Gate Delay Estimation with Library Compatible Current Source Models and Effective Capacitance

Dimitrios Garyfallou, Stavros Simoglou, Nikolaos Sketopoulos, Charalampos Antoniadis,
Christos P. Sotiriou, Nestor Evmorfopoulos, and George Stamoulis
Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece
Email: {digaryfa, ssimoglou, sketopou, haadonia, chsotiriou, nestevmo, georges}@e-ce.uth.gr

***Abstract*—As process geometries shrink below 45nm, accurate and efficient gate-level timing analysis becomes even more challenging. Modern VLSI interconnects are more resistive, signals no longer resemble saturated ramps, and gate input pins exhibit significant Miller effect. Over recent years, the semiconductor industry has adopted Current Source Models (CSMs) for accurate gate modeling. Industrial gate models, however, are precharacterized assuming capacitive loads, which poses significant challenges to the approximation of the highly resistive load interconnect with an effective capacitance (C_{eff}). In fact, most related works are either computationally expensive or unable to approximate the output slew. Furthermore, they require additional precharacterization and ignore the Miller effect. In this paper, we present an iterative methodology for fast and accurate gate delay estimation. The proposed approach accurately computes the driver output waveform, using closed-form formulas to calculate a C_{eff} per waveform segment, while accounting for their interdependence. Thus, it allows for variable analysis resolution exploiting an accuracy/runtime trade-off. In contrast to prior works, our approach is compatible with conventional CSMs and considers the impact of Miller capacitance. We evaluate our method on representative driver-load test circuits consisting of interconnects with arbitrary RC characteristics and ASU ASAP 7nm standard cells. The proposed method achieves 1.3% and 2.5% delay and slew Root Mean Square Percentage Error (RMSPE) against SPICE, respectively. In addition, it provides high efficiency, as it converges in 2.3 iterations on average.**

***Index Terms*—Gate delay estimation, Current Source Models (CSMs), effective capacitance, resistive shielding, Miller effect**

I. Introduction

With continuous technology scaling, accurate and efficient timing analysis plays an ever increasing role in the successful design of complex ICs. Transistor-level electrical simulators [1] may offer golden accuracy results, however, they fail to meet performance and memory requirements for full-scale analysis of modern IC designs. Thus, timing analysis is typically abstracted at the gate-level, where circuit delay is analyzed in stages [2]. Each stage consists of a driver gate, one or multiple receiver gate(s), and an interconnect. The objective of this work is the fast and accurate gate delay and slew estimation, which is essential for timing analysis. Interconnect delay plays a big role in modern nanometer-scale technologies, but it also depends on gate slew estimation. At the same time, the accuracy and performance of gate delay and slew estimation depend not only on the driver and receiver gate models, but also on the interconnect load model.

Gate models are generated by performing transistor-level simulations, per library standard cell, for a set of input signal slews and output loads. This standard cell characterization information is stored in Look-Up Tables (LUTs) of technology libraries, and is used during timing analysis to compute driver gate delay and output slew, given the input slew and output load. For simplicity and speed, a lumped capacitive load is assumed for LUT characterization. Thus, this single capacitance value must be used to represent both the interconnect load, as well as the nonlinear receiver input pin capacitance. However, at 45nm and below, interconnects are becoming increasingly resistive, while nonlinear transistor and Miller capacitances imply that signals no longer resemble smooth, saturated ramps [3]. As a consequence, conventional Voltage Response Models (VRMs), such as the Non Linear Delay Model (NLDM), are inadequate to accurately capture the nonlinear driver waveform, thus leading to significant errors in delay and slew computations. To address this key challenge, Current Source Models (CSMs) [4]–[8] have been proposed, which capture more detail compared to VRMs. Hence, the classical NLDM, used in EDA for decades, has now been replaced by the Composite Current Source (CCS) model [7] and the Effective Current Source Model (ECSM) [8].

The interconnect load model is itself an issue, as modeling the driving point admittance of highly resistive on-chip interconnects is challenging. It is worth noting that due to the high resistance of on-chip interconnects, inductive effects are not significant in timing analysis, and thus only RC interconnect load models are typically considered [2]. A reduced-order π-model [9] of the distributed RC interconnect may provide sufficient accuracy, however, it is not part of the technology library characterization process. On the other hand, modeling the complex RC network using total interconnect capacitance (C_{total}) is overly pessimistic, due to the resistive shielding effect [10]. For accurate gate delay estimation, previous approaches [10]–[18] compute an effective capacitance (C_{eff}) to account for resistive shielding, while maintaining compatibility with precharacterized gate models. However, most of these approaches, whether iterative [10]–[14], [17], [18] or non-iterative [15], [16], are either computationally expensive or inadequate to approximate the output slew. Moreover, they require explicit instantiation of a Thevenin equivalent gate model, as well as precharacterization of information that is not part of standard cell libraries. Few works propose library compatible methods for gate delay estimation using C_{eff} [11], [18], however [11] uses NLDM and only [18] exploits CSMs. A common shortcoming of all the aforementioned methods is that they do not consider the Miller effect, thus ignoring the impact of receiver input pin capacitance on C_{eff}, delay and slew estimation.

In this paper, we focus on improving gate delay estimation by considering the receiver Miller capacitance, as well as the behavior of C_{eff} in multiple regions, while exploiting library compatible CSMs and being very computationally efficient. The contributions of this work can be summarized as follows:

- We propose a methodology to estimate the driver output voltage waveform and C_{eff} in multiple waveform regions. To achieve this, we implemented an iterative algorithm that considers their interdependence, while taking into account the impact of Miller effect. The proposed approach is compatible with CSMs widely adopted by industry [7], [8]. Our approach is computationally efficient, relying on closed-form formulas, while achieving convergence in very few iterations. At the same time, accuracy is not compromised. Experimental results on stages implemented in 7nm FinFET technology show that our method results in 1.8% and 2.5% delay and slew Root Mean Square Percentage Error (RMSPE) over SPICE, respectively, while it achieves convergence in 2.3 iterations on average.
- We investigate the impact of resistive shielding and Miller effect on gate delay estimation, by comparing our method with six methods that adopt different gate and load models. Our results indicate that the proposed method achieves greater accuracy, especially for output slew, compared to single C_{eff} methods [19], while C_{total} is extremely inaccurate for highly resistive loads. For stages with low impedance interconnects and significant receiver Miller capacitance, our method further improves delay and slew estimation.

The rest of the paper is organized as follows. We present other research approaches for gate delay estimation in Section II. In Section III, we describe the fundamentals of library compatible CSMs, C_{eff} computation, as well as the challenges of exploiting them. Section IV presents our methodology for accurate and efficient gate delay estimation using CSMs and C_{eff}. In Section V, we evaluate the accuracy and performance of our approach. Finally, conclusions are drawn in Section VI.

II. Related Work

During the past two decades, various works have focused on improving gate and load models to enable accurate driver output waveform estimation in the presence of RC interconnects.

