# Convex Delay Models for Transistor Sizing *

Mahesh Ketkar
Department of ECE,
University of Minnesota
Minneapolis, MN 55455, USA
ketkar@ece.umn.edu

Kishore Kasamsetty
Department of ECE,
University of Minnesota
Minneapolis, MN 55455, USA
kishore@mail.ece.umn.edu

Sachin Sapatnekar
Department of ECE,
University of Minnesota
Minneapolis, MN 55455, USA
sachin@ece.umn.edu

## 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. The delay model is incorporated into a transistor sizing algorithm based on TILOS. The models were characterized by using a set of grid points and then validated using a disjoint data set. The models were found to be within about 10% of SPICE for nearly all of the gate types considered. Also presented are the experimental results of sizing various test circuits.*

## 1 Introduction

Transistor sizing, an important problem in designing high performance circuits, has traditionally been formally defined as [1]:

$$\begin{array}{ll} \text{minimize} & \textit{Area or Power} \\ \text{subject to} & \textit{Delay} \leq T_{spec}. \end{array} \quad (1)$$

There have been many significant attempts to solve this problem, for example, [1, 2]. Most published approaches use the Elmore delay model [3] 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.

It is generally accepted that the use of the Elmore delay model at the transistor level is very inaccurate for modern designs. This inaccuracy can be attributed to its failure to accurately consider important factors such as input transition times, position of the switching transistor, sizes of fighting complementary transistors, temporal relation between inputs and transistor nonlinearities. As a result, 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.

Several approaches for accurate timing modeling have been proposed in the past. For example, one could model gate delays by developing closed form expressions [4]. Much work has been done in the development of closed form models for inverters and then mapping other gates to an equivalent inverter [5, 6]. An alternative approach uses, a look-up table constructed using experimentally derived delay data for various configurations, with intermediate data points being derived by interpolation methods, as in the delay model in [7]. However, this approach requires storage of large number of data points to guarantee accuracy and hence is very 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 functions that are shown to work well for modeling circuit delays. These functions are provably convex under a variable transformation that is explained in next section. The set of functions from which these functions are chosen includes the set of posynomials as a proper subset, and therefore, we refer to these functions as *generalized posynomials*. 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.

## 2 Background

### 2.1 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. A problem of the type

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

is a convex programming problem if $f(\mathbf{x})$ and $g_i(\mathbf{x}), 1 \leq i \leq m$, are convex functions. In the context of transistor sizing, this requires the derivation of convex closed-form expressions for the path delay; as a result, this will satisfy the requirement of relation (2) that each timing constraint is of the form $g_i(\mathbf{x}) \leq 0$. All of these statements constitute well-known facts [1, 2].

### 2.2 Posynomial delay modeling

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.

*This work is supported in part by a gift from Intel Corporation, by the NSF under contract CCR-9800992 and the SRC under contract 99-TJ-692. We would like to thank Dr. Priyadarsan Patra from Intel.

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 $\gamma_j \in \mathbf{R}^+$. 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 [2], 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}.

# 3 Modeling using generalized posynomials

## 3.1 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.

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

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

   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 represents a posynomial function. Therefore, a generalized posynomial of first order is similar to a posynomial, except that the place of the $x_i$ variables in Equation (3) is taken by a posynomial. 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).

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

**Theorem 1**: If the range of interest of $\mathbf{x}$ 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 mapped to a convex function in the $\mathbf{z}$ domain.

**Proof**: 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 [8]. 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 $\alpha_{ij}$ 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

$$L = P \prod_{i=1}^{m} (G_{k-1,i})^{\beta_i}, \beta_i \geq 0, \tag{8}$$

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

$$\begin{aligned} \bigtriangledown L &= P \sum_{i=1}^{m} \left\{ \left( \prod_{j=1, i \neq j}^{m} (G_{k-1,j})^{\beta_j} \right) \beta_i (G_{k-1,i})^{\beta_i - 1} \bigtriangledown G_{k-1,i} \right\} \\ &= L \sum_{i=1}^{m} \frac{\beta_i \bigtriangledown G_{k-1,i}}{G_{k-1,i}} \end{aligned} \tag{9}$$

$$\begin{aligned} \bigtriangledown^2 L &= L \left\{ \left( \sum_{i=1}^{m} \frac{\beta_i \bigtriangledown G_{k-1,i}}{G_{k-1,i}} \right) \left( \sum_{i=1}^{m} \frac{\beta_i \bigtriangledown G_{k-1,i}^T}{G_{k-1,i}} \right) + \right. \\ &\left. \sum_{i=1}^{m} \frac{\beta_i}{G_{k-1,i}^2} \left( G_{k-1,i} \bigtriangledown^2 G_{k-1,i} - \bigtriangledown G_{k-1,i} \bigtriangledown G_{k-1,i}^T \right) \right\} \end{aligned} \tag{10}$$

