TAU 2021 Contest: Stage Delay Calculator using Current Source Model

iTimer Team

Members: Zi-Sheng Lin[1], Shih-Kai Lee[2], Yu-Kang Lin[2], Pei-Huan Tsai[2]
Advisors: Pei-Yu Lee[3], Iris Hui-Ru Jiang[12]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
[3]Cadence Design Systems, Inc
Email: {r08943090, b06901146, b07901027, b07901087}@ntu.edu.tw;
{palacedeforsaken, huiru.jiang}@gmail.com

***Abstract*—A stage delay calculator serves as the fundamental engine in Static Timing Analysis (STA) tools, which provides the delay and slew information of timing arcs for both cells and interconnects. While existing open-source STA tools use the Non-Linear Delay Model (NLDM) to compute timing arcs, the industry has already established Current Source Models (CSM) to deal with complicated effects emerging in advanced process nodes. We propose a stage delay calculator framework that exploits the CCS timing model to construct accurate waveforms at both the driver end and the receiver end. With the constructed waveforms, delay and slew results can be solved easily. Also, an RC reduction algorithm is developed to improve not only the efficiency but also the stability in large RC networks. Experimental results show the effectiveness of the proposed delay calculator framework with 0.998 correlation against SPICE simulation in tree-structured circuits.**

1. Introduction

In static timing analysis (STA), timing arcs are the most fundamental components defining the delay between two points. *Gate delay* and *net delay* are two major timing arcs used in gate-level STA, where gate delay indicates the delay from input of a logic gate to its output, and net delay indicates the delay from the output of a logic gate, through interconnect wires, to input of a fan-out gate. Furthermore, the *slew* at the endpoint of a timing arc is also concerned because the input transition time dramatically affects the switching speed of logic gates.

In traditional STA tools, gate delay is obtained by the Non-Linear Delay Model (NLDM) and so-called effective capacitance techniques. In such techniques, the driver cell is modeled as a Thevenin circuit, and then the equivalent capacitance of the RC load is solved iteratively. Nevertheless, due to the resistive effects of interconnects emerging in advanced technology nodes, it is difficult to obtain both accurate gate delay and slew at the same time via NLDM and effective capacitance techniques.

On the other hand, Elmore delay is commonly used in STA tools to calculate net delay and slew. In such interconnect modeling, input slew (of the interconnect) is used to estimate the response at the output of the interconnect with moment-based approximation. Although many works focused on more reasonable modeling of interconnects, one pitfall is the highly non-linear behavior of the driver cell such as the long-tail effect, which results in a very slow charging speed at the tail part of the driver waveform. In this situation, using a single input slew is not capable to attain accurate net delay and slew.

To deal with the aforementioned challenges, the industry established its advanced timing models way back in the 2000s, including the Effective Current Source Model (ECSM) and the Composite Current Source model (CCS). The two models resemble each other in modeling the driver cell as a current source, so they are called Current Source Models (CSM). With CSM enabled, commercial STA tools can deliver more accurate timing analysis, in terms of correlation to transistor-level SPICE simulation. Thus far, there have been no open-source STA tools, however, support either form of CSM libraries.

TAU 2021 Contest [1] aims at designing a stage delay calculator that is compatible with CCS libraries. The main task of the delay calculator is to calculate gate delay, gate slew, net delay, and net slew of a stage of logic gates. An example of the circuit of interest is shown in Figure 1, which consists of a driver cell, a tree-structured RC circuit representing interconnect wires, and receiver cells at the other end of the interconnect.

The goal of the stage delay calculator is to achieve high correlation against SPICE simulation while keeping the high efficiency, which is the spirit of STA. Therefore, we tackle the following challenges in development of the stage delay calculator:

(1) To derive the driver output waveform based on the CCS timing library, in order to accurately calculate gate delay and slew of highly resistive nets.