The simplest approximation of the driving point admittance of an RC interconnect is C_{total}, which is computed by summing all interconnect capacitance values. However, this results in pessimistic gate delay estimation, as it totally ignores interconnect resistance which shields a part of total capacitance. A more accurate approximation is a reduced-order model. Authors in [9] propose a π model, which may be computed by matching the first three moments of the driving point admittance using a moment-matching technique [20]. It follows that for accurate gate delay estimation, a four-dimensional LUT indexed by input slew and the π model parameters ($C_{near}; R; C_{far}$) may be used. However, this is costly in terms of storage and computational requirements. Additionally, it is incompatible with gate models precharacterized in standard cell libraries assuming lumped capacitive loads (e.g. NLDM, CCS, ECSM) [7], [8]. To address these limitations, the concept of C_{eff} is introduced [10].

Early research on gate delay estimation using C_{eff} focuses on computing a single capacitance value to approximate the output waveform [10], [14], [15]. Authors in [10] use a two-piece output waveform and propose a C_{eff} calculation method for single stage gates. C_{eff} is calculated by equating the average current at the gate output, (i) when using the driving point admittance as a load, and (ii) when using the estimated C_{eff} as a load. Average output currents are equated until the output voltage reaches the 50% threshold. However, this approach involves an expensive iterative algorithm which requires 5 to 10 iterations to converge, and uses empirical equations which assume fast input transitions. Aiming at modeling complex gates, the approach in [14] introduces an empirical time-varying Thevenin equivalent gate model, independent of the input signal thresholds. Similarly to [10], they equate the average currents for a specific output voltage waveform region, from 20% to 50%, denoted as the "active region". A disadvantage of this method is that it is computationally expensive, as it uses Newton-Raphson iteration to calculate C_{eff} and the Thevenin voltage source parameters. To sidestep the performance limitations, authors in [15] propose a non-iterative method for C_{eff} estimation. Regarding the gate model, the main difference is that they use a Thevenin model composed of a fixed linear ramp and a fixed resistance, which yields a delay error up to 15%. The main drawback of the above methods [10], [14], [15] is that they are inadequate to accurately match the output waveform as they focus on gate delay estimation, thus leading to slew errors up to 50% [12].

To overcome this limitation, many works propose the use of a single or two C_{eff} values for matching the output slew directly [12], [13], [17], or matching specific output voltage thresholds (e.g. 10% and 50%) separately [16]. In [12], authors compute a single C_{eff} by tuning an osculating Thevenin model, until the delay when the model is loaded by the original RC interconnect and the delay when the model is loaded by C_{eff} are approximately equal. Authors in [13] use a Thevenin model and present an iterative method to estimate a single C_{eff}. In contrast to previous approaches, this method does not equate the average currents or gate delays, but minimizes the error between the output voltage waveforms from $0.2V_{dd}$ to $0.8V_{dd}$. The approach in [17] adopts a multi-ramp driver model and uses two C_{eff} values to model different slew rates of the nonlinear output waveform in the presence of process variations. However, this method needs to perform complicated statistical precharacterization. In a different approach, authors in [16] use two C_{eff} values to match the lower (e.g. 10%) and upper (e.g. 50%) output voltage thresholds instead of matching slew directly. Although this method is non-iterative and may be reasonably fast, it induces inaccuracy in slew estimation since it is based on empirical gate modeling and assumes a fixed switching resistance for complex gates.

Note that most of the aforementioned schemes [10], [12]–

[17] apply moment-matching techniques and approximate the interconnect load admittance with poles and residues, either to reduce the load to a π-model (e.g. using [9]) or to compute C_{eff} and the Thevenin model parameters. However, our proposed methodology does not include expensive techniques and can be extended to handle distributed interconnects without requiring computation of moments and reduction to π-model.

Common to all these approaches [10], [12]–[17] is that they require explicit instantiation of a Thevenin equivalent model and precharacterization of non-standard information such as the Thevenin model parameters and the delays to arbitrary output voltage thresholds. Besides being inefficient and unable to accurately capture the output voltage waveform, these approaches are also incompatible with standard cell libraries.

A library compatible load model is presented in [11]. However, this method cannot approximate the output slew, as it uses a single C_{eff} and assumes that driver output voltage is a ramp with fixed slew estimated using NLDM. Instead, our method uses library compatible CSMs [7], [8], to accurately estimate a piecewise linear (PWL) driver voltage waveform, by computing a different C_{eff} value for each linear segment. The approach in [18] is the one closest to ours, in the sense that it uses Multiple Voltage Threshold Models (MVTM), a variant of CSM, and multiple C_{eff}. However, the computation of the driver output waveform involves moment-matching techniques and is not straightforward like our proposed approach.

Additionally, none of the previous works [10]–[18] accounts for the Miller effect, ignoring the impact of receiver input pin capacitance on C_{eff} and driver output waveform. On the contrary, our proposed method takes this effect into account, thus leading to better delay and output slew accuracy results.

III. Background and Motivation

In this section, we present library compatible CSMs, and provide an example of the CCS model. Moreover, we describe the traditional way to compute C_{eff} and the challenges arising when exploiting CSMs and C_{eff} for gate delay calculation.

A. Gate modeling and library compatible CSMs

In a gate-level design, almost every gate acts as a driver in one stage and a receiver in an other one. As a result, both driver and receiver models are essential to accurately capture the electrical behavior of a gate. More specifically, the driver model should capture the timing characteristics of a gate (e.g. gate delay and output slew), while the receiver model should capture the capacitive load that is presented to the driver gate.

Traditional gate modeling includes a VRM (which defines characteristics of the driver output voltage response) as a driver model, and a single capacitance value (or two values, for rise and fall) as a receiver model [21]. In nanometer technologies, however, this modeling is highly inaccurate, as signal waveforms and input capacitances are nonlinear. On the driver side, the main issue is that interconnect resistance can become several kOhms, resulting in much higher interconnect impedance compared to the resistance of the driver gate. In such cases, VRMs, such as Thevenin models and NLDM, tend to produce an output waveform which is the same as the input waveform, dwarfing the effect of the output load. On the receiver side, input pin capacitance actually depends both on the input signal slew and the receiver output load, while it also varies considerably during a transition, due to the Miller effect. This effect describes the increase in input pin capacitance caused by the presence of input-to-output coupling capacitance (known as Miller capacitance). As gate-to-drain capacitance increases, Miller effect becomes even more pronounced [3]. Thus, since the load seen by the driver depends on the receiver input pin capacitance, it is evident that constant capacitance models are insufficient for accurate gate delay estimation.

In contrast to the traditional gate models, CSMs use a time-varying voltage-controlled current source as a driver model, and a complex voltage-controlled capacitor as a receiver model. As a consequence, they are able to approximate the nonlinear waveforms and yield SPICE-accurate results within reasonable time. Although several CSMs have been proposed in the literature [4]–[6], our methodology focuses on CSMs adopted by industry, such as ECSM and CCS, which are precharacterized in standard cell libraries. In this paper, we refer to such models as *library compatible CSMs*. These CSMs store the driver output voltage or current waveforms into two-dimensional LUTs, indexed by input signal slew (tr^{in}) and output load (c^{out}), for each timing arc (i.e. input-to-output connection). More specifically, ECSM represents the voltage waveform in the form of $V(t) = F_v(tr^{in}; c^{out})$, while CCS represents the current waveform as $I(t) = F_i(tr^{in}; c^{out})$. ECSM and CCS precharacterized waveforms are equivalent, as the voltage waveform can be derived by integrating the corresponding current waveform. Further on, both models are able to account for the Miller effect by providing multiple capacitance values in similar LUTs. To better explain library compatible CSMs, we provide a brief demonstration of the CCS model and the differences compared to NLDM.

