A New Class of Convex Functions for Delay Modeling and their Application to the Transistor Sizing Problem

Kishore Kasamsetty, Mahesh Ketkar and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455, USA

Abstract— This paper derives a methodology for developing accurate convex delay models to be used for transistor sizing. A new rich class of convex functions to model gate delay is presented and the circuit delay under such a model is shown to be equivalent to a convex function. The richness of these functions is exploited to accurately model gate delay for modern designs. Since the delay under this model is a convex function, optimal sizing algorithms based on convex programming techniques are applied with the new delay model. Experimental results demonstrating the accuracy of proposed model are presented along with results of sizing various test circuits.

I. Introduction

The transistor sizing problem is a circuit optimization problem that is important in high performance designs. By suitably selecting the sizes of the transistors in the circuit, a designer may meet the required timing goals. The problem has traditionally been formally defined as [1]:

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

There have been many significant attempts to solve this problem, for example, [1], [2], [3]. Most published approaches use the Elmore delay model [4], [5] for timing calculations, and a breakthrough observation in [1] was that the circuit delay under this model is a posynomial function (to be defined later) of the transistor sizes. The advantage of this functional form is that under a simple variable transformation, the problem can be transformed into a convex optimization problem for which it is guaranteed that any local minimum is also a global minimum. This is a powerful property since it precludes the need for any hill-climbing approaches to get out of local minima. Based on this property, the method in [1] used a greedy heuristic that tried to achieve this global minimum. A more formal and exact convex optimization algorithm was employed in [3], guaranteeing that the global minimum would be found.

It is generally accepted that the use of the Elmore delay model at the transistor level is very inaccurate for modern designs, and even exact optimization under this model may lead to a wrong solution to the sizing problem since the timing model has a bad correlation with reality. More precisely, the solution may be suboptimal in that it meets the timing specification without minimizing the cost function, or entirely inaccurate, in the sense that it may not meet the timing constraints at all. Therefore, it is immensely important to use more accurate timing models in sizing.

Several approaches for accurate timing modeling have been proposed in the past. For example, one could model gate delays by developing closed form expressions (for example, [6], [7]). Alternatively, a look-up table could be constructed using experimentally derived delay data for various configurations, with intermediate data points being derived by interpolation methods, as in the delay model in [8]. The data point spacing determines the accuracy of the model and hence, there is a tradeoff between desired levels of accuracy and the amount of data storage, and achieving good accuracy may be prohibitively expensive in terms of memory requirements. Neither the closed-form modeling approach nor the table-look-up modeling method is particularly well suited for optimization since the modeling functions typically do not possess any convexity properties and cannot be used in the context of a formal optimization algorithm that is guaranteed to find the global minimum in a reasonable time. Moreover, it is not necessarily true that these models will have continuous derivatives, or, in the case of look-up tables, any derivative at all. Therefore, there is a need for new models that permit accurate delay computations, while maintaining convexity properties suited for optimization. This work derives a methodology for developing such models.

The theoretical underpinning of this approach is a result that defines a new class of provably convex functions that are shown to work well for modeling circuit delays. A formal proof of convexity is shown, and it is to be emphasized that the implications of this proof are more far-reaching than the immediate problem of delay modeling. The set of functions from which these convex functions are chosen is more general than posynomials in that the class of posynomials is a subset of this class; therefore, we refer to these functions as *generalized posynomials*. It is likely that these functions may have applications in other optimization problems in other fields, just as posynomials have been used in many fields other than circuit optimization. This work uses a curve-fitting approach to find a least-squares fit from the delay function, computed by SPICE over a grid, to a generalized posynomial in order to provide guarantees on accuracy of the delay model while using functions that are well-behaved in an optimization context.

Before describing the model, a few words on the mathematical notation are in order. While these are standard notational techniques, their explicit explanation at this point is intended to remove any possibly ambiguities. The set of real numbers is denoted by $\mathbf{R}$, and the set of positive reals by $\mathbf{R}^+$. An n-dimensional real space is represented by the symbol $\mathbf{R}^n$, and an n-dimensional vector, $\mathbf{x} = [x_1, \cdots x_n]$, as $\mathbf{x} \in \mathbf{R}^n$.

The outline of this paper is as follows. Section II presents

This work is supported in part by a gift from Intel Corporation, by the NSF under contract CCR-9800992 and the SRC under contract 98-DJ-609.

some general mathematical background on posynomials and convexity. The idea of a generalized posynomial is presented in Section III and its application to circuit delay modeling is discussed. The actual problem we solve is given in the Section IV and the proof of convexity of the delay model for the entire circuit is provided in Section V. This is followed by experimental results in Section VI and concluding remarks in Section VII.

II. Background

A. Delay abstractions

The delay characteristics of the output waveform at a gate may be represented by two numbers:
(1) the *delay*, i.e., the difference in the time when the output waveform crosses 50% of its final value, and the corresponding time for the input waveform.
(2) the *output transition time*, i.e., the time required for the waveform to go from 10% to 90% of its final value.

For a capacitive load, the delay and output transition time of a gate of a fixed size can be precharacterized as a function of input signal transition time, τ, and the load capacitance, C_L.

B. Convex optimization

A convex programming problem, also referred to as a convex optimization problem, involves the minimization of a convex function over a convex set (for definitions of these terms, the reader is referred to [9]). A problem of the type

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{such that} & g_i(\mathbf{x}) \leq 0, 1 \leq i \leq m \\ & \mathbf{x} \in \mathbf{R}^n \end{array} \tag{2}$$

is a convex programming problem if $f(\mathbf{x})$ and $g_i(\mathbf{x}), 1 \leq i \leq m$, are convex functions. Since all linear functions are convex, a linear programming problem is a special case of a convex program, where the functions f and g_i are all linear.

It is desirable to attempt to express any optimization problem using a convex formulation, as far as possible, under the caveat that the accuracy of the modeling functions for the objective and the constraints must be preserved. In the context of transistor sizing, this requires the derivation of convex closed-form expressions for the path delay.