(2) To adopt efficient circuit simulation to solve net delay

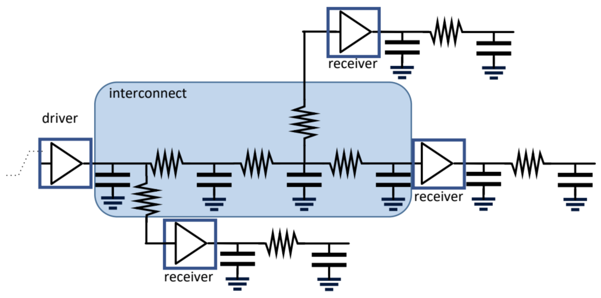


Figure 1. Example of the setup of TAU 2021 circuits.

and slew as accurate driver waveform is available.

(3) To apply RC network reduction to enhance both efficiency and stability of the delay calculator, which is desired for large RC interconnects.

2. Proposed Framework

Figure 2 shows the framework of the proposed delay calculator. First, the interconnect RC network is reduced by a TICER based RC network reduction algorithm. The reduced RC network contains fewer nodes and is used in subsequent both gate and net delay calculation stages. The RC reduction stage is crucial to not only the efficiency but also the solution quality in later stages. Due to the lower complexity of the reduced RC networks, both speed and stability can be improved in numerical solving procedure.

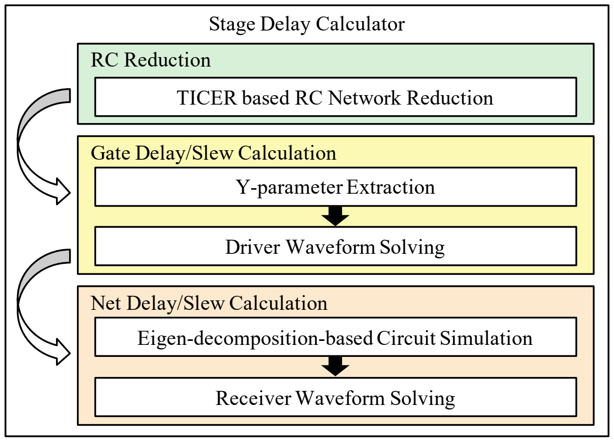
In the second stage, the reduced RC network is paired up with driver and receiver information, and then the waveform at the driver cell's output is solved. Both CCS driver and receiver data are exploited to construct accurate driver waveforms in this stage. While the gate slew is directly derived from the solved driver waveform, the gate delay, however, is calculated based on effective capacitance techniques for better accuracy.

After the driver waveform construction, modified nodal analysis (MNA) based circuit simulation is adopted to construct the waveform at the receiver end. Since the explicit RC reduction is applied in the first stage, no extra model-order-reduction techniques are required in this stage. With the computed driver and receiver waveforms, net delay and net slew can be calculated easily.

The proposed techniques and implementation are detailed in the following sections.

2.1. RC Network Reduction

To expedite delay/slew calculation, RC reduction can be performed on the input network. Time Constant Equilibration Reduction (TICER) [2] is a classic RC reduction method. By eliminating nodes with extreme time constant, TICER produces smaller, less-stiff RC networks. It has some desirable properties such as realizable RC circuits, controllable accuracy, and linear time complexity on most nets, thus being a fairly good choice to be applied to our model.


Stage Delay Calculator
RC Reduction
TICER based RC Network Reduction
Gate Delay/Slew Calculation
Y-parameter Extraction
Driver Waveform Solving
Net Delay/Slew Calculation
Eigen-decomposition-based Circuit Simulation
Receiver Waveform Solving
Figure 2. Overview of the proposed stage delay calculator.

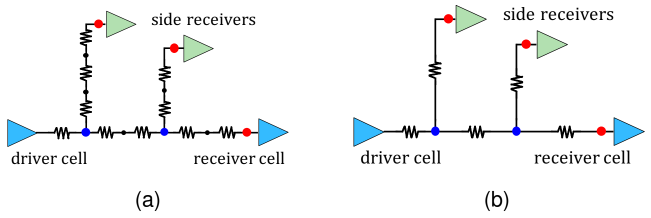


Figure 3. (a) A typical tree-structured net, where the blue nodes are *branch* nodes and the red nodes are *sink* nodes. (b) Reduced network that preserves the branch and sink nodes.