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

$$G_k = \sum_{i=1}^{r} L_{k,i} = \sum_{i=1}^{k} P_i \prod_{j=1}^{m_i} (G_{k-1,i,j})^{\beta_{i,j}}, \tag{11}$$

so that each $L_{k,l}$ is of the form of the function $L$ defined in Equation (8). We may use the expressions for the gradient and Hessian of $L$ in Equations (9) and (10) 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

$$\vec{u}_j = \sum_{j=1}^{m} \frac{\beta_j \bigtriangledown G_{k-1,i,j}}{G_{k-1,i,j}}, \tag{12}$$

this may be rewritten as

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

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

## 3.2 Delay estimation

### 3.2.1 Outline of the delay modeling approach

Our characterization approach uses sizes of transistors belonging to the gate along with the traditional cell characterization parameters, namely input transition time and load capacitance. We refer to these input parameters 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 $\tau$, and the output load capacitance, $C_L$. Therefore, for an inverter, $w_p$, $w_n$, $\tau$, 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 to achieve the desired levels of accuracy. The general form of expression that provided consistently good results for different gate types is as follows

$$\text{Delay} = \sum_{j=1}^{m} P_j \cdot \prod_{i=1}^{n} (x_i^{\Delta} + c_{ij})^{\beta_{ij}} + C \tag{13}$$

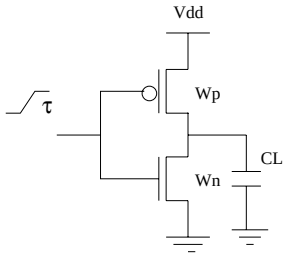


Figure 1: Inverter circuit

Here, the $x_i$'s are characterization variables, and the $c_{ij}$'s, $\beta_{ij}$'s, $C$, and $P_j$'s are real constants, referred to collectively as *characterization constants*. The parameter $\Delta$ is set to either -1 or 1, depending on the variable, as will soon be explained. The problem of characterization is that of determining appropriate values for the characterization constants. We will show in Section 4 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.

Due to the curve-fitting nature of the characterization procedure 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 $\tau$ are increased, and decreases as $w_n$ is increased, implying that an appropriate choice for the parameter $\Delta$ for the first three variables is 1, and that for $w_n$ is -1. Note that this is not as restrictive as the Elmore form since, among other things, the $\beta_{ij}$'s and $c_{ij}$'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.

### 3.2.2 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 (13).

A series of simulations is performed to collect the experimental data using the HSPICE circuit simulator. 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 $d^4$. Therefore, it is important to choose the data points carefully; in particular, 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 perform ed by solving the following nonlinear program that minimizes the sum of the squares of the percentage errors over all data points.

$$\text{minimize} \sum_{i=0}^{N} \left[\frac{D_{estim}(i) - D_{actual}(i)}{D_{actual}(i)}\right]^2 \tag{14}$$

where $N$ is the number of data points, $D_{estim}(i)$ and $D_{actual}(i)$, respectively, represent the values given by Equation (13), and the corresponding measured value at the $i^{\text{th}}$ data point. This nonlinear programming problem is solved using the MINOS optimization package [9] to determine the values of characterization constants.

## 3.3 Characterization of a set of primitives

For a library-based design, a full characterization of all cells is a viable alternative and its complexity is comparable to characterizing the library using any other means. For general full custom designs, the number of SPICE data points to be generated for the curve fit increases exponentially with the number of characterization variables. It is computationally expensive to perform such a large number of simulations and hence an alternative strategy is suggested.

An alternative strategy is to precharacterize a set of logic structures such that any gate can be mapped to one of the elements of

this set with some acceptable loss of accuracy. It is important to note that even under this procedure, the transistor sizing approach will size each transistor individually, and this method is only used for delay estimation.

One straightforward technique that may be used is to map all of the gates to an "equivalent inverter" [5, 6], 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 $\left[w_1^{-1} + w_2^{-1}\right]^{-1}$.

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. This issue is addressed in the section 3.3.1.