C. Posynomial delay modeling

In much of the previous work on transistor sizing, the circuit delay has been expressed in the form of a class of functions known as posynomials. A posynomial is a function p of a positive variable $\mathbf{x} \in R^n$ that has the form

$$p(\mathbf{x}) = \sum_j \gamma_j \prod_{i=1}^{n} x_i^{\alpha_{ij}} \tag{3}$$

where the exponents $\alpha_{ij} \in \mathbf{R}$ and the coefficients $\gamma_j \in \mathbf{R}^+$. Simply stated, this means that a posynomial is similar to a polynomial, with the difference that each term must have a positive coefficient, and the exponents associated with the variables may be arbitrary real numbers that could be positive, negative, or zero. In the positive orthant in the $\mathbf{x}$ space, posynomial functions have the useful property that they can be mapped onto a convex function through an elementary variable transformation, $(x_i) = (e^{z_i})$.

The Elmore delay model used, for example, in TILOS [1] and iCONTRAST [3], used the following form of expressions for the path delay.

$$D(\mathbf{x}) = \sum_{i,j=1}^{n} a_{ij} \frac{x_i}{x_j} + \sum_{i=1}^{n} \frac{b_i}{x_i} + K \tag{4}$$

where $a_{ij}, b_i, K \in \mathbf{R}^+$ are constants and, $\mathbf{x} = [x_1, \cdots, x_n]$ is the vector of transistor sizes. Notice that the Elmore delay expressions are a subset of the set of posynomials; specifically they are posynomials whose exponents belong to the set {-1,0,1}.

III. Delay modeling using generalized posynomials

A. Generalized posynomials

Posynomials and convex functions are a rich class of functions and the basic motivation for this work is that better delay estimates can be obtained by fully exploiting this richness. In this work, we introduce a generalized posynomial form that is useful for delay modeling and can be proven to have convexity properties.

A generalized posynomial function $G_k(\mathbf{x}), \mathbf{x} \in \mathbf{R}^n$, where $k \geq 0$ is called the order of the function, is defined recursively as follows:
1. A generalized posynomial of order 0, G_0, is the posynomial form defined earlier:

$$G_0(\mathbf{x}) = \sum_j \gamma_j \prod_{i=1}^{n} x_i^{\alpha_{ij}}, \tag{5}$$

where the exponents $\alpha_{ij} \in \mathbf{R}$ and the coefficients $\gamma_j \in \mathbf{R}^+$.
2. A generalized posynomial of order $k \geq 1$ is defined as

$$G_k(\mathbf{x}) = \sum_j \gamma_j \prod_{i=1}^{n} [G_{k-1,i}(\mathbf{x})]^{\alpha_{ij}}, \tag{6}$$

where the exponents $\alpha_{ij} \in \mathbf{R}^+$ and the coefficients $\gamma_j \in \mathbf{R}^+$, and $G_{k-1,i}(\mathbf{x})$ is a generalized posynomial of order $k-1$.

Specifically, the generalized posynomial of first order, is given by

$$f(\mathbf{x}) = \sum_i \gamma_i \prod_{j=1}^{m} \left(\sum_{l=1}^{p_i} \omega_{ijl} \prod_{s=1}^{n} x_s^{a_{ijls}} \right)^{\beta_{ij}} \tag{7}$$

where each $\beta_{ij} \in \mathbf{R}^+$, each $a_{ijls} \in \mathbf{R}$, each $\gamma_i \in \mathbf{R}^+$, and each $\omega_{ijl} \in \mathbf{R}^+$.

Stripping Equation (7) of its complicated notation, one may observe that the term in the innermost bracket,

$$\sum_{l=1}^{m} \omega_{ijl} \prod_{s=1}^{p} x_s^{a_{ijls}}, \tag{8}$$

represents a posynomial function. Therefore, a generalized posynomial is similar to a posynomial, except that the place of the x_i variables in Equation (3) is taken by an arbitrary posynomial. Clearly, since x_i is a posynomial, the class of posynomials is a proper subset of the class of generalized posynomials. Similarly, a generalized posynomial of order k uses a generalized posynomial of order $k-1$ in place of the x_i variables in Equation (3).

Example: Consider a posynomial function

$$f_1(x_1, x_2, x_3) = 5x_1^{0.3}x_2^{-1.732} + 2.4x_2^2x_3^{-1}$$

This is also a generalized posynomial since posynomials are a subset of this class. An example of a generalized posynomial that is *not* a posynomial is the function

$$f_2(x_1, x_2, x_3) = 9.8(7.6x_1^{7.6} + 5.4x_2^{5.4}x_3^{-5.4} + 3.2)^{1.09}$$
$$2.7(x_1x_2^{-1} + x_1^2x_2^3)^{0.9} + 34.5(1.2x_1x_2^{-3.3} + 3.4x_1x_2x_3)^{1.732}$$

The following theorem parallels the relationship between posynomials and convex functions.

Theorem 1: If the range of interest of $f(\mathbf{x})$ is restricted to the positive orthant where each $x_i > 0$, then under the variable transformation from the space $\mathbf{x} \in \mathbf{R}^n$ to the space $\mathbf{z} \in \mathbf{R}^n$ given by $x_i = e^{z_i}$, the generalized posynomial function f of equation (6) is a convex function in the variable set $\mathbf{z}$.

Proof: Refer to Appendix A.

B. Delay estimation

B.1 Outline of the delay modeling approach

Many commonly utilized delay estimation approaches, such as those used for standard cell characterization, estimate the delay of a gate for a given input transition time and output load capacitance, with the sizes of the transistors inside the gate being kept constant. However, for our purposes, any sizing procedure requires the timing model to capture the effects of varying the transistor sizes on the gate delays. This causes the number of parameters for the delay function to increase, making the problem of delay modeling for sizing algorithms more complex. These input parameters will be referred to as characterization variables.

