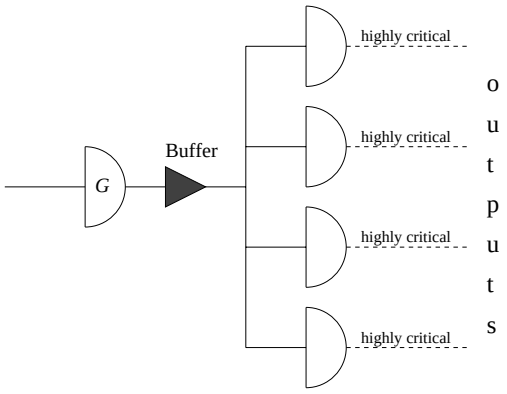

Figure 3: Type A buffer insertion.

3.2 Types of buffer insertion strategies

We identify two situations in which the insertion of buffers is advantageous, which we will refer to as Type A and Type B buffer insertion scenarios, respectively. As shorthand notation, we will refer to an output H of a gate G being highly critical if some highly critical path passes through gates G and H; similarly, we also refer to mildly critical and noncritical outputs.

Type A If a gate whose outputs are all *highly critical* drives a large capacitive fanout, buffer insertion can help in reducing the delays of these paths. Figure 3 shows the situation of type A buffer insertion[3]. By choosing an appropriate size of buffer, the fanout capacitance of Gate G may become smaller, and sum of the delays of the buffer and Gate G may be smaller than the delay of Gate G in the unbuffered circuit.

Type B If a gate has some highly critical outputs and some mildly critical and noncritical outputs, then one may isolate the capacitance of the noncritical outputs from the highly critical path by inserting a buffer, as shown in Figure 4. The mildly critical paths constitute a gray area and must be assigned to be either critical or noncritical, based on measures that we will

[3]The essential idea here is not dissimilar to the Mead-Conway idea of using chains of inverters to drive a large load, with a ratio of e minimizing the delay. However, we differ in the following ways: (a) our objective is not to minimize the delay but to meet a specification (b) if the circuit as a whole is anything other than a chain of inverters, it is not possible to use the constant-ratio idea to minimize the delay of the circuit.

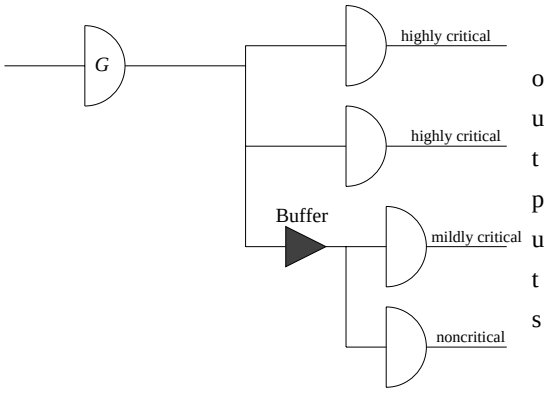

Figure 4: Type B buffer insertion.

develop later in this paper. Since the fanout capacitance of gate G becomes smaller, the RC delay of G is reduced, and therefore, the delay along the highly critical paths is reduced.

As a side-effect, the delay along the noncritical paths may be increased. The additional delay introduced along noncritical paths that became critical after buffer insertion, can be made to meet specifications through a small amount of sizing.

The challenge here is to quantify measures of criticality, and to use them to determine appropriate locations for buffer insertion.

Interestingly, the work in [9] that was performed independently also uses similar terminology for Type A and Type B buffers. However, that work concentrates on reducing wire delay at the post-layout phase, an issue that we do not address here.

3.3 Examining the effect of buffer insertion

We will examine the effect of buffer insertion through a simple example. Consider the gate G shown in Figure 5 driving gates $G_a, G_b, G_c, \cdots, G_f$. If all of the outputs are on highly critical paths, then we would insert a Type A buffer immediately after G, driving all fanouts. The insertion of this buffer would change the fanout capacitance of G and therefore, its delay would change from $D_{G,old}$ to $D_{G,new}$. If the delay of the buffer is D_{buf}, then the change in the delay to the most critical output would be $D_{G,new} + D_{buf} - D_{G,old}$, since all other gate delays in the circuit would be unaffected. For the buffer insertion to be advantageous, this value must be negative.

Note also that this transformation would change the delay of *any* path passing through G by the same amount, and would therefore decrease it; for any path that does not pass through G, the

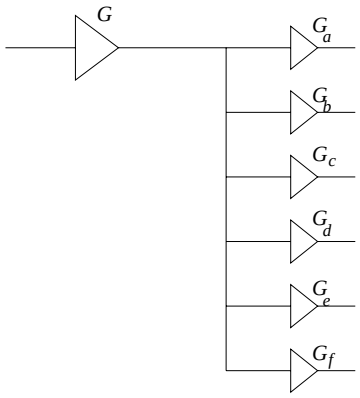

Figure 5: The effect of buffer insertion.

delay remains unaffected. Therefore, this transformation either reduces the delay at each primary output, or leaves it unaffected.

