Signal Delay in Coupled, Distributed RC Lines in the Presence of Temporal Proximity

V. Chandramouli†, A. I. Kayssi‡†, and K. A. Sakallah†

†Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI-48109-2122

‡†Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon

Abstract

With improvements in technology, accurate delay modeling of interconnects is becoming increasingly important. Due to decreasing feature sizes, the spacing between the signal lines is also decreasing. Consequently, the switching activities on the neighboring lines can have a significant impact on the delay of the line of interest, and can no longer be ignored. Accurate modeling of this phenomenon, which we call the proximity effect, is the subject of this paper. This is similar to the state-dependency of logic gate delays, where signal delay can be affected by the switching activities on the side inputs of a gate. We describe an efficient and accurate delay computation method using precomputed interconnect moments that treats the coupled lines as uniform, distributed RC lines and does not make any lumped approximations. This allows the proposed delay model to be used in a timing analysis tool operating over both gate and interconnect domains while accounting for state-dependency.

1 Introduction

With the advent of deep submicron technologies, interconnect delays can no longer be neglected and may in fact dominate gate delay in some cases [3, 16]. These improvements in technology, while increasing system performance and putting more functionality on a single chip, are also reducing the spacing between adjacent lines [16]. As a result of this physical proximity, if the signals on the adjacent lines switch simultaneously or near simultaneously (temporal proximity) with the signal on the line of interest, the delay of the line may be affected significantly [27]. We refer to this as the proximity effect in the paper. As a result, the interconnect delays, like logic gate delays are state-dependent, where the delay of an interconnect line depends on the state (rising, falling, or quiescent) of the signals on adjacent signal lines. The state-dependency of logic gate delays has been recognized for some time [8, 21] and in this paper, we show how to model the state-dependent interconnect delay.

Interconnect analysis has been the subject of intense research in recent years ([3, 17, 19, 22, 25, 26] to name a few; we refer the reader to [5] for an introduction to these methods). Recog-
nizing the linear nature of interconnects, almost all of these techniques are based on a reduced order modeling in the Laplace (s) domain based on moment matching [15]. However, in many cases the emphasis is on finding a stable lower order approximation and using it in a traditional circuit simulation environment [3, 22, 25, 26]. While circuit simulation is useful at the transistor level of abstraction, we need efficient and accurate delay models for use in a timing analysis tool that operates over both gates and interconnects at higher levels of abstraction. Therefore, we seek fast, compact models that can be used to compute delay “on demand” by the timing analyzer. We draw a parallel to the gate delay models, where the model is either a lookup-table or a simple analytical formula (obtained through precharacterization).

Of course, there have been other research efforts at providing fast and accurate delay values for interconnects that are not intimately linked with circuit simulation. In [17], a technique is presented for the rapid evaluation of moments for large tree or near-tree like interconnects, such as a clock distribution net. However, the moments are computed using lumped segments to approximate the distributed lines and coupling between adjacent lines is not considered. For long sets of lines that are physically close and with increasing clock frequencies, a large number of lumped elements may be required that will increase the memory and computation requirements. In [19], capacitive coupling is considered but restrictions are placed on the nature of coupling. Moreover, transmission lines are modeled by lumped approximations, and though a technique for modeling distributed elements directly is also described in the paper, no example with coupled distributed lines is presented. Recently, a technique was given for fast and accurate delay computation of a single RC line that could be used in conjunction with a timing analyzer [23]. However, the proximity effect and the consequent state-dependence of delay was not considered. Another technique that computes the delay of a single RC line under a ramp input and that does not rely on moment matching is described in [10]. However, it ignores coupling between adjacent lines and the resulting formulas are complicated and not in a form that can be used with a timing analysis tool.

Our approach for modeling the proximity effect is based on moment matching; however, we precompute moments directly from the distributed lines without making any lumped approximations. We then provide a computationally efficient technique for calculating delay that takes into account the switching activities on the neighboring lines. We deal exclusively with loaded Uniform RC (URC) transmission lines [7], ignoring inductive effects and dielectric leakage, both of which are reasonable assumptions for on-chip interconnects. We make no assumptions about the load at the output of the line; it could be either capacitive (if driving a gate) or a \( \pi \)-circuit (if driving other interconnect), the two cases seen most often in digital ICs. As noted in [12], a \( \pi \)-circuit is a good approximation of the loading due to an interconnect structure. We consider coupling only between signal layers on the same level, since typically signal wires on the layers above and below run in an orthogonal direction. As a result, the overlap capacitance due to a single line on the line of interest will be small compared to the coupling capacitance to other lines on the same layