Propagation Delay Approximation considering Effective Capacitance and Slew Degradation

Santiago Mok, advised by Prof. Puneet Gupta, EE Dept., UCLA

Introduction

Due to technological scaling and high frequency circuits, fast and effective timing algorithm is a desirable component for timing-driven optimization tasks. Elmore delay is widely adopted for interconnect delay approximation attributed for its simple analytical function of circuit parameters. In this report, I will present method to increase accuracy in estimating propagation delay using Elmore value. In each of the following sections, I will give an introduction to the methodology to increase interconnect estimation accuracy. In the first three sections I will talk about RC-π modeling and effective capacitance, Elmore computation and 50% point delay scaling, and slew degradation. Each of the aforementioned sections is followed with my implementation of the algorithm. The fourth section is dedicated to an add-on utility to compute leakage power that serves as a component to the next phase of the project.

I) Effective Capacitance

The effective capacitance is a more accurate approximation to the total net capacitance seen by the driving gate. In RC-interconnect modeling, the total net capacitance is often overestimated due to the presence of interconnect resistance. The net resistance tends to block some of the net load seen by the source gate. While the overestimated total capacitance is often insignificant or unnoticed in most circuit, but in particular, the effect is noticeable in high-frequency signals and large fan-out pin such as the clock. The higher order RC-π model (Figure 1) can be used to approximate interconnect resistive effect [1].

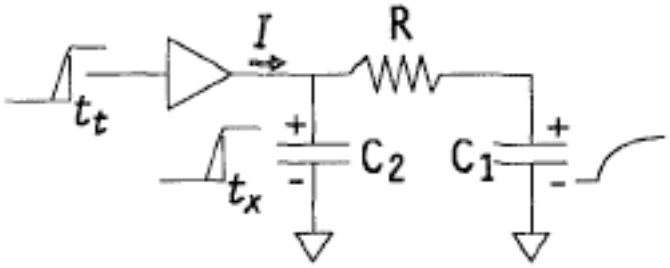

Figure 1 RC-π model with inverter gate [2]

The RC-π model is the 3rd-order circuit approximation for the driving-point admittance of a general RC tree where $C_1 = 5/6*C_{total}$, $C_2 = 1/6*C_{total}$, and $R=15/25R_{total}$ as explained in [1]. From the CRC model in figure 1, Abbaspour and Pedram [3] proposed an initial approximation to the effective capacitance:

$$C_{eff} = C_1 + \frac{R_d}{R_d + R_\pi} C_2 \qquad (2)$$

The equation is a function of the driving gate and the resistive effect that ranges between 0 and 1. If the driving strength were weak, as $R_d \rightarrow \infty$, the driver would see the entire capacitive load (C_1+C_2). On the other hand when the driving strength is strong, as $R_d \rightarrow 0$, the driver would only see C_1 as it load. Instead of obtaining the driver resistance R_d from a characterization table, I approximated R_d by solving the step response of a first order RC circuit when excited with a step input and assuming 10% to 90% rise time:

$$\frac{C * dV_{out}}{dt} + \frac{V_{out} - V_o}{R_d} = 0 \quad (3)$$

$$V_{out}(t) = (1 - e^{(-t/R_{dC})}) * V_o \quad (4)$$

$$R_d = \frac{t}{C * (\ln(0.9) - \ln(0.1))} \quad (5)$$

- Implementation and Algorithm

  In getWireCapEff(…) from oagTimerElmoreWireModel class. Effective capacitance is computed after output transition time of the relative driving gate is calculated.

  Algorithm

  1) Pass originally calculated total capacitance, resistance, and input transition time value into getWireCapEff function.
  2) Obtain library upper and lower RC slew threshold value that was previously store in oagTimerTPoint class. (Default to 20/80 if not defined in library)
  3) Calculate C1, C2, R as defined in the π model in figure 1
  4) Calculate driving gate resistance R_d from (5)
  5) Return Ceff computed using (2)

II) Elmore Delay

Elmore delay is a fast and simple interconnect delay estimation model computed through an RC-tree network. RC (resistor-capacitor) network has been widely used in electronic circuit design automation for modeling the driving gate and interconnect circuits. RC network model the circuit with capacitors from all nodes to ground, no capacitors connected between non-ground nodes, and no resistors connected to ground as shown in figure 2. In general, the Elmore value is computed by traversing the RC-tree and summing all resistors along the shortest path to the output node; each resistor sum is then multiplied by the grounded capacitor in the subset path:

$$T_{D_i} = \sum_{k=1}^{N} R_{ki} * C_k \quad (6)$$

In particular, the Elmore value at the output node "i" in figure 2 is computed by:

$$\text{Elmore value} = R_1C_1 + R_1C_2 + (R_1 + R_3)C_3 + (R_1 + R_3)C_4 + (R_1 + R_3 + R_i)C_i$$