If we consider Type B buffer insertion in Figure 5, and let us now assume that G_a and G_b are highly critical, G_e and G_f are noncritical, and G_c and G_d are critical, but not highly critical. We will refer to a gate G_i as being buffered if the Type B buffer is placed between the output of G and G_i, and we will consider it unbuffered otherwise. We can now place a single Type B buffer using the following ideas:

- G_a and G_b must certainly be unbuffered.
- G_e and G_f should be buffered, since they only add to the capacitance being driven by gate G. Although it is possible that G_e and G_f may become critical outputs after buffer insertion, they would, at worst, probably be very mildly critical since they were noncritical before buffer insertion.
- The key issue is the status of G_c and G_d. If they are buffered, then they may become highly critical after buffer insertion. On the other hand, if they are not buffered, the capacitance at G may be too high and the delay of gate G may not be reduced sufficiently by buffer insertion. Therefore, the best solution may either buffer off none, one, or two of the gates G_c and G_d, and a good criterion is required to determine which of these should be chosen.

3.4 On complexity and convexity issues

The transistor sizing problem is well known to be equivalent to a convex programming problem [1, 2] when the topology of the circuit is fixed, since the area objective and the circuit path delays can be represented as posynomial [10] functions of the transistor sizes[4].

However, when the structure of the circuit is allowed to change, this is no longer true. If there are b possible buffer locations associated with a path, then there are b possible delay functions $f_1, f_2 \cdots f_b$ (each a posynomial), of which one (or at most a few) is optimal. Note that an optimal circuit is the circuit with the minimum area for the given delay specification. The path delay is thus f_1 or f_2 or $\cdots f_b$, which cannot be represented as a convex programming problem (it may, however, be written as an mixed integer nonlinear programming problem), and its solution is not easy to find. A second pointer to its difficulty is that even a special restriction of the problem, that of finding the optimal locations for Type B buffers in an unsized circuit, is NP-complete [5]. Therefore, we resort to heuristic methods for solving the problem.

4 Outline of the algorithm

The procedure developed here enhances the TILOS algorithm [1] which operates iteratively, identifying the most critical path in every iteration. The sensitivity of the path delay, D, to the area, A, given by $\partial D/\partial A$, is computed for all of the transistors along the critical path, and the transistor with the most negative sensitivity is bumped up by a factor, `Bumpsize`. `Bumpsize` is typically set to a value that is just larger than one, and values between 1.1 and 1.5 have been seen to work well. The procedure continues until all timing specifications are met.

As in the TILOS algorithm, we begin with the unsized circuit as provided to us. We continue optimizing the circuit until all the delay constraints are met at every circuit output. Until that is achieved, in each iteration, we identify the most critical path, i.e., the path with the largest violation of the timing specification. We attempt to improve the delay along this path by one of several possible transformations

- bumping up the size of some transistor along the path

[4] A posynomial is a function g of a positive variable $\mathbf{w} \in \mathbf{R}^n$ that has the form $g(\mathbf{w}) = \sum_j \gamma_j \prod_{i=1}^{n} w_i^{\alpha_{ij}}$, where the exponents $\alpha_{ij} \in \mathbf{R}$ and the coefficients $\gamma_j > 0$. Roughly speaking, a posynomial is a function that is similar to a polynomial, except that (a) the coefficients γ_j must be positive, and (b) an exponent α_{ij} could be any real number, and not necessarily a positive integer, unlike the case of polynomials. A posynomial has the useful property that it can be mapped onto a convex function through an elementary variable transformation [10] $(w_i) = (e^{x_i})$

- inserting a Type A buffer along the critical path
- inserting a Type B buffer to isolate noncritical paths from critical paths

The general philosophy behind the algorithm is shown below. It should be stressed that although this is the general philosophy, the actual implementation is somewhat different and will be elaborated on in subsequent sections.

```
minimum_delay = minimum-sized circuit delay
Initialization (all gate sizes are set to minimum values)
While (delays at all primary outputs are not ≤ T_spec) {
      Compare path delays with T_spec and find the most critical path
      For all gates on the critical path {
          Estimate figure of merit of bumping up a transistor
          Estimate figure of merit for inserting a Type A buffer
          Estimate figure of merit for inserting a Type B buffer
      }
      If (bumping up a transistor has the best figure of merit)
          increase the size of a selected transistor
      if (inserting a Type A buffer has the best figure of merit)
          insert a Type A buffer
      if (inserting a Type B buffer has the best figure of merit)
          insert a Type B buffer
      Recompute circuit delays
      if (circuit_delay < minimum_delay) minimum_delay = circuit_delay
      if (circuit_delay > 1.1 × minimum_delay)
          /* failed to meet specifications */
          exit
}
```

The iterations end if futher sizing does not result in delay reduction, and in fact, increases the circuit delay by a significant amount.

In the following sections, we will consider the problems of developing figures of merit for bumping up a transistor and for inserting a Type A or a Type B buffer. Since we have followed the TILOS template, we will also use the most negative sensitivity $S_T = \partial D/\partial A$ of a transistor to compare the

relative figures of merit. Note that S_T corresponds to the delay reduction caused by bumping up the transistor size; this fact will be used when we develop comparable figures of merit for inserting Type A and Type B buffers.

4.1 Type B buffer insertion

4.1.1 Rationale behind the procedure

The purpose of using Type B buffers is to insulate the noncritical paths from the highly critical paths, thereby enabling greater amounts of delay reduction for the circuit as a whole.

