Library Compatible C_{eff} for Gate-Level Timing

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

Abstract

Accurate gate-level static timing analysis in the presence of RC loads has become an important problem for modern deep-submicron designs. Non-capacitive loads are usually analyzed using the concept of an effective capacitance, C_{eff}. Most published algorithms for C_{eff}, however, require special cell characterization or supplemental information that is not part of standard timing libraries. In this paper we present a novel C_{eff} algorithm that is strictly compatible with existing timing libraries. It is also fast, easily implemented, and quite accurate—within 3% of transistor-level simulation in our tests. The method is based on approximating a gate by a current source, estimating the delay difference when the gate drives the actual RC load and a reference capacitor, and then converting the delay discrepancy into a C_{eff} value. Central to carrying out this program is the innovative concept of delay correction transfer function.

1. Introduction

Given the overwhelming complexity of modern digital IC designs, gate-level static timing analysis is often performed as a conservative way to verify timing closure without the onerous task of generating test-vectors. At the same time, with the extreme miniaturization of deep-submicron technology, interconnect resistance must be accounted for during timing analysis. But handling resistance poses a problem for standard gate-level timing methodology, which is built on the capacitive load assumption. Traditionally, cells are characterized in the form of tables or equations that give cell delay and output slew as functions of load capacitance and input slew:

$$t_{delay} = f_1(C_L, t_{slewIn})$$
$$t_{slewOut} = f_2(C_L, t_{slewIn}) \qquad (1.1)$$

The problem we consider in this paper is how to apply capacitive characterization data (1.1) to distributed RC interconnect.

2. Prior Ceff Methods

To get a handle on resistive shielding, most authors use the concept of **effective capacitance** C_{eff}—which we might define as *the capacitance that produces the same gate delay as the actual RC load*

Various approaches to estimating Ceff have been published. One popular approach by Pileggi *et al* [1][2][3][4]calculates Ceff by first creating a Thevenin model as a proxy for the cell and then setting Ceff equal to the capacitance which sinks the same current as the RC load when driven by this Thevenin model. Like most Ceff algorithms, this one is iterative: one guesses a Ceff, computes a Thevenin model, then uses this Thevenin model to compute a better Ceff, etc. As presented, the Thevenin approach requires in addition to (1.1) either a user-supplied value for the Thevenin resistance R_d or non-standard characterization data such as delay times to *three* threshold levels—20%, 50% and 90%. Besides being computationally expensive, therefore, this method has the drawback of being incompatible with standard timing libraries.

To side-step the computational complexity point, Kahng and Muddu in [5] generalize [6] by proposing a Ceff algorithm which also uses a Thevenin circuit to drive a CRC π load. Differences from Pileggi's approach are: (1) [5] does not iteratively adjust the Thevenin model, but simply assumes an R_d and a ramp time T_R; (2) an approximation to the effective load during a *fast* gate transition, C_{ramp}, is computed by matching gate delays (Pileggi matches current delivery) for the π and C_{ramp} loads when driven by the Thevenin model; finally (3), adjustment for the actual speed of the cell is then made by taking a weighted average of C_{ramp} and total capacitance C_{tot}:

$$C_{eff} = C_{ramp} + (C_{tot} - C_{ramp})\frac{1}{1 + D_{LD}/D_{NL}} \qquad (2.1)$$

Here D_{LD} is the gate delay with C_{tot} as load and D_{NL} is the gate delay with no load—i.e. f_1 in (1.1) with C_L=C_{tot} and C_L=0. Drawbacks of this approach are the relative large error (the authors cite 15%) and murkiness about how to choose R_d and T_R.

A third approach of some interest is [7], which

postulates the existence of a technology-dependent C_{eff} function

$$\frac{C_{eff}}{C_1+C_2} = f_{Ceff}\left(\frac{C_2}{C_1+C_2}, \frac{t_{slewOut}}{R_2C_2}\right) \qquad (2.2)$$

where C_1, R_2, and C_2 are the parameters of the π network representing the load. For a given CMOS technology, this function is obtained during cell characterization by fitting to the data from numerous SPICE simulations of a "representative" gate: the gate first drives various π loads and then, for each π load, a capacitance is found which when simulated yields the same delay. Given (2.2), delay calculation proceeds by fixed point iteration between f_{Ceff} in (2.2) and f_2 in (1.1) until consistent values for C_{eff} and $t_{slewOut}$ are found. Merits of this approach are efficiency (the authors report a speed-up of at least 2 times over Pileggi's Thevenin scheme) and possibly improved accuracy. The drawback, of course, is the characterization effort required to get f_{Ceff}.