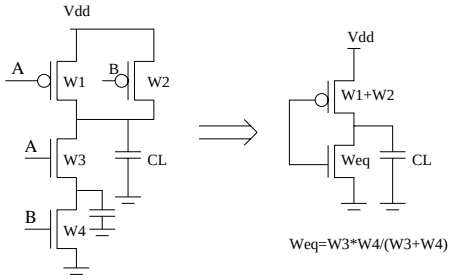


Figure 2: Mapping of a nand gate

We attempt to reduce the errors caused because of these approximations in our mapping procedure by defining a set of basic primitives and mapping arbitrary complex gates to these primitives. We have developed primitives for three distinct types of logic structures namely *simple gates*, *complex gates* and *sequential elements* for both fall and rise transition.

### 3.3.1 Simple gates

For simple gates, we have developed one input, two input and three input primitives. Single input primitive is basically an inverter. We refer to an inverter as a primitive because of the fact that mapping procedure along with inverters also maps NOR gates for fall transition and NAND gates for rise transition on an inverter. Since this primitive is identical to the inverter described in Section 3.2.1, it is not discussed any further.

Here we emphasize that an *n*-input primitive does not mean that it is a primitive only for the *n*-input gates. Any gate having equal to or more than *n* inputs would be mapped to an *n*-input primitive depending upon the position of the switching transistor.

The set of two input primitives for fall transition at the output 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.

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

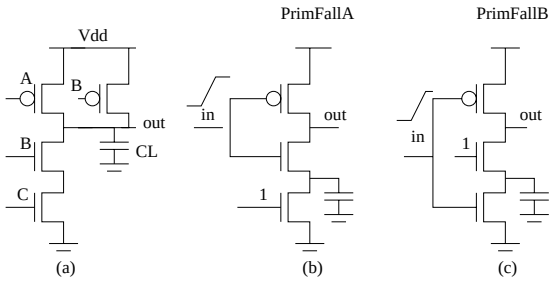


Figure 3: 2-input primitives for fall transition

B of Figure 3(a), the gate is mapped to Figure 3(c). 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 contributes only as a loading capacitance, and hence for rise delay calculation, the nand gate is mapped to an inverter. Similarly, 2-input primitives, containing two pmos transistors in series with an nmos transistor, namely PrimRiseA and PrimRiseB, are developed that can accurately model NOR gates and NOR gate-like structures.

For simple gates with more than two inputs and complex gates, an expanded set of primitives is necessary. The set of primitives used to approximate such gates is shown in Figure 4. It should be noted that these are not the only primitives on which gates with more than three inputs will be mapped. For example, consider a three input NAND gate and the case where the latest arriving input is the one connected to the topmost transistor in the nmos chain. In this case, the NAND gate will be mapped to the two input primitive PrimFallA shown in Figure 3(a); the two nmos transistors at the bottom are collapsed into one transistor of equivalent width.

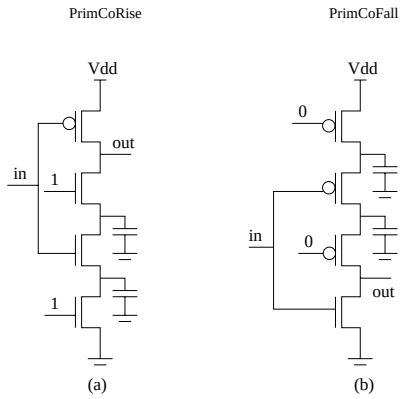


Figure 4: Primitives for mapping of simple and complex gates

### 3.3.2 Complex gates

In case of simple gates with only one transistor chain, the internal node capacitances are inherently taken into account during the modeling phase. For example, in the case of AOI gates there is more than one parallel chains of transistors. Hence if AOI gates are mapped (except when all the transistors connected to the output and belonging to the nonconducting chains are off) on to the primitives developed for simple gates, then the internal node capacitances would not be correctly accounted for, resulting in inaccurate delay values. We solve this problem by developing another set of primitives. For 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 5. Similarly, a limited set of primitives can be developed for general complex gates.

