Fast Interconnect and Gate Timing Analysis for Performance Optimization

Soroush Abbaspour, Massoud Pedram, Amir Ajami, Chandramouli Kashyap

IBM Corp., University of Southern California, Magma Design Automation, Intel Corp.

Abstract

Static timing analysis (STA) is a key step in the physical design optimization of VLSI designs. The lumped capacitance model for gate delay and the Elmore model for wire delay have been shown to be inadequate for wire-dominated designs. Using the effective capacitance model for the gate delay calculation and model order reduction techniques for wire delay calculation is prohibitively expensive. In this paper, we present sufficiently accurate and highly efficient filtering algorithms for interconnect timing as well as gate timing analysis. The key idea is to partition the circuit into low and high complexity circuits, whereby low complexity circuits are handled with efficient algorithms such as total capacitance algorithm for gate delay and the Elmore metric for wire delay and high complexity circuits are handled with sign-off algorithms. Experimental results on microprocessor designs show accuracies that are quite comparable with sign-off delay calculators with more than of 65% reduction in the computation times.

1. Introduction

As CMOS process technologies scale down towards nanometer regimes, the accuracy and efficiency of static timing analysis (STA) has become increasingly important to the successful timing closure of an integrated circuit design flow. Most STA tools break the analysis into two parts: 1) gate delay calculation, and 2) interconnect or wire delay calculation. It is widely accepted that computing gate delays using a lumped total capacitance and computing the wire delay using the Elmore model are grossly inadequate for the wire dominated designs of today. To address this drawback, various model order reduction techniques such as AWE [2], PRIMA [13], etc. have been proposed to accurately model the interconnect delay. On the other hand, the load as seen by the driving gate is modeled by a reduced-order model such as the π-model [8]. Therefore, an "effective capacitance" technique was proposed [9] which provides a way to map the π-load to an equivalent capacitance (in the sense of gate propagation delay).

While these approaches exhibit good accuracies and are used for sign-off level, they can be too computation-extensive to be used in the context of physical design optimization. Recognizing this shortcoming, there has been much research on deriving closed-form formulas or delay metrics for wire delay estimation [3][5][12][17]. However, these delay metrics introduce lots of error to the STA results and are not reliable enough to be used for optimization. On the other hand, not much attention has been given on speeding up the gate delay calculation, which as we show next, accounts for a significant portion of the overall STA run time.

We measured the time spent in various parts of a commercial sign-off STA tool on many designs including two 90nm technology microprocessor designs, which we call them Design#1 and Design#2. Table 1 presents important statistics for these two designs. It also reports the time that the tool spends on the "gate timing analysis", "interconnect timing analysis", and the whole STA runtime. We find that, on average, about 60 percent of the CPU time of STA is spent on the gate timing analysis.

For accuracy purposes, Figure of Merit (FOM) metric has been used to measure how poor is the distribution of negative slacks (i.e. worst-negative slack at end points) in the design. "FOM integral" represents the summation of all the negative slack endpoints in the designs. This metric is chosen because it captures how many paths are timing critical and need to be fixed. In comparison, the worst slack gives one number that indicates the worst negative slack of the design that need to be fixed. "FOM Number" is the number of negative slack end points.

Table 1: 90nm Microprocessor Design Specifications

Design	Gates	Nets	Gate TA	Interconnect TA	STA
#1	1.1M	1.4M	1146(s)	241(s)	2010(s)
#2	2M	2.3M	1945(s)	388(s)	3235(s)

We applied different combinations of interconnect timing analysis algorithms (AWE or Elmore) and gate timing analysis algorithms (effective capacitance and lumped capacitance) on many designs including both Design#1 and Design#2. As an example, we have provided the FOM results for Design#2 in Table 2. It can be derived that although Elmore metric is efficient but can change the FOM results by orders of magnitude. In addition, C_{total} can change the FOM results by orders of magnitude with respect to the golden FOM results (i.e. using AWE for interconnect timing analysis and C_{eff} for gate timing analysis). Thus, it is important to have new interconnect and gate timing analysis algorithms which are capable of accurately and efficiently calculating interconnect and gate delay and slew along a path.

Our first contribution in this paper is to present a filtering technique for speeding up the interconnect timing analysis step in an STA tool, while maintaining a reasonable level of accuracy. As we will see later in this work, Elmore delay based algorithms could be accurate for some cases of nets or interconnects and we do not need to use higher order moments based algorithms for delay and slew calculation. For some other cases, we may need to use two moments for the interconnect delay and slew calculation where a new efficient metric has been proposed. Finally, for other cases, we may need to use AWE based algorithm for interconnect delay and slew calculation.