Common to all these approaches is that they presuppose supplementary data or non-standard cell characterization. While acceptable, perhaps, to silicon vendors or vertically-integrated companies which do their own library characterization and have in-house timing tools, these methods are problematic for design houses and general-purpose CAD tools which must work with no-frills, industry-standard cell libraries.

By contrast, we seek in this paper a method for estimating C_{eff} that is strictly manacled to (1.1) and does not suppose additional or non-standard cell characterization[1]. We say such a delay-calculation methodology is **Standard Library Compatible**. Clearly there is a practical need for Standard Library Compatible delay calculation methods—*even if* these methods are not as accurate as techniques that exploit additional, non-standard information. In fact, we shall see that compatible methods can have *very* competitive accuracy. The methods summarized above are not standard library compatible.

These preliminaries aside, the problem we wish to solve is the following. A CMOS gate or cell drives a distributed RC load. We are given the timing model for the gate—e.g. equations (1.1)—and are given the load in the form of a driving point admittance $Y_{RC}(s)$. We wish to find a capacitance C_{eff} such that the gate has the same delays for the RC and the C_{eff} loads.

The plan of attack for our C_{eff} algorithm, details of which follow, involves conceptually the following 4 steps:

1) Assume the gate delivers the same current *i(t)* to an RC load $Y_{RC}(s)$ and to a reference capacitance C_0; in other words, treat the gate as a (time-varying) current source.
2) For want of better information, assume the gate's output voltage when loaded by C_0 is a linear ramp with slew time given by f_2 in (1.1).
3) Compute the difference in gate delays for the two loads, using the concept of **delay-correction transfer function.**
4) Convert the delay correction into a C_{eff} value.

3. Driving Point Admittance Moments

The RC network loading a gate can be characterized by it's driving point admittance moments,

$$Y_{RC}(s) \equiv \frac{I(s;RC)}{V(s;RC)} = y_0 + y_1 s + y_2 s^2 + y_3 s^3 + \ldots \qquad (3.1)$$

In what follows we shall require only the three **admittance moments** y_1, y_2, y_3. These moments can be efficiently calculated by the methods in [8][9][10]. If the RC load itself has no dc path to ground, as we assume here, then y_0 vanishes. Moreover, it can be shown that y_1 in (3.1) is the *total capacitance* to ground of the RC network as seen from the gate; we denote this quantity by C_{tot}.

When the load is represented by a C1-R2-C2 π network, the first three moments are given by,

$$y_1 = C_1 + C_2 = C_{tot},\ y_2 = -R_2C_2^2,\ y_3 = R_2^2C_2^3 \qquad (3.2)$$

4. Current Source Approximation

Central to our algorithm is the assumption that a gate can be reasonably represented as a current source, at least for purposes of cell delay calculation. The advantage of this is that it obviates the need for a Thevenin resistance R_d. To motivate this assumption, consider the i-v characteristics of a typical CMOS inverter in Figure 1. Path A-B-C-D is the trajectory of the gate's output, and gate delay is determined by how quickly the output moves from A to C. The details of the path in i-v space will vary from load to load, of course, but if the loads are not *too* wildly different, then the currents will be almost the same at each instance of time because of the relatively flat aspect of the iv characteristics in region A to C. Without committing too grave an error, we will treat the currents as if they were *exactly* the same.

[1] Actually, there *is* one iota of additional information we must know. In order to interpret (1.1) quantitatively with precision, we must know the threshold levels used in defining or measuring gate delay and output slew. These are not always given as part of timing libraries, but our algorithm will assume that they are known.