Inspired by TICER, a realizable RC reduction algorithm is implemented as the first stage of our delay calculator.

In our RC reduction algorithm, the sink and branch nodes are preserved to maintain the structure of the RC tree as shown in Figure 3. It is possible to annotate fixed receiver capacitances on the sink nodes and then delete them using the typical TICER algorithm. By doing so, a higher reduction rate thus more speedup in the delay calculation can be achieved. Preservation of sink nodes, on the other hand, allows the CCS receiver model to be used in later stages, which is especially critical to the accuracy when the RC value on the interconnect itself is small. The branch nodes are also preserved during the reduction in order to maintain the tree topology of the interconnect, which is helpful to the analysis in the later stages. Preservation of branch nodes also improves the overall accuracy in our experiments.

The following outlines our implementation of the RC reduction algorithm. First, we implemented the graph-based TICER algorithm, instead of matrix-based. The advantage of using graphs is to improve the efficiency since the RC interconnect trees are extremely sparse. Furthermore, we consider only *quick nodes* in the TICER algorithm, which is found to be more effective in both accuracy and reduction rate for the circuit of interest. To delete quick nodes, a user-defined time-constant is set as the threshold. We conclude that the threshold can be as high as possible, i.e., deleting almost all nodes except branch and sink nodes. As a result, the RC reduction algorithm always tries to reduce an RC wire segment into a single π-shaped model. In our experi-

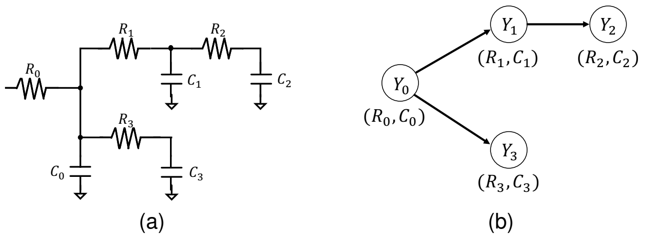


Figure 4. (a) A tree-structured RC interconnect. (b) Its RC-tree representation. The admittance polynomial at root, Y_0, is calculated by post-order traversal on the tree.

ments, even if 99% of quick nodes are deleted, the accuracy loss is less than 1% observed in SPICE simulations.

2.2. Y-parameter Extraction Algorithm

Algorithm 1 Gate Delay/Slew Calculation

```
Input: Circuit information, Spef RC, CCS timing library
Output: Driver waveform, Gate delay/slew
1: Construct RC-Tree
2: Extract $Y$-parameters
3: Construct driver waveform by MVTM
4: Calculate gate/delay slew
```

Algorithm 1 illustrates the proposed gate delay/slew calculation procedure. In the first two steps, Y-parameters are extracted from the interconnect and used to solve the driver waveform and gate delay/slew in the next two steps. Given an RC interconnect network in *Spef* format, the algorithm first constructs its RC-tree representation. Figure 4. demonstrates how an RC-tree representation is constructed from the interconnect circuit. For each node v_i, a tuple of RC value is given by (R_i, C_i) where C_i is the capacitance on the node, and R_i is the resistance on the edge $v_i v_j$ that connects to v_i's parent v_j.

After the construction of RC-tree T, the admittance polynomial $Y_i(s) = sB_i(s)/A_i(s)$ is computed for every node in T. There are three types of operations used to compute the admittance polynomials.

$$\begin{aligned} \text{leaf}: \quad & Y_i = \frac{sC_i}{sR_iC_i + 1} \\ \text{series}: \quad & Y_i = \frac{s(C_iA_j + B_j)}{sR_i(C_iA_j + B_j) + A_j} \\ \text{parallel}: \quad & Y_j + Y_k = \frac{s(A_jB_k + A_kB_j)}{A_jA_k} \end{aligned}$$

For example, in Figure 4(b), admittance of a leaf node, such as v_3, can be calculated by the *leaf* formula. If the node has one child, e.g., v_1, its admittance is given by the *series* formula, where A_j, B_j are the admittance polynomials of the child node. Finally, if v_i has multiple children nodes, i.e., v_0, its children admittance, Y_1 and Y_3, have to be merged by the *parallel* formula and performing *series* with v_0 afterwards.

