Osculating Thevenin Model for Predicting Delay and Slew of Capacitively Characterized Cells

Bernard N. Sheehan
Mentor Graphics Corporation, Wilsonville, Oregon USA

ABSTRACT

To extrapolate from one point to another using a line, one had better get the slope right. In this paper we apply a similar concept to the important problem in Static Timing Analysis (STA) of predicting cell timing for RC loads using capacitive characterization data. Instead of a line we have a Thevenin circuit, and instead of matching slopes we match load sensitivities. We present a table driven, highly accurate cell delay and slew prediction procedure that can improve STA when interconnect effects dominate. It is significantly more accurate (especially slew) than previously published C_{eff} methods.

Categories and Subject Descriptors
B.6.3 [**Logic Design**]: Design Aids – *simulation, verification.*

General Terms Algorithms, Verification.

Keywords Static Timing Analysis, Effective Capacitance

1. INTRODUCTION

While static timing analysis (STA) has traditionally treated interconnect as a lumped capacitor, the realities of deep sub-micron design require interconnect to be treated as a distributed RC or RLC network. Various methods for extrapolating cell timing to non-capacitive interconnect have been proposed: [1] suggests characterizing cells for π loads; [2] proposes an empirically derived C_{eff} function; [3] models the cell as a Thevenin circuit and derives a C_{eff} from this; [4] uses a Thevenin model with a piecewise linear ramp; [5] extends the Thevenin approach to networks with coupling; [6] uses a two region driver/load model; recently, [7] computes a C_{eff} from a current source approximation.

This paper presents a new method for calculating Thevenin-C_{eff} pairs based on a simple but effective idea—the recognition, namely, that the Thevenin model should not only match delay and slew at C_{eff}, but should also match delay and slew *sensitivities*. By analogy with fitted curves that match both ordinates and slopes, we call such a model an *osculating* model. In this paper we present a practical, accurate C_{eff} procedure based on the concept of osculating Thevenin models.

2. TIMING MODELS

In order to lend precision to our discussion, we introduce notation for the data normally available in a gate-level timing library. A **timing model** for a cell or gate G consists of a set of tables or equations which give delay and output slew as functions of capacitive load and input slew, i.e.,

$$t^G_{delay} = f_1(C_L, t_{slewIn}), \quad t^G_{slewOut} = f_2(C_L, t_{slewIn}) \qquad (2.1)$$

In practice, delay and slew are computed from the times a cell's input and output waveforms cross certain voltage thresholds-- $a_L V_{DD}$=.2, $a_D V_{DD}$=.5, $a_H V_{DD}$=.8 typically. Slew as we use the term is an interval, not a rate: it is the difference between output crossing times at $a_L V_{DD}$ and $a_H V_{DD}$.

3. THEVENIN TEMPLATE

As in [3], our Thevenin model M(t_0,t_r,R) consists of a resistance R in series with a saturating voltage ramp that begins at time t_0 and rises to V_{DD} after an additional time tr. The response of M when driving a capacitor C is

$$v_C(t) = V_{DD} v(t - t_0, t_r, RC) \qquad (3.1)$$

where $v(t, t_r, RC) =$

$$\begin{cases} \dfrac{t}{t_r} - \dfrac{RC}{t_r}\left(1 - e^{-t/RC}\right) & 0 \le t < t_r \\ 1 - \dfrac{RC}{t_r}\left(e^{-(t-t_r)/RC} - e^{-t/RC}\right) & t_r \le t \end{cases} \qquad (3.2)$$

4. OSCULATION

In a nutshell, here is what we will be attempting. A Thevenin model can have various admixtures of tr and R and still replicate the observed $t^G_{slewOut}$. Among *all* pairs of tr and R which correctly match *slew*, we choose that pair which also replicates *slew sensitivity*. This is the crux of our idea.

To develop this idea we start with the recognition that (2.1) tells us more than just delay and slew for a given t_{slewIn} and capacitive load C_L. Functions (2.1) give us, in effect, *curves* of t_{delay} and $t_{slewOut}$ versus C_L. We therefore know the tangents to these curves; in other words, we know load sensitivities.