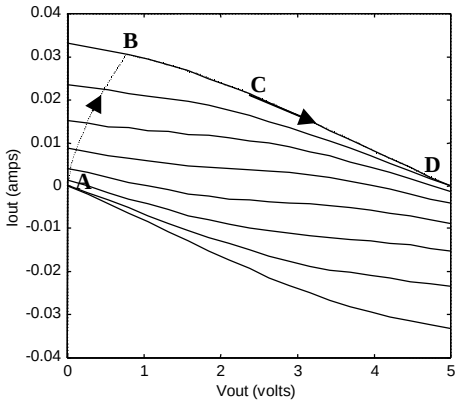

Figure 1. IV Characteristic of a CMOS Inverter

5. Delay Correction Transfer Function

We next develop the concept of delay correction transfer function, which is a natural consequence our current-source approximation for the gate. Mathematically, our current source approximation takes the form

$$i(t;RC) = i(t;C_0) \tag{5.1}$$

where C_0 is a reference capacitance against which we compare the RC response. (We will discuss later how to choose C_0.) Taking Laplace transforms and expressing current as admittance times voltage, (5.1) becomes

$$Y_{RC}(s)V(s;RC) = Y_{C_0}(s)V(s;C_0) \tag{5.2}$$

This last equation can be written suggestively as

$$H(s) \equiv \frac{V(s;RC)}{V(s;C_0)} = \frac{C_0 s}{y_1 s + y_2 s^2 + y_3 s^3 + \ldots} \tag{5.3}$$

where we have used (3.1) and $Y_{C_0}(s) = C_0 s$. H(s), the **delay correction transfer function**, tells us how to correct or alter the voltage response to the lumped load C_0 in order to get the voltage response for the RC load.

Although we could work directly with (5.3), it is convenient for calculations to approximate H(s) by

$$\tilde{H}(s) = \left\{1 + \frac{ks}{(as+1)}\right\}\left(\frac{C_0}{y_1}\right) \tag{5.4}$$

with *k* and *a* being chosen so that the first 3 moments of H and $\tilde{H}$ match:

$$k = -\frac{y_2}{y_1}, \quad a = -\left(\frac{y_3}{y_2} + k\right) \tag{5.5}$$

Using the monotonicity property of admittance moment ratios [11], it can be shown that *k* and *a* are always positive for RC networks.[2]

[2] For π loads, the positivity of *a* and *k* follows immediately from (3.2), which substituted into (5.5) yields the intuitively appealing expressions

$$k = (R_2C_2)\frac{C_2}{C_1 + C_2}, \; a = (R_2C_2)\frac{C_1}{C_1 + C_2}$$

The response of $\tilde{H}$ to an infinite ramp is

$$L^{-1}\left[\frac{\tilde{H}_{LD}}{Ts^2}\right] = \left\{\frac{t}{T} + \frac{k}{T}(1 - e^{-t/a})\right\}\left(\frac{C_0}{y_1}\right) \tag{5.6}$$

a result we will use in a moment.

6. Output Waveform for Capacitive Loads

In order to calculate V(s;RC) from (5.3), we need V(s; C_0). Timing model (1.1) does not give us details about the shape of a gate's output waveform. The only thing we know from (1.1) about the waveshape is the slew time between two reference thresholds—20% and 80%, say. Lacking any better information, a natural choice for $V(s;C_0)$ is a saturating ramp

$$V(s;C_0) \approx V_{DD}\frac{(1 - e^{-Ts})}{Ts^2} \tag{6.1}$$

where the rise time T is inferred from timing model (1.1)

$$T = \frac{V_{DD}}{0.8V_{DD} - 0.2V_{DD}} f_2(C_0, t_{slewIn}) \tag{6.2}$$

Note that $V(s;\ C_0)$ crosses threshold αV_{DD} at time αT. We will use this fact, too, in a moment.

7. Delay Correction

Substituting (6.1) into (5.3) and approximating $H(s)$ by $\tilde{H}(s)$, the response for the RC load can be estimated as

$$V(s;RC) \approx \tilde{H}(s)V_{DD}\frac{(1 - e^{-Ts})}{Ts^2} \tag{7.1}$$

If $\bar{a} = y_1 a / C_0 < 1$, we are guaranteed that $v(t;RC)$ will reach threshold before T, since *k* positive.[3] Hence (5.6) applies, and the threshold crossing time t^* can be found by solving

$$\frac{t^*}{T} + \frac{k}{T}(1 - e^{-t^*/a}) = \frac{y_1 a}{C_0} \tag{7.2}$$

8. C_{eff} Algorithm

According to the above discussion, the gate output voltage for an RC load reaches threshold αV_{DD} at a time

$$\Delta t = aT - t^* \tag{8.1}$$

earlier than it would if the gate were loaded by lumped capacitor C_0. This same reduction in delay can be