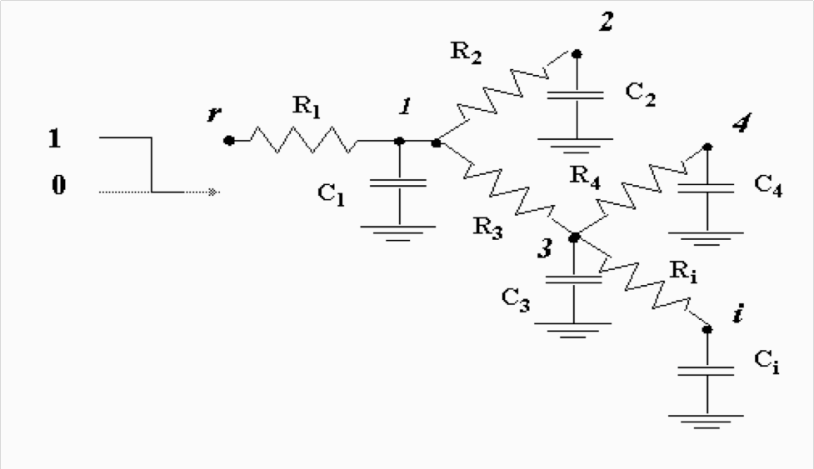

Figure 2 RC-Tree (courtesy of Prof Markovic)

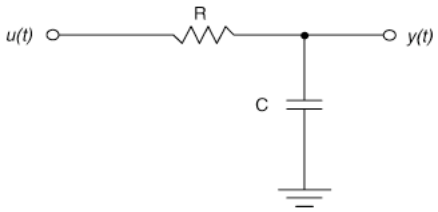

Figure 3 RC-Chain circuit [4]

Considering the first order RC circuit in figure 3, the 50% delay can be calculated by measuring the step response at y(t). Since the step response is the integral function over the impulse response, solving (7) for the 50% delay yield the result in (8).

$$v(t) = 1 - e^{-t/RC} \quad (7)$$

$$t = \ln(2) * RC \quad (8)$$

The Elmore value can be scales by ln(2) or 0.69 to effectively compute the 50% point delay.

- Implementation and Algorithm

  The Elmore delay algorithm is implemented in the class oagTimerElmoreWireModel and is incorporated into oagTimerTimer. The interconnect delay is included in the static timing analysis when instructed by the user. The Elmore algorithm follows an in-depth traversal of all the resistors and capacitors explained in the following algorithm. ‘

Algorithm

1. Collect all output nodes for the given parasitic network
2. Perform an in-depth traversal starting from the output node of the source driver
3. Store the resistor value and obtain the other end of the node
4. If the other node connects to more than one RC-tree branch:
    a) Compute and store the upper path partial computed Elmore value
5. Sum resistors and multiply capacitors in the Elmore path until:
    a) It hit a branch, go to 3 or
    b) Reaches the end of a branch or end of the Elmore Path, goto 6
6. If the other node is the end of a branch or Elmore path:
    a) Compute and store the partial computed Elmore value
    b) Sum the capacitors downstream and multiply it by the resistors sum in upstream path
    c) Recurse upstream, if there are non-visited branch
        i. Store the previous partial Elmore value into an array relative to each Elmore path
        ii. Goto 3
    d) Else the recursion reaches the top, compute all the subset value of the different Elmore paths.
7. Scales the delay value by 0.69

III) Slew Degradation

The slew rate or transition time refers to the rising or falling time when a signal switches state. Commonly used definitions of slew are 10/90 and 20/80 slews. The latter one is used in this project. The 20/80 slew refers to the time difference when the waveform crosses the 20% point and the 80% point. In order to simplify calculations, original OA Gear timing engine considered equal transition time when signals propagated from the output of the driver gate to the input of the connected gate. However, due to resistive and capacitive effect over the net, the input transition at the destination gate never matches exactly to the output transition time from the driver gate. That is, the time it takes to cross the 20% point and 80% point at the input of the next gate is delayed, which is refer to slew degradation. According to [5], the degraded slew value at the input of the next gate can be computed by taking the root mean square of the output slew and step slew over the net:

$$S_i = \sqrt{S_{out}^2 + S_{ED}^2} \quad (8)$$

where the step response (S_{ED}) of the RC circuit (Elmore) is computed by:

$$S_{ED} = \ln(4) * T_{RC} \quad (9)$$

for 20-80 transition time. T_{RC} is the Elmore delay value of the wire.