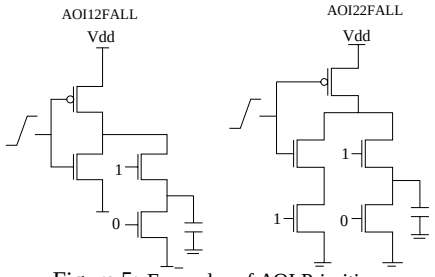


Figure 5: Examples of AOI Primitives

### 3.3.3 Sequential elements

A static sequential element normally consists of a set of pass transistors and a few inverters. An example sequential element is shown in Figure 6. Since an inverter that drives a transmission gate forms a single channel connected component, as shown in Figure 6, we develop a separate model for this component, and in conjunction with the inverter model explained earlier, we are now able to model every channel connected component in this sequential element. An advantage of the ability to develop accurate models for the sequential elements is the simplicity in constraint formulation in the across-latch optimization.

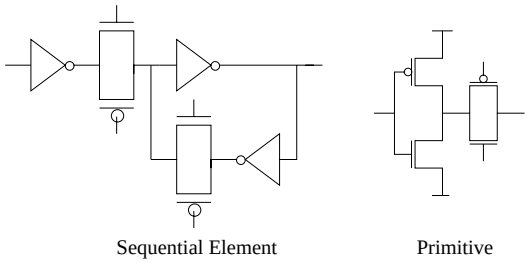


Figure 6: Sequential Element and Primitive

# 4 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 (13), we see that the fall delay of the gate corresponding to stage $l$ has the following form:

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

and the output fall transition time of the gate in stage $l$ has the form[1]

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

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

where $P_i > 0$, $Q > 0$, $c_{ni}$,$c_{pi}$,$k_{ni}$,$k_{pi}$,$\beta_{ni}$,$\beta_{pi}$, $\gamma_{ni}$,$\gamma_{pi}$ $\forall i, k_{C_j}$,$c_{C_j}$ $\forall j$,$k_\tau$,$c_\tau$, $\beta_{C_j}$,$\gamma_{C_j}$,$\beta_\tau$,$\gamma_\tau$ are real constants. The $w_{ni}$ and $w_{pi}$ values, as usual, refer to the nmos and pmos transistor sizes, $\tau$ 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

$$C_i = \sum_j k'_j w_j + k'' \tag{15}$$

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 (15) we can see that output transition time is represented by a generalized posynomial. Additionally, the loading capacitance given by equation (15) has the form of a generalized posynomial. Using Theorem 1, it can be seen that when the input transition time and loading capacitance expressions are substituted in Equation (15), the resulting expression is also a generalized posynomial. 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, 2], 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. For gates that do not adhere to the template, the mapping techniques described in Section 3.3 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

$$w_i'^{-1} = \sum_{q \in \{1 \cdots n\}} w_q^{-1}, 1 \le i \le m \tag{16}$$

All occurrences of value of $w_i'^{-1}$, which is a basic variable in the characterization equation (see the last paragraph of Section 3.2.1), can be substituted as above in Equation (15), maintaining the generalized posynomial property of the delay equation.

# 5 Experimental Results

Table 1 shows the validation results of different primitives, proposed in Section 3.3, with respect to SPICE. The purpose of listing these validation results on the primitives is to emphasize that transistor nonlinearities can be effectively modeled by convex functions and to test the validity of our basic idea of modeling delay as convex functions. Referring to Equation (13), a value of $j = 1$ was chosen, and it was observed that the use of higher values for $j$ did not offer significant improvements in accuracy. The characterization was performed in a $0.25\mu$m technology by varying transistor widths to up to $80\mu$m, $\tau$ from 20 to 300 ps (10% to 90%) and $C_L$ up to 800 fF. We emphasize that accurate fits are required only in the region where sizing constraints are satisfied. For example, if output transition time violates the specification then the optimizer will ensure that its value is reduced to a point in the feasible region, and the convexity of the functions will force the optimization to move to this region after some iterations.

Table 2 shows the validation results of various gates with respect to SPICE. We stress here that all the possible mappings for a gate are considered and the worst case results are shown in the table. For example, the fall transition on gate Nand3 can map on to either primitive PrimFallA, PrimFallB or PrimCoFall. It was found that PrimCoFall provided the best results, while PrimFallA and PrimFallB provided a smaller degree of accuracy due to the fact that a

| Primitive | Delay | |
|---|---|---|
| | Mean | Deviation |
| InvRise | 0.31 % | 2.84 % |
| InvFall | 1.29 % | 2.82 % |
| PrimFallA | -1.28 % | 4.74 % |
| PrimFallB | 1.07 % | 2.95 % |
| PrimRiseA | -0.67 % | 3.59 % |
| PrimRiseB | 0.13 % | 0.93 % |
| PrimCoFall | -0.68 % | 2.96 % |
| PrimCoRise | -0.35 % | 1.79 % |
| AOI12Fall | 0.87 % | 6.27 % |
| SeqFall | 7.46 % | 4.73 % |