We begin with an explanation of the timing model for an inverter, such as the one shown in Figure 1; this model is generalized to complex gates in subsequent sections. The aim is to be able to estimate delay as a function of the pmos and nmos transistor widths, w_p and w_n, the input transition time τ, and the output load capacitance, C_L. Therefore, for an inverter, w_p, w_n, τ, and C_L form the set of characterization variables. These variables reflect the set of variables that are generally considered to be important in defining the delay of a gate in most models.

We attempted the use of several types of functions that have generalized posynomial form, to achieve the desired levels of accuracy. The general form of expression that provided consistently good results for different gate types is a first order generalized posynomial given by

$$\text{Delay} = \sum_{j=1}^{m} P_j \cdot \prod_{i=1}^{n} (x_i^{\Delta} + c_{ij})^{\beta_{ij}} + C \qquad (9)$$

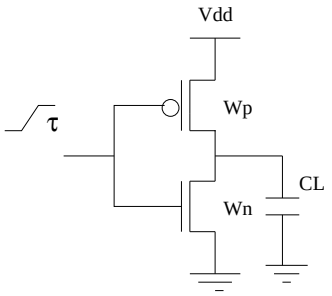

Fig. 1. Inverter circuit

Here, the x_i's are characterization variables, and the c_{ij}'s, β_{ij}'s, C, and P_j's are real constants, referred to collectively as *characterization constants*. The parameter Δ is set to either -1 or 1, depending on the variable, as will soon be explained. We will show in Section V that the use of this form of function implies that the circuit delay can be expressed as a generalized posynomial function of the transistor widths. The problem of characterization is that of determining appropriate values for the characterization constants.

Due to the curve-fitting nature of the characterization procedure (akin to standard cell characterization), it is not possible to ascribe direct physical meanings to each of these terms. However, it can be seen that the fall delay increases as C_L, w_p and τ are increased, and decreases as w_n is increased, implying that the value of Δ for the first three variables must be 1, and that for w_n should be -1. Note that this is not as restrictive as the Elmore form since, among other things, the β_{ij}'s provide an additional degree of freedom that was not available for the Elmore delay form. A similar argument may be made for the rise delay case.

B.2 Circuit simulations and curve-fitting

A two-step methodology is adopted to complete the characterization. In the first step, a number of circuit simulations are performed to generate points on a grid. In the second, a least-squares procedure is used to fit the data to a function of the type in Equation (9).

A series of simulations is performed to collect the experimental data using the HSPICE circuit simulator [10]. The total number of data points, N, increases exponentially with the number of characterization variables. For the inverter circuit with four characterization variables and d data points for each variable to cover the range of interest, the total number of data points, N would be d^4. The issue of choosing the set of data points is similar to the problem of choosing data points for the characterization of standard cells, where the delay is characterized in terms of the output load and input slope. In this case, we have the same problem but with more variables, and similar methods are used to cope with the problem. For example, it is not necessary to choose an even grid for the transistor widths and a smaller granularity of points can be chosen for larger w_n's in case of the fall transition.

The determination of the characterization constants was performed by solving the following nonlinear program that

minimizes the sum of the squares of the percentage errors over all data points.

$$\text{minimize} \sum_{i=0}^{N} \left[\frac{D_{estim}(i) - D_{actual}(i)}{D_{actual}(i)} \right]^2 \qquad (10)$$

where N is the number of data points, $D_{estim}(i)$ and $D_{actual}(i)$, respectively, represent the values given by Equation (9), and the corresponding measured value at the i^{th} data point. This nonlinear programming problem is solved using the MINOS optimization package [11] to determine the values of characterization constants.

C. Characterization of a set of primitives

To illustrate the problem of directly extending the above methodology from inverters to arbitrary gates, we consider a three-input NAND gate circuit. The characterization variables for this gate will be the sizes of the three pmos transistors, sizes of the three nmos transistors, τ and C_L. If five data points were chosen to cover each of these variables, we would have total $5^8 = 32768$ data points. It is computationally expensive to perform such a large number of simulations and generate this database for curve fitting. For more complex gates, as the number of data points increases exponentially with the number of transistors in the gate, this would lead to a large overhead, both in terms of the simulation time and the time required to perform the curve-fit. Therefore, an alternative strategy is suggested. We emphasize even under this procedure, the transistor sizing approach will size each transistor individually, and this method is only used for delay estimation. It is important to emphasize that the use of these mapping strategies only serves to reduce the complexity of the characterization procedure. If one is willing to invest the CPU time required to perform the characterizations for each gate type, then this procedure is unnecessary. In a library-based design, this full characterization of all cells is a viable alternative for a fixed library and its complexity is comparable to characterizing the library using any other means.

In our approach, the gate whose delay is to be measured is mapped to the closest precharacterized logic structure and the delay is calculated. A secondary advantage of this approach is that the delay model is not restricted to a fixed library type, and arbitrary gate types can be handled.

One straightforward technique that may be used is to map all of the gates to an "equivalent inverter" [12], and use the inverter characterization to estimate delays; the sizes of the pull-down nmos transistor and the pull-up pmos transistor of this inverter reflect the real pull-down or pull-up path in the gate. The widths of these new transistors are referred to as the equivalent widths. The equivalent width calculation is based on modeling the 'on' transistors as conductances, and the equivalent width corresponds to the effective conductance of the original structure. Accordingly, if two transistors of widths w_1 and w_2 are connected in parallel, the equivalent width is defined as $w_1 + w_2$ and if the transistors are connected in series, the equivalent width is defined as $\left[w_1^{-1} + w_2^{-1}\right]^{-1}$.