As stated in the outline of the algorithm, the objective is to determine a figure of merit that can reasonably be compared to the figure of merit for sizing, namely, the sensitivity of the most sensitive gate. Let us temporarily assume that we have developed a way of measuring the criticality of a gate output, and that we can recognize the highly critical outputs. We will later show the precise method by which this is achieved in Section 4.1.3.

While considering a candidate Type B buffer location at one of the outputs of a gate, we first consider the delay along its highly critical fanouts. By definition, a Type B buffer will always reduce the delay to a highly critical fanout, and this is achieved at the expense of an increase in the area; the area increase corresponds to the area of the inserted minimum-sized buffer.

Therefore, a reduction in the delay by an amount ΔD can be effected by an area increase of ΔA. We must now estimate the amount of area, ΔA_T, required by the sizing procedure to achieve the same delay reduction. If $\Delta A < \Delta A_T$, then we insert the Type B buffer.

To fairly compare the effects of sizing and Type B buffer insertion, let us consider the following problem:

> `For the same reduction in delay,` ΔD`,`
> `what is the increase in the area required by the two procedures?`

It is tempting to estimate ΔA_T as $-\frac{\Delta D}{S_T}$ (recalling that S_T, the figure of merit for transistor sizing, is the most negative sensitivity of a critical path transistor), as shown by curve (a) in Figure 6. However, the corresponding change in area, $\Delta \mathrm{A}_{(\mathrm{a})}$, is only a lower bound on the value of ΔA_T and is typically not a very tight bound. This is because the sensitivity S_T corresponds to a *small* perturbation, whereas the change ΔD is large. A linear approximation would provide

optimistic estimates of ΔA_T. Moreover, for a transistor with size x, it has been shown that

$$S_T = K_1 - \frac{K_2}{x^2} \tag{6}$$

where K_1, K_2 are independent of x [1]. The above equation is accurate for the delay of the current critical path, but not for the delay of the entire circuit, which is the maximum path delays; note that the critical path may change when a transistor size is altered.

A second idea would be to use Equation(6) to estimate $\Delta \mathrm{A_T}$, as shown by curve (b) in Figure 6. However, this estimate, $\Delta \mathrm{A_{(b)}}$, is accurate only if this same transistor is critical in every sizing step involved in reducing the delay by ΔD. This is typically not true, and therefore, such an expression would provide an upper bound on the area.

In most cases, the actual area-delay curve would lie between the two bounds as shown by curve (c) in Figure 6, and our problem is to determine the shape of this curve and the value of ΔA_T.

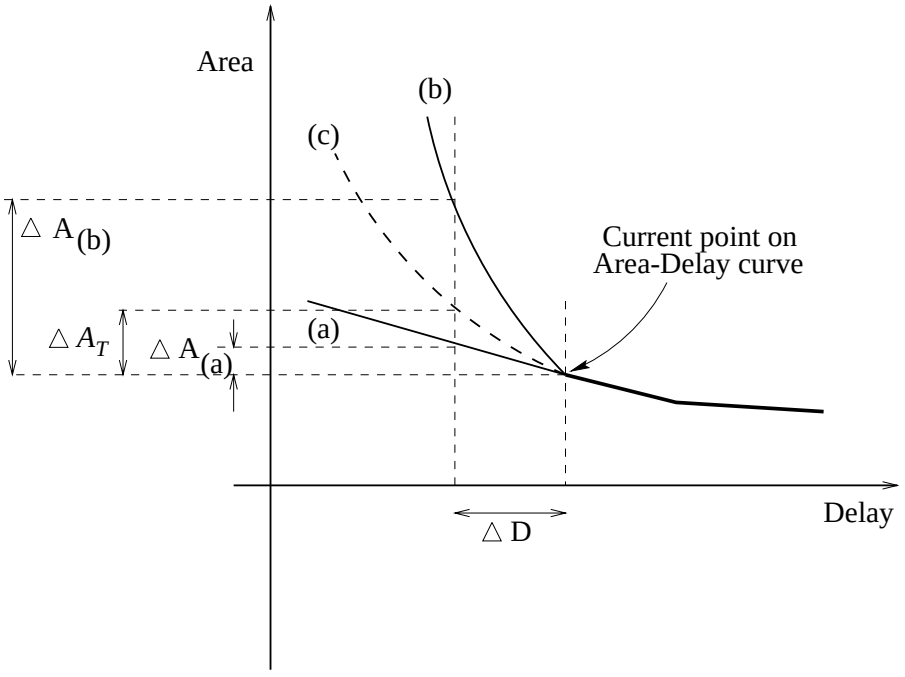

Figure 6: Estimating $\Delta \mathrm{A_T}$

An additional complication is as follows. Consider, for a moment, the TILOS algorithm for transistor sizing, and let D be the delay of the circuit during the current iteration. Then, bumping

up the size of an individual transistor causes a delay reduction of δD on the current most critical path and an area increase of δA. However, the *circuit* delay is not necessarily reduced by δD since the bumping operation could cause a different path to become the most critical path. This approximation is justifiable in TILOS because the area and delay change in each iteration are very small. Note that if we wanted to be exact, we should have considered the area increase required to constrain *all* path delays to $D - \delta D$.

However, if the delay is changed from D by a *large* amount, ΔD, as is the case in our situation, such an approximation is invalid, and we must find the area increase required by sizing to ensure that the delays of *all* paths (and not just the current most critical path) are less than $D - \Delta D$. In other words, all of these paths must be sized appropriately, and ΔA must be computed by considering the effect of all of these paths, and not just the most critical path as in TILOS.

To take care of this problem, we consider all primary outputs, and find the area increase required to ensure that the maximum delay over *all* outputs (and not just at the critical output) is no larger than $D - \delta D$.

4.1.2 Details of the procedure

As indicated above, calculating ΔA_T for a gate with size w as $\left[\Delta D / \frac{\partial D}{\partial A}\right]$ is very inaccurate since

(1) the sensitivity $\frac{\partial D}{\partial A}$ varies nonlinearly with the gate size.

(2) a large delay reduction of ΔD is probably best achieved by sizing not just this single gate, but several other gates too.

To estimate the value of ΔA_T, given a specific buffer insertion point, we first calculate the change in the delay of the circuit due to the insertion of a minimum sized Type B buffer. At each such primary output i, we use an extrapolation method to estimate the area increase, Δa_i, required to match the circuit delay reduction. We then calculate the figure of merit for sizing as

$$\Delta A_T = \sum_{i \in po} \Delta a_i \tag{7}$$

The extrapolation procedure is implemented as follows. For each primary output, we store the effect of the most recent sizing steps as a delay vs. circuit area table. We use those data to extrapolate the change in area corresponding to ΔD at every output; these values are summed up

to give ΔA_T, as described in the last section. Specifically, we use Lagrangian extrapolation [11] to estimate ΔA_T for ΔD. We found that a fourth order polynomial approximation was adequate.

If ΔA_T, the estimated area required to achieve the delay reduction through sizing alone, is lower than ΔA_B, the area of a minimum sized Type B buffer to be inserted at the chosen point, then the buffer is inserted. If not, the algorithm abandons the Type B buffer insertion in the current iteration, and then chooses either a Type A buffer insertion or a sizing step.

4.1.3 Finding an appropriate location for buffer insertion

The criterion for Type B buffer insertion is to isolate the less critical paths from the more critical ones; if the total capacitance of the less critical paths is substantial, then significant delay improvements are possible. Therefore, our first challenge is to develop a measure for criticality, which is key to the success of this algorithm and is required to partition the fanouts of a gate into a "critical" and "noncritical" set, as shown in Figure 7. We will also quantify the criticality of the mildly critical path, which constitute a gray area, and develop measures to decide whether they should be considered critical or noncritical.

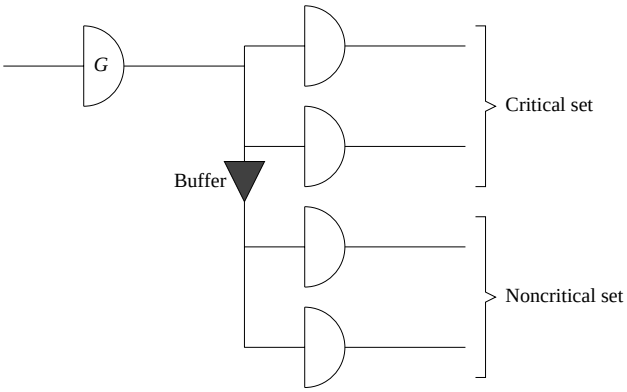

Figure 7: Partitioning the fanout gates into critical and noncritical sets for Type B buffer insertion.

Consider the sensitivity, $\frac{\partial d}{\partial x_i}$ for each gate i, where x_i is the size of the gate, and d is the delay of the most critical path through the gate. We consider the possibility of bumping up the size of gate i, and recognize that it is pointless to size a gate with positive sensitivity as that would increase the circuit delay. We maintain the number

$$\sigma_i = \min(0, \frac{\partial d}{\partial x_i} \cdot \Delta x_i) \tag{8}$$

for each gate i, where Δx_i is the amount by which the gate size would be increased if it were to be bumped up. Therefore, σ_i estimates the reduction in the gate delay through a possible bumping up operation. Note that gates with a positive sensitivity are assigned a σ_i of zero since the gate size would be left unchanged if the bumping operation were to increase the delay.

We define a measure for the criticality that we call χ, associated with each gate fanout. This measure is related to the amount by which the delay of a circuit can be reduced and to the delay along a path. Fanouts with larger χ values are less critical than those with smaller χ values.

A backward PERT traversal[5] is performed from the primary outputs towards the primary inputs (PI's) to calculate the value of χ for each gate. The χ value at each primary output is set to be the difference between the maximum delay at the primary output and the actual delay to that point. Therefore, increasing the path delay to that primary output by χ will leave the circuit delay unchanged.

If we know the χ value for all the fanouts of a given gate i, its own χ value is calculated as

$$\chi_i = \min_{j \in \text{fanouts}(i)} [\chi_j + \text{slack}_j] + |\sigma_i| \tag{9}$$

where slack_j represents the slack at fanout j. The slack is defined as the amount by which the delay along this path may be increased before it becomes the longest delay path in the circuit. Note that all elements in this equation have dimensions of delay. Therefore χ_i is a measure of the amount of delay increase along a path from the gate to any primary output that can be absorbed "easily," either by the slack or by a small amount of sizing. This is consistent with the idea that a high value of χ_i at the fanout j of a gate i implies that the maximum delay path from i to a primary output through j is not very critical.