Definition 4.1. A circuit model M for a cell is called an **osculating model** at operating point C_0 if the model predicts the same delays and slews at C_0 as (2.1), i.e. if

$$t^M_{delay} = f_1(C_o, t_{slewIn}) \qquad (4.1)$$

$$t^M_{slewOut} = f_2(C_o, t_{slewIn}) \tag{4.2}$$

and, moreover, it replicates one or both of the gate's load sensitivities at C_0:

$$\frac{d}{dC_L} t^M_{delay}\bigg|_{C_L=C_0} = \frac{df_1}{dC_L}(C_o, t_{slewIn}) \tag{4.3}$$

$$\frac{d}{dC_L} t^M_{slewOut}\bigg|_{C_L=C_0} = \frac{df_2}{dC_L}(C_o, t_{slewIn}) \tag{4.4}$$

We expect an osculating model to closely follow cell timing not just at the single load C_0 but also for loads in the *vicinity* of C_0.

5. OSCULATING THEVENIN MODEL

We now elaborate the concept of osculating model into an efficient design procedure for creating an osculating model *based on a Thevenin template.* Our procedure involves constructing special design tables from which we read off the Thevenin model parameters that match delay, slew, and *slew* sensitivity. We do not match *delay* sensitivity, since our Thevenin model $M(t_0,t_r,R)$ has only 3 parameters and therefore cannot satisfy all 4 conditions in definition 4.1.

6. DESIGN TABLES

The specification of suitable design tables is central to making our osculating approach practical. These tables are only computed *once* for a given technology. We use them to *directly* lookup of Thevenin parameters, rather than having to compute them for each cell by (3.2), which would be expensive.

The trick is to collapse *all* design problems (for given a_L, a_D, a_H) into *one* set of tables. We do this by scaling time. Specifically, we divide all time quantities by

$$UnitTime \equiv \frac{t^G_{slewOut}}{a_H - a_L} \tag{6.1}$$

We denote the resulting normalized quantity by placing a bar over its symbol. With this scaling, (6.1) takes the form

$$\bar{t}^G_{slewOut} = a_H - a_L \tag{6.2}$$

This, of course, is also the slew of a linear ramp with $\bar{t}_r = 1$, which is the motivation for our choice of *UnitTime.*

Consider now the following argument. Given (3.2) and a set of relative timing thresholds a_L, a_D, a_H, the equations

$$\begin{aligned} v(\bar{t}_L, \bar{t}_r, \overline{R}\overline{C}) &= a_L \\ v(\bar{t}_D, \bar{t}_r, \overline{R}\overline{C}) &= a_D \\ v(\bar{t}_H, \bar{t}_r, \overline{R}\overline{C}) &= a_H \end{aligned} \tag{6.3}$$

for threshold crossing times, together with the normalization constraint (6.2),

$$\bar{t}_H - \bar{t}_L = a_H - a_L \tag{6.4}$$

give 4 equations in 5 unknowns (a_L, a_D, a_H are known); such a system can, in theory, be solved for $\bar{t}_L$, $\bar{t}_D$, $\bar{t}_H$, $\overline{R}\overline{C}$ in terms of $\bar{t}_r$. Since delay, slew, and load sensitivities, in turn, can be computed from $\bar{t}_L$, $\bar{t}_D$, $\bar{t}_H$, $\overline{R}\overline{C}$ and $\bar{t}_r$, they also can be regarded as functions of $\bar{t}_r$.

Definition 6.1. **Osculating Thevenin design tables** consist of tabulations of the following four functions of $\bar{t}_r$:

$$\begin{aligned} \overline{R}\overline{C} &= A_1(\bar{t}_r) \\ \bar{t}_D &= A_2(\bar{t}_r) \\ \overline{R}\overline{C}(\partial\bar{t}_{delay} / \partial\overline{R}\overline{C}) &= A_3(\bar{t}_r) \\ \overline{R}\overline{C}(\partial\bar{t}_{slewOut} / \partial\overline{R}\overline{C}) &= A_4(\bar{t}_r) \end{aligned} \tag{6.5}$$

obtained from (3.2), (6.3), and (6.4).

Normalization moves our design into the space of Thevenin models with $\bar{t}_r$ in [0,1]. At one extreme is $\bar{t}_r$ =1: here R= 0 and the gate's output slew is entirely accounted for by the ramp time of the Thevenin voltage source. At the other extreme is $\bar{t}_r$ =0: here the Thevenin model consists of a *step* voltage and a resistor such that the gate's output slew is fully accounted for by the time constant of R charging C. Between these extremes, A_1 tells how to trade off R and $\bar{t}_r$ so as to hold output slew constant; A_4 gives the *slew* sensitivities of these various admixture of R and $\bar{t}_r$.