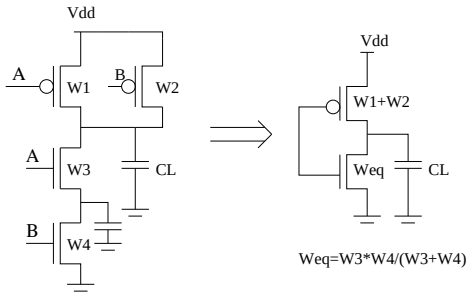

Fig. 2. Mapping of a nand gate

However, such a reduction has shortcomings. Consider the nand gate in Figure 2(a), whose equivalent inverter approximation is illustrated in Figure 2(b). The node capacitances at nodes other than the output are not accounted for in this approximation. Also, the same mapping will be used irrespective of whether input A or B is switching, whereas in reality, these two cases correspond to different delay values. We attempt to reduce the errors caused because of these approximations in our mapping procedure by defining a set of basic primitives, where each primitive corresponds to a different position of swithcing transistor, and mapping arbitrary complex gates to these primitives. The ensuing discussion describes each of these primitives.

C.1 Simple gates

The set of two-input primitives is shown in Figure 3 (the presence of a load capacitance at the output is implicit and is not shown). Timing analysis procedure in our tool assumes only single input transitions, and hence there can only be one pair of pmos and nmos transistors switching at a time.

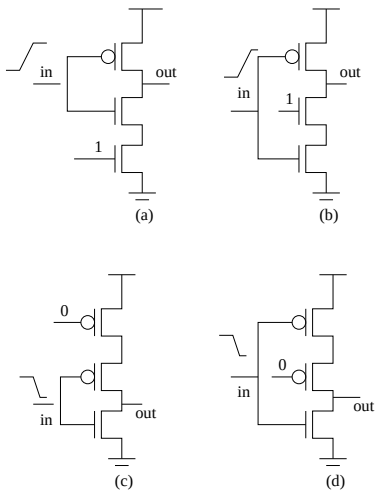

Fig. 3. Set of primitives

Consider the two-input nand gate shown in Figure 4(a). For the fall delay, if the input transition occurs at input A, then the gate is mapped to Figure 3(a). Note that since the output is being pulled down in the case of a fall delay calculation, the pull-down is retained while pull-up is replaced by a single transistor, and the characterization equations of Figure 3(a) are used to estimate the delay.

In a similar fashion, when the input transition occurs at input B of Figure 4(a), the gate is mapped to Figure 3(b). A similar procedure is applied for rise delays, i.e., the pull-up part is retained while the pull-down part is replaced by an equivalent nmos transistor. If we assume single input transitions, only one of the pmos transistors will be on during the rise output transition. The pmos transistor that is off acts only as a loading capacitance, and hence for rise delay calculation, the nand gate is mapped to an inverter; the loading capacitance of the inverter consisting of the fanout load of the gate and the drain capacitance of the off pmos transistor. In a similar fashion, for the nor gate in Figure 4(b), when the transition is at input A, the gate is mapped to the primitive in Figure 3(d), and when the transition is at input B, the gate is mapped to the primitive in Figure 3(c).

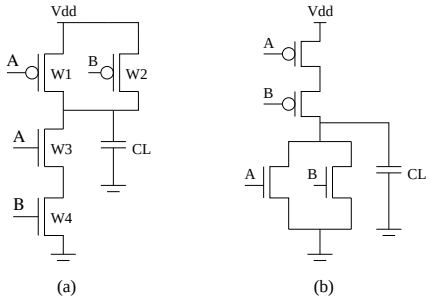

Fig. 4. nand and nor gates

The set of three-input primitives is shown in the Figure 5.

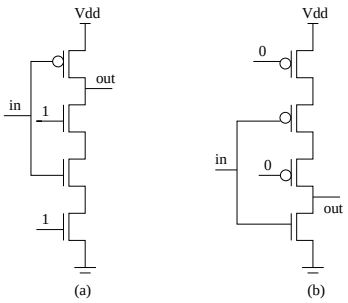

Fig. 5. Primitives for mapping of complex gates

Before explaining the procedure of delay modeling, we introduce the notion of the *largest resistive path (LRP)*. In the worst case switching scenario for a gate, there is exactly one path from the output node to the ground [V_{dd}] node for a fall [rise] transition. This path may be formed by calculating equivalent widths for the 'largest resistive path' from different nodes to ground/V_{dd} nodes [3]. The complex gate is represented by a directed graph with an edge from the drain and source nodes of each transistor in the gate[1]. Since the 'on' resistance of a transistor is, crudely speaking, proportional to the reciprocal of its width, the edge weights are the reciprocals of the widths of corresponding transistors. The largest resistive path between nodes n_1 and n_2 is the path of largest weight from n_1 to n_2. The LRP and the weight of the LRP which corresponds to the width of equivalent transistor is found using a longest path algorithm [14]. Note that the LRP computation based on the crude estimations of resistance are only required to predict the identity of the worst-case path, and more accurate delay modeling is carried out for the actual delay computation.

[1]In a static CMOS gate, it is always possible to uniquely identify the source node and the drain node. This may not be true in circuits with pass gates, which are not handled in our work. The Jouppi rules [13], for example, could be used to extend this work to circuits with pass gates.

Armed with this definition, we will now explain the computation of fall and rise delays for the gate.

FALL DELAYS: For fall delay estimation, the switching nmos transistor in the complex gate is identified; denote this transistor by N_{switch}.

Case A: If the source/drain node of the transistor N_{switch} is connected to the output node, the LRP from drain/source of N_{switch} to ground is replaced by its equivalent transistor, and the pull-up network by its equivalent transistor. The gate is thus mapped to a primitive as shown in Figure 6; this corresponds to a mapping to the primitive in Figure 3(a).

Case B: If the source/drain node of the transistor N_{switch} is connected to ground, the LRP from drain/source of N_{switch} to the output node is replaced by its equivalent transistor, and the pull-up network by its equivalent transistor. The gate is thus mapped to the primitive shown in Figure 3(b).