- Implementation and Algorithm

  Considering the slew model in (8) and (9), the degraded input slew value is computed under updateCellPathArr(…) function in oagTimerTimer class.
  Algorithm

a) Obtain the output slew value
b) Compute S_{ED}
c) Take the root mean square of the output slew and S_{ED}
d) Store the new input slew value to compute its cell delay and output slew.

IV) Power Calculations

After incorporating Elmore delay model and propagation delay scaling, we have further extended the timing engine with power calculation functionalities. The added functions include querying for leakage power of an individual cell and total leakage power of a circuit design. Additionally, we have added a function that query for the cell area. The purpose of the added power functionalities is to use power constraints as one of the parameter for the next phase of the project, gate-sizing algorithm. We will add further functionalities to the OAGear Timer API, as the benchmark circuits require.

- Implementation and Algorithm

The new functions are defined in oagTimerUtil class. There are three ways to query for cell leakage power: a) by oaInst, b) by oaModule, and c) by cell name. Total leakage power is queried by passing in the oaDesign parameter. Cell area can only be queried by oaInst.

a. getCellLeakagePower(…)
   a) If oaInst, find its master module
   Else If oaModule, goto b)
   Else if cell name, find it design and topModule
   b) Get the TPointMaster from the cell module
   c) Return the leakage power value previously stored from the .lib library

b. getTotalLeakagePower(…)
   a) Get the blocks for the given design
   b) Iterate over all oaInst
   c) Sum each individual oaInst leakage power

c. getCellArea(…)
   a) Get the topModule and masterModule from oaInst parameter
   b) Get the TPointMaster from the masterModule
   c) Return the cell area previously stored from the .lib library

Results:

To test my algorithm efficiency and correctness, I will run timing analysis against primetime, an industry timing analysis engine. I will use benchmark circuits from ISCAS85 and ISCAS89 design. In the following tables, design starting with c* are from ISCAS85 benchmarks and design starting with s* are ISCAS89 examples. I will report the result for different stages of the implementation. In table 1, the results are tabulated for timing without net delay. Then, I ran timing analysis considering the propagation delay of the interconnect; the Elmore value is scaled by ln(2) or 0.69. The results are tabulated in table 2. The third timing analysis is run with effective capacitance and the results are shown in table 3. Slew degradation computation and comparison are shown in table 4. Table-5 compares Leakage power calculation versus hand-calculated value.

Design	# of Cells	OA Gear Timer (ns)	Primetime (ns)
C880	212	5.60	5.61
C1355	581	3.56	3.57
S5378	1483	2.30	2.30
S9234	1296	2.847	2.85
S13207	2618	4.25	4.25

Table 1: Timing analysis excluding interconnect delay

Design	# of Cells	Max_Fanout	OA Gear Timer (ns)	Primetime (ns)	% Error
C880	212	6	6.75	6.77	0.30
C1355	581	4	4.31	4.32	0.20
S5378	1483	18	3.09	3.10	0.30
S9234	1296	23	4.01	3.96	1.26
S13207	2618	29	5.61	5.55	1.08

Table 2: Timing Analysis including interconnect propagation delay

Design	# of Cells	Max_Fanout	OA Gear Timer (ns)	Primetime (ns)	% Error
C880	212	6	6.75	6.77	0.30
C1355	581	4	4.31	4.32	0.20
S5378	1483	18	2.92	3.10	5.81
S9234	1296	23	3.88	3.96	2.27
S13207	2618	29	5.31	5.55	4.32