It turns out that A_3 (our normalized *delay* sensitivity) is nearly flat for much of $\bar{t}_r$ in [0,1]; it does not, therefore, provide a good basis for selecting $\bar{t}_r$. This is why in practice we match slew sensitivity rather than delay sensitivity.

7. BUILDING THE TABLES

In this section we describe how the design tables can be computed. In the next two sections we explain their use.

To calculate a set of design tables, we begin with the trivial solution of (6.3) and (6.4) at $\bar{t}_r = 1$:

$$\mathbf{x} \equiv \begin{bmatrix} \bar{t}_L \\ \bar{t}_D \\ \bar{t}_H \\ \overline{R}\overline{C} \end{bmatrix} = \begin{bmatrix} a_L \\ a_D \\ a_H \\ 0 \end{bmatrix} \tag{7.1}$$

If $\bar{t}_r$ is now replace by $\bar{t}_r - d\bar{t}_r$, a new vector **x** is found by iteratively solving

$$\mathbf{Mdx} = -\mathbf{F} \tag{7.2}$$

where

$$\mathbf{M} = \begin{bmatrix} \partial v_L / \partial\bar{t} & 0 & 0 & \partial v_L / \partial\overline{R}\overline{C} \\ 0 & \partial v_D / \partial\bar{t} & 0 & \partial v_D / \partial\overline{R}\overline{C} \\ 0 & 0 & \partial v_H / \partial\bar{t} & \partial v_H / \partial\overline{R}\overline{C} \\ -1 & 0 & 1 & 0 \end{bmatrix} \tag{7.3}$$

is the differential of (6.3) and (6.4) and

$$\mathbf{F} = \begin{bmatrix} v_L - a_L \\ v_D - a_D \\ v_H - a_H \\ \bar{t}_H - \bar{t}_L - (a_H - a_L) \end{bmatrix} \tag{7.4}$$

is the residual error. Our notation here is that the subscript on v indicates the crossing time at which a quantity is evaluated—e.g., $\partial v_L / \partial \bar{t}$ is the partial of equations (3.2) with respect to t evaluated at $\bar{t} = \bar{t}_L$.

Once Newton-Raphson iterations (7.2) converge, values of A_1 and A_2 at $\bar{t}_r - d\bar{t}_r$ are immediately available from the 2nd and 4th components of **x**. The delay and slew sensitivities--and hence A_3 and A_4--can be calculated from (3.2) using

$$\frac{\partial \bar{t}_{delay}}{\partial \overline{R}\overline{C}} = -\frac{\partial v_D / \partial \overline{R}\overline{C}}{\partial v_D / \partial \bar{t}},$$
$$\frac{\partial \bar{t}_{slewOut}}{\partial \overline{R}\overline{C}} = \frac{\partial v_L / \partial \overline{R}\overline{C}}{\partial v_L / \partial \bar{t}} - \frac{\partial v_H / \partial \overline{R}\overline{C}}{\partial v_H / \partial \bar{t}} \tag{7.5}$$

with the subscript on v again denoting the crossing-time at which a partial of v is evaluated.[1] Continuing in this way we compute the design tables decrementally for all values of $\bar{t}_r$ in [0,1].

8. USING THE TABLES

Given the design tables, it is a simple procedure to construct an osculating Thevenin model $M(t_0, t_r, R)$ for gate targets

$$t^G_{delay} = f_1(C_0, t_{slewIn}), \quad t^G_{slewOut} = f_2(C_0, t_{slewIn}) \tag{8.1}$$

$$S^G_{slewOut} = \frac{df_2(C_o, t_{slewIn})}{dC_L} \tag{8.2}$$

Normally, $S^G_{slewOut}$ is had by numerical differential of (2.1).

Algorithm 8.1. Osculating Thevenin Model Construction

Given: $C_0, t^G_{delay}, t^G_{slewOut}, S^G_{slewOut}$, and design tables.

Find: Osculating Thevenin model $M(t_0, t_r, R)$.

1) Inverse interpolate table A_4 to get $\bar{t}_r$ such that[2]

$$A_4(\bar{t}_r) = \frac{C_0 S^G_{slewOut}}{UnitTime} \tag{8.3}$$

2) $t_r = \bar{t}_r * UnitTime$ (8.4)

3) $R = \dfrac{\overline{R}\overline{C}}{C_0} * UnitTime$ (8.5)

4) $t_0 = t^G_{delay} - \bar{t}_D * UnitTime$ (8.6)

where $\overline{R}\overline{C} = A_1(\bar{t}_r)$ and $\bar{t}_D = A_2(\bar{t}_r)$.