Table 2: Design #2 FOM Results

Interconnect Algorithm	Gate Algorithm	FOM No.	FOM Slack Worst	FOM Integral
AWE*	Ceff*	260947	-1.345	-57138
AWE	Ctotal	352716	-1.807	-98780
Elmore	Ceff	443123	-2.012	-99877
Elmore	Ctotal	466654	-2.254	-101234

*The first row is the golden result.

Our second contribution in this paper is to present a filtering technique for speeding up the gate delay calculation step in an STA tool, while maintaining a reasonable level of accuracy. The filtering technique resorts to a necessary condition check to determine if C_{total} can be used for the gate delay and/or output slew calculations without introducing a significant inaccuracy.

The remainder of this paper is organized as follows: In section 2, we present the threshold based filtering algorithm for fast interconnect timing analysis. In section 3, the fast gate timing analysis is presented. Section 4 presents the conclusion remarks.

2. Fast Interconnect Timing Analysis

In this section, we focus on the interconnect timing analysis. Elmore [1] used the first moment of the impulse response transfer function and approximated the median (the desired delay) by the mean of the impulse response. It is well established that the Elmore delay metric can be off by orders of magnitude in some cases. To conquer the accuracy problem, different delay metrics has been proposed by using higher moments [3][5][12][17]. These delay metrics try to use a fixed number of moments to find the delay and slew, accordingly. Using the fact that, for some nets, Elmore based delay is accurate enough and for some nets, the delay metrics

based on two or higher moments should be used; therefore, none of the previous works would give accurate, yet efficient, results.

This section presents TFA, a threshold-based filtering algorithm, for propagation delay and output slew calculation of high-speed VLSI interconnects. The TF algorithm partitions the circuit nets into three groups based on their top-level characteristics: one group of nets – called *low complexity nets* - lend themselves to accurate delay calculation with the Elmore delay whereas the second and third groups of nets – called *medium* and *high complexity nets* – demand more sophisticated and time-consuming delay calculations based on the first two or higher moments of the impulse response transfer function, respectively. The idea of dividing the circuit nets into different classes for the purpose of minimizing the computational workload of a delay calculation engine while providing reasonable accuracy for the computed delays is quite intuitive and straightforward. The key challenge, however, is in being able to do the examination and classification of the nets accurately. This is precisely what we accomplish in this section by our threshold-based filtering algorithm, as will be shown later.

The remainder of this section is organized as follows. In section 2.1, by using the circuit theory, a new analytical closed-form equation for calculating the delay and output slew of an interconnect line under step and ramp inputs is presented. Section 2.2 uses these analytical equations as a signature function to sort the nets into simple and complex ones. Experimental results are reported in section 2.3.

2.1 Analysis of the Threshold-Based Filtering Algorithm

The ratio of the voltage of the output node, $V_o(s)$, to the input voltage, $V_i(s)$, for a linear time-invariant (LTI) system is called the voltage transfer function, $H(s)$. For an *RC* tree, this ratio can be written as:

$$H(s) = V_o(s)/V_i(s) = \sum_{k=0} m_k s^k \qquad (1)$$

where m_i is called the i^{th} moment of the voltage transfer function. If a unit ramp input with α–β% rise time of $T_{in(\alpha-\beta)}$ is applied to such an *RC* segment, then the α–β% output transition time can be written as [5][10][17]:

$$T_{out(\alpha-\beta)} = \sqrt{\left(T_{in(\alpha-\beta)}\right)^2 + \left(RC\ln\left(\frac{100-\alpha}{100-\beta}\right)\right)^2} \qquad (2)$$

Based on Eqn. (2), if the ratio of the input slew to the corresponding *RC* value for two different *RC* circuits is the same, then the ratio of their output transition times to the *RC* values will be the same. Considering *RC* value to be an indicator for Elmore delay of more general resistive-capacitive tree, this fact implies that the *Input_slew/Elmore* is a key characteristic for the delay calculation, and interestingly, one of the most important factors when determining the degree of accuracy of an Elmore delay calculator. Therefore, for an *RC* tree, the output slew can be calculated as:

$$T_{out(\alpha-\beta)} \cong \sqrt{\left(T_{in(\alpha-\beta)}\right)^2 + \left(Elmore\times\ln(\frac{100-\alpha}{100-\beta})\right)^2} \qquad (3)$$

where T_{out} and T_{in} denote the transition times at the output and input nodes of the *RC* tree and Elmore denotes the Elmore delay.