The next challenge is to use this measure of criticality to determine the best location at which a Type B buffer should be inserted to improve the current most critical path. The steps involved in determining the buffer location can now be summarized as follows:

1. Find the gate i with the maximum fanout capacitance along the most critical path of the circuit. We will consider inserting a Type B buffer at the output of this gate to partition the critical and noncritical fanouts[6].

[5] As pointed out in [12], the method referred to as PERT in the CAD literature is actually the critical path method (CPM). However, we persist with the prevailing incorrect usage to avoid confusion.

[6] Several other criteria for selecting gate i were tried out; however, this criterion was seen to be the most successful across all circuits.

2. Find the maximum value of χ_j of all fanouts of gate i; let χ_{max} be the maximum value of χ_j. All fanouts j whose χ_j is $\geq c_1 \cdot \chi_{max}$ (where $c_1 < 1$ is an empirically tuned number) are placed in the noncritical set[7] This has the effect of placing all fanout gates with high χ values in the noncritical set.

   However, if too many gates are placed in the noncritical set, the capacitance to be driven by the type B buffer may become too high. This could cause its delay to be very large, so that path through i and some of the gates in the noncritical set may become very critical. Therefore, the above classification may be too optimistic, and we apply the next criterion described below.

3. Having determined the noncritical set, we next estimate the effect of inserting a minimum-sized buffer. When a buffer is inserted, the delay of gate i is reduced by an amount ΔD_{dec}, which is the delay reduction along the critical paths. Along a noncritical fanout j, the delay is increased by $\Delta D_{inc} - \Delta D_{dec}$, where ΔD_{inc} is the increased delay due to the insertion of a buffer.

   Recall that χ_j is an estimate of the amount by which the delay from j to the primary outputs may be increased before j lies on the most critical path of the circuit. Of this amount, $\Delta D_{inc} - \Delta D_{dec}$ is consumed by the insertion of a buffer.

   Therefore, with the insertion of the buffer, we may say that the delay from j to the primary outputs may be increased by $\chi_j - (\Delta D_{inc} - \Delta D_{dec})$ before j would lie on the most critical path. The larger this amount, the less critical the path would be after buffer insertion. Therefore, we calculate this quantity for each fanout and if its value is small, then we remove the fanout j from the noncritical set.

4. For any fanout j, if $\chi_j - (\Delta D_{inc} - \Delta D_{dec}) < \beta$ for some empirically determined β, then the gate is moved from the noncritical set to the critical set.

   The value of β was chosen as $c_2 \cdot d_{minsize}$, where $d_{minsize}$ is the delay of a minimum size inverter driving a minimum size load. The use of $d_{minsize}$ is purely for normalization purposes to ensure that the value of β is of the correct order of magnitude. The value of c_2 is then determined empirically.

It was experimentally found that approximate values of $c_1 = 0.8$ and $c_2 = -0.5$ work well on the circuits that we tested. Note that a negative value of c_2 causes a negative value for β.

[7] We reiterate that some of the mildly critical paths may also be quantized as being "noncritical."

Recall that a mildly critical path (using the terminology of Figure 4) may also be buffered off. The negative value for β corresponds to a mildly critical path where the insertion of a buffer may actually increase the path delay enough to cause the value of $\chi_j - (\Delta D_{inc} - \Delta D_{dec})$ to be negative. When this value is negative, it might seem that the path delay would increase after buffer insertion. However, in our calculations, we assumed a minimum sized buffer, and if the value of $\chi_j - (\Delta D_{inc} - \Delta D_{dec})$ is very slightly negative, we may recover from this easily by sizing the buffer by a small amount. Therefore, in practice, we found that a small negative value for c_2 gave good results.

4.2 Type A buffer insertion

During the iterative procedure, we observed that inserting a minimum-sized Type A buffer almost always caused the path delay to increase. However, by appropriately choosing the size of the Type A buffer, delay reductions can be effected. The following procedure is used to estimate the potential delay reduction through Type A buffer insertion at each gate output:

1. Find the minimum (most negative) sensitivity among the gates along the most critical path, denoted as $\left.\frac{\partial D}{\partial x}\right|_{best}$.

2. For each gate on the most critical path, we calculate the values of ΔD_{rise} and ΔD_{fall}, the changes in the rise and fall delays, respectively, if a Type A buffer were to be inserted at that gate output. For the buffer insertion step to achieve a useful purpose, it must be ensured that both $\Delta D_{rise} < 0$ and $\Delta D_{fall} < 0$. Keeping the topology of the rest of the circuit constant (in particular, keeping the sizes of the gates fanning into and out of the proposed buffer constant), the sizes of transistors in the buffer are estimated[8].

   Only those gates at which both the rise and fall delays can be reduced are considered as candidates for buffer insertion. For these gates, the sensitivity of the buffer, $\left.\frac{\partial D}{\partial x}\right|_{buffer}$, is determined for the calculated size.