By post-order traversal on T, the admittance at the root r, denoted $Y_r(s)$, can be computed recursively using the above three formulas.

Next, the first-order pole approximation of $Y_r(s)$ is calculated by partial fraction decomposition,

$$Y_r(s) = \frac{sB_r}{A_r} = \sum_{i=1}^{m} \frac{z_i}{s - p_i}, \tag{1}$$

where A_r is a polynomial of degree m and (p_i, z_i) is the pair of decomposed pole-zero applicable for driver waveform analysis in the next stage.

To exploit the CCS receiver models, multiple sets of Y-parameters can be extracted using different values of receiver capacitance. In our implementation, only two sets of Y-parameters were extracted: One uses all receiver's capacitance 1 (as they are charged less than 50%), and the other uses their capacitance 2. This approach is based on the assumption that all receivers are switching almost at the same time, which is valid if the RC value on the interconnect itself is relatively small. In this case, behavior of the RC load is dominated by the receivers.

2.3. Driver Waveform Solving

After extraction of first-order Y-parameters, multiple voltage threshold driver model (MVTM) algorithm [3] is adopted to solve the output waveform of the driver cell. In short, MVTM extends the concept of effective capacitance. It uses an equivalent capacitance to approximate the load RC network. While the traditional effective capacitance techniques try to fit a certain delay point, say $0.5V_{dd}$, MVTM divides the output voltage range into multiple voltage thresholds and the *dynamic* capacitance of each threshold is solved. In this manner, not only delay but also the full slew range can be covered at the same time.

After solving the driver waveform, corresponding to a voltage-time sequence, the gate slew can be calculated easily. As for the gate delay, however, we exploit the relationship between the effective capacitance C_e and the dynamic capacitance $C_{d,l}$:

$$0.5V_{dd}C_e = \sum_{\{l | V_{l+1} \le 0.5V_{dd}\}} C_{d,l}(V_{l+1} - V_l). \tag{2}$$

Then, the NLDM table is used to lookup gate delay with conventional interpolation techniques. The reason why we do not use the time difference of delay points between the input waveform and the output waveform is to suppress small error in the solved waveform. Such error exists because of non-smoothness of the CCS waveform data. CCS waveform data do not always sample at the delay point, and thus the MVTM algorithm may give a less accurate delay time. Instead, accumulated dynamic capacitance over the delay range is more accurate in our experiments.

2.4. Circuit Simulation Techniques

After solving the driver waveform, circuit simulation techniques are applied to solve the waveform at the receiver end. In this stage, modified nodal analysis (MNA) is adopted to construct the voltage waveform at the input of the specified receiver. An s-domain MNA formulation for RC networks can be defined as follows,

$$(\mathbf{G} + s\mathbf{C})\mathbf{V} = \mathbf{I_d}, \tag{3}$$

where $\mathbf{G}, \mathbf{C}$ are the conductance and capacitance matrices, $\mathbf{V}$ is the nodal voltage vector, and $\mathbf{I_d}$ is the driver vector for external voltage and current sources. After constructing the MNA equation, eigen-decomposition is adopted to solve the nodal voltages.