For an *RC* tree, considering only the first order moment in delay calculation implies that the second order moment is the square of the first moment, which is not always true due to the shielding effect of the wires. In general, this m_2/m_1^2 ratio varies from a number smaller than 1 to almost 50. Therefore, we need to consider the effect of higher moments. By considering the first two moments of the impulse response transfer function, we can approximate $H(s)$ by:

$$\tilde{H}(s) = \frac{1}{1-m_1 s+(m_1^2-m_2)s^2} = 1+m_1 s+m_2 s^2+(2m_1m_2-m_1^3)s^3+\ldots \qquad (4)$$

As a result, we approximate the α-β% output transition as:

$$T_{out(\alpha-\beta)} \cong \sqrt{\left(T_{in(\alpha-\beta)}\right)^2 + \left(Elmore\times\gamma_{\alpha-\beta}\right)^2} \qquad (5)$$

where γ is a function of m_2/m_1^2. In addition, by approximating the step response of a second order system, we calculate the γ value in Eqn. (5) as a linear function of m_2/m_1^2 as follows:

$$\gamma_\alpha = \lambda_\alpha(m_2/m_1^2)+\kappa_\alpha \qquad (6)$$

This linear approximation is accurate enough for the analysis and helps us to understand the sensitivity of the delay and slew calculation to the shielding effect. However, one can use higher order terms and get a more accurate γ value.

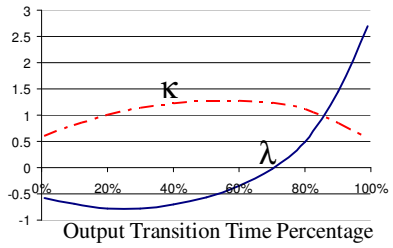

Figure 1: λ and κ vs. output transition percentage

The values of γ and κ are calculated and shown in Figure 1. From Figure 1 and Eqns. (5) and (6), since λ is multiplied by m_2/m_1^2, it is obvious that the 10% to 90% of the transition time is sensitive to the m_2/m_1^2 change. It also shows that the around 70% point is not as sensitive to the value of m_2/m_1^2 (and thereby to shielding effect) as the 50% transition or any other points are. Figure 2 shows this scenario for different values of m_2/m_1^2. More precisely, if m_2/m_1^2changes by 20%, the 10% point to 90% point transition time changes by as much as 43% whereas the 70% point output transition time changes slightly. Figure 1 also can help us to understand how much error we can incur in our delay/slew analysis if we do not consider higher moments (m_2, m_3, …) for calculating the propagation delay and slew.

Based on Eqn. (4), considering only the first two moments of the impulse response transfer function is equivalent to assuming that the third moment is equal to $2m_1m_2-m_1^3$. Interestingly, the output transition times are not sensitive to $m_3/(2m_1m_2-m_1^3)$ as much as they are sensitive to the m_2/m_1^2. However, to have an accurate interconnect timing analyzer, when $m_3/(2m_1m_2-m_1^3)$ becomes larger than a critical value, the AWE method need to be used to find the delay and slew.

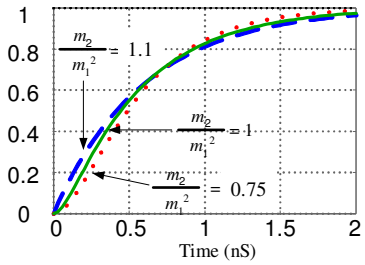

Figure 2: Step response of a second order system for three values of m_2/m_1^2

The advantage of this methodology is that the latter scenario occurs rarely in today's high frequency digital circuits. Indeed, the $m_3/(2m_1m_2-m_1^3)$ is linearly dependent on the m_2/m_1^2. Thus, whenever m_2/m_1^2 value exceeds a critical limit the effect of third moment should also be taken into account by using the AWE method. This critical limit can change according to the degree of precision needed during the path timing analysis.