   If $\left.\frac{\partial D}{\partial x}\right|_{buffer} < \left.\frac{\partial D}{\partial x}\right|_{best}$, then this location is designated as a permitted buffer insertion location. The rationale behind this is that at this point, after buffer insertion,

[8] For purposes of this estimation only, we make a simplification where we consider that the size of the p-transistor is K times that of the n-transistor, and use a simple iterative loop to solve for the transistor sizes in the buffer that minimize the average of the rise and fall delays.

(a) the delays of all the paths driven by the buffer are smaller than those of prior to buffer insertion, and

(b) the most negative sensitivity value on that critical path is made even more negative than before, implying that the increase in area due to buffer insertion could be recouped in future steps through sizing. In other words, the potential for reducing the path delays with buffer insertion is better than the ability without buffer insertion.

3. Among the permitted buffer insertion points in Step 2, the output of gate k with the best delay reduction is chosen to be the best Type A buffer insertion location. The value of $(\Delta D_{rise} + \Delta D_{fall})/2$ is used to estimate the effect of buffer insertion on the delay.

4. Having performed a Type A buffer insertion, the buffer and its predecessor gate k are now reset to the minimum size to correct for any over-sizing in k in the past. The sizing procedure is permitted to size these gates back up again in subsequent iterations to their optimal sizes, so that the solution is not unduly bound by any incorrect sizing choices that were made before the buffer was added. During this process, we prohibit further Type A buffer insertion until the iterations reach the point where the circuit delay becomes smaller than that before the insertion of this Type A buffer.

4.3 The final algorithm

The pseudocode shown in Section 4 was only a general outline of our procedure, and we may now describe the pseudocode of the algorithm more accurately as follows:

```
minimum_delay = minimum-sized-circuit-delay
while (current_delay > T_spec)
{
  if the criteria of type B buffer insertion are satisfied
         then perform type B buffer insertion.
  else if the criteria of type A buffer insertion are satisfied
         then perform type A buffer insertion.
  else
         bump up the most sensitive transistor.
  recalculate current_delay
  if (current_delay < minimum_delay) minimum_delay = delay
```

```
  if (current_delay > 1.1 × minimum_delay) exit.
}
```

The algorithm chooses to consider the option of inserting a Type B buffer first, and then considers the Type A buffer, finally defaulting to transistor sizing if neither is viable. It is possible to consider these in any order, but it was found that this ordering worked best for the circuit examples that we tried.

We now attempt to provide an estimate of the amount of computation involved in each iteration. While a detailed complexity analysis is unrealistic due to the unpredictability of the number of iterations, it is useful to count the number of computations involved in each iteration of this algorithm.

We assume that the number of gate fanins and fanouts are bounded by a constant, which implies that delay and sensitivity calculation for each gate can be carried out in constant time. The timing analysis required to calculate the circuit delay is $O(|V| + |E|)$, where $|V|$ is the number of vertices in the circuit graph, corresponding to the number of gates in the circuit, and $|E|$ is the number of edges in the circuit graph, where each edge corresponds to an interconnection from one gate to one of its fanouts. After the first time, however, the computation is significantly reduced since incremental techniques are used. In the worst case, we only process all edges in the fanout cone of the predecessor of the gate that is sized or the buffer that is inserted. The worst-case complexity of this step is also $O(|V| + |E|)$, but the typical update is empirically seen to occur in much less time. During delay calculation, the slack at each node is also calculated at no additional increase in the computational complexity.

The next step involves the calculation of gate sensitivities along the critical path. If D_c is the depth of the circuit (largest number of gates on any path) then the number of gates on the critical path is bounded above by D_c, and the amount of time required to compute the sensitivities and to find the maximum sensitivity is $O(D_c)$. This step is the only computation required for the sizing operation, and is also required by the criteria of Type A and Type B insertion.

For type B buffer insertion, the calculation of σ values is required in the fanout cone of gates in the critical path. This can be carried out in $O|V|$ time. This is followed by a PERT procedure that uses the slacks to compute the χ values in $O(|V| + |E|)$ time. The gate on the critical path with the highest capacitance is found in $O(D_c)$ time. Since the number of fanouts is bounded, the use of the χ values to partition the fanouts into critical and noncritical fanouts is completed in constant time. Note that the comparison with sizing, illustrated in Figure 6 is performed in constant time.

For type A buffer insertion, the amount of time required for steps 1 through 4 in Section 4.2 is $O(D_c)$, assuming (as is seen in practice), that the iterations of step 2 take constant time.

Therefore, in summary, each iteration requires $O(|V| + |E|)$ time for timing analysis and slack calculation, $O(D_c)$ time for sensitivity calculation, $O(|V| + |E|)$ time to evaluate type B buffer insertion, and $O(D_c)$ time to evaluate Type A buffer insertion, and since $D_c < |V|$, the overall complexity of each step is $O(|V| + |E|)$. We emphasize that due to the incremental techniques used, this is a pessimistic estimate of the complexity.

5 A brief note on unsuccessful strategies

For purposes of completeness (and since a negative result is also sometimes a worthwhile result), we believe that it is also worthwhile to point out a few strategies that seem sound on the surface, but were found to be unsuccessful in our experiments.