Fig. 1: NLDM vs CCS timing model.

CCS driver model captures the nonlinear current waveforms in output_current_rise/fall LUTs, as demonstrated in Fig. 1, which are used during timing analysis to estimate driver delay and output slew. Also, the time instant when the corresponding driver input waveform crosses the de-

(a) Linear ramp input voltage (b) Time domain (c) Frequency domain (d) Effective capacitance

Fig. 2: Effective capacitance calculation for a π-model with linear ramp input voltage waveform.

lay threshold (usually $0.5V_{dd}$), which is necessary to calculate gate delay, is stored as reference_time. On the contrary, NLDM captures fixed delay and output slew values (stored in cell_rise, rise_transition LUTs), and thus provides inferior accuracy in delay estimation compared to CCS.

CCS receiver model typically uses two different input pin capacitance (C_p) values, $C1$ and $C2$, to model the nonlinear receiver input transistor capacitance and the Miller effect. $C1$ is considered up to the delay threshold of the driver output waveform, while $C2$ is considered past this point. More than two regions and corresponding capacitance values can be used to improve the accuracy. As shown in Fig. 1, CCS receiver capacitance values are stored in receiver_capacitance_1/2_rise/fall LUTs. Contrary to CCS, NLDM provides only a single capacitance value (stored in rise_capacitance attribute) and ignores the Miller effect.

B. Modeling the RC interconnect load with a single C_{eff}

It has been shown that a distributed RC interconnect may be replaced by an equivalent π-model, without a significant loss of accuracy [9]. Traditionally, interconnect loads had been highly capacitive and less resistive. Hence, resistance R had negligible impact on delay calculation, and the use of C_{total} (i.e. the sum of near capacitance C_{near} and far capacitance C_{far}) was sufficient to achieve accurate results. With technology scaling, however, interconnect loads are becoming more and more resistive, which complicates the approximation of the RC interconnect with a single capacitance. Considering the π-model of Fig. 2b, as R tends to zero, the π-model capacitors are effectively connected in parallel, and can be summed together without loss of accuracy. However, when R possesses a significant value, it acts as an open-circuit, shielding C_{far}, and thus only C_{near} is "seen" by the driver. Therefore, we cannot neglect the shielding effect when substituting the π-model with an equivalent C_{eff} (shown in Fig. 2d).

Let us now describe how C_{eff} is calculated. The following are modifications of the approach proposed in [19]. Consider the example of Fig. 2b, where the π-model is driven by a voltage source $V_i(t)$ which represents the driver output waveform. Assuming that $V_i(t)$ is a linear ramp with rise transition time t_r, as shown in Fig. 2a, the waveform equation is:

$$V_i(t) = \begin{cases} \frac{V_{dd}}{t_r} t; & t < t_r \\ V_{dd}; & t \geq t_r \end{cases}$$

The circuit representation in the frequency domain is depicted in Fig. 2c. Using Kirchhoff's current law and Ohm's law, the supplied current can be expressed as a function of the input voltage and the π-model RC parameters, as follows:

$$\begin{aligned} I(s) &= I_1(s) + I_2(s) = \frac{V_i(s)}{1/(sC_{near})} + \frac{V_i(s)}{R + 1/(sC_{far})} \\ &= V_i(s)\left(sC_{near} + \frac{sC_{far}}{1 + sRC_{far}}\right) \end{aligned} \quad (1)$$

Additionally, if we transform $V_i(t)$ into the frequency domain, we obtain:

$$V_i(s) = \frac{V_{dd}}{s^2 t_r}\left(1 - e^{-st_r}\right) \quad (2)$$

Now, by substituting Eq. (2) in Eq. (1), when $t < t_r$, we get:

$$\begin{aligned} I(s) &= \frac{V_{dd}}{t_r}\left(1 - e^{-st_r}\right)\left(\frac{C_{near}}{s} + \frac{C_{far}}{s(1 + sRC_{far})}\right) \\ &= \frac{V_{dd}}{t_r}\left(1 - e^{-st_r}\right)\left(\frac{C_{near}}{s} + \frac{C_{far}(1 + sRC_{far}) - sRC_{far}^2}{s(1 + sRC_{far})}\right) \\ &= \frac{V_{dd}}{t_r}\left(1 - e^{-st_r}\right)\left(\frac{C_{near}}{s} + \frac{C_{far}}{s} - \frac{C_{far}}{s + 1/(RC_{far})}\right) \end{aligned}$$

Finally, after transforming the current equation back to the time domain, the resulting equation is given by:

$$I(t) = \underbrace{\frac{V_{dd}}{t_r} C_{near}}_{I_1(t)} + \underbrace{\frac{V_{dd}}{t_r} C_{far}\left(1 - e^{-\frac{t}{RC_{far}}}\right)}_{I_2(t)} \quad (3)$$

At this point, C_{eff} can be defined as a capacitance seen by the driver voltage source $V_i(t)$, which requires the same charge transfer as that required by the π-model. Typically, C_{eff} is calculated up to a specific voltage threshold $V_\ell = \ell V_{dd}$, with factor ℓ representing a percentage of V_{dd}. For this reason, we denote this effective capacitance by C_ℓ. Note that V_ℓ corresponds to a time instant T_ℓ, where $V_i(T_\ell) = V_\ell$, which also represents the driver output slew from $0V_{dd}$ to ℓV_{dd}, assuming that the output transition begins at $t = 0$ (i.e. $T_{0V_{dd}} = 0$).

The equation for the charging of the π-model, up to T_ℓ, can be derived, using Eq. (3), as follows:

$$Q_\ell = \int_0^{T_\ell} I(t)\,dt = \int_0^{T_\ell} \frac{V_\ell}{T_\ell}\left(C_{near} + C_{far}\left(1 - e^{-\frac{t}{RC_{far}}}\right)\right) dt$$

Since the charge of C_ℓ is given by $Q_\ell = C_\ell V_\ell$, equating the two charge transfer equations yields:

$$V_\ell C_{near} + V_\ell C_{far}\left(1 - \frac{RC_{far}}{T_\ell}\left(1 - e^{-\frac{T_\ell}{RC_{far}}}\right)\right) = C_\ell V_\ell$$

By solving for C_ℓ, we obtain:

$$C_\ell = C_{near} + C_{far}\left(1 - \frac{RC_{far}}{T_\ell}\left(1 - e^{-\frac{T_\ell}{RC_{far}}}\right)\right) \quad (4)$$

or in a more compact form $C_{`} = C_{near} + K_{`} C_{far}$, where

$$K_{`} = 1 \quad \frac{RC_{far}}{T_{`}} \quad 1 \quad e^{\frac{T_{`}}{RC_{far}}}$$

As shown in the above formula, $K_{`}$ factor, which is the capacitance shielding factor, depends on the time constant $RC_{far}$ and the input slew $T_{`}$ of the -model interconnect. It is evident, from Eq. (4), that when the -model interconnect is highly resistive, $K_{`}$ tends to zero. On the contrary, when $R$ is close to zero, $K_{`}$ approaches 1. This con rms the intuition that $C_{`}$ is equivalent to the parallel connection of C_{near} and C_{far}, when R is negligible, while it effectively accounts for a virtually disconnected C_{far}, when R is very large.

