Complexity of Minimum-delay Gate Resizing

Supratik Chakraborty
Dept. of Computer Science & Engineering
Indian Institute of Technology, Bombay, India
supratik@cse.iitb.ernet.in

Rajeev Murgai
Fujitsu Laboratories of America, Inc.
Sunnyvale, California, USA.
murgai@fla.fujitsu.com

Abstract

Gate resizing for minimum circuit delay is a fundamental problem in the performance optimization of gate-level circuits. In this paper, we study the complexity of two different minimum-delay gate resizing problems for combinational circuits composed of single-output gates. The first problem is that of gate resizing for minimum circuit delay under the load-dependent delay model. The second problem is a variant of the first, where we relax the delay model to a load-independent one, but impose load constraints instead, i.e., each gate output is not allowed to drive a capacitive load that exceeds its drive capacity. The goal, as before, is to minimize the delay through the circuit. To the best of our knowledge, there has been no published result on the complexity of these problems. In this paper, we prove that both problems are NP-complete. The proofs are inspired by Murgai's work [6], in which the global fanout optimization problem under a fixed net topology was shown to be NP-complete. These results, along with previously published ones, establish that gate resizing is a hard problem except under the most simplistic assumptions.

1 Motivation

Gate resizing for minimum circuit delay is a fundamental problem in performance optimization of gate-level circuits. Ideally, each gate should be optimally sized during technology mapping. However, exact technology mapping is expensive in practice due to the large size of the technology library and due to the complex interaction between the gate being mapped and the unmapped portion of the logic. In addition, wire loads often cannot be estimated with sufficient accuracy during technology mapping to make the best choices for gate sizes. As a result, heuristics are used, which, among other things, may not select the best sizes for gates from a delay perspective [3]. This leaves scope for improving the circuit delay by resizing gates after technology mapping. Being an in-place optimization technique, gate resizing is also layout-friendly (i.e., it does not disturb placement and routing of cells) and can be used during or after layout when more accurate wire load and delay information is available. Thus, gate resizing has become an important optimization problem in its own right.

The complexity of timing optimization problems, such as gate resizing or technology mapping for minimum circuit delay, depends on the delay model used in the analysis. Two commonly used delay models in gate-level circuits are: the **load-independent** delay model (**LIDM**) and the **load-dependent** delay model (**LDDM**).[1] Kukimoto *et al.* [4] have recently shown that under LIDM, technology mapping (hence gate resizing) for minimum delay in a combinational circuit of single-output gates can be solved in time polynomial in the number of gates and the number of library patterns using dynamic programming. In this paper, we resolve the question of the complexity of this problem under LDDM, and show that it is NP-complete. To the best of our knowledge, there is no published result on its complexity. A related problem is that of minimizing the circuit delay by gate resizing under the simpler LIDM, but with additional constraints called **load constraints**. Each output pin of a gate in the library has a **drive capacity**, which is the maximum capacitive load the pin can drive without compromising the signal integrity or the delay specification (typically a gate is calibrated for delay only up to a certain load capacitance, beyond which the gate delay parameters are not guaranteed to be accurate). The load constraints mandate that the total capacitive load driven by every pin must not exceed the pin's drive capacity. We show that gate resizing under these constraints is also NP-complete.

The inspiration behind both of our results is Murgai's work [6], in which the fanout optimization problem for a circuit with fixed net topologies was shown to be NP-complete. It turns out that the proof strategy used there is powerful and can be modified to prove the current results.

The paper is organized as follows. In Section 2, we describe the LDDM and LIDM models and formulate the gate resizing problem. Section 3 presents a brief summary of related timing optimization problems and their complexities. Section 4 outlines the construction used in [6] to prove NP-completeness of the global fanout optimization problem. We use this idea in proving NP-completeness of gate resizing/technology mapping under LDDM in Section 5. The NP-completeness proof of gate resizing under LIDM with load constraints is presented in Section 6. Section 7 presents experimental results using a greedy resizing heuristic. Finally, we conclude the paper in Section 8.

2 Preliminaries

In this section, we describe two widely used gate delay models and formalize the gate resizing problem for minimum circuit delay. We consider circuits of only **single output gates**. In addition, **all wire capacitances and delays are assumed to be zero.**

2.1 Gate Delay Models

Given a single-output gate g, let $\delta(i, g)$ denote the delay from an input pin i of g to the gate output. We will use g to denote the output of g. The load c_g refers to the cumulative capacitance seen at the output of g. It is the sum of the input pin capacitances γ_p of all the fanout pins p of g. The drive capacity of the output of g, denoted λ_g, is maximum cumulative capacitance that g can drive without compromising the signal integrity or the delay specifications.

[1] Input-slew based delay models are also popular [1]; we will comment on them in Section 8.