5.1 Incorporating rollback

Since the insertion of a buffer is a drastic step, we considered including rollback, where after each buffer insertion, a certain number of prior sizing steps were nullified. The idea behind rollback is that any sizing steps performed immediately prior to the buffer insertion step may have been suboptimal since they were performed under the assumption that no buffer would be inserted. Therefore, it was thought to be a good idea to consider rolling back to an earlier iteration and resuming the process from there.

However, in practice, we tried several criteria for incorporating rollback and found that the results using rollback were seldom better, while the memory requirements and execution times were phenomenally large. Therefore, we abandoned the idea of incorporating rollback into the optimization process.

5.2 Gate cloning

An alternative to buffer insertion would be to clone gates to perform Type B buffer insertion. The primary idea is that instead of creating a new buffer, the use of cloned versions of a gate could be useful. For example, if a Type B buffer is to be inserted at the output of an inverter, then cloning the inverter amounts to an additional expense of two transistors, while inserting a buffer (which consists of two inverters) amounts to an expense of four transistors. Secondly, buffer

Table 1: Comparison of Sizing vs Sizing+Buffer Insertion

Circuit	$\|G\|$	D_u	A_u	T_{spec}	*Sizing*		*Sizing+Buffer Insertion*		Area Ratio
					Area	CPU time(s)	Area	CPU time(s)	
cc	58	61.4	248	23	900	6.7	706 (A:1; B:6)	4.9	1.27
cm163	43	43.4	160	14	692	3.0	428 (A:1; B:5)	2.6	1.62
f51m	136	82.5	548	50	2627	17.6	1627 (A:1; B:4)	13.3	1.62
i135	269	121.1	1252	36	4307	29.9	2183 (A:0; B:13)	32.8	1.97
c499	202	177.3	816	51	3004	30.2	2571 (A:1; B:15)	62.4	1.17
c1355	546	324.5	2128	100	5001	145.9	4279 (A:1; B:38)	192.3	1.17
c2670	1193	456.0	4152	88	9000	481.3	8586 (A:1; B:94)	595.7	1.05
c5315	2307	831.2	8772	190	15000	987.2	13619 (A:1; B:125)	1013.3	1.10

insertion increases the number of levels in a circuit and therefore may cause unnecessary delay increases along paths that lead to outputs of moderate criticality. The use of cloning may resolve this problem by avoiding the insertion of an additional level of logic in the form of a buffer.

In our implementations we found that when we added gate cloning to the list of strategies used here, we never obtained better results on any of the benchmark circuits. This can be attributed to the fact that situations such as the above do not occur sufficiently often in these circuits. However, one could certainly generate an artificial example where gate cloning could be useful.

6 Experimental results

The algorithms described above have been implemented in C on an HP 735 workstation. In Table 1, we present the results on some circuits from the ISCAS85 [13] and LgSynth91 [14] benchmark suites.

For each circuit, the number of gates $|G|$, the unsized delay D_u, and the unsized area A_u are shown. For a given (moderate) timing specification T_{spec}, the area of our approach is compared with the area from our implementation of TILOS, which is a direct implementation from [1]. The differences between our implementation of TILOS and [1] is that we replace each gate by an equivalent inverter, characterized by gate sizes W_n and W_p, and solve the circuit to find the optimal W_n and W_p and, in case of our algorithm, the optimum buffer locations too. Moreover, we use a timing model that takes input slew times into consideration. Next to the area numbers the table are also shown (in brackets) the number of Type A and Type B buffers. It is seen here that

most of the buffers in this table are of Type B. Although not shown in this table, it was observed that for tighter specifications, a larger number of Type A buffers were added. The CPU times for both methods are very similar. The area ratio shown in the last column shows the ratio of the area required by sizing alone as compared to the area required by our method. Therefore, this number should be at least equal to 1 (as it always is here) and a larger magnitude implies a better improvement over sizing alone. Significant improvements are possible in most cases. We point out that as the delay specification is tightened further, larger area savings are possible for each circuit for tightened constraints.

For example, for circuit c5315 with a timing specification of 190ns, our approach provided an area of 13619, while the area of the circuit using sizing alone was 15000. This corresponds to a savings of about 10%. Our approach requires the insertion of 1 Type A buffer and 125 Type B buffers to meet this specification.

The entire area-delay tradeoff for this algorithm for three different benchmark circuits is shown in Figures 8 through 10. In each case, it is seen that significant improvements are possible from the use of our approach, particularly for tighter specifications. The reader is cautioned that although some curves, such as the one in Figure 9, seem to be close to each other, in the steep region of the curves, even small differences are greatly magnified on the y-axis, and our approach gives significant cost savings in that region.