The main advantage of the described method is that it provides a closed-form formula for C_{eff} estimation, rather than applying expensive moment-matching techniques [20]. It is important to note that although $C_{`}$ of Eq. (4) does not include the receiver input pin capacitance C_p, it may be easily extended to account for it, by adding a constant C_p (obtained by NLDM) together with C_{far}, as they are in parallel. However, even including a constant C_p, it still ignores the Miller effect.

C. Challenges in gate delay estimation using CSMs and C_{eff}

Two main challenges arise when attempting to estimate gate delay and output slew using CSMs and C_{eff}. First, there is an interdependence between driver output slew and receiver input pin capacitance. As described in Section III-A, driver output current and voltage waveforms depend on the load seen by the driver, which in turn depends on input pin capacitance of receiver gate(s). On the other hand, receiver input capacitance is a function of receiver input slew, which depends both on interconnect RC parasitics and on driver output slew. Second, the modeling of C_{eff} is essential for accurate delay calculations. However, it is infeasible to obtain a single C_{eff} value, that is suitable for both delay and slew calculations, as it cannot exactly match the actual load in terms of driver output current at any time instant. In addition, as shown in Eq. (4), $C_{`}$ is a function of driver output slew, and thus there is an interdependence between them as well.

The above challenges dictate the use of an iterative approach, which can handle both interdependencies simultaneously. To the best of our knowledge, such an approach has not yet been proposed in the literature.

IV. Proposed Approach

In this section, we present our approach for accurate and ef cient gate delay estimation. To this end, we propose an iterative algorithm that effectively addresses the aforementioned challenges, exploiting library compatible CSMs and the dynamic behavior of C_{eff}, while considering the Miller effect.

A. Computation of a single C_{eff} considering the Miller Effect

As discussed in Section III-A, the Miller effect may be very strong, especially at technology nodes below 45nm [3]. In contrast to the resistive shielding effect, the Miller effect impact on gate delay and slew is higher in stages with small impedance interconnects, where receiver input pin capacitance dominates C_{eff}. As a result, for accurate C_{eff} estimation, we must consider the entire driver RC load (-model and receiver input pin capacitance C_p), while taking into account the Miller effect. In our proposed approach, we exploit the dynamic behavior of C_p, instead of using a constant value, using library compatible CSMs which model the Miller effect.

Fig. 3: Output voltage waveform and slew calculation for a -model with linear ramp input voltage waveform.

To compute the receiver pin capacitance up to a speci c voltage threshold $V_{`}$, which is denoted as $C_{p(`)}$, the slew at the output of the interconnect, $T^0_{`}$, must be calculated. The most accurate estimation may be obtained by performing interconnect transient analysis using the driver output waveform as excitation. However, this is prohibitive even for small circuits. A closed-form formula for the interconnect output slew can be derived as follows. Fig. 3 depicts an approximation of the output voltage waveform for a -model, given a ramp input voltage waveform $V_i(t) = \frac{V_{`}}{T_{`}} t$, up to $V_{`}$. In this case, the output voltage $V_o(t)$ can be calculated by:

$$V_o(t) = V_i(t) \quad I_2(t)R$$

$$= \frac{V_{`}}{T_{`}} t \quad \frac{V_{`}}{T_{`}} (C_{far} + C_{p(`)}) \quad 1 \quad e^{\frac{t}{R(C_{far} + C_{p(`)})}} \quad R$$

In the above equation, $I_2(t)$, which is given in Eq. (3), has been updated to include $C_{p(`)}$, which is parallel to C_{far}.

Therefore, at $t = T_{`}$, when the input voltage waveform crosses $V_{`}$, the output voltage is given by:

$$V_k = \frac{V_{`}}{T_{`}} \quad T_{`} \quad R(C_{far} + C_{p(`)}) \quad 1 \quad e^{\frac{T_{`}}{R(C_{far} + C_{p(`)})}} \quad \# \tag{5}$$

Also, from Fig. 3, it can be seen that:

$$\tan(\;) = \frac{V_k}{T_{`}} = \frac{V_{`}}{T^0_{`}} =) \quad T^0_{`} = \frac{V_{`} T_{`}}{V_k} \tag{6}$$

Substituting Eq. (5) in Eq. (6) yields:

$$T^0_{`} = \frac{T_{`}}{1 \quad \frac{R(C_{far} + C_{p(`)})}{T_{`}} \quad 1 \quad e^{\frac{T_{`}}{R(C_{far} + C_{p(`)})}}} \tag{7}$$

Now, given $T^0_{`}$ and the receiver output load, $C_{p(`)}$ is computed by accessing the CSM receiver capacitance LUTs (e.g. CCS receiver_capacitance_1/2_rise/fall to derive $C1$, $C2$ values). In order to account for $C_{p(`)}$, the effective capacitance formula of Eq. (4) is nally updated to:

$$C_{`} = C_{near} + ( C_{far} + C_{p(`)}) \quad 1 \quad \frac{R(C_{far} + C_{p(`)})}{T_{`}} \quad 1 \quad e^{\frac{T_{`}}{R(C_{far} + C_{p(`)})}} \quad \# \tag{8}$$

Fig. 4: Comparison between the linear ramp voltage waveform computed using a single C_{eff} and the actual SPICE waveform.

Fig. 5: Comparison between the PWL voltage waveform computed using multiple C_{eff} and the actual SPICE waveform.

or in a compact form $C_\ell = C_{near} + K_\ell (C_{far} + C_{p(\ell)})$, where

$$K_\ell = 1 - \frac{R(C_{far} + C_{p(\ell)})}{T_\ell}\left(1 - e^{-\frac{T_\ell}{R(C_{far} + C_{p(\ell)})}}\right)$$

Even though C_ℓ of Eq. (8) improves accuracy by considering the Miller effect, it is still insufficient to accurately estimate gate delay and output slew, as it is a single C_{eff} and assumes that driver output voltage is a linear ramp. As can be seen in Fig. 4, this approach totally ignores the nonlinear shape of the actual driver waveform obtained using SPICE. The resulting estimation error is much higher for driver output slew, as it is measured between the time instants when the output voltage waveform crosses the lower (V_{low}) and upper (V_{high}) thresholds, compared to driver delay which is measured between the time instants when the input and output voltage waveforms cross the delay threshold (V_{delay}). However, accurate driver slew computation is essential for interconnect delay and slew estimation, which impacts delay and slew estimation for the receiver gate(s).