Theorem 8.2. The Thevenin model so constructed matches the target delay, slew, and slew sensitivity at C_0.

Proof. Normalization ensures that M and G have the same output slew. Steps (1) and (4) above ensure that M and G have the same slew sensitivities and delays, respectively.

9. EFFECTIVE CAPACITANCE

We now apply the above theory to calculate the effective capacitance of an arbitrary RC load.

Algorithm 9.1. Effective Capacitance Using Osculating Models

Given: A gate timing model like (2.1), an RC load, and an initial estimate C_0 for C_{eff} (e.g. total capacitance to ground).

Find: The effective capacitance C_{eff} of an RC network driven by the gate, and a Thevenin model M for the gate when loaded by the RC network.

1) Calculate the osculating Thevenin model M for the gate at C_0 (Alg. 8.1). Let $t^M_{delay}(C_0)$ be the gate delay (same as M's delay, by construction) when loaded by C_0.

2) Drive the RC network with M to get $t^M_{delay}(RC)$, the delay of the Thevenin model when loaded by the RC network. One option here is to compute the pole/residue representation of the RC network, express the ramp response analytically, then find threshold crossings by Newton-Raphson.

3) Estimate C_{eff} from the following equation
$$t^M_{delay}(RC) - t^M_{delay}(C_0) \approx S^M_{delay} * (C_{eff} - C_0) \tag{9.1}$$
where
$$S^M_{delay} \equiv \partial t^M_{delay} / \partial C = A_3(\bar{t}_r)/C_0 \tag{9.2}$$

4) If $C_{eff} \approx C_0$ and $t^M_{delay}(RC) \approx t^M_{delay}(C_{eff})$, stop, otherwise set $C_0 = C_{eff}$ and go to step 1.

Actually, osculation is so powerful that the above algorithm gives excellent results without iteration (step 4). The iterative version almost always converges in 2 or 3 cycles even with stringent convergence criteria in step 4. Although we do not develop the point here, the same algorithm applies to RLC and coupled networks, with appropriate modifications to step (2).

10. RESULTS

We apply algorithm 9.1 to a set of drivers and RC networks of various lengths and topologies representative of interconnect for a 0.18 μm process. Figure 1 plots (o) gate delays predicted by our algorithm using a timing model like (2.1) against delays obtained by full SPICE-level simulation using transistor models

[1] Equations (7.5) make use the identity,

$$F(x,y) = 0 \Rightarrow \frac{dy}{dx} = -\left(\frac{\partial F}{\partial x}\right) \Big/ \left(\frac{\partial F}{\partial y}\right)$$

[2] Since R is constant (but unknown at this point in the calculation), we have

$$\overline{R}\overline{C}\frac{\partial \bar{t}_{slewOut}}{\partial \overline{R}\overline{C}} = \overline{C}\frac{\partial \bar{t}_{slewOut}}{\partial \overline{C}} = C\frac{\partial \bar{t}_{slewOut}}{\partial C}$$

an equivalence which underlies (8.3)

for the gates. Also plotted are: (*) delays from algorithm [3], and (+) delays obtained by substituting total load capacitance into (2.1). Figure 2 is a similar set of plots for gate output slew. Both figures are based on threshold levels of $a_L = .2, a_D = .5, a_H = .9$, as preferred by [3][3].