In the **load-independent delay model (LIDM)**, the delay from input pin i of gate g to its output is independent of the load capacitance and is given by the intrinsic delay:

$$\delta(i,g) = \alpha_{i,g}$$

In the **load-dependent delay model (LDDM)**, however,

$$\delta(i,g) = \alpha_{i,g} + \beta_{i,g} c_g,$$

where,
$\alpha_{i,g}$ = intrinsic delay from i to g,
$\beta_{i,g}$ = load coefficient of the path from i to g,
c_g = load capacitance at the output of gate g.

The **delay on a path** from a circuit primary input (PI) to a circuit primary output (PO) is the sum of the pin-to-pin delays through the gates on the path. The **circuit delay** is the maximum of all such path delays in the circuit. Given the arrival times at the primary inputs, a **forward delay trace** through the circuit computes signal **arrival times** at every pin of every gate and at the primary outputs. Given the required times at the primary outputs, a **backward delay trace** computes required times of signals at every pin of every gate and at the primary inputs.

2.2 The Gate Resizing Problem

The input to our problem is a combinational circuit composed of single-output gates chosen from a library in which each gate is available in different sizes. For each size s of a gate, the area A_s, drive λ_s, capacitance γ_i of each input pin i, load coefficient $\beta_{i,s}$ and intrinsic delay $\alpha_{i,s}$ from each input pin i to the gate output are specified. *The gate resizing problem is to determine the size of each gate in the circuit such that the circuit delay is minimized.*

With LIDM, only $\alpha_{i,s}$ is used (equivalently, $\beta_{i,s}$ is assumed to be 0); with LDDM, all of γ_i, $\alpha_{i,s}$, and $\beta_{i,s}$ are relevant. When load constraints are considered, λ_s and γ_s values are relevant; without load constraints, λ_s for all gates are effectively assumed to be $+\infty$.

3 Earlier Results

3.1 Load-independent Delay Model

The problems of gate resizing and technology mapping under different constraints can be divided into two groups: those that are solvable in polynomial time and those that are NP-complete.

3.1.1 Polynomial-time Solvable Problems

Kukimoto *et al.* [4] have shown that under LIDM, technology mapping (hence gate resizing) for minimum circuit delay can be solved in polynomial time using a dynamic programming algorithm when all gates have single outputs. Their algorithm is similar to the one Rudell proposed in his thesis [8] for covering trees for minimum delay, with the exception that Kukimoto's work does not divide a graph into trees. As a result, a match at a circuit node can cover a multi-fanout node.

3.1.2 NP-complete Problems

Li *et al.* have shown that gate resizing under area constraints is NP-complete. It remains NP-complete even for single-output gates and chain circuits (a special class of trees) [5]. The problem of gate resizing in circuits with multi-output gates has also been shown to be NP-complete. However, if gate replication is allowed, it is easy to see from Kukimoto's result [4] that the multi-output case can be solved in polynomial time.

Chan showed that when both min and max delay constraints are present, gate resizing is NP-complete even for single-output gates [2].

Recently, Murgai has shown that gate resizing/technology mapping for minimizing the maximum delay under separate rise and fall delay parameters is also NP-complete [7].

3.2 Load-dependent Delay Model

Since LIDM is a special case of LDDM, constrained versions of gate resizing that are NP-complete under LIDM remain NP-complete under LDDM. However, in the absence of constraints (like area, min-max delay constraints or separate rise and fall delays), the complexity of resizing gates for minimizing the delay of circuits with single output gates under LDDM has not been resolved in the literature yet.

4 Complexity of Global Fanout Optimization: A Starting Point

Our proofs in this paper are based on the NP-completeness proof of the global fanout optimization problem with fixed net topologies (GFO-NTF), as described by Murgai [6]. We, therefore, briefly describe GFO-NTF and sketch its NP-completeness proof here.

The GFO-NTF problem for a circuit can be stated as follows:

- *Given*
  1. *A Boolean network η consisting of gates belonging to a cell library $\mathcal{L}$, and nets with fixed topologies,*
  2. *A set of buffers and inverters in $\mathcal{L}$,*
  3. *Parameters $\alpha_{i,g}$, $\beta_{i,g}$, γ_i for each gate g in $\mathcal{L}$ ($\lambda_g = +\infty$),*
  4. *Required times & capacitive loads at POs of η,*
- *Insert buffers and inverters on the edges (segments) of nets of η such that the minimum required time at the primary inputs of η is maximized. The use of LDDM is implicit in fanout optimization.*

The following assumptions are made in solving the problem:

1. A buffer can be inserted only on an internal edge (net segment), i.e., an edge not directly connected to a PI or a PO.
2. At most one buffer is inserted on any edge.