Let $\mathbf{G^{-1}C} = \mathbf{W\Lambda W^{-1}}$, where $\mathbf{\Lambda}$ is the diagonal eigenvalue matrix and $\mathbf{W}$ is the eigenvector matrix. Then, equation (3) can be rewritten as,

$$\mathbf{GW}(\mathbf{I} + s\mathbf{\Lambda})\mathbf{W^{-1}V} = \mathbf{I_d}. \tag{4}$$

Next, the nodal voltage can be solved by inverse the left-hand-side of equation (4),

$$\mathbf{V} = \mathbf{W}(\mathbf{I} + s\mathbf{\Lambda})^{-1}\mathbf{W^{-1}G^{-1}I_d} = \mathbf{W}(\mathbf{I} + s\mathbf{\Lambda})^{-1}\mathbf{Y}. \tag{5}$$

Since $(\mathbf{I} + s\mathbf{\Lambda})$ is diagonal, its inverse is done by taking reciprocal of the diagonal entries, that is, $(\mathbf{I} + s\mathbf{\Lambda})^{-1} = diag([\frac{1}{1+s\lambda}])$. This gives a nice equation for the s-domain voltage at node i,

$$V_i(s) = \sum_j \frac{(\mathbf{W})_{i,j}(\mathbf{Y})_j}{1 + s\lambda_j}. \tag{6}$$

In the receiver waveform solving stage, the driver cell is modeled as a current source whose output is given by the driver waveform, $I_d(t)$, which is a piecewise constant waveform solved by the MVTM algorithm. Let $(I_0, I_1, ..., I_{m+1})$ and $(t_0, t_1, ..., t_m)$ be the segments and time of change of $I_d(t)$. Then $I_d(t)$ can be decomposed as summation of changes and then be transformed to s-domain,

$$I_d(s) = \sum_{k=0}^{m} \frac{I_{k+1} - I_k}{s} e^{t_k s}. \tag{7}$$

Combining equations (6) and (7), the time-domain nodal voltage function can be derived by inverse Laplace transform,

$$V_i(t) = \sum_j \sum_k (\mathbf{E})_{jk} \left[1 - e^{-\frac{t-t_k}{\lambda_j}}\right] \cdot u(t - t_k), \tag{8}$$

where $\mathbf{E}$ is result of matrix multiplications and $u(t)$ is the unit step function.

2.5. Receiver Waveform Solving

While equation (8) gives an analytical solution to nodal voltages, the desired net delay and net slew have to be calculated by the inverse function of $V_i(t)$. The net delay/slew solving algorithm is listed in Algorithm 2. The algorithm applies the *bisection* method to solve the time of the receiver waveform reaches given voltage levels. Here, the validity of the bisection method is based on the assumption of monotonic response of the interested RC interconnect, that is, $(t_{s2} > t_d > t_{s1}$ for the receiver waveform, and the receiver waveform shall always lag behind the driver waveform $T^{drv}(v) \leq T^{rcv}(v), \forall\ 0 \leq v \leq 1$.

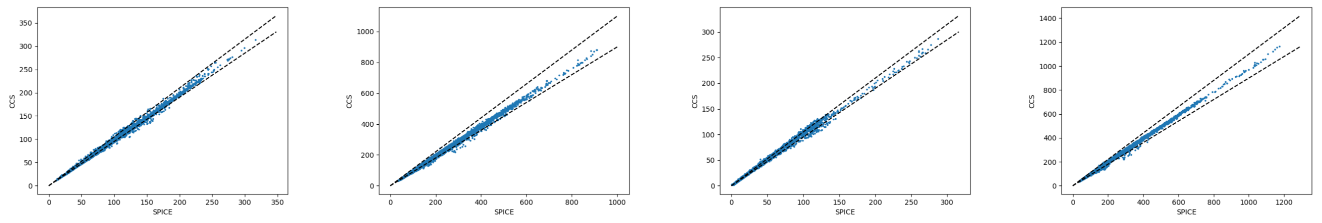
Algorithm 2 Net Delay/Slew Solving

```
Input: Driver time-voltage waveform $T^{drv}(v)$
    delay/slew voltage level $v_d, v_{s1}, v_{s2}$
Output: net delay $t$, net slew $s$
1: $t_{low} \leftarrow T^{drv}(v_{s2}), t_{up} \leftarrow 2t_{low}$
2: while $V^{rcv}(t_{up}) < v_{s2}$ do
3:     $t_{low} \leftarrow t_{up}, t_{up} \leftarrow 2t_{up}$
4: end while
5: $t_{s2} \leftarrow$ Bisection$(v_{s2}, t_{low}, t_{up})$
6: $t_d \leftarrow$ Bisection$(v_d, T^{drv}(v_d), t_{s2})$
7: $t_{s1} \leftarrow$ Bisection$(v_{s1}, T^{drv}(v_{s1}), t_d)$
```