[3] If $\bar{a} \geq 1$, we have chosen too small of a reference capacitance C_0.

achieved by replacing C_0 with

$$C_{eff} = C_0\left(1-\frac{\Delta t}{aT}\right) \qquad (8.2)$$

This equation follows from the defining relation for a capacitor,

$$C\frac{\Delta V}{\Delta t} = i \qquad (8.3)$$

On the one hand, we know that C_0 charges to αV_{DD} in a time αT—see 6.1; on the other hand, we want C_{eff} to charge to αV_{DD} in a time $aT - t^*$. Hence, invoking our load-invariant current assumption,

$$C_0\frac{aV_{DD}}{aT} = C_{eff}\frac{aV_{DD}}{aT - t^*} \qquad (8.4)$$

which rearranged gives (8.2).

We have yet to describe how to choose our reference capacitance C_0. One natural choice C_{tot}. However, we can do better. Equation (7.2) is based on matching *three* moments of $Y_{RC}(s)$. If we match only *two* moments instead, we get a simple and quick guess for C_{eff} , which we take as our C_0.

With this wrinkle for choosing C_0 added in, our C_{eff} algorithm takes the following form. Let α_L, α_M, α_H be the voltage thresholds used in measuring (defining) delay and slew in (1.1). To compute C_{eff} and the associated gate delay for a load $Y_{RC}(s)$, perform the following sequence of calculations:

1) Compute k, a from (5.5)

2) $T = \dfrac{1}{a_H - a_L} f_2(y_1, t_{slewIn})$

3) $C_0 = y_1 \max\left\{\dfrac{a}{a+k}, 1-\dfrac{k}{a_M T}\right\}$

4) $T = \dfrac{1}{a_H - a_L} f_2(C_0, t_{slewIn})$

5) Solve $\dfrac{t^*}{T} + \dfrac{k}{T}\left(1-e^{-t^*/a}\right) = \dfrac{y_1 a_M}{C_0}$ for t*

6) $\Delta t = t^* - a_M T$

7) $C_{eff} = C_0\left(1+\dfrac{\Delta t}{a_M T}\right)$

8) $t_{delayGate} = f_1(C_{eff}, t_{slewIn})$

9. Experimental Results

To examine the validity of our Ceff algorithm for a particular case, consider a CRC π network with elements of 2.0pF, 400 ohms, and 2.0pF. We drive this network with a gate whose IV characteristics are in Figure 1. For comparison, we also drive with the same gate a lumped capacitor C_{tot}=4.0pF and capacitor C_{eff} as given by our algorithm. Figure 2 plots currents and voltages versus time for the three loads. Although (5.1) is not satisfied during the entire transition, currents for C_{eff} and CRC *do* match closely during the first 0.5ns, which is sufficient to accurately predict gate delay.

To further test our method of calculating C_{eff}, we apply it to a diverse set of RC topologies representing global interconnect for a typical 0.18 micron process. We drive each RC network first with an inverter based on the Schichman-Hodges MOSFET equations [12], then with one based on the α power law model proposed by T. Sakurai and R. Newton[13]; channel-length modulation effects are included for both drivers.

Figure 3 plots the error between the gate delays predicted by our algorithm and those obtained by transistor-level simulation. For comparison we also plot the delay errors obtained using (1.1) with $C_L=C_{tot}$ and using the algorithm in [1]. We do not give results for [5], since that algorithm as published is not fully defined (free parameters R_d and T_r appear in the formulation); nor do we give results for [7], since we do not have a f_{Ceff} function available to us. Looking at Figure 3, we see that our algorithm is accurate to within 2% or 3% and is more accurate than the Thevenin method in [1].