B. Computation of multiple C_{eff}

To accurately approximate the nonlinear driver waveform, we compute a PWL ramp, exploiting library compatible CSMs. Fig. 5 demonstrates that this CSM waveform is able to closely match the actual waveform, leading to great accuracy in delay and slew estimation. To compute this waveform, we use multiple C_{eff} values, i.e. one C_{eff} value per each linear segment. The effective capacitance $C_\ell^{\ell+1}$ for a specific voltage region $[V_\ell; V_{\ell+1}]$ of the driver output waveform can be derived by using the equivalent charge $Q_\ell^{\ell+1}$ equation, given by:

$$Q_\ell^{\ell+1} = \int_{T_\ell}^{T_{\ell+1}} I(t)\,dt = Q_{\ell+1} - Q_\ell$$

Since $Q = CV$ and $V_\ell^{\ell+1} = V_{\ell+1} - V_\ell$, we have:

$$C_\ell^{\ell+1} = \frac{Q_\ell^{\ell+1}}{V_\ell^{\ell+1}} = \frac{Q_{\ell+1} - Q_\ell}{V_{\ell+1} - V_\ell} = \frac{C_{\ell+1}V_{\ell+1} - C_\ell V_\ell}{V_{\ell+1} - V_\ell} \qquad (9)$$

Eq. (9) describes C_{eff} in a specific region as a function of the C_{eff} values corresponding to the lower and upper thresholds of the region. Note that $C_\ell; C_{\ell+1}$ are computed using Eq. (8), to account for the Miller effect.

The detailed CSM waveform may also be used for a more accurate interconnect analysis, in order to compute a PWL receiver input waveform, using Eq. (7). This improves the estimation accuracy for receiver delay, output slew, and input pin capacitance. Furthermore, driver output slew $T_\ell^{\ell+1}$ and interconnect output slew $T_\ell^{(\ell+1)'}$, for a specific region $[V_\ell; V_{\ell+1}]$, can be derived as:

$$T_\ell^{\ell+1} = T_{\ell+1} - T_\ell \quad \text{and} \quad T_\ell^{(\ell+1)'} = T'_{\ell+1} - T'_\ell$$

As can be seen in Fig. 5, three C_{eff} values computed in three voltage regions, i.e. $[0V_{dd}; V_{low}]$, $[V_{low}; V_{delay}]$ and $[V_{delay}; V_{high}]$, are typically sufficient to accurately compute driver delay and output slew. Computing a more detailed CSM waveform, using more C_{eff} values, may lead to improved accuracy results, inducing a small performance overhead.

C. Algorithm for gate delay and output slew estimation, using CSMs and multiple C_{eff}

In this subsection, we present the iterative algorithm that implements our proposed approach. The purpose of this algorithm is to estimate both gate delay and output slew for a specific driver timing arc of a given <driver, π-model, receiver> stage. An example of such stage is shown in Fig. 1.

Given the driver input slew (tr_d^{in}) for the respective timing arc, a receiver output capacitive load (C_L^{out}) (usually set to C_{total}), the π-model parameters ($\{C_{near}; R; C_{far}\}$), and the non-controlling values for the driver and receiver side inputs, our method iteratively computes driver delay (d) and output slew (T_{low}^{high}), until output slew converges. This is done by computing the CSM driver output voltage waveform, and C_{eff}, in n regions provided as a set $V = \{V_a; \dots; V_b\}$ of $n+1$ subsequent voltage thresholds. The C_{eff} values in these regions are stored into a set $C = \{C_a^{a+1}; \dots; C_{b-1}^{b}\}$, while the time instants when driver output waveform crosses the specified voltage thresholds are stored into a set $T = \{T_a; \dots; T_b\}$.

The details of the proposed algorithm are described in Algorithm 1. First, $C_\ell^{\ell+1}$ for each specified region $[V_\ell; V_{\ell+1}]$ is initialized to C_{total}, using the NLDM receiver input pin capacitance (steps 2-5). Second, the CSM output voltage waveform is computed, using C_{total} (step 6), and is used to obtain the initial estimation of d and T_{low}^{high} (step 7). In the

Algorithm 1: Compute driver delay and output slew for a $\langle$driver, π-model, receiver$\rangle$ stage

Input: $V = \{V_a, \ldots, V_b\}$, tr_d^{in}, c_r^{out}
Output: d, T_{low}^{high}
1 Function compute_driver_CSM_timing(V, tr_d^{in}, c_r^{out}):
2 foreach voltage region $[V_\ell, V_{\ell+1}]$ in V do
3 $C_\ell^{\ell+1} = C_{near} + C_{far} + C_{nldm}$
4 update C with $C_\ell^{\ell+1}$
5 end
6 $\{T, t_{ref}\}$ = compute_CSM_waveform(V, C, tr_d^{in})
7 $\{d, T_{low}^{high}\}$ = compute_CSM_delay_slew(T, V, t_{ref})
8 while T_{low}^{high} not_converged do
9 foreach voltage region $[V_\ell, V_{\ell+1}]$ in V do
10 compute $T_\ell^0, T_{\ell+1}^0$ using Eq. (7)
11 compute $C_{p(\ell)}, C_{p(\ell+1)}$ by accessing CSM LUTs using (T_ℓ^0, c_r^{out}) and $(T_{\ell+1}^0, c_r^{out})$, respectively
12 compute $C_\ell^{\ell+1}$ using Eq. (9)
13 update C with $C_\ell^{\ell+1}$
14 end
15 $\{T, t_{ref}\}$ = compute_CSM_waveform(V, C, tr_d^{in})
16 $\{d, T_{low}^{high}\}$ = compute_CSM_delay_slew(T, V, t_{ref})
17 end
18 End Function

Algorithm 2: Compute CSM driver output waveform in specified voltage regions, using multiple C_{eff}

Input: $V = \{V_a, \ldots, V_b\}$, $C = \{C_a^{a+1}, \ldots, C_{b-1}^{b}\}$, tr_d^{in}
Output: $T = \{T_a, \ldots, T_b\}$, t_{ref}
1 Function compute_CSM_waveform(V, C, tr_d^{in}):
2 foreach voltage region $[V_\ell, V_{\ell+1}]$ in V do
3 $c_d^{out} = C_\ell^{\ell+1}$
4 compute driver output waveform by accessing CSM LUTs using (tr_d^{in}, c_d^{out})
5 if (CSM used is CCS) then
6 transform waveform from current to voltage
7 compute $T_\ell, T_{\ell+1}$ using voltage waveform and $V_\ell, V_{\ell+1}$
8 update T with $T_\ell, T_{\ell+1}$
9 end
10 compute t_{ref} by accessing CSM LUTs using tr_d^{in}
11 End Function

Algorithm 3: Compute driver delay and output slew, using CSM driver output waveform

Input: $T = \{T_a, \ldots, T_b\}$, $V = \{V_a, \ldots, V_b\}$, t_{ref}
Output: d, T_{low}^{high}
1 Function compute_CSM_delay_slew(T, V, t_{ref}):
2 compute $T_{low}, T_{delay}, T_{high}$ using CSM waveform $\{T, V\}$
3 $d = T_{delay} - t_{ref}$
4 $T_{low}^{high} = T_{high} - T_{low}$
5 End Function

main iterative refinement loop (steps 8-17), for each region $[V_\ell, V_{\ell+1}]$, the algorithm computes the receiver input slew values $T_\ell^0, T_{\ell+1}^0$ (step 10), in order to update the corresponding $C_{p(\ell)}$ values (step 11), and computes the new $C_\ell^{\ell+1}$ value to update C (steps 12-13). Then, driver output waveform, delay and output slew are re-calculated (steps 15-16). This iterative refinement is performed until T_{low}^{high} converges within a specified tolerance (e.g. $|T_{low}^{high}(new) - T_{low}^{high}(old)| <$ tolerance). Thus, the overall time complexity of our method is $O(|V|)$, where $|V|$ is the number of voltage thresholds. The CSM operations (LUT accesses, current-to-voltage transformations, interpolations, and driver waveform computation), which differ across various CSMs and are related to their characteristics, are of constant time complexity, since they do not depend on $|V|$.

To compute the CSM driver output waveform, we developed Algorithm 2. Given a set of voltage thresholds, C_{eff} per region and the driver input slew tr_d^{in}, this algorithm computes the driver output waveform (i.e. voltage for ECSM or current for CCS) for each region $[V_\ell, V_{\ell+1}]$, by setting the driver output load c_d^{out} to the corresponding $C_\ell^{\ell+1}$ (step 3), and accessing CSM LUTs (e.g. CCS output_current_rise/fall) using (tr_d^{in}, c_d^{out}) (step 4). In case the CCS model is used, the corresponding voltage waveform is obtained by integrating the current waveform (e.g. using the Trapezoidal rule) (steps 5-6). Then, $T_\ell, T_{\ell+1}$ are computed and T is updated (steps 7-8). Finally, after computing the CSM waveform, described by $\{T, V\}$, the algorithm computes the reference_time t_{ref}, by accessing CSM LUTs using tr_d^{in} (step 10).

Driver delay and output slew are computed using the operations described in Algorithm 3. Given the CSM output voltage waveform $f(T, V)$, this algorithm computes the time instants $T_{low}, T_{delay}, T_{high}$, when the output waveform crosses $V_{low}, V_{delay}, V_{high}$ (step 2). Then, using these values and the input reference time t_{ref}, it computes d and T_{low}^{high} (steps 3-4).

At this point, we can elaborate on a key aspect regarding the efficient implementation of the proposed approach. The most computationally expensive step in our methodology is the CSM driver output waveform computation, described in Algorithm 2. This is because it involves interpolation between the closest precharacterized voltage waveforms, to compute the non-precharacterized waveform for arbitrary slew, capacitance values. Additionally, in the case of CCS, the closest current waveforms have to be transformed to voltage waveforms prior to interpolation, which may also be costly.

To improve performance, the CCS current-to-voltage transformation and the computation of $T_\ell, T_{\ell+1}$ values for each precharacterized CSM waveform may be performed only once. In order to reduce memory requirements, we may compute and store only the required set of time instants T for the specified set of voltage thresholds V. This may be performed either off-line before the delay calculation for all precharacterized waveforms, or only the first time we process each waveform. In case this is performed off-line for all standard cells, multiple threads may be used in parallel to speedup the procedure. Thus, to compute $T_\ell, T_{\ell+1}$ in Algorithm 2, we may interpolate between these time instants, instead of the entire waveforms.

The proposed approach may be extended to handle stages with distributed RC interconnects and multiple receiver gates, by exploiting the forward-backward traversal algorithm presented in [19], in order to update C (Algorithm 1, steps 10-15). This algorithm computes the delay of an RC interconnect, which is handled as connected π-models, assuming single slew and C_{eff} on each node. However, it can be modified to compute slew and C_{eff} per specified region. Specifically, during the forward traversal, the slew for each region may be propagated, using breadth-first search (BFS), from the driver output pin (source) towards the receiver input pins (sinks).

For each -model output node, $T^{\ell+1}$ may be computed with Eq. (7), using the $C^{\ell+1}$ of this node as C_{far} (considering also $C_{p(\ell)}$ for the -models connected to sinks). Then, during the backward traversal, $C^{\ell+1}$ may be recalculated using Eq. (9), and propagated backwards from sinks to source, to update C.

Moreover, it is worth mentioning that our algorithm may be integrated into the delay calculator of any gate-level Static Timing Analysis (STA) [2] or Dynamic Timing Analysis (DTA) [22] method based on library compatible CSMs.

V. EXPERIMENTAL EVALUATION

To evaluate our method, we implemented Algorithm 1 using CCS as CSM, and three regions for C_{eff} and driver waveform computation. We also implemented six alternative methods (M1-M6) that differ in three key features, i.e. (i) the driver model, where CCS or NLDM is utilized, (ii) the load model, where C_{total} or C_{eff} is used, and (iii) the receiver model, where CCS is used to consider the Miller effect, or NLDM is used otherwise. Table I summarizes the key differences of the investigated methods. Note that for the methods which use three regions (3 C_{eff}), without loss of generality, we set $V = \{0V_{dd}, 0.1V_{dd}, 0.5V_{dd}, 0.9V_{dd}\}$. We selected these regions because the standard cell library used for our experiments is precharacterized using $V_{low} = 0.1V_{dd}$, $V_{delay} = 0.5V_{dd}$, $V_{high} = 0.9V_{dd}$. Similarly, for the methods which use 1 C_{eff}, the single region $V = \{0V_{dd}, 0.5V_{dd}\}$ is used. Moreover, the convergence tolerance for T_{low}^{high} had been set to 10^{-3}.

In more detail, M1 uses NLDM and computes C_{total}, while it ignores the Miller effect. Among all the examined methods, M1 is the only non-iterative method. All the other methods, i.e. M2-M6, are implemented with modifications of the iterative method described in Algorithm 1. M2 assumes a single C_{eff} (Algorithm 1, steps 9-14) and implements a function similar to compute_CSM_waveform(), that computes a ramp waveform with fixed slew, by using the NLDM LUTs. However, it cannot model the Miller effect. M3 considers the Miller effect, but computes C_{total} using two input pin capacitance values, C1 and C2, for the receiver model (Algorithm 1, steps 12-13). The rest of the methods (M4, M5, M6 and ours) compute C_{eff}, however, differ in the number of driver voltage waveform regions selected to be matched, and Miller effect consideration. More specifically, M4 (i.e. the method of [19]) and M5 (i.e. a variant of [18]) compute C_p using NLDM, (Algorithm 1, step 11), and use 1 C_{eff} and 3 C_{eff}, respectively. Finally, M6 is identical to our method, but applies interconnect transient simulation to estimate receiver input slew more accurately (Algorithm 1, step 10).

To evaluate the accuracy of the above methods, we measured their Root Mean Square Percentage Error (RMSPE) against Synopsys® HSPICE [1]. For each timing metric x (delay or output slew), the RMSPE of each method, across all measurements, is calculated by:

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{x}_i - x_i}{x_i} \cdot 100\%\right)^2}$$

where $\hat{x}_i$ is the measurement of the examined method for stage i, x_i is the corresponding HSPICE measurement, and n is the total number of measurements for all stages considering both rise and fall transitions (i.e. $n = 2 \cdot \#\,stages$).