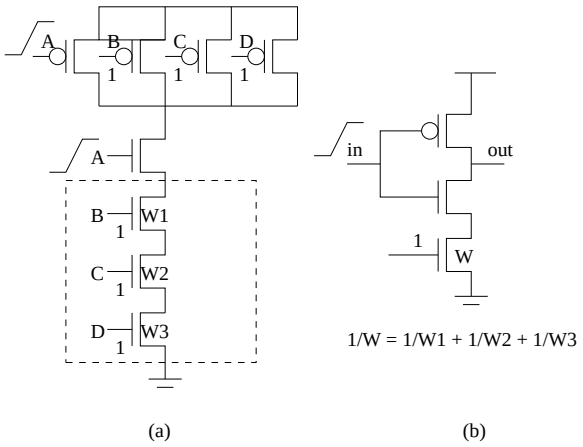

Fig. 6. An illustration of the mapping procedure

Case C: If neither the drain nor the source nodes of N_{switch} is connected to the V_{dd} or ground node, then the LRP 's from drain/source of N_{switch} to V_{dd}/ground nodes are replaced by their corresponding equivalent transistors. The pull-up network is replaced by its equivalent transistor, thereby mapping the gate to the primitive shown in Figure 5(a).

RISE DELAYS: For rise delay estimation, the switching pmos transistor in the complex gate is identified. Let it be identified by P_{switch}.

Case A: If the source/drain node of the transistor P_{switch} is connected to the output node, the LRP from drain/source of P_{switch} to V_{dd} is replaced by its equivalent transistor, and

the pull-down network by its equivalent transistor. The gate is thus mapped to the primitive shown in Figure 3(c). *Case B:* If the source/drain node of the transistor P_{switch} is connected to V_{dd}, the LRP from drain/source of P_{switch} to output node is replaced by its equivalent transistor, and the pull-down network by its equivalent transistor. The gate is thus mapped to the primitive shown in Figure 3(d). *Case C:* If neither the drain nor the source nodes of P_{switch} are connected to the V_{dd} or ground node, then the LRP 's from drain/source of P_{switch} to V_{dd}/ground nodes are replaced by their corresponding equivalent transistors. The pull-down network is replaced by its equivalent transistor, thereby mapping the gate to the primitive shown in Figure 5(b).

C.2 Complex Primitives

One can see that above mentioned methodology can very efficiently handle various simple gates as well as complex gates. However, the methodology will not always give accurate delays for the gates with disjoints paths to power supply or ground (AOI). In case of simple gates with only one transistor chain, the internal node capacitances are inherently taken into account during the modeling phase. But in the case of AOI gates there is more than one parallel chains of transistors. Hence if AOI gates are mapped on to the primitives developed for simple gates, then the internal node capacitances in the nonconducting chains would not be correctly accounted for resulting in inaccurate delay values. When the internal node capacitances are charged then these capacitances need to be considered and hence a set of complex (AOI) primitives is developed. The beauty of our approach is that these primitives can be developed as simple extensions of the primitives of simple gates. For the AOI gates we make use of the observation that the worst case delay corresponds to one conducting chain of transistors between the output and supply, while all other chains are nonconducting. This shows that primitives for AOI gate can be developed by addition of a nonconducting transistor chain in parallel to the transistor chain in the simple gate primitive. A few example primitives for AOI gates are shown in the Figure 7. Similarly, a limited set of primitives can be developed for general complex gates.

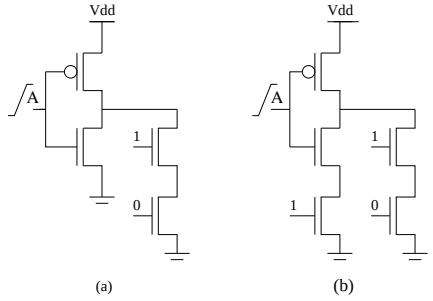

Fig. 7. Examples of AOI primitives

IV. Statement of the problem

The problem that we solve is an altered form of the traditionally stated problem of (1). In addition to minimizing the cost function under delay constraints at primary output nodes, ratioing constraints are also used. Ratioing constraints are important in balancing rise and fall delays which have an impact on the noise margin (Section 2.3 of [12].

$$\begin{array}{ll} \text{minimize} & \textit{Area} \text{ or } \textit{Power} \\ \text{subject to} & \textit{Delay} \leq T_{spec} \text{ at all primary outputs} \\ & K_1 \leq \dfrac{\text{Pull-down strength}}{\text{Pull-up strength}} \leq K_2 \; \forall \text{ gates } i \end{array}$$

V. Proof of convexity of the delay model

The ensuing discussion shows that the delays of individual paths satisfy the property of convexity, and uses this fact to prove the convexity of the optimization problem. It is to be emphasized that this discussion is purely for expository purposes; the optimizer used in this work for sizing *does not* require the enumeration of all paths, and performs the optimization efficiently by checking, through a timing analysis, whether the constraints are satisfied or not. For details, the reader is referred to [1].

Let the critical path of the circuit be represented by a set of stages, where each stage represents a gate. Let us first consider a scenario with fully characterized gates where no primitives are used, but the delay is characterized in terms of the size of each transistor. Then, substituting the characterization variables explicitly into Equation (9), we see that the fall delay of the gate corresponding to stage l has the following form:

$$\begin{aligned} \text{Delay}_l = \sum_i P_i \cdot & (w_{n1}^{-1} + c_{n1})^{\beta_{n1}} \cdots (w_{nm_n}^{-1} + c_{nm_n})^{\beta_{nm_n}} \\ & (w_{p1} + c_{p1})^{\beta_{p1}} \cdots (w_{pm_p} + c_{pm_p})^{\beta_{pm_p}} \\ & (\tau_{i-1} + c_\tau)^{\beta_\tau} \Pi_j (C_j + c_{C_j})^{\beta_{C_j}} \end{aligned}$$

and the output fall transition time of the gate in stage l has the form[2]

$$\begin{aligned} \tau_l \;=\; & Q \cdot (w_{n1}^{-1} + k_{n1})^{\gamma_{n1}} \cdots (w_{nm_n}^{-1} + k_{nm_n})^{\gamma_{nm_n}} \\ & (w_{p1} + k_{p1})^{\gamma_{p1}} \cdots (w_{pm_p} + k_{pm_p})^{\gamma_{pm_p}} \\ & (\tau_{i-1} + k_\tau)^{\gamma_\tau} \Pi_j (C_j + k_{C_j})^{\gamma_{C_j}} \end{aligned}$$

where $P_i > 0$, $Q > 0$, c_{ni},c_{pi},k_{ni},k_{pi},β_{ni},β_{pi}, γ_{ni},γ_{pi} $\forall i$, k_{C_j},c_{C_j} $\forall j$,k_τ,c_τ, β_{C_j},γ_{C_j},β_τ,γ_τ are real constants. The w_{ni} and w_{pi} values, as usual, refer to the nmos and pmos transistor sizes, τ refers to the transition time, and the C_j's correspond to the capacitances at the gate output and at internal nodes. We will show that the delay and transition time functions have the form of generalized posynomials.