In the net delay/slew solving algorithm, equation (8) is intensively computed and its computation time is proportional to the number of piecewise linear segments of the driver waveform. To speed up the solving process, Algorithm 3 is proposed to reduce the number of waveform segments before solving the net delay and slew. The main idea of Algorithm 3 is to merge adjacent segments that have similar slopes. A user-defined tolerance T is used to control how large slope difference is acceptable to merge two adjacent segments.

Algorithm 3 Driver Waveform Reduction

```
Input: change points $((t_0, v_0), \cdots, (t_{m-1}, v_{m-1}))$,
    tolerance $T$
Output: slopes $(s_0, \cdots, s_k)$, starting times $(\tau_0, \cdots, \tau_{k-1})$
1: $s_0 \leftarrow 0$; $\tau_0 \leftarrow 0$; $j \leftarrow 0$; $l \leftarrow 0$; $\sigma \leftarrow 0$;
2: for $i = 1$ to $m - 1$ do
3:     $s \leftarrow (v_i - v_l)/(t_i - t_l)$
4:     if $|\sigma - s|/s > T$ then
5:         $j \leftarrow j + 1$; $s_j \leftarrow s$; $\tau_j \leftarrow t_l$; $l \leftarrow i - 1$;
6:     end if
7:     if $i = m - 1$ then
8:         $\sigma \leftarrow (v_i - v_{i-1})/(t_i - t_{i-1})$
9:         $s_{j+1} \leftarrow \sigma, \tau_{j+1} \leftarrow t_i$
10:    else
11:        $\sigma \leftarrow s$
12:    end if
13: end for
```



(a) Gate Delay (b) Gate Slew (c) Net Delay (d) Net Slew

Figure 5. The correlation between our results and SPICE results in 10k randomly generated circuits. Here, the dashed lines indicate $\pm 5\%$ and $\pm 10\%$ outlier boundaries for delays and slews, respectively.

3. Experimental Results

We implemented the proposed algorithms in the C++ programming language. GSL [4] library is used to solve pole-residues, and Eigen [5] library is adopted to handle matrix operations. Our program was evaluated on an AMD Ryzen Threadripper 3990X 2.9GHz Linux Workstation with 197GB memory. We linked our program to the contest's Delay Calculator Took Kit (DCTK), and the ASU 7nm PDK [6] is used to generate the benchmarks.

We randomly generated 10k circuits to evaluate our approach. The accuracy and performance results are shown in Table 1. Runtime of the delay calculator is reduced by 180% when using the proposed RC reduction algorithm, and the RC reduction algorithm takes less than 4% of the overall runtime. While the memory usage increases from 608 KB to 1682 KB, the overhead is insignificant, which is less than 0.01% of the main DCTK program.

Interestingly, the accuracy of both gate delay and net delay are improved when RC reduction is applied. Root-mean-square (RMS) error of gate delay against SPICE is improved from 2.82% to 2.53% and RMS error of net delay is improved from 5.21% to 5.01%. Moreover, the number of gate delay outliers is reduced from 5.86% to 4.56%, and the number of net delay outliers is also reduced from 14.78% to 13.54%. Although the RC reduction slightly degrades the RMS error of gate slew from 4.15% to 4.29%, the accuracy of the net slew, which propagates to the next stage in actual STA applications, is not affected much. As a result, the RC reduction algorithm can improve the overall efficiency as well as accuracy of the proposed stage delay calculator.