2.2 The Filtering Algorithm

As observed earlier, the *Input_slew/Elmore* is an extremely important factor in determining the propagation delay and slew. When the value of

Input_slew/Elmore becomes greater than a critical limit, then there is one dominant pole in the voltage transfer function, and therefore, the first moment would be sufficiently accurate for calculating the output delay and transition time. It can be observed that the Elmore delay and Elmore slew errors are functions of the *Input_slew/Elmore*. If the *Input_slew/Elmore* is greater than the critical threshold, the Elmore delay error is quite negligible. However, when *Input_slew/Elmore* is less than this threshold, the Elmore delay may result in a considerable error. The proposed filtering algorithm makes use of this behavior to classify the stage delays based on the critical value of *Input_slew/Elmore.* The parameters used in the filtering algorithm are defined as follows:

ϕ is defined as the *Elmore threshold* value. When the first moment of the voltage transfer function is less than this threshold, then the estimation errors of the slew and delay (which are calculated based on Elmore metric) are small because the critical path delays are not sensitive to these estimation errors.

μ is defined as the *dominant-pole cut off ratio*. When the value of the input slew over Elmore delay is greater than μ, then the Elmore-based timing analysis is accurate enough.

η is defined as the *second moment filtering-threshold* value. If the value of m_2/m_1^2 is less than this threshold, Eqn. (5) becomes the basis of the timing analysis. For an interconnect line with m_2/m_1^2 greater than this threshold, the AWE method should be used to calculate the higher moments. As η goes towards 1, the delay and slew calculations become more accurate but the runtime increases.

Therefore, given the input slew T_r, the TF algorithm for calculating the stage delay is as follows:

Threshold-based Filtering Algorithm

1. Calculate the first moment m_1 ;
2. if ($|m_1| \le \phi$ || $T_r / |m_1| \ge \mu$) {
   Calculate Elmore-based delay and slew;
   Return; }
3. Calculate m_2 ;
4. if ($m_2/m_1^2 \le \eta$) {
   Use Eqn. (5) to calculate delay and slew;
   Return; }
5. Calculate higher moments;
6. Use *AWE* to calculate the delay and slew;
7. Return delay and slew values;

2.3 Experimental Results

To verify the accuracy of the proposed filtering technique, the algorithm was applied to many high-performance designs including Design#1 and Design#2. The design specifications are shown in Table 1. All the experimental runs of the proposed algorithm were done on a 2.0 GHz X86-based PC with 2GB of RAM. The sign-off FOM results (using AWE for interconnect timing analysis and C_{eff} for gate timing analysis) are shown in the first row of Table 3 for Design#1 and Table 4 for Design#2. We changed the interconnect timing analysis algorithm from AWE to Elmore and D2M and reported the results in the above-mentioned tables. As it is shown, the FOM results change by orders of magnitude when we apply Elmore and D2M, however, the runtime decreases significantly. We also applied the TFA algorithm using ϕ = 4*ps*, μ =7 and η = 1.44. The proposed filtering algorithm improves the interconnect timing analysis runtime by 65%. In addition, TFA resulted in a very small amount of error in FOM results comparing to AWE-based delay calculator results. For Design#1, the max/average/min errors are 6% / 1%/-2% while for Design#2 the max/average/min errors are 8% / 1%/-3%. Decreasing ϕ and η and increasing μ tends to increase the accuracy at the expense of higher runtime. In fact, the filtering algorithm with $\phi \to 0$, $\mu \to \infty$, and $\eta \to 0$ simply resort to the AWE-based timing analysis. Similarly, with $\mu \to 0$, the proposed filtering algorithm reduces to the Elmore-based for delay and slew calculation.

Evidently, there is a trade off between efficiency and accuracy when choosing the threshold parameter values. As an example, ϕ is the threshold value that filters those cases where the Elmore delay calculator returns a small delay value. Definition of "small" is however design and technology dependent. 10ps in 180nm technology may be a small value while in 90nm technology it may not be considered small anymore. One can choose different ϕ values in different stages of the design flow, starting with a large ϕ value but choosing smaller ones as he/she proceeds from earlier design stages toward the sign-off stage.

Table 3: Design#1 Experimental Results

Algorithm	FOM #	FOM Slack Worst	FOM Integral	Intercon. TA
AWE	1160	-0.196	-63	241
TFA	1170	-0.198	-64	89
D2M	1834	-0.222	-93	87
Elmore	5699	-0.361	-630	47

Table 4: Design#2 Experimental Results

Algorithm	FOM #	FOM Slack Worst	FOM Integral	Intercon. TA
AWE	260947	-1.345	-57138	388
TFA	262385	-1.343	-57847	139
D2M	322543	-1.544	-79176	132
Elmore	443123	-2.012	-99877	90

* C_{eff} algorithm is used for gate timing analysis for all experiments in Tables 3 and 4.