Stated as a decision problem, the GFO-NTF problem can be formulated as follows:

INSTANCE: *Given a combinational circuit η consisting of PIs, POs, and gates interconnected with fixed-topology net-trees, required times r and load capacitances for all POs, a buffer library, and a number D.*
QUESTION: *Does there exist a buffering of the nets of η such that the required time at each PI is at least D?*

Theorem 4.1 *GFO-NTF is NP-complete [6].*

Sketch of Proof: Given η and a buffering scheme (a list of buffered edges and the corresponding buffers used), the required times at the PIs can be computed in time linear in the size of η (number of gates + number of edges) by traversing η in reverse topological order, starting from POs. Then, one can check if each PI required time is greater than or equal to D. Thus GFO-NTF is in NP.

To prove NP-completeness, 3SAT is transformed to GFO-NTF. The 3SAT problem is as follows.
INSTANCE: *Collection $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ of clauses on a finite set of n variables x_1 through x_n, such that $|\mathcal{C}_j| = 3$ for $1 \leq j \leq m$.*

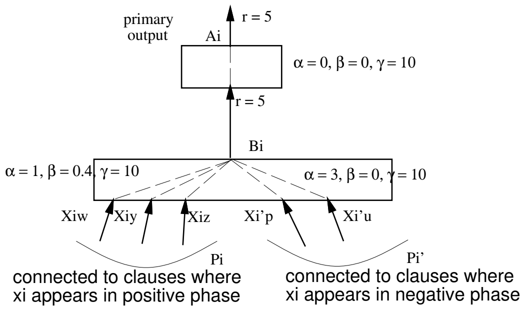

Figure 1: Variable circuit-assembly for variable x_i

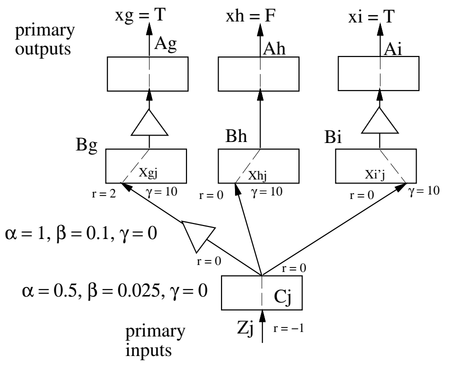

Figure 2: Buffering and required times for clause $\mathcal{C}_j$

QUESTION: *Is there a truth assignment of the n variables (true T or false F) that satisfies all the clauses in* $\mathcal{C}$*?*

From an instance of 3SAT, an instance of GFO-NTF is constructed as follows. The circuit η contains two kinds of components: *variable circuit-assemblies* (n of these) and *clause circuit-assemblies* (m of these). For each variable x_i, a variable circuit-assembly, as shown in Figure 1, is used. It has two cells A_i and B_i. The output pin of A_i is a PO of the circuit η. Its input pin is connected to the output of B_i. The single input-to-output path within A_i has zero α and β values, but an input pin capacitance γ of 10 units. B_i has as many inputs as there are clauses in which x_i and x_i' appear. The inputs to the variable circuit-assembly for x_i are divided into two sets: $P_i = \{X_{i_j}\}$ and $P_i' = \{X_{i_j}'\}$. Each input in P_i (P_i') is connected to the output of a clause assembly in which x_i appears uncomplemented (complemented). For all $X_{i_j} \in P_i, \alpha_{X_{i_j},B_i} = 1, \beta_{X_{i_j},B_i} = 0.4, \gamma_{X_{i_j}} = 10$. Also, for all $X_{i_j}' \in P_i', \alpha_{X_{i_j}',B_i} = 3, \beta_{X_{i_j}',B_i} = 0, \gamma_{X_{i_j}'} = 10$.

The clause circuit-assembly for a clause $\mathcal{C}_j$ consists of a single-input cell C_j. The input Z_j of C_j is a PI of η. The output of C_j fans out to three pins. If the clause $\mathcal{C}_j$ has literals x_g, x_h and x_i', the output of C_j is connected to pins X_{g_j}, X_{h_j} and X_{i_j}' of cells B_g, B_h, and B_i respectively (see Figure 2 – ignore the buffers). For all j, C_j has $\alpha = 0.5, \beta = 0.025$, and $\gamma = 0$.

The circuit η thus constructed has m primary inputs, Z_j, (number of clauses in the 3SAT instance) and n primary outputs, A_i, (number of variables in 3SAT). The net topologies are fixed, as shown in Figure 2. The construction of η clearly takes time polynomial in n and m. We now consider a buffer library with a single buffer type b that has $\alpha_b = 1, \beta_b = 0.1, \gamma_b = 0$. Thus, there are 2 buffering choices for each net (B_i, A_i), and $2^3 = 8$ choices for the net rooted at C_j. The required time $r(A_i)$ at each PO A_i is set to 5, the capacitive load at each PO is set to 0, and we choose $D = -1$. This completes the instance definition for GFO-NTF.