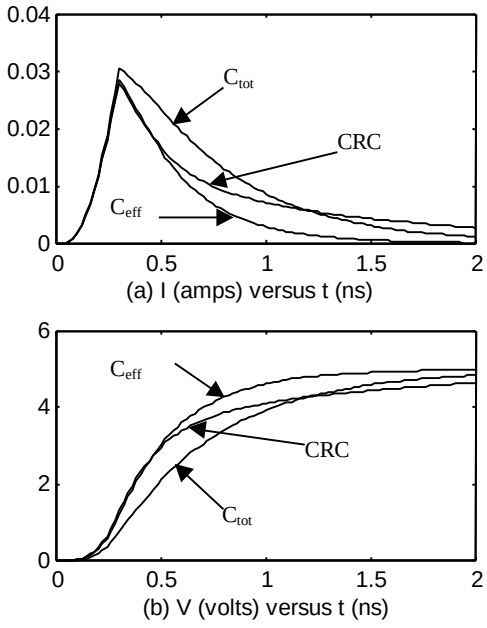

Figure 2. Gate driving CRC, Ctot, and Ceff

Also note that our algorithm is almost always conservative—it predicts a (slightly) longer gate delay than the actual delay. This property is desirable when doing timing analysis to check for long paths.

10. Conclusion

By way of conclusion, let us consider the advantages of the C_{eff} algorithm we have presented here. That the algorithm is fast can be judged by looking at the summary in Section 8: only two evaluations of f_2 and one of f_1 are required, and the only non-trivial step is (5), which involves iteratively solving a simple transcendental equation. It turns out that one can establish *a priori* a narrow search interval for t*, so this equation in practice converges quickly and reliably. The algorithm is also easy to implement—just a few formula evaluations and the iterative solution of one simple non-linear equation. Despite these advantages on the side of simplicity, our method is also surprisingly accurate—within 3% for our test set and more accurate than the Thevenin method of [1], even though the latter method demands additional characterization data to furnish an R_d. Finally and most importantly, our method is Standard Library Compatible. It doesn't need delays to *three* threshold levels, like [1]; it doesn't ask the user to supply an R_d and T_R, like [5]; it doesn't need an empirical f_{Ceff} function, like [7]. Fast, simple, and accurate, the algorithm works with *any* run-of-the-mill gate-level timing library. This sets it apart from most published methods and gives it very wide applicability.

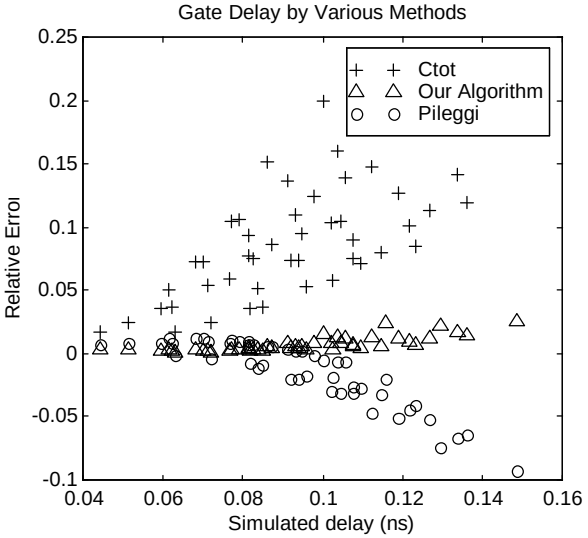

Figure 3. Gate Delay Test Set

1 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.

2 C. Ratzlaff, S. Pullela, and L. T. Pillage, "Modeling the RC interconnect effects in a hierarchical timing analyzer," *Proc. Custom Integrated Circuits Conf.*, May 1992.

3 F. Dartu, N. Menezes, J. Aian, L. T. Pillage, "A Gate Model for High-Speed CMOS Circuits,", *DAC 94*, pp. 576-80.

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

5 A. B. Kahng, S. Muddu, "New Efficient Algorithms for Computing Effective Capacitance," *ISPD 98*, pp. 147-51.

6 S. P. McCormick, "Modeling and Simulation of VLSI Interconnections with Moments," *PhD Thesis*, MIT, June, 1989.

7 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

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

9 C. Ratzlaff and L. Pillage, "RICE: Rapid Interconnect Circuit Evaluation Using AWE," *IEEE Trans. on CAD*, June 1994.

10 B. N. Sheehan, "An AWE Technique for Fast Printed Circuit Board Delays," *DAC 96*, pp. 539-43.

11 A. Odabasioglu, E. Acar, M. Celik, and L. T. Pileggi, "S2P: A new accurate metric for RC interconnects," *CMU-Technical Report*, 1998.

12 H. Schichman and D. A. Hodges, "Modelling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits," *IEEE J. Solid-State Circuits*, SC-3, 1968.

13 T. Sakurai and R. Newton, "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas,", *IEEE J. Solid-State Circuits*, April, 1990.