3. Fast Gate Timing Analysis

In this section, we present a filtering technique for speeding up the gate delay calculation step in an STA tool, while maintaining a reasonable level of accuracy: The filtering technique resorts to a necessary condition check to determine if C_{total} can be used for the gate delay and/or output slew calculations without introducing a significant inaccuracy. The motivation for the filtering approach is given in Figure 3, where it is shown that the distribution of the "actual effective capacitance" over the "total capacitance" in a design is highly skewed towards one. As shown in Figure 3, for "Design#2", the mean of the distribution of C_{eff}/C_{total} ratio is equal to 0.97. We have observed similar behavior in many other large industrial designs.

This section is organized as follows: Section 3.1 reviews the background and previous work in the area of gate timing analysis. Section 3.2 describes the filtering technique mentioned above for speeding up the gate delay and slew analysis. Experimental results are reported in section 3.3.

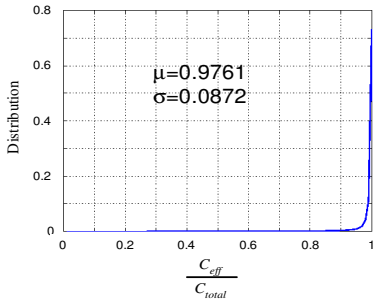

Figure 3: Distribution of the C_{eff}/C_{total} ratio for Design#2

3.1 Background

In VDSM technologies, we cannot neglect the effect of interconnect resistances of the output loads. Using the sum of all load capacitances as the capacitive load is simple, but can be quite pessimistic [16]. A more accurate approximation for an n^{th} order load seen by the gate (i.e., a load with n distributed capacitances to ground) is to use a second order RC-π model [8]. Therefore, the "effective capacitance" approach has been proposed [9][14][16] whereby the RC-π load is approximated by an *equivalent capacitance*, C_{eff}.

All of effective capacitance approaches resort to the iterative calculation of C_{eff} for the given circuit scenario, which can be costly in the context of physical design optimization tools. In this section, we present a filtering

approach that resorts to a necessary condition check to determine if C_{total} algorithm is sufficient for evaluating the delay and/or output slew of logic gates, and thereby, avoid effective capacitance calculations.

As shown in Figure 3, for most cases of the gate timing analysis, C_{eff} is very close to C_{total} i.e., if we are able to identify these cases, it will then be possible to use C_{total} algorithm for the gate delay and/or output slew calculation for these cases, and employ the C_{eff} algorithm for the remaining cases. To find out the type of the STA case that we must perform on a circuit configuration, we resort to an efficient and accurate condition check.

Problem statement: Given is a CMOS driver whose input rise time is T_{in} and drives an output *RC*-π load. The problem is to find a robust and efficient necessary condition check to distinguish between cases that can be accurately handled by using the C_{total} algorithm and those cases that need the iterative C_{eff} algorithm for gate propagation delay and/or output slew calculation during the physical design optimization process.

3.2 The Proposed Filtering Technique

In our quest for a robust and efficient necessary condition, we start with the effective capacitance definition. Based on its definition, the effective capacitance, C_{eff}, is a pure capacitance that replaces an *RC*-π load and has the property that it stores the same amount of charge as the *RC*-π load until a certain point of the output voltage transition (e.g., the 50% point of the output transition). We assume that the output voltage waveform for the CMOS driver behaves as a combination of ramp and exponential waveforms and therefore, actual C_{eff} must be obtained as a simple average of the C_{eff} obtained for ramp output waveform and the C_{eff} obtained for exponential output waveform.

In the following, we calculate C_{eff} for ramp and exponential waveforms of the gate output voltage. Modeling gate output waveform as exponential voltage waveform, we have shown that the iterative effective capacitance equation for matching any θ% point of the gate output transition time can be written as (derivations are omitted for brevity):

$$C_{eff}^{Exp}(\theta) = C_n + k_{Exp}(\theta)C_f \quad where$$
$$k_{Exp}(\theta) = \left[1 + \frac{y}{\theta}\left(e^{\ln(1-\theta)/y} - 1\right)\right] \quad and \quad y = \ln\left(\frac{1-\alpha}{1-\beta}\right) \times \frac{R_\pi C_f}{T_{R(\alpha-\beta)}} \tag{7}$$