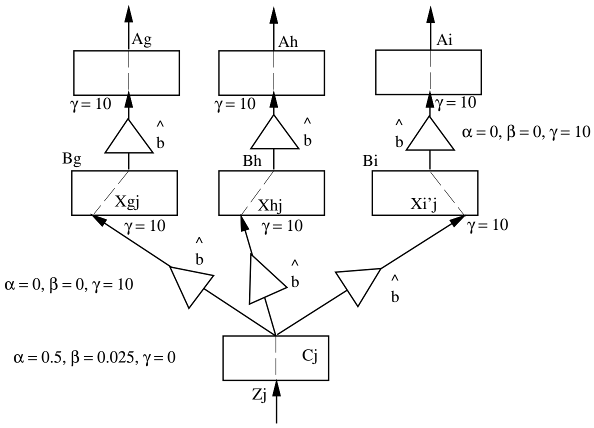

Figure 3: Circuit-assembly for clause $\mathcal{C}_j$

It was shown in [6] that $\mathcal{C}$ *is satisfiable if and only if there exists a buffering of the nets of* η *such that for all primary inputs* Z_j *of* η*,* $r(Z_j) \geq -1$.

5 Gate Resizing Under LDDM

Although gate resizing under LIDM in circuits with single-output gates is polynomially solvable, Section 3.2 leaves open the question of the complexity of the problem under LDDM. We settle this question by showing that it is NP-complete. For LDDM, this is the strongest complexity result, since our problem instance is a special case of the constrained versions of the problem mentioned in Section 3.1.2. Thus, our proof also proves that those problems are NP-complete under LDDM.

It must be noted here that Rudell [8] and Touati [9] have addressed the problem of optimum technology mapping for tree circuits under LDDM. They proposed an optimum dynamic programming algorithm that considered the possibility of different capacitive loads at the fanout of a gate. However, no result or algorithm was presented for general combinational circuits, which are directed acyclic graphs.

We proceed by formulating GATE RESIZING, the decision problem version of gate resizing.

INSTANCE: *Given a combinational circuit* η *consisting of gates from a gate library* $\mathcal{L}$ *that contains several sizes for each gate, and a number* D*.*

QUESTION: *Does there exist a resized circuit, obtained by replacing each gate of* η *with one of a different or same size from* $\mathcal{L}$*, with its delay at most* D*?*

Theorem 5.1 *GATE RESIZING is NP-complete under LDDM.*

Proof: It is easy to see that GATE RESIZING is in NP. Given each gate size, the circuit delay can be computed in linear time by a forward trace and then compared with D.

For NP-completeness, we use a transformation from 3SAT. We construct an instance of η and a gate library with sizes and parameters for each gate. Intuitively, the proof parallels the NP-completeness proof of GFO-NTF with two main differences:

1. The circuit η is the same as the one in the GFO-NTF proof, except that we introduce a gate $\hat{b}$ on each internal edge, i.e., edges (B_i, A_i) and (C_j, B_i). This is shown in Figure 3. The

gate $\hat{b}$ has one input and one output pin, and has the following parameters:

$$\alpha_{\hat{b}} = 0, \beta_{\hat{b}} = 0, \gamma_{\hat{b}} = 10.$$

The gate $\hat{b}$ can be resized to b, where b is the buffer used in the GFO-NTF circuit, i.e.,

$$\alpha_b = 1, \beta_b = 0.1, \gamma_b = 0.$$

We assume each gate A_i, B_i, and C_j has only one size available in the library, with the same α, β, and γ parameters as in the GFO-NTF circuit.

Since $\gamma_{\hat{b}}$, the input capacitance of $\hat{b}$, is the same as that of the input pins of gates A_i and B_i (i.e., 10 units), and since the delay through $\hat{b}$ is zero (since $\alpha_{\hat{b}} = \beta_{\hat{b}} = 0$), *$\hat{b}$ models the absence of a buffer on the corresponding edge in the GFO-NTF circuit.* Resizing a particular $\hat{b}$ to b models the insertion of b on the corresponding edge.

2. In gate resizing, the goal is to minimize the maximum arrival time at the primary outputs. In fanout optimization, the goal is to maximize the minimum required time at the primary inputs. It is easy to see that these are two different ways of looking at the same circuit delay. In the GFO-NTF problem, the required times at the primary outputs were set to 5, and we wished to determine if the minimum required time at the primary inputs was at least -1. For the gate resizing problem, we set the arrival times of all primary inputs to -1 and ask: *Does there exist a resized circuit with the maximum primary output arrival time at most 5 (i.e., circuit delay at most 6)?*

With these two modifications, the correspondence between the GFO-NTF circuit and the GATE RESIZING circuit is established. It can be checked that *required times at the primary inputs of the GFO-NTF circuit are at least -1 if and only if there exists a resized circuit with delay at most 6.* From the proof of NP-completeness of GFO-NTF, it follows that the instance of 3SAT is satisfiable if and only if there exists a resized circuit with delay at most 6. This completes the NP-completeness proof of GATE RESIZING.