The correlation coefficients between our results and SPICE results are listed in Table 2. The correlation coefficients of delays and slews are greater than 0.998. Figure 5 depicts the correlation plot between our results and SPICE results. We have about 5.52% outliers in both gate and net slew, and it can be seen that the outliers are located nearby the tolerance range.

The parallelization of our program was implemented via OpenMP. Figure 6 shows the resource requirement of our algorithm under parallelization. When using 16 threads, we achieved 8.25X runtime speedups with almost no memory overhead.

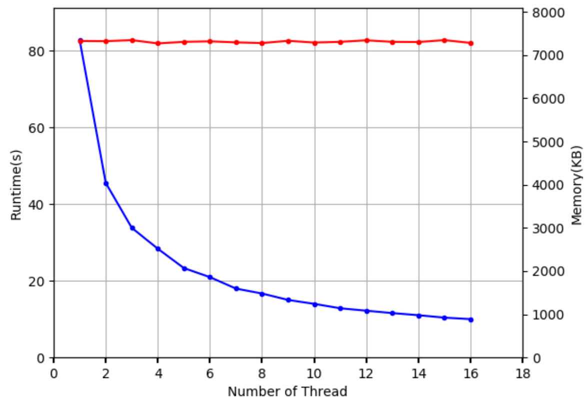


Figure 6. Runtime & incremental memory w.r.t number of threads in 1M random tree circuits.

TABLE 1. Quality and Performance Comparison

Algorithm	w/o RC Reduction	w/ RC Reduction
Runtime (sec.)	0.23	0.08
Incr. Mem (KB)	608	1620
RMS Gate Delay Error	2.82%	2.53%
RMS Gate Slew Error	4.15%	4.29%
RMS Net Delay Error	5.21%	5.01%
RMS Net Slew Error	2.99%	3.02%
No. Gate Delay Outliers	5.86%	4.56%
No. Gate Slew Outliers	2.40%	2.46%
No. Net Delay Outliers	14.78%	13.54%
No. Net Slew Outliers	1.50%	1.52%

TABLE 2. Correlation between SPICE and Our results

Measurement	Correlation Coefficient
Rise Gate Delay	0.9989
Fall Gate Delay	0.9987
Rise Gate Slew	0.9984
Fall Gate Slew	0.9986
Rise Net Delay	0.9981
Fall Net Delay	0.9983
Rise Net Slew	0.9986
Fall Net Slew	0.9990

4. Conclusion

In this work, we proposed a stage delay calculator framework compatible with CCS timing libraries. Our delay calculator gives accurate delay and slew results in typical tree-structured nets. The proposed stage delay calculator can serve as the fundamental engine in gate-level STA tools. In our framework, an RC reduction algorithm was proposed to reduce the complexity of RC interconnects, which brought a significant efficiency improvement. Also, the framework exploited CCS timing and receiver models to construct accurate driver waveforms, which enabled MNA-based circuit simulation to calculate more accurate net delay/slew results. In the experiments, our delay calculator achieved 0.998 correlation against transistor-level SPICE simulation.

Acknowledgment

Special thanks go to contest organizers for their constant support and valuable feedbacks.

References

[1] TAU 2021 Timing Contest: Stage Delay Calculator using Current Source Model, https://sites.google.com/view/tau-contest-2021/home

[2] B. N. Sheehan, ”TICER: Realizable reduction of extracted RC circuits,” *1999 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers*, 1999.

[3] P. Feldmann et. al., ”Driver waveform computation for timing analysis with multiple voltage threshold driver models,” *45th Design Automation Conference*, 2008.

[4] GSL - GNU Scientific Library, https://www.gnu.org/software/gsl/

[5] Gaël Guennebaud, Benoît Jacob et. al., Eigen v3, http://eigen.tuxfamily.org

[6] L.T. Clark, et. al., “ASAP7: A 7-nm FinFET Predictive Process Design Kit,” Microelectronics Journal, vol. 53, 2016.