The capacitance at each internal or gate output node i, C_i is modeled by

$$C_i = \sum_j k'_j w_j + k'' \tag{11}$$

[2]The rise delay and rise transition time expressions are similar, with the roles of w_n and w_p interchanged.

where the $k_j^{'}$ and k'' values are real constants, and w_j's represent the equivalent transistor widths in the circuit.

From the Equation (11) we can see that output transition time is represented by a generalized posynomial. Also the loading capacitance given by equation (11) has generalized posynomial form. Using Theorem 1, it can be seen that when the input transition time and loading capacitance expressions are substituted in the delay Equation (11), the resulting expression is also a generalized posynomial. Thus, the path delay is a generalized posynomial function of the device widths, that can be mapped on to a convex function. The objective function is chosen as a weighted sum of the transistor sizes, which is clearly a generalized posynomial form. Using identical arguments to [1], [3], since the maximum of convex functions is convex, the problem of area minimization under delay constraints for "template" gates can be shown to be a convex programming problem.

The problem of power minimization can be dealt with in a similar fashion; since the edge rates are being explicitly controlled in our formulation, the short-circuit power is implicitly controlled and, unlike [15] can be neglected from the cost function. As a result, the power objective merely requires minimization of the dynamic power, which is well known to be a weighted sum of the device sizes [15].

For gates that do not adhere to the template, the mapping techniques described in Section III-C may be used to model the delay function. We will now show that in such a case, the delay function continues to remain in the generalized posynomial form. Let $w_1^{'}, \cdots, w_m^{'}$ represent transistor widths in the primitives the gates are mapped to. In the process of mapping the gates, the transistor widths in the primitives can be expressed in terms of the actual transistor widths in the circuit. Let $w_1, \cdots, w_n$ represent the actual transistor widths in the circuit. Then $w^{'}$'s can be expressed as

$$w_i'^{-1} = \sum_{q \in \{1 \cdots n\}} w_q^{-1}, 1 \leq i \leq m \qquad (12)$$

All occurrences of value of $w_i'^{-1}$, which is a basic variable in the characterization equation (see the last paragraph of Section III-B.1), can be substituted as above in Equation (11), maintaining the generalized posynomial property of the delay equation.

VI. Experimental Results

The accuracy of the new model for different gates with respect to the SPICE is demonstrated in Table I. Referring to Equation (9), a value of $m = 1$ was chosen, and it was observed that the use of higher values for j did not offer significant improvements in accuracy, and it was found that these values of m and j were adequate for our needs. The characterization was performed in a $\lambda = 0.25\mu$m technology by varying transistor widths to up to 30 λ, τ from 20 to 300 ps and C_L up to 600 fF. Characterizations over other ranges also showed similar behavior.

The table shows a comparison of the accuracies obtained (relative to a SPICE simulation) using generalized posynomials with those from correspondingly fitted parameters (for the same data) for the Elmore model. In each case, the average error was significantly smaller (from half to a quarter of the error), and the standard deviation in the error values was substantially smaller. This reinforces the fact that Elmore delays cannot be considered reliable, and the proposed approach presents more accurate delay models.

Logic Gate	Gen. Posy. fit		Elmore fit	
	Error	σ_{Error}	Error	σ_{Error}
Inverter	1.2%	1.4%	8.2%	9.0%
Nand2	7.2%	6.9%	23.3%	17.9%
Nor2	6.1%	6.8%	23.5%	18.3%
Nand3	13.2%	11.5%	28.2%	20.4%
Nor3	5.8%	4.5%	19.3%	16.5%

TABLE I
Accuracy of the delay model

Output Capcitance = 30fF
Delay without any constraint = 934ps

Model Delay (ps)	SPICE Delay (ps)	Error	Area
840	835	0.59 %	6.67
745	752	-0.94 %	7.75
655	670	-2.30 %	10.01
560	594	-6.07 %	14.05

TABLE II
Comparison of Model Delay for C17 with SPICE

The delay models developed in this paper were incorporated into iCONTRAST [3] in a C program. The input to the program is a SPICE-like transistor-level netlist. The results of running the algorithm on various test circuits are shown in Table III. The cost function is set to be the area of the circuit, estimated as the sum of the transistor sizes.

An important point in judging the correctness of our model is correspondence of the model delay of a sized circuit with that of its SPICE delay. We used the circuit C17 from the ISCAS85 benchmark suit. First we measure the input-to-output delay of the circuit with all transistor sizes set to minimum, that is unsized delay. In our case this input-to-optput delay was 934ps. Once this delay is known we enforce a tighter delay requirement and optimize the circuit. We then measure the delay of the optimized circuit using SPICE and compare it with the model delay. The results of this comparison, for the target delay of 90%, 80%, 70% and 60% of unszied delay, are shown in the Table II. Also given in the table is the associated area. It can observed, as expected, that as the specification is made tighter, the required area to meet these specifications increases.