5.1 Complexity of Technology Mapping

We now consider the complexity of minimum-delay technology mapping (or DAG covering) under LDDM. Traditionally, in technology mapping, the given circuit is decomposed into simple two-input gates before it is covered with library patterns. If we allow gates with an arbitrary number of inputs to be present in the acyclic graph on which covering is done, gate resizing can be seen as a special case of technology mapping. Of course, we also need to ensure that the library does not contain a gate that is a composition of two or more gates in the circuit. Under these conditions, minimum-delay technology mapping under LDDM reduces to the gate resizing problem, and hence is NP-complete.

6 Gate Resizing with Load Constraints

We now focus on the gate resizing problem with load constraints specified for all gate outputs. As described in Section 2.1, the load constraint λ of an output pin specifies the maximum cumulative capacitance the pin can drive without compromising electrical characteristics of the signal (voltage level, slew, delay, etc). Any practical gate resizing solution must satisfy the load constraints of all gates in the circuit.

Theorem 5.1 implies that gate resizing under load constraints is NP-complete under LDDM. In this section, we show that the

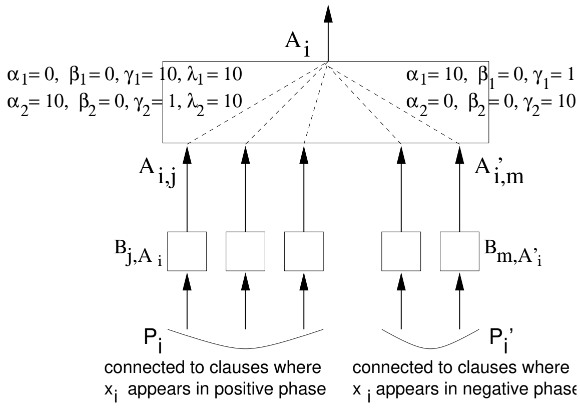

Figure 4: Variable circuit-assembly for variable x_i.

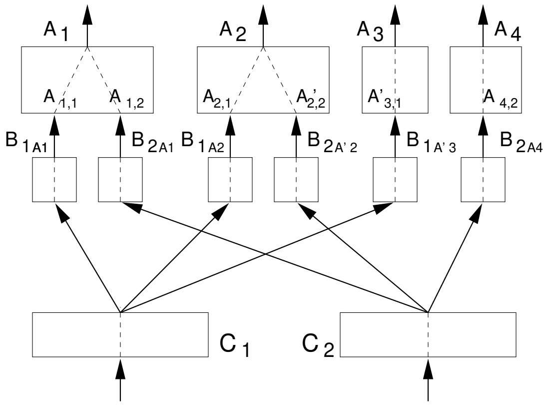

Figure 5: Circuit for $\mathcal{C} = (x_1 + x_2 + x_3')(x_1 + x_2' + x_4)$.

problem is NP-complete even under LIDM. We use GATE RESIZING, the decision problem described in Section 5, for the NP completeness proof.

Theorem 6.1 *GATE RESIZING with load constraints is NP-complete under LIDM.*

Proof: Given a solution (i.e., sizes of all gates in the circuit), the circuit delay can be computed in time linear in the size of the circuit (number of gates and wires). This can then be compared with D (the number in the GATE RESIZING problem) in constant time. Also, for each gate, the cumulative load driven by its output can be computed in time linear in the size of the circuit, and then compared with its load constraint in constant time. Therefore, GATE RESIZING with load constraints is in NP.

To prove NP-completeness, we reduce 3SAT to GATE RESIZING with load constraints. Given an instance $\mathcal{C}$ of 3SAT, we construct a circuit η similar to that used in the proof of Theorem 4.1. However, for each variable x_i, the variable circuit-assembly is now as shown in Figure 4.

If there are k clauses in which x_i appears in complemented or uncomplemented form, the variable circuit-assembly for x_i consists of one $k-$input cell labeled A_i and k single-input cells labeled B_{j,A_i} or $B_{j,A_i'}$. The inputs of A_i are divided into two sets: P_i, consisting of all inputs $A_{i,j}$ connected (through cells of type B) to a clause assembly C_j in which x_i appears in uncomplemented form, and P_i' consisting of all other inputs of A_i. The output of B_{j,A_i} (or

B_{j,A'_i}) is connected to input $A_{i,j}$ (or $A'_{i,j}$) of A_i. As an example, Figure 5 shows the circuit for $\mathcal{C} = (x_1 + x_2 + x'_3)(x_1 + x'_2 + x_4)$.