Table 3: Timing Analysis including interconnect propagation delay with Ceff

	i_14/Q -> i_1/A				i_1/O -> i_3/A		
s27	OA Gear	PrimeTime	%error		OA Gear	PrimeTime	%error
Load:	0.0139785	0.013978	0.00%		0.0067914	0.006792	0.01%
Net Delay	0.000136	0.000151	9.93%		8.13E-05	8.80E-05	7.61%
Output Transistion	0.0167741	0.017557	4.46%		0.00814966	0.008549	4.67%
Input Transition	0.0167752	0.017573	4.54%		0.00815044	0.008557	4.75%
Net Slew Degration	1.10E-06	1.60E-05	93.12%		7.80E-07	8.00E-06	90.25%
	i_3/O -> i_6/A				i_6/O -> i_7/B		
s27	OA Gear	PrimeTime	%error		OA Gear	PrimeTime	%error
Load	0.0115397	0.01154	0.00%		0.007319	0.00732	0.01%
Net Delay	0.000160762	0.000163	1.37%		9.00E-05	9.70E-05	7.22%
Output Transistion	0.0138476	0.014525	4.66%		0.0175655	0.018335	4.20%
Input Transition	0.0138494	0.01456	4.88%		0.017566	0.018343	4.24%
Net Slew Degration	1.80E-06	3.50E-05	94.86%		5.00E-07	8.00E-06	93.75%
	i_7/O -> i_8/A				i_8/O -> i_9/A		
s27	OA Gear	PrimeTime	%error		OA Gear	PrimeTime	%error
Load	0.0074509	0.007451	0.00%		0.0238035	0.023804	0.00%
Net Delay	9.19E-05	0.000101	9.01%		5.20E-04	5.28E-04	1.44%
Output Transistion	0.0178821	0.018661	4.17%		0.0285642	0.030062	4.98%
Input Transition	0.0178825	0.018671	4.22%		0.0285733	0.030077	5.00%
Net Slew Degration	4.00E-07	1.00E-05	96.00%		9.10E-06	1.40E-05	35.00%

Table 4: Slew Degradation Comparison

Design	Cells	Hand Calculation (pW)	OA Gear Computation (pW)	%Error
C880	212	120068.232	119861	0.17%
C1355	581	252281.55	252281	0.0002%
S27	16	13367.0416	13367	0.0003%

Table 5: Design leakage power calculations

Analysis

Interconnect delay have a major impact in computing propagation delay. Observing the results, interconnect contribute approximately 20%-30% of the total propagation delay. The 50% delay computed with Elmore value scaled by ln (2) is an overestimate of the actual delay shown in Primetime. The overestimate is shown in table-2 results; the values computed by OA Gear tend to be higher than Primetime timing values. To further reduce error in approximation, we incorporated effective capacitance and slew degradation into the delay computation. Observing table-3, the margin of error has increased because primetime used total load instead of effective capacitance for its timing calculations. Thus, the timing values reported by OA Gear is now an underestimate of the actual timing values in Primetime. In table-4, we compared slew

degradation results with Primetime. As you can observe the relative error between OA Gear and Primetime is very high. One reason is that Primetime added slew-derate to the output slew value even though slew_derate was set to '1' (meaning not to add slew-derate). We debugged individual cells without slew-derate and the transition time matches exactly with OA Gear. If we could have remove slew-derate from Primetime timing analysis, we would have reported a more accurate slew degradation comparison. Additionally, we have compared that cell leakage power calculations matches with hand calculations.

Conclusion

In this report, we modeled the wire in a logic circuit as a series of lumped RC chains. This type of model simplifies wire delay calculations through Elmore analysis, which is a fast and simple approximation of the delay propagated over the wire connecting two gates. We then computed the 50% time point by calculating the response assuming a step function is applied at the input. The result of the 50% point was that the Elmore delay computation was scaled by ln(2) (0.69). Since the resistive effect across the wire shields some of the total load seen by the driver, we computed the effective capacitance and incorporated it into OA Gear timing calculations. Since Primetime selectively use effective capacitance in it computation, we were unable to accurately measure our results. Furthermore, we added slew degradation to model the time difference when the signal switches at the output of the driver gate and at the input of the connected gate. Last, we incorporated leakage power calculation in OA Gear timing engine. Furthermore, even though we were unable to accurately compare our results, we have shown that interconnect delay add major time overhead to overall circuit timing.

References:

[1] P.R. O'Brien and T.L. Savarino, "Modeling the Driving-Point Characteristic of Resistive Interconnect for Accurate Delay Estimation," IEEE Explore, 1989.

[2] C.L. Ratzlaff, S. Pullela, and L.T. Pillage, "Modeling The RC-Interconnect Effects in a hierarchical Timing Analyzer," IEEE 1992 Custom IC Conf, 1992.

[3] R. Gupta, B. Tutuianu, L.T. Pileggi, "The Elmore Delay as a Bound for RC Trees with Generalized Input Signals", IEEE Explore, 1997

[4] R.Mita, G.Palumbo, M.Poli, "Propagation Delay of an RC-Chain With a Ramp Input", IEEE Transactions on circuits and systems-II: Express briefs, Vol. 54 No.1, January 2007.

[5] S. Hu, C. Alpert, J. Hu, S.K. Karandikar, Z. Li, W. Shi, C.N. Sze, "Fast Algorithms for Slew-Constrained Minimum Cost Buffering", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, No.11, November 2007