We have also derived that if the output voltage of a gate is approximated with a ramp voltage waveform with α% to β% rise time of $T_{R(\alpha-\beta)}$, the iterative C_{eff} equation for matching any θ% output transition can be written as (derivations are omitted for brevity):

$$C_{eff}^{Ramp}(\theta) = C_n + k_{Ramp}(\theta)C_f \quad \text{where}$$
$$k_{Ramp}(\theta) = \left[1 - \frac{x}{\theta}\left(1 - e^{-\theta/x}\right)\right] \quad \text{and} \quad x = (\beta - \alpha)\frac{R_\pi C_f}{T_{R(\alpha-\beta)}} \tag{8}$$

Now, based on the assumption made above, the iterative equation for actual C_{eff} calculation for any θ% point of the output transition time can be represented as:

$$C_{eff}(\theta) = C_n + [\zeta k_{Exp}(\theta) + (1-\zeta)k_{Ramp}(\theta)]C_f \tag{9}$$

where $0 \le \zeta \le 1$ is the linear combination factor for exponential and ramp waveforms. However, we observed that using ζ=0.5 shows the minimum error between the iterative C_{eff} equation in Eqn. (9) and the actual sign-off C_{eff} value. We will refer to single iteration of Eqn. (9) as the condition check formula. Figure 4 compares the plots of C_{eff}^{Exp}, C_{eff}^{Ramp} and C_{eff} for delay calculation using single iteration of Eqn. (9) over "C_{total}" on the y-axis versus the "actual sign off C_{eff}" for delay calculation over "C_{total}" on the x-axis. To do single iteration of Eqn. (9), we use the output slew of the gate, when the gate sees the total capacitance as the load. Subsequently, we calculated "k_{Ramp}" and "k_{Exp}" and "$(k_{Ramp}+k_{Exp})/2$." As shown in this figure, the single-iteration C_{eff} using Eqn. (9) is reasonably close to the actual sign-off C_{eff} value.

Before starting the discussion for filtering algorithm, we define two threshold parameters; η and γ. η is the threshold parameter which separates the cases that utilize efficient delay calculation from the cases that employ iterative C_{eff} for delay calculation. γ is the threshold parameter that has the same property for output slew calculation. To find out the type of the STA scenario that we encounter in practice, we resort to Eqn. (9). First, we calculate the slew of the gate for the total capacitive load. Next, we find C_{eff} by using a single-iteration of Eqn. (9) and the output slew from the previous step. If C_{eff}/C_{total} is greater than a pre-specified threshold value, η, then we call the gate library and find the gate propagation delay for the obtained C_{eff}. Furthermore, if $C_{eff}/C_{total} \le \gamma$, then the output slew (which was obtained previously for a load capacitance of C_{total}) will be updated. Notice that the recalculation of output slew is unnecessary, when the C_{eff}/C_{total} is large. So under normal threshold settings (say η=0.8), the slew recalculation will never be performed. However, if one sets η to a small value (say 0.3) in order to reduce the CPU time for STA, then it is possible that the slew recalculation will be performed when a larger γ value (say 0.5) is used in order to limit the error. Finally, if $C_{eff}/C_{total} \le \eta$, then we will have to resort to a more accurate way of calculating C_{eff} (use of the Thevenin equivalent circuit for the driver) and obtain the gate propagation delay and output slew values. The summary of the above discussion is captured in the following algorithm.

FTAG: Filtering-based Effective Capacitance Algorithm **(η,γ) {**
Call the "gate library" and find the gate output slew for C_{total}
Find the new C_{eff} using a single iteration of Eqn. (9).
If $(C_{eff}/C_{total} > \eta)$
Call the gate library and find the gate propagation delay for C_{eff}.
If $(C_{eff}/C_{total} \le \gamma)$ // *enter this code for* $\eta \le \gamma$
Update output slew using Eqn. (10).
Else
Perform the sign-off iterative C_{eff} calculation.}