The gate library $\mathcal{L}$ is defined as follows. For each clause circuit-assembly C_j, there are two cell sizes, C^1 and C^2, with parameters as shown in Table 1 (a). For each cell B_{j,A_i} or B_{j,A'_i}, there are also two choices of cell sizes, with parameters as shown in Table 1 (b). Finally, for each variable x_i, there are two cell sizes for A_i, with parameters as shown in Table 2.

Cells for C_j	α	β	γ	λ	Cells for $B_{j,A_i/A'_i}$	α	β	γ	λ
C^1	0	0	0	5	B^1	5	0	2	10
C^2	7	0	0	6	B^2	0	0	1	1

(a) (b)

Table 1: Cell sizes for C and B type cells.

Cells for A_i	Input pin X	α_{X,A_i}	β_{X,A_i}	γ_{X,A_i}	λ
A_i^1	$X = A_{i,j} \in P_i$	0	0	10	10
	$X = A'_{i_j} \in P'_i$	10	0	1	
A_i^2	$X = A_{i_j} \in P_i$	10	0	1	10
	$X = A'_{i_j} \in P'_i$	0	0	10	

Table 2: Cell sizes for A type cells

We now show that $\mathcal{C}$ *is satisfiable if and only if there exists a gate resizing satisfying all load constraints such that the circuit delay is at most 10*. In other words, $D = 10$ in the GATE RESIZING problem.

[Only if]: Assume $\mathcal{C}$ is satisfiable and let $\mathcal{A}$ be a satisfying truth assignment. We derive a gate resizing of the circuit η from $\mathcal{A}$ as follows. If variable x_i is assigned the value *true*, we choose A_i^2 as the cell size for A_i. Otherwise, we choose A_i^1 for A_i. Given this choice of A cells, we choose B cells as follows. For each input pin of A_i that has $\gamma = 10$, we choose B^1 as the cell size for the B cell connected to it. For all other B cells connected to inputs A_i (these have $\gamma = 1$), we choose B^2 for the cell size. Finally, we choose C^1 for all clause circuit-assemblies C_j. Note that our choice of B cell sizes guarantees that load constraints of all B cells are satisfied.

Consider a clause $\mathcal{C}_j = (x_g + x_h + x'_i)$ in $\mathcal{C}$. At least one literal in $\mathcal{C}_j$ must evaluate to *true* under the assignment $\mathcal{A}$. Without loss of generality, let this be x_g. From our choice of cell sizes for A_g, it is now easy to see that: (i) the delay from input $A_{g,j}$ to the output of A_g is 10, and (ii) the capacitance of input $A_{g,j}$ is 1. Our choice of B cells then implies that B^1 is chosen for B_{j,A_g}. From Table 1 (b), the delay of B^1 is seen to be 0, so the delay from the input of B_{j,A_g} to the output of A_g is $10 + 0 = 10$. Also, the capacitance at the input of B_{j,A_g} is 1. The same analysis holds for all literals in $\mathcal{C}_j$ that evaluate to *true* under $\mathcal{A}$.

Now consider the case where x_g evaluates to *false* under A. Using the same notation as above, we find that: (i) the delay from input $A_{g,j}$ to the output of A_g is 0, and (ii) the capacitance of input $A_{g,j}$ is 10. With our choice of B cells, the delay from the input of B_{j,A_g} to the output of A_g is now $0 + 5 = 5$ and the input capacitance of B_{j,A_g} is 2. The same analysis holds for all literals in $\mathcal{C}_j$ that evaluate to *false* under $\mathcal{A}$.

Since at least one literal in $\mathcal{C}_j$ must evaluate to true, the sum of the input capacitances of B_{j,A_g}, B_{j,A_h} and B_{j,A'_i} is bounded above by $2+2+1 = 5$. Therefore, by choosing C^1 for the cell size of C_j, the load constraint of C_j is satisfied. In addition, since C^1 has 0 input to output delay, the maximum delay from the input of C_1 to the outputs of A_g, A_h or A_i is $\max(10, 5, 5)$ or $\max(10, 10, 5)$ or $\max(10, 10, 10)$, depending on the number of literals that evaluate to *true*. Thus, in each case, the maximum delay is 10.

The above analysis holds for all clauses C_j. So the circuit delay is 10 and all load constraints are satisfied.

[If]: Assume there exists a gate resizing solution such that the circuit delay is at most 10. We first show that the smallest value of the circuit delay satisfying all load constraints is at least 10, so the gate resizing solution has a circuit delay of exactly 10.