TABLE I: Characteristics of the investigated methods and Root Mean Square Percentage Errors (RMSPE) against HSPICE

Method	Driver Model	Load Model	Receiver Model	RMSPE Delay	RMSPE Slew
M1	NLDM	C_{total}	NLDM	19.19 %	21.60 %
M2	NLDM	1 C_{eff}	NLDM	5.86 %	8.92 %
M3	CCS	C_{total}	CCS	19.08 %	21.51 %
M4	CCS	1 C_{eff}	NLDM	1.65 %	10.81 %
M5	CCS	3 C_{eff}	NLDM	1.99 %	2.68 %
M6*	CCS	3 C_{eff}	CCS	1.01 %	2.10 %
Ours	CCS	3 C_{eff}	CCS	1.32 %	2.48 %

* Computes receiver input slew using transient simulation

To compare the investigated methods for both accuracy and runtime, we integrated them into the TAU 2020 contest C++ framework [23], which generates representative driver, -model, receiver stages. For our experiments, the driver and receiver gates are selected from the ASU ASAP 7nm Predictive PDK [24], which is publicly available in the OpenROAD GitHub repository [25]. Some basic MOSFET SPICE model parameters of the PDK are provided in Table II. However, the TAU framework only supports gates with one input/output pin, i.e. buffers and inverters, thus we also had to extend this framework to support multiple input/output gates (e.g. NAND, XOR, AOI, HA, DFFRS, DLatch) and account for any combinational, sequential, and asynchronous timing arc.[1]

TABLE II: ASU ASAP 7nm SPICE MOSFET Parameters

SPICE Parameter	Value
Structure Selector (GEOMOD)	1 (triple-gate)
Channel Length (L)	21 nm
Fin Height (H_{fin})	32 nm
Fin Thickness (T_{fin})	6.5 nm
Threshold Voltage ($V_{th,n}$; $V_{th,p}$)	0.25 V; 0.2 V
Oxide Permittivity (ϵ_{ox})	34.53 pF/m
Physical Oxide Thickness (T_{oxp})	21 nm

The examined -model loads are representative input admittance models of real IC interconnects of varying length, routed on different metal layers (up to 16 layers), and their resistance and capacitance values cover an exhaustive range of 0.2 to 100 kOhm and 0.0001 to 0.25 pF, respectively. Moreover, for the driver timing arc, we used an input voltage waveform with slew varying from 0.005 to 0.32 ns, which represents an ASU ASAP pre-driver gate. Finally, receiver output capacitance values in the range of 0.0004 to 1.473 pF are used, which along with driver input slew, cover the entire ranges used in the CCS model precharacterization.

A. Accuracy Results

To compare the accuracy of the investigated methods, in terms of delay and output slew RMSPE against HSPICE, a set of 50k stages was used. In Fig. 6, the horizontal axis corresponds to the time constant over input slew metric of the driver RC load (-model and receiver capacitance), i.e. $\frac{R(C_{far}+C_p)}{T_{low}^{high}}$, for all stages arranged in buckets. This metric

[1] Our delay calculator is available at https://github.com/digaryfa/UTH-Timer

Fig. 6: Gate delay and output slew RMSPE against HSPICE on a testcase with 50k stages.

was used to represent different RC and input waveform characteristics of driver loads. Fig. 6 clearly shows that the methods which use C_{total} as load model, i.e. M1 and M3, lead to extremely inaccurate results. For example, M1 results in 3.27 and 2.42 greater RMSPE, for delay and slew, respectively, compared to M2 (as shown in Table I).

Moreover, Table I and Fig. 6 present a comparison of the NLDM and CCS gate models. In general, methods using CCS present high delay and slew accuracy. For example, M4 presents 1.65% delay RMSPE, by using CCS as driver model, while M2 results in 5.86% delay RMSPE, by using NLDM. However, M4 may lead to slightly higher slew error, as it uses NLDM as a receiver model.

At this point, we can evaluate the impact of multiple C_{eff} values on the accuracy of delay and slew estimation. As depicted in Table I, the use of multiple C_{eff} values (i.e. in M5, M6 and ours) does not significantly influence the delay accuracy, compared to M4 which uses the same driver model and a single C_{eff}. On the other hand, output slew accuracy can be dramatically improved using multiple C_{eff} values. As can be seen in Fig. 6, specifically in bucket [0.50, 0.74], M4 results in approximately 24% slew RMSPE, while our method achieves 4% error.

As for the impact of Miller effect on delay and slew calculation, our proposed method leads to better results compared to M5, which ignores this effect. Fig. 6 demonstrates that for small values of $\frac{R(C_{far}+C_p)}{T^{high}_{low}}$, i.e. in the range [0, 0.09], the Miller effect has a significant impact on delay calculation, while slew calculation is slightly influenced. For example, in bucket [0, 0.01], our method achieves 0.87% delay RMSPE, compared to M5 which leads to 2.48% error.

Finally, we compare our method against M6, which provides the highest accuracy among all the examined methods. As shown in Table I, our method results in 1.32% delay RMSPE and 2.48% slew RMSPE, while M6 leads to 1.01% and 2.1% errors, respectively. However, M6 is significantly slower than our methodology, as interconnect transient simulation is time-consuming, rendering this method prohibitive for large designs. Therefore, considering only the investigated methods that are efficient for gate-level timing analysis, our method achieves the best accuracy results. In addition, it is worth mentioning that the proposed iterative method achieves less than 2.6% and 4% delay and slew RMSPE, respectively, even from the first iteration.

B. Runtime Results

To examine the scalability of our method, we generated various testcases, from 10 to 200k stages. For runtime evaluation, we used a Linux workstation with an Intel® 4-core, 8-thread CPU running at 3.60GHz, and 16 GB memory. Note that all the examined methods, as well as the HSPICE simulations, estimate the delay and slew for all stages in parallel, using multiple threads. Table III reports the detailed runtimes of the investigated methods, while Fig. 7 demonstrates their scalability with the number of stages. As shown in Table III, M6 is prohibitive even for small number of stages, while its execution time for 200k stages is 626.32 seconds (2328x slower than ours). On the contrary, all the other methods are quite fast, as they compute driver delay and slew for 200k stages in less than 0.27 seconds, while presenting similar scalability, as depicted in Fig. 7. In more detail, the NLDM-based methods, i.e. M1 and M2, need only 0.15 and 0.17 seconds for 200k stages, respectively, while M3 requires 0.21 seconds. The runtime overhead of using either one or three C_{eff} values is negligible, as methods M4 and M5 present, i.e. 0.21 and 0.24 seconds, respectively. The proposed method computes gate delay and output slew in 0.27 seconds for 200k stages, adding only a small overhead compared to `M5` which ignores the Miller effect. In general, our iterative method converges in 2.3 iterations on average and always in less than 4 iterations.

Note that for optimal results, the appropriate method may be applied based on its runtime and the characteristics of the stage to be analyzed (*i.e.* the relative bucket, as shown in Fig. 6).