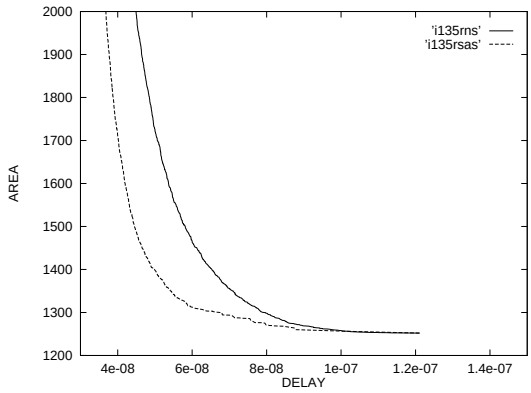

Figure 8: Area-delay tradeoff for circuit i135

For some circuits, such as c499 (Figure 9) and c1355 (Figure 10) the area from our approach for loose delay specifications is very slightly worse than that from sizing alone. The explanation for this can be seen by examining our approach in a different light. In each step, the approach

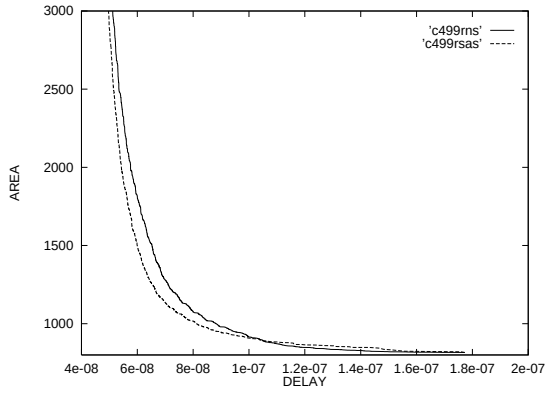

Figure 9: Area-delay tradeoff for circuit c499

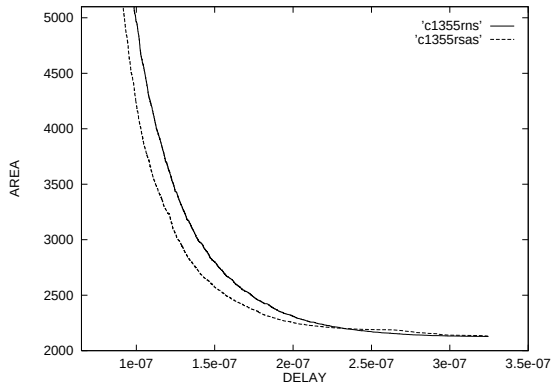

Figure 10: Area-delay tradeoff for circuit c1355

attempts to reduce the delay of the circuit, going along an area-delay tradeoff curve that is similar in nature to that shown in Figure 1, with a smaller value of d_{min}. As the algorithm progresses, each iteration represents a motion to the left along this tradeoff curve. The method typically "looks ahead" to determine if a buffer will be required to meet a delay specification (with the best area) several iterations in the future. Therefore, for a few iterations after the buffer is inserted, the results are likely to be slightly suboptimal, and some of this is manifested in the results. Since we are primarily interested in sizing circuits to meet tight delay specifications, which is a region where our algorithm works well, we have not taken any steps to remedy the occasional minor problem with loose delay specifications.

7 Conclusion

In this paper, we have aimed to support the basic idea that buffer insertion can help to improve the area-delay tradeoff curve and have presented heuristic algorithms for the purpose. The gate sizing procedure is known to be power-conscious since it sizes gates only when necessary and reduces the dynamic power; in this work, the efficacy of this is further improved by considering buffer insertion to achieve the delay goal for the circuit with a smaller area/power cost. Additionally, it is ensured that buffers are added only as needed so as to minimize the area and the power dissipation, and the process of buffer insertion is targeted towards meeting a given specification, rather than towards minimizing the circuit delay. The techniques developed herein are supported by experimental results that demonstrate that improvements can be achieved both in the area and the minimum achievable delay in comparison with an algorithm that performs sizing alone.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] J. Fishburn and A. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 326–328, 1985.

[2] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.

[3] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor sizing for low power CMOS circuits," *IEEE Transactions on Computer-Aided Design*, vol. 15, pp. 665–671, June 1996.

[4] K. J. Singh and A. Sangiovanni-Vincentelli, "A heuristic algorithm for the fanout problem," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 357–360, 1988.

[5] C. L. Berman, J. L. Carter, and K. L. Day, "The fanout problem: From theory to practice," in *Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conference*, pp. 69–99, 1989.

[6] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, Jan. 1948.

[7] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[8] K. O. Jeppson, "Modeling the influence of the transistor gain ratio and the input-to-output coupling capacitance on the CMOS inverter delay," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 646–654, June 1994.

[9] K. Sato, M. Kawarabayashi, H. Emura, and N. Maeda, "Post-layout optimization for deep submicron design," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 740–745, 1996.

[10] J. Ecker, "Geometric programming: methods, computations and applications," *SIAM Review*, vol. 22, pp. 338–362, July 1980.

[11] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis.* New York: McGraw-Hill, 1978.

[12] T. M. Burks, K. A. Sakallah, and T. N. Mudge, "Critical paths in circuits with level-sensitive latches," *IEEE Transactions on VLSI Systems*, vol. 3, pp. 273–291, June 1995.

[13] F. Brglez and H. Fujiwara, "Accelerated ATPG and fault grading via testability analysis," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 695–698, 1985. (Available at `http://www.cbl.ncstate.edu/CBL_Docs/iscas85.html`).

[14] S. Yang, "Logic synthesis and optimization benchmarks user guide," tech. rep., Microelectronics Center of North Carolina, Research Triangle Park, NC, 1991. (Available at `http://www.cbl.ncsu.edu/CBL_Docs/lgs91.html`).