We report the results of the filtering technique for different threshold values for Design#1 and Design#2 in the next section. What remains is how to update the output slew of the gate when $\eta \le C_{eff}/C_{total} \le \gamma$. This is done with the following equation (derivations are omitted for brevity):

$$T_{R(\alpha-\beta)}^{new} = \left\{\zeta h_{Ramp}(\alpha\% - \beta\%) + (1-\zeta)h_{Exp}(\alpha\% - \beta\%)\right\}T_{R(\alpha-\beta)} \quad \text{where}$$
$$h_{Exp}(\alpha\% - \beta\%) = \frac{\frac{C_{eff}^{Exp}(\alpha)}{C_n + C_f}\ln(1-\alpha) - \frac{C_{eff}^{Exp}(\beta)}{C_n + C_f}\ln(1-\beta)}{\ln\left(\frac{1-\alpha}{1-\beta}\right)} \quad \text{and} \tag{10}$$
$$h_{Ramp}(\alpha\% - \beta\%) = \frac{\frac{C_{eff}^{Ramp}(\beta)}{C_n + C_f}\beta - \frac{C_{eff}^{Ramp}(\alpha)}{C_n + C_f}\alpha}{\beta - \alpha}$$

3.3 Experimental Results

To compare the accuracy and performance of the proposed technique, the algorithm is applied on many high-performance industrial designs, including Design#1 and Design#2. Some of the characteristics of these two designs are shown in Table 1. For accuracy purposes, Figure of Merit (FOM) metric has been used. We performed several experiments on Design#1 (c.f. Table 5) and Design#2 (c.f. Table 6). For the gate timing characteristics, we used the sign-off level gate library which contains detailed and accurate k-factor equations for describing the timing behavior of the logic gates. These equations are functions of the input transition time, the output load, V_{dd}, temperature, process parameters, etc. Since, we observed ζ=0.5 introduce minimum error with respect to sign-off C_{eff} calculation, in this section, we set ζ=0.5 in Eqns. (9) and (10).

Experiment 1 is the golden experiment in terms of accuracy since it uses sign off STA for the timing analysis of the design. Experiments 2-7 apply the proposed filtering approach with different filtering threshold values. As experiment 4 indicates, η=0.95 and $\gamma \le \eta$ gives a reasonable accuracy of within 1% error, while it improves the runtime a lot. Experimental results indicate that FTAG improves the runtime of the sign-off C_{eff} by about 50%, while introducing an error of only 1% to the FOM results.

Experiment 9 makes use of C_{total} algorithm. As shown in Table 5, the FOM results for experiment 9 suffer from very large errors. The single-iteration effective capacitance is used in experiment 8. As it is shown, the error in the results is much less compared to the C_{total} algorithm while the runtime is comparable to the runtime of C_{total} algorithm.

As mentioned before, the threshold values in the filtering algorithm are designer, technology, "step in design flow" dependent and a designer can choose these threshold parameter values based on his/her own trade off between desired accuracy and runtime, starting with a small η value but choosing larger ones as he/she proceeds from earlier design stages toward the sign-off stage. One can run a few test cases for each class of designs and in each technology node to obtain the threshold parameter values for the filter. So deriving these parameter values is rather straight-forward, but must be tailored to a particular design and technology.

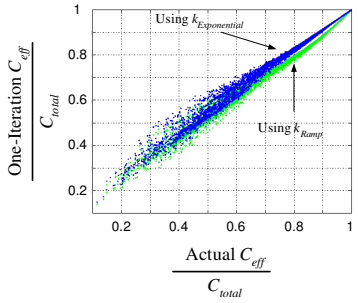

Figure 4: Using k_{Ramp} and k_{Exp} coefficients for C_{eff} Calculation (one iteration)

4. Conclusion

In this paper, first, a threshold-based filtering algorithm for estimating the interconnect delay and slew in high performance interconnects was presented. The proposed algorithm filters a set of nets for timing evaluation using the Elmore-based delay and slew calculation engine. Furthermore, a closed-form expression for calculating the delay and slew was provided for those interconnect lines with m_2/m_1^2 less than a certain critical threshold. Experimental results on large industrial designs show that the filtering technique resulted in a negligible error of 1% error while exhibiting about 65% improvement in the interconnect timing analysis runtime. Next, a threshold based filtering technique was proposed to speed up the gate delay and slew calculation in VDSM technologies. It was observed that the distribution of the "actual C_{eff} over C_{total}" ratio in industrial designs is highly skewed toward one which led us to a novel filtering algorithm. This algorithm utilizes the C_{total} for most circuit scenarios and a C_{eff} algorithm for the remaining rare scenarios. Experimental results on large industrial designs show that the filtering technique resulted in a negligible error of 1% error while exhibiting about 50% improvement in the gate timing analysis runtime.

5. References

[1] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, 19, Jan. 1948, pp. 55-63.

[2] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. on Computer Aided Design,* vol. 9, 1990, pp. 352-366.