To show that our model is computationally efficient we optimized several datapath as well as ISCAS85 benchmark circuits. We measured the unsized delay of each circuit and then forced the target delays of 80%, 70% and 60% of the unsized delay. The run times remain reasonable;

Circuit	Unsized Area	Unsized Delay	T_{spec}	Sized Area	Exec Time
Inv10	5	1.68ns	1.35ns	7.52	0.24s
			1.20ns	10.07	0.34s
			1.00ns	16.06	0.33s
Add2	13	1.80ns	1.45ns	20.07	3.8s
			1.26ns	30.84	4.1s
			1.08ns	71.62	4.7s
Add8	52	6.3ns	5.0ns	98.1	7.2s
			4.4ns	148.0	5.4s
			3.8ns	286.0	24.0s
Add32	208	24.2ns	21.8ns	271	240s
			19.4ns	378	68s
			17.0ns	716	150s
C432	251	9.3ns	7.4ns	424	129s
			6.5ns	625	54s
			5.6ns	954	25s
C880	441.5	7.4ns	5.9ns	665	830s
			5.2ns	915	257s
			4.4ns	1565	48s
C499	701	10.7ns	8.5ns	1125	2897s
			7.4ns	1308	1851s
			6.3ns	1783	911s

TABLE III
RESULTS OF SIZING VARIOUS CIRCUITS

a thousand transistor circuit, c432, is seen to be sized in about 2 min.

VII. Conclusion

We have presented a new delay model for CMOS gates that is better suited for modern technologies than the Elmore model, but maintains the convexity properties. A new class of functions called generalized posynomials is proposed and are shown to have the same relation to convex functions as posynomials. Experimental results illustrating the effectiveness of this model have been reported, and the results of running the sizing algorithm with the improved model have also been included.

Although we have not done so in this work, the accuracy of the model may be increased by enhancing the richness of the modeling functions. For the transition time expression in Equation 11, a sum of terms could be used and this could be shown to maintain the convexity properties of the formulation. For delay modeling, one could use the "max" operator to further enhance accuracy, if required. If $f_1, f_2, \cdots, f_k$ are convex functions then the function $F = \max(f_1, f_2, \cdots, f_k)$ is also a convex function and hence can be used to improve accuracy of the approximation, while retaining the convexity properties of the delay function. However, to implement such a scheme, the number of characterization constants will increase, thereby increasing the characterization effort. Nevertheless, it may be a useful extension if the accuracy achieved using current model is not deemed sufficient. Since the current modeling approach is a special case of this, where $k = 1$, the extension is guaranteed to do no worse than our approach.

Acknowledgements

The authors are grateful to Prof. M. P. Desai at IIT Bombay for initiating this train of thought, and to Prof. S. Boyd at Stanford for his comments on Theorem 1, and to the anonymous reviewers for their 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] J.-M. Shyu, A. L. Sangiovanni-Vincentelli, J. Fishburn, and A. Dunlop, "Optimization-based transistor sizing," *IEEE Journal of Solid-State Circuits*, vol. 23, pp. 400–409, Apr. 1988.

[3] 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.

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

[5] 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.

[6] H. Y. Chen, *Design Automation for High Performance CMOS VLSI Circuits*. PhD thesis, University of Illinois at Urbana-Champaign, Dec. 1988.

[7] S. Dutta, S. S. M. Shetti, and S. L. Lusky, "A comprehensive delay model for CMOS inverters," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 864–871, August 1995.

[8] V. B. Rao, T. N. Trick, and I. N. Hajj, "A table-driven delay-operator approach to timing simulation of MOS VLSI circuits," in *Proceedings of the 1983 International Conference on Computer Design*, pp. 445–448, 1983.

[9] D. G. Luenberger, *Linear and Nonlinear Programming*. Reading, Massachusetts: Addison-Wesley, 2nd ed., 1984.

[10] Meta-Software, Inc., *HSPICE User's Manual: Volume II: Elements and Device Models*, 1996.

[11] Department of Operations Research, Stanford University, *MINOS 5.4 USER'S GUIDE*, 1995.

[12] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*. Reading, MA: Addison-Wesley, 2nd ed., 1993.

[13] N. P. Jouppi, "Timing analysis and performance improvement of MOS VLSI design," *IEEE Transactions on Computer-Aided Design*, vol. CAD, pp. 650–665, July 1987.

[14] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, ch. 6, pp. 527–532. McGraw-Hill Book Company, Cambridge, MA, 1992.

[15] S. S. Sapatnekar and W. Chuang, "Power vs. delay in gate sizing: Conflicting objectives?," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 463–466, 1995.

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

Appendix

I. Proof of Convexity of Generalized Posynomials

It is well known that a generalized posynomial of order 0, $G_0(\mathbf{x})$, is transformed to a convex function, $G_0(\mathbf{z})$ in the $\mathbf{z}$ domain [16]. Since the functional form of the functions $G_k(\mathbf{x}), k > 0$, is different from that of $G_0(\mathbf{x})$ due to the additional nonnegativity constraint on the α_{ij} variables, they are treated separately.

The proof of Theorem 1 proceeds by considering $G_k(\mathbf{z})$ for $k \geq 1$; to prove its convexity, it is enough to prove the

convexity of

$$L = P\prod_{i=1}^{m}(G_{k-1,i})^{\beta_i}, \beta_i \geq 0, \tag{13}$$

since a sum of convex functions is convex. The gradient and Hessian of this function are, respectively, given by

$$\begin{aligned}\bigtriangledown L &= P\sum_{i=1}^{m}\left\{\left(\prod_{j=1,i\neq j}^{m}(G_{k-1,j})^{\beta_j}\right)\beta_i(G_{k-1,i})^{\beta_i-1}\right. \\ &\left.\bigtriangledown G_{k-1,i}\right\} = L\sum_{i=1}^{m}\frac{\beta_i \bigtriangledown G_{k-1,i}}{G_{k-1,i}} \end{aligned}\tag{14}$$

$$\begin{aligned}\bigtriangledown^2 L &= L\left\{\left(\sum_{i=1}^{m}\frac{\beta_i \bigtriangledown G_{k-1,i}}{G_{k-1,i}}\right)\left(\sum_{i=1}^{m}\frac{\beta_i \bigtriangledown G_{k-1,i}^T}{G_{k-1,i}}\right)+\right. \\ &\left.\sum_{i=1}^{m}\frac{\beta_i}{G_{k-1,i}^2}\left(G_{k-1,i}\bigtriangledown^2 G_{k-1,i} - \bigtriangledown G_{k-1,i}\bigtriangledown G_{k-1,i}^T\right)\right\}\end{aligned}\tag{15}$$

We will prove that L is a convex function by showing that the matrix $\bigtriangledown^2 L$ is positive semidefinite. Since the first term is easily seen to be positive semidefinite, the function L is convex if $(G_{k-1,i}\bigtriangledown^2 G_{k-1,i} - \bigtriangledown G_{k-1,i}\bigtriangledown G_{k-1,i}^T)$ is positive semidefinite. We will now show this by proving the following result, by induction and the proof of Theorem 1 follows as an immediate consequence. The matrix $\left(G_k \bigtriangledown^2 G_k - \bigtriangledown G_k \bigtriangledown G_k^T\right)$ is positive semidefinite for all $k \geq 0$.

Basis case Consider a zeroth order generalized posynomial given by

$$G_0 = \sum_{i=1}^{p}\omega_i\prod_{j=1}^{n}e^{a_{ij}z_j} = \sum_{i=1}^{p}h_i,$$

where $h_i = \omega_i\prod_{j=1}^{n}e^{a_{ij}z_j}$. It is easy to see that the value of each h_i is positive for all $\mathbf{z}$; this observation is used later in the proof.

Now consider the matrix $H = \left(G_0 \bigtriangledown^2 G_0 - \bigtriangledown G_0 \bigtriangledown G_0^T\right)$. The $(q,l)^{\text{th}}$ term of this matrix is given by

$$\begin{aligned}H_{ql} &= \left(\sum_{i=1}^{p}h_i\right)\left(\sum_{i=1}^{p}h_i a_{iq}a_{il}\right) - \left(\sum_{i=1}^{p}h_i a_{iq}\right)\left(\sum_{i=1}^{p}h_i a_{il}\right) \\ &= \sum_{i=1}^{p}\sum_{j=1,j\neq i}^{p}\left[h_i h_j\left(a_{iq}-a_{jq}\right)\cdot a_l\right] \\ &= \sum_{i=1}^{p}\sum_{j=i+1}^{p}\left[h_i h_j\left(a_{iq}-a_{jq}\right)\cdot\left(a_{il}-a_{jl}\right)\right]\end{aligned}$$

Therefore, we can write

$$H = \sum_{i=1}^{p}\sum_{j=i+1}^{p}h_i h_j\left(\vec{a}_i-\vec{a}_j\right)\cdot\left(\vec{a}_i-\vec{a}_j\right)^T$$

where $\vec{a}_i = [a_{i1}, a_{i2}, \cdots a_{in}]^T$. Therefore, H is positive definite since each $h_i > 0$.

Induction hypothesis: For a generalized posynomial $G_{k-1}(\mathbf{z})$ of order $k-1$, where $k \geq 1$,

$$G_{k-1}(\mathbf{z})\bigtriangledown^2 G_{k-1}(\mathbf{z}) - \bigtriangledown G_{k-1}(\mathbf{z})\bigtriangledown G_{k-1}(\mathbf{z})^T$$

is positive semidefinite.

For the inductive step, we write

$$G_k = \sum_{i=1}^{r}L_{k,i} = \sum_{i=1}^{k}P_i\prod_{j=1}^{m_i}(G_{k-1,i,j})^{\beta_{i,j}}, \tag{16}$$

so that each $L_{k,l}$ is of the form of the function L defined in Equation (13). We may use the expressions for the gradient and Hessian of L in Equations (14) and (15) to write

$$\begin{aligned}&G_k \bigtriangledown^2 G_k - \bigtriangledown G_k \bigtriangledown G_k^T \\ &= \left(\sum_{l=1}^{r}L_{k,l}\right)\left(\sum_{l=1}^{r}\bigtriangledown^2 L_{k,l}\right) - \left(\sum_{l=1}^{r}\bigtriangledown L_{k,l}\right)\left(\sum_{l=1}^{r}\bigtriangledown L_{k,l}\right)^T \\ &= \sum_{l=1}^{r}\sum_{q=1}^{r}\left(L_{k,l}\bigtriangledown^2 L_{k,q} - \bigtriangledown L_{k,l}\bigtriangledown L_{k,q}^T\right)\end{aligned}$$

If we set

$$\vec{u}_j = \sum_{j=1}^{m}\frac{\beta_j \bigtriangledown G_{k-1,i,j}}{G_{k-1,i,j}}, \tag{17}$$

this may be rewritten as

$$\begin{aligned}&\sum_{l=1}^{r}\sum_{q=1}^{r}L_{k,l}(L_{k,q}\{\vec{u}_q\vec{u}_q^T + \sum_{i=1}^{m}\frac{\beta_i}{G_{k-1,q,i}^2}(G_{k-1,q,i} \\ &\bigtriangledown^2 G_{k-1,q,i} - \bigtriangledown G_{k-1,q,i}\bigtriangledown G_{k-1,q,i}^T)\}) - L_{k,l}L_{k,q}\vec{u}_l\vec{u}_q^T \\ &= \sum_{l=1}^{r}\sum_{q=1}^{r}L_{k,l}L_{k,q}\sum_{i=1}^{m}\frac{\beta_i}{G_{k-1,q,i}^2} \\ &\left(G_{k-1,q,i}\bigtriangledown^2 G_{k-1,q,i} - \bigtriangledown G_{k-1,q,i}\bigtriangledown G_{k-1,q,i}^T\right) + \\ &\sum_{l=1}^{r}\sum_{q=l+1}^{r}L_{k,l}L_{k,q}\sum_{i=1}^{m}(\vec{u}_q-\vec{u}_l)(\vec{u}_q-\vec{u}_l)^T,\end{aligned}$$

which is positive semidefinite by the induction hypothesis. **QED**.