TABLE III: Runtime results of the examined methods for testcases with number of stages varying from 10 to 200k

#Stages	Runtime (sec)						
	M1	M2	M3	M4	M5	M6	**Ours**
10	0.002	0.002	0.002	0.002	0.002	0.033	0.002
100	0.002	0.002	0.002	0.002	0.002	0.307	0.002
1000	0.002	0.002	0.003	0.002	0.003	3.017	0.003
2500	0.005	0.005	0.005	0.005	0.006	5.701	0.006
5000	0.006	0.006	0.007	0.007	0.008	15.135	0.009
10000	0.010	0.011	0.013	0.012	0.014	30.617	0.016
25000	0.022	0.024	0.029	0.027	0.032	67.357	0.035
50000	0.040	0.044	0.056	0.053	0.062	155.803	0.069
100000	0.077	0.085	0.109	0.104	0.121	322.373	0.135
200000	0.149	0.167	0.215	0.207	0.242	626.327	0.269

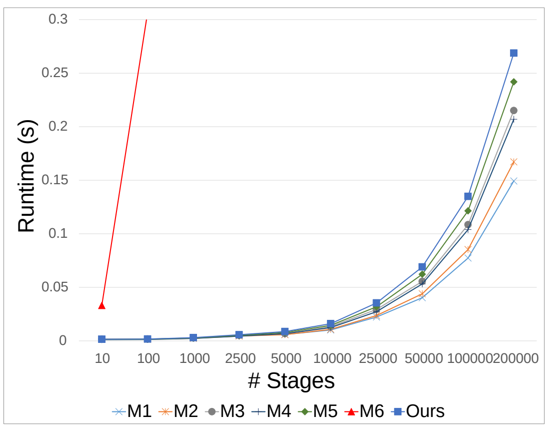

Fig. 7: Graphical comparison between the runtimes of the examined methods for testcases varying from 10 to 200k stages.

VI. Conclusions

In this paper, we presented an iterative method for fast and accurate gate delay estimation. The proposed approach estimates the driver output voltage waveform and C_{eff} in multiple waveform regions, while considering their interdependence. In contrast to prior works, it exploits library compatible CSMs, employs closed-form formulas, and considers the impact of Miller effect. Its high accuracy and fast convergence make it appealing for use either within early design stage or sign-off timing analysis. To evaluate our approach, we integrated our method into the TAU 2020 contest framework and generated 200k test stages, composed of representative π-models and ASU ASAP 7nm gates. Experimental results indicate that our approach achieves 1.3% and 2.5% delay and slew RMSPE, respectively, while converging in 2.3 iterations on average.

References

[1] Synopsys - HSPICE. Accessed: Dec. 22, 2020. [Online]. Available: https://www.synopsys.com/verification/ams-verification/hspice.html
[2] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer, 2009.
[3] C. Bittlestone, A. Hill, V. Singhal, and A. NV, "Architecting ASIC Libraries and Flows in Nanometer Era," in *Proc. of the 40th annual DAC*, 2003, pp. 776–781.
[4] J. F. Croix and D. Wong, "Blade and Razor: Cell and Interconnect Delay Analysis Using Current-Based Models," in *Proc. of the 40th annual DAC*, 2003, pp. 386–389.
[5] N. Menezes, C. Kashyap, and C. Amin, "A "True" Electrical Cell Model for Timing, Noise, and Power Grid Verification," in *Proc. of the 45th annual DAC*, 2008, pp. 462–467.
[6] C. Knoth, H. Jedda, and U. Schlichtmann, "Current Source Modeling for Power and Timing Analysis at Different Supply Voltages," in *Proc. of the DATE Conference & Exhibition*, 2012, pp. 923–928.
[7] Synopsys - Composite Current Source (CCS). Accessed: Dec. 22, 2020. [Online]. Available: http://www.opensourceliberty.org/ccspaper/
[8] Cadence - Effective Current Source Model (ECSM. Accessed: Dec. 22, 2020. [Online]. Available: https://www.cadence.com/en_US/home/alliances/standards-and-languages/ecsm-library-format.html
[9] P. R. O'Brien and T. L. Savarino, "Modeling the Driving-Point Characteristic of Resistive Interconnect for Accurate Delay Estimation," in *Proc. of the ICCAD*, 1989, pp. 512–515.
[10] J. Qian, S. Pullela, and L. Pillage, "Modeling the "Effective Capacitance" for the RC Interconnect of CMOS Gates," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 13, no. 12, pp. 1526–1535, 1994.
[11] B. N. Sheehan, "Library Compatible Ceff for Gate-Level Timing," in *Proc. of the DATE Conference & Exhibition*, 2002, pp. 826–830.
[12] ——, "Osculating Thevenin Model for Predicting Delay and Slew of Capacitively Characterized Cells," in *Proc. of the 39th annual DAC*, 2002, pp. 866–869.
[13] S. Abbaspour and M. Pedram, "Calculating the Effective Capacitance for the RC Interconnect in VDSM Technologies," in *Proc. of the ASP-DAC*, 2003, pp. 43–48.
[14] F. Dartu, N. Menezes, and L. T. Pileggi, "Performance Computation for Precharacterized CMOS Gates with RC Loads," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 15, no. 5, pp. 544–553, 1996.
[15] A. B. Kahng and S. Muddu, "Improved effective capacitance computations for use in logic and layout optimization," in *Proc. of the 12th International Conference on VLSI Design*, 1999, pp. 578–583.
[16] Y. Zhou, Z. Li, R. N. Kanj, D. A. Papa, S. Nassif, and W. Shi, "A More Effective Ceff for Slew Estimation," in *Proc. of the ICICDT*, 2007, pp. 1–4.
[17] J. M. Wang, J. Li, S. Yanamanamanda, L. K. Vakati, and K. K. Muchherla, "Modeling the Driver Load in the Presence of Process Variations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2264–2275, 2006.
[18] P. Feldmann, S. Abbaspour, D. Sinha, G. Schaeffer, R. Banerji, and H. Gupta, "Driver Waveform Computation for Timing Analysis with Multiple Voltage Threshold Driver Models," in *Proc. of the 45th annual DAC*, 2008, pp. 425–428.
[19] R. Puri, D. S. Kung, and A. D. Drumm, "Fast and Accurate Wire Delay Estimation for Physical Synthesis of Large ASICs," in *Proc. of the 12th GLSVLSI*, 2002, pp. 30–36.
[20] R. W. Freund, "Krylov-subspace methods for reduced-order modeling in circuit simulation," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1-2, pp. 395–421, 2000.
[21] I. Keller, K. H. Tam, and V. Kariat, "Challenges in Gate Level Modeling for Delay and SI at 65nm and Below," in *Proc. of the 45th annual DAC*, 2008, pp. 468–473.
[22] D. Garyfallou, I. Tsiokanos, N. Evmorfopoulos, G. Stamoulis, and G. Karakonstantis, "Accurate Estimation of Dynamic Timing Slacks using Event-Driven Simulation," in *Proc. of the ISQED*, 2020, pp. 225–230.
[23] TAU 2020 Timing Contest - Delay Calculator using Current Source Models. Accessed: Dec. 22, 2020. [Online]. Available: https://sites.google.com/view/taucontest2020/
[24] L. T. Clark *et al.*, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.
[25] ASU - ASAP7 PDK. Accessed: Dec. 22, 2020. [Online]. Available: https://github.com/The-OpenROAD-Project/asap7/