We prove that the smallest value of the circuit delay is ≥ 10 by contradiction. Assume it is < 10. This implies that no input to output path of any A cell has a delay of 10. Given the choices for A cells, this implies that the capacitance of each input of each A cell is 10. Since B^2 type cells have a drive capacity of 1, all B cells must be of type B^1 (drive = 10) to satisfy load constraints. Therefore, the delay from the input of a B cell to the output of an A cell is $5 + 0 = 5$, and the input capacitance of all B cells is 2. Since each C cell drives three B cells, each C cell must have a drive capacity at least 6 to satisfy load constraints. Therefore, C^2 is the only choice for C cells. But, with this choice, the delay from the input of any C cell to the output of an A cell is $5 + 7 = 12 > 10$ – a contradiction. Therefore, the smallest value of the circuit delay must be ≥ 10.

We now obtain a truth assignment from the gate resizing solution as follows. If A_i^1 is the chosen size of cell A_i, then variable x_i is assigned the value *false*. Else, x_i is assigned the value *true*. We claim that the resulting truth assignment, $\mathcal{A}$, satisfies each clause $\mathcal{C}_j$ of the 3SAT expression $\mathcal{C}$.

Following the arguments used in the proof of the smallest value of the circuit delay, it can be shown that the maximum delay, δ_j, from the input of C_j to the outputs of A cells in its transitive fanout is ≥ 10. Since the circuit delay under the given resizing is 10, δ_j must be ≤ 10 as well. Therefore, $\delta_j = 10$, and this applies to all clause circuit-assemblies C_j. The proof is now completed by showing that if δ_j equals 10, clause $\mathcal{C}_j$ in the original 3SAT expression is satisfied under $\mathcal{A}$.

We prove this by contradiction. Assume that the maximum delay, δ_j, from the input of clause assembly C_j to the outputs of A cells in its transitive fanout is 10, but clause $\mathcal{C}_j$ is not satisfied under $\mathcal{A}$. Since $\mathcal{C}_j$ is not satisfied, all three literals in $\mathcal{C}_j$ must evaluate to *false*. Without loss of generality, consider a literal x_g in $\mathcal{C}_j$. Since this evaluates to *false*, cell type A_g^1 must have been chosen for cell A_g. Recall that $A_{g,j}$ is the input of A_g that is connected to C_j through B_{j,A_g}. Given our choice of A cells, the capacitance at input pin $A_{g,j}$ is 10, and the input to output delay from input pin $A_{g,j}$ to the output of A_g is 0. In order to satisfy load constraints, cell size B^1 (drive = 10) must have been chosen from cell B_{j,A_g}. This has a delay of 5 and an input capacitance of 2. So the delay from the input of B_{j,A_g} to the output of A_g is $5 + 0 = 5$. The same reasoning holds for all three literals in clause $\mathcal{C}_j$. Thus, the total capacitance driven by cell C_j is $3 \times 2 = 6$. From Table 1 (a), it is clear that cell size C^2 (drive = 6) must have been selected for C_j. However, since this has an input to output delay of 7, the maximum delay from the input of C_j to the output of A_g (A_i) is $5 + 7 = 12$, which contradicts our premise of a maximum delay of 10. This proves the claim and hence the theorem.

6.1 A Heuristic Solution

Although gate resizing under load constraints is NP-complete, practical gate resizing algorithms must consider load constraints in order to avoid insidious signal integrity problems. To address

ex	#gates	area (BC)	o.d. (ns)	n.d. (ns)	ΔA (BC)	CPU (sec)
ex1	356	1567	7.84	5.16	187	3
ex2	17.1K	122.6K	7.44	7.32	112	33
ex3	40.0K	200.2K	11.38	9.03	676	75
ex4	86.7K	381.6K	18.41	12.13	1157	196
ex5	172.2K	718.6K	56.36	48.62	583	167

1K = 1000, 1 BC = area of smallest inverter in the library.
o.d. = original delay; n.d. = new delay after resizing.
ΔA = area penalty due to resizing.
All benchmarks are in 0.35-μ technology.

Table 3: Experimental Results

this issue, we outline an iterative heuristic technique for gate resizing under load constraints. It is a simple enhancement of well-known greedy resizing schemes (e.g., [3]), using load constraints to eliminate certain replacements.

In our solution, we first perform a static timing analysis of the circuit and identify candidate gates for potential resizing. These are the ϵ-critical gates in the circuit, i.e., gates whose slack lie within ϵ of the circuit slack. For each candidate gate, a set of replacement gates is chosen from the library. Each replacement gate is then checked for load constraint violations:

1. The drive of the replacement gate must be at least as large as the total capacitance driven by the gate output, and
2. input pin capacitances of the replacement gate must not cause load constraint violations for the corresponding driver gates.