Table 1: Primitive Validation

| Gate | Delay | | |
|---|---|---|---|
| | Output Transition | Mean | Deviation |
| Inv | Rise | 0.31% | 2.83 % |
| | Fall | 1.29 % | 2.82 % |
| Nor2 | Rise | 1.82 % | 2.56 % |
| | Fall | 11.10 % | 5.06 % |
| Nand2 | Rise | 5.18 % | 6.17 % |
| | Fall | -0.46 % | 3.58 % |
| Nor3 | Rise | 0.24 % | 1.76 % |
| | Fall | 24.2 % | 7.64 % |
| Nand3 | Rise | 9.21 % | 5.98 % |
| | Fall | 1.26 % | 2.29 % |
| AOI3 | Rise | 5.21 % | 6.38 % |
| | Fall | 0.86 % | 6.27 % |

Table 2: Gate Validation

three input gate was mapped to a two-input primitive using the concept of an equivalent width of two series transistors. It is observed that all of the errors that are above 2% are obtained when an $n$-input gate is mapped to a $k$-input primitive where $k < n$. If some gate type gives unacceptable results for some input transition we can further enhance the accuracy by characterizing the model for that specific scenario over a range of parameter values.

To verify whether we get reasonable accuracy with our model we optimized the C17 benchmark from ISCAS85 benchmark suite with an accurate convex optimizer, and then ran SPICE on the optimized circuit. Table 3 shows the validation results. The unsized delay corresponds to the circuit with all transistor sizes set to minimum. The circuit is optimized for the target delays that vary from 60% to 90% of the unsized delay, as listed in column one. Columns two and three show the SPICE delay of the optimized circuit and worst case errors in the output delay measured by our model as compared to SPICE. The area of the circuit is shown in the last column.

The delay models developed in this paper were incorporated into the TILOS algorithm described in [1] in a C program. The results of running the algorithm on various test circuits are shown in Table 4. The cost function is set to be the area of the circuit, estimated as the sum of the transistor sizes. We first measured unsized delays using our model. The circuits are then optimized for target delays of 70% to 95% of the unsized delay. The results show that our convex model, in addition to being very accurate is also computationally efficient when used in the inner loop of a TILOS-like iterative transistor sizing algorithm.

## 6 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 its members are shown to have the same relation to convex functions as posynomials. Experimental results illustrating the effectiveness of this model have been

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 3: Comparison of Model Delay for C17 with SPICE

| Circuit | Unsized Delay (ns) | Unsized Area ($\mu$m) | $T_{spec}$ (ns) | Sized Area ($\mu$m) | Execution Time |
|---|---|---|---|---|---|
| C432 | 2.517 | 403.5 | 2.391 | 458.09 | 69s |
| | | | 2.265 | 499.39 | 130s |
| | | | 2.175 | 595.06 | 173s |
| | | | 2.136 | 603.08 | 179s |
| | | | 2.013 | 787.57 | 270s |
| C880 | 2.295 | 721 | 2.180 | 722.63 | 4s |
| | | | 2.066 | 727.74 | 11s |
| | | | 1.950 | 735.34 | 21s |
| | | | 1.836 | 752.94 | 42s |
| | | | 1.721 | 775.31 | 65s |
| | | | 1.606 | 847.84 | 89s |
| C499 | 3.644 | 1023 | 3.462 | 1023.35 | 3s |
| | | | 3.279 | 1025.68 | 9s |
| | | | 3.097 | 1031.27 | 18s |
| | | | 2.915 | 1048.99 | 51s |
| | | | 2.733 | 1104.73 | 166s |
| | | | 2.551 | 1233.98 | 384s |

Table 4: Results of sizing various circuits

reported, and the results of running the sizing algorithm with the improved model have also been included.

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

[4] 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.

[5] A. Nabavi-Lishi and N. C. Rumin, "Inverter models for cmos gates for supply current and delay evaluation," *IEEE Transactions on CAD*, vol. 13, pp. 1271–1279, October 1994.

[6] A. Chatzigeorgiou, S. Nikolaidis, and I. Tsoukalas, "A modeling technique for cmos gates," *IEEE Transactions on CAD*, vol. 18, pp. 557–575, May 1999.

[7] 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.

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

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