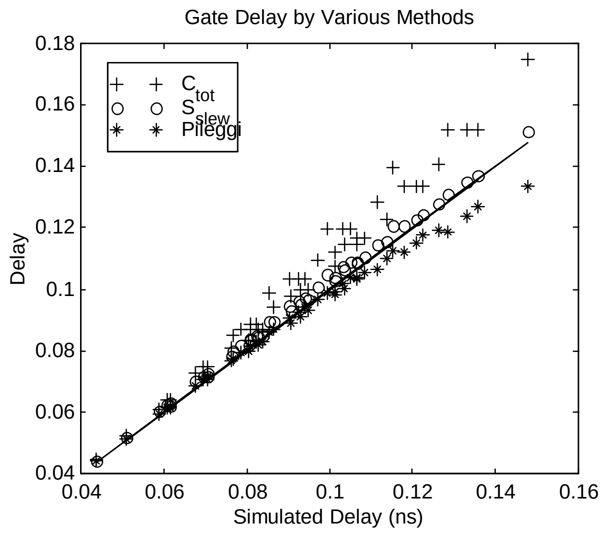

Figure 1. Cell Delay by Various Methods

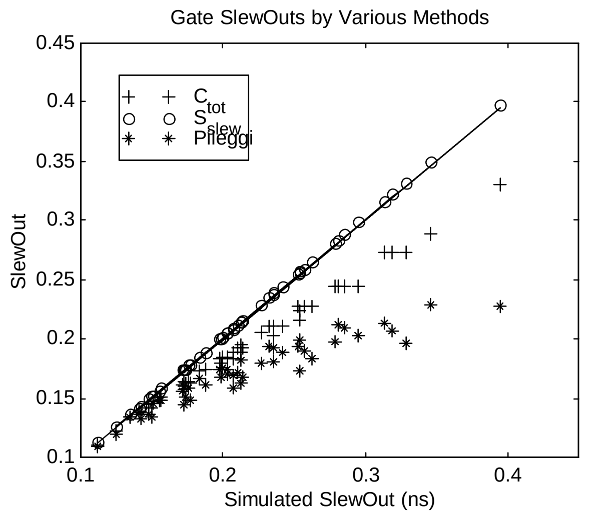

Figure 2. Output Slew by Various Methods

In all cases the osculating algorithm converged to C_{eff} in at most 3 iterations (usually 1 or 2 iterations).

Judging from Figures 1 and 2, we conclude that the osculating Thevenin algorithm is quite robust over a wide range of interconnect scales and topologies. It consistently predicts *slightly* longer delays than simulation--a desirable circumstance when timing long paths. Moreover, it predicts slews as accurately as delays. By contrast, most published C_{eff} algorithms do not even report on the accuracy of their slew predictions. Finally, it does not produce outlying points--a well-tempered consistency that makes it ideal for timing verification.

For the same data-set, algorithm [3] is inferior in all these respects; especially, [3] underestimates slew by as much as 50%. Slew, of course, influences downstream delays and is a key quantity governing noise, so correct slew is important for accurate interconnect analysis.

11. CONCLUSION

The problem we address in this paper is vitally important to Static Timing Analysis of deep sub-micron designs—namely, how to predict delays *and* slews of a cell for *distributed* loads using characterization data for *capacitive* loads. Previous researchers have attacked this problem in various ways, mostly by constructing a Thevenin model that *interpolates* cell timing.

Our contribution is to design a Thevenin model to approximate not only cell timing but also slew sensitivity. In this way we tune the Thevenin model to better mimic cell behavior in a *region* of loads; as confirmed by our tests, the resulting circuit is better at *extrapolating* to non-capacitive loads.

The osculating Thevenin method presented here has many advantages. It is efficient because it uses normalized design tables, the construction and use of which is described in the paper. The method gives surpassingly accurate results for both delay and slew. As an important pragmatic merit, the method does not require special, non-standard cell characterization, unlike many published techniques.

12. ACKNOWLEDGMENTS

I wish to thank Ellis Cohen and Archie Lachner for various discussions during commercial integration of this method.

[3] In applying algorithm [3], we computed the Thevenin resistance anew for each load using the formula $R = \dfrac{t_{90}(C_L) - t_{50}(C_L)}{C_L \ln 5}$ as given in equation (22) of [3].

[1] P. O'Brien and T. Savarino, "Modeling the Driving Point Characteristic of Resistive Interconnect for Accurate Delay Estimation," ICCAD 89, pp. 512-15

[2] R. Macys, S. McCormick, "A New Algorithm for Computing the 'Effective Capacitance' in Deep Submicron Circuits," IEEE Custom Integrated Circuits Conf. 1998, pp. 313-16

[3] F. Dartu, N. Menezes, L. T. Pileggi, "Performance computation for Precharacterized CMOS Gates with RC Loads," ," IEEE Trans. Computer Aided Design, May 1996, pp. 544-53.

[4] R. Arunachalam, F. Dartu, and L. Pileggi, "CMOS Gate Delay Models for General RLC Loading," ICCAD 97.

[5] F. Dartu, L. Pileggi, "Calculating Worst-Case Gate Delays Due to Dominant Capacitance Coupling,", DAC97, pp.46-51.

[6] J. Qian, S. Pullela, L. Pillage, "Modeling the 'Effective Capacitance' for the RC Interconnect of CMOS Gates," IEEE Trans. Computer Aided Design, Dec. 1994, pp. 1526-35.

[7] B. Sheehan, "Library Compatible Ceff for Gate-Level Timing", DATE 2002, pp. 826-30.