If both conditions are satisfied, the replacement gate is evaluated based on an appropriately chosen cost function. The function takes into account the local performance gain (i.e., improvement in slack at the gate output) potentially obtainable by replacing the current gate with the replacement gate, and the associated area penalty. The "best" replacement is then chosen: this is the one that gives the minimum value of the cost function. Candidate gates and their best replacements are then ranked, and the replacements are applied in order of their ranks. During each replacement, load constraint violation checks are made again, and any replacement that fails these checks (because of earlier replacements) is rejected.

Finally, we perform timing analysis to compute the new circuit delay after the replacements. If the circuit delay improves, the resizing iteration is repeated (since critical gates may have changed). Otherwise, a suitable subset of resizings from the last iteration are undone, until the circuit delay becomes equal to or less than what it was before applying the replacements.

Clearly, this is a greedy strategy and may fail to find an optimal gate resizing solution. The quality of the results depends on the cost function used and the order in which replacements are applied.

7 Experimental Results

We have implemented the heuristic solution proposed in Section 6.1, and have performed experiments with a set of optimized, mapped, placed and globally routed industrial designs. The delay model used in our experiments incorporates both gate and wiring delay. Elmore delay was used for estimating wire delays, while gate delays were measured using a generalization of LDDM [1]. In the generalized model, the pin-to-pin delay of a gate is a linear function of its intrinsic delay, the load capacitance at the output and the slew of the input signal.

Table 3 shows the relevant statistics of the designs used in our experiments. This includes the number of gates (a gate could be as simple as an inverter or as complex as an 8-bit adder), the total gate area of the design (in terms of the smallest inverter in the library), and the original circuit delay. The two largest designs, `ex4` and `ex5`, are *hi-vision TV encoder/decoder* designs; `ex5` has 172K gates. In each case, the original circuit satisfied the load constraints of all gate outputs, where the capacitive load includes the input pin capacitances of the fanout gates as well as the wire load. The last three columns of the table show the results of applying our heuristic. The optimized delay, area penalty due to resizing, and CPU time for resizing each circuit are shown in these columns. Our experiments show that appreciable delay reduction (up to 35%) can be obtained at a very small area penalty (less than 0.4% for all benchmarks except `ex1`) using a greedy gate sizing heuristic. The experiments were done on an Ultrasparc 60 with 768MB RAM. The run-times are quite reasonable: each example completes in less than four minutes. We have also verified that all pins of the final circuit after gate resizing satisfy their respective load constraints. However, we do not know how far our solutions are from the optimum (absolute minimum delay).

8 Conclusions

The complexity results in this paper focussed only on load-independent and load-dependent delay models. As mentioned in the last section, there is a more general class of delay models in which the delay of a gate depends both on the capacitive load at its output and on the slew of input signals [1]. Since the slew-dependent delay models are a generalization of the load-dependent and load-independent models, the complexity results presented here continue to hold under the input-slew based models.

Based on earlier results (see Section 3) and the new ones presented in this paper, we find that gate resizing is polynomially solvable only under the most simplistic and restricted scenario: single-output gates, load-independent delay model, identical rise and fall delays, same min and max constraints, no area constraints and no load constraints. Introduction of any reasonable practical constraints renders the problem intractable. Thus, we conclude that practical gate resizing is inherently a hard problem!

References

[1] D. Auvergerne, N. Azemard, D. Deschacht, and M. Robert. Input Waveform Slope Effects in CMOS Delays. In *IEEE J. of Solid-State Circuits*, 1990.

[2] P. Chan. Algorithms for Library-specific Sizing of Combinational Logic. In *DAC*, pages 353–356, 1990.

[3] O. Coudert, R. Haddad, and S. Manne. New Algorithms for Gate Sizing: A Comparative Study. In *DAC*, pages 734–739, 1996.

[4] Y. Kukimoto, R. K. Brayton, and P. Sawkar. Delay-Optimal Technology Mapping by DAG Covering. In *DAC*, pages 348–351, 1998.

[5] W. N. Li, A. Lim, P. Agarwal, and S. Sahni. On the Circuit Implementation Problem. In *DAC*, pages 478–483, 1992.

[6] R. Murgai. On The Global Fanout Optimization Problem. In *ICCAD*, pages 511–515, 1999.

[7] R. Murgai. Performance Optimization Under Rise and Fall Parameters. In *ICCAD*, pages 185–190, 1999.

[8] R. Rudell. *Logic Synthesis for VLSI Design*. PhD thesis, UC Berkeley, April 1989. UCB/ERL M89/49.

[9] H. Touati. *Performance-oriented Technology Mapping*. PhD thesis, UC Berkeley, November 1990. UCB/ERL M90/109.