[3] C. Alpert, A. Devgan, and C. Kashyap, "A two moment RC delay metric for performance optimization," *Proc. of International Symposium on Physical Design*, 2000.

[4] C. Kashyap, C. J. Alpert, and A. Devgan, "An effective capacitance based delay metric for RC interconnect," Proc. of International Conference on Computer Aided Design, pp. 229 – 234, 2000.

[5] C. Alpert, A. Devgan, and C. Kashyap, "RC delay metrics for performance optimization," IEEE Trans. on Computer Aided Design, vol. 20, pp. 571-582. 2001.

[6] C. L. Ratzlaff, N. Gopal, and L. T. Pillage, "RICE: rapid interconnect circuit evaluator," Proc. of 28th ACM/IEEE Design Automation Conference, pp. 555-560, 1991.

[7] R. Gupta, B. Tutuianu, L. Pileggi, "The Elmore delay as a bound for RC trees with generalized input signals, "IEEE Trans. on Computer Aided Design, vol. 16, 1997, pp. 95-104.

[8] P. R. O'Brien and T. L. Savarino, "Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation," Proc. of International Conference on Computer Aided Design, pp. 512-515, 1989.

[9] J. Qian, S. Pullela, and L. Pillage, "Modeling the "effective capacitance" for the RC interconnect of CMOS gates," IEEE Trans. on Computer Aided Design, vol. 13, 1994, pp. 1526-1535.

[10] H. L. Trentelman, A. A. Stoorvogel, and M. Hautus, *Control Theory of Linear Systems*, Springer, NY, 2001.

[11] J. Rubinstein, P. Penfield, and M. Horowitz, "Signal delay in RC networks", IEEE Trans. on Computer Aided Design, pp. 202-211, 1983.

[12] F. Liu, C. Kashyap, and C. J. Alpert, "A delay metric for RC circuit based on the weibull distribution," Proc. of International Conference on Computer Aided Design, pp. 620-624, 2002.

[13] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: passive reduced-order interconnect macromodeling algorithm," Proceedings of the IEEE/ACM international conference on Computer-aided design San Jose, pp. 58 – 65, 1997.

[14] F. Dartu, N. Menezes, and L. Pillegi, "Performance computation for precharacterized gates with RC loads", IEEE Trans. on Computer Aided Design 15(5):544-553,1996.

[15] R. Puri, D.S. Kung, A.D. Drumm, "Fast and accurate wire delay estimation for physical synthesis of large ASICs," Proc. of GLSVLSI, 2002

[16] S. Abbaspour, M. Pedram, "Calculating the effective capacitance for the RC interconnect in VDSM technologies," Proc. of Asia and South Pacific Design Automation Conference, January 2003.

[17] C. Alpert, C. Kashyap, F. Liu, A. Devgan, "Delay and slew metrics made simple" IEEE Transactions on Computer-Aided Design 2004.

Table 5: Results for Design#1 Using Detailed Library

Exps	η	γ	FOM No.	Slack Worst	FOM Integral	Gate TA (s)	STA (s)
Exp1	1	≤η	1160	-0.196	-63	1146	2010
Exp2	0.99	≤η	1165	-0.197	-64	730	1564
Exp3	0.98	≤η	1174	-0.197	-64	703	1535
Exp4	0.95	≤η	1170	-0.198	-64	586	1418
Exp5	0.90	≤η	1285	-0.211	-73	535	1366
Exp6	0.80	≤η	1534	-0.222	-93	505	1337
Exp7	0.50	≤η	3018	-0.294	-224	467	1327
Exp8	0	1	3699	-0.331	-330	481	1313
Exp9	C_{total}		6146	-0.418	-702	439	1247

Table 6: Results for Design #2 Using Detailed Library

Exps	η	γ	FOM No.	Slack Worst	FOM Integral	Gate TA (s)	STA (s)
Exp1	1	≤η	260947	-1.345	-57138	1945	3235
Exp2	0.99	≤η	261188	-1.345	-57215	1392	2509
Exp3	0.98	≤η	261285	-1.347	-57320	982	2069
Exp4	0.95	≤η	262385	-1.343	-57847	931	2048
Exp5	0.90	≤η	264848	-1.352	-58817	892	2009
Exp6	0.80	≤η	273693	-1.427	-63210	798	1915
Exp7	0.50	≤η	322543	-1.544	-79176	730	1847
Exp8	0	1	324484	-1.616	-81268	452	1569
Exp9	C_{total}		352716	-1.807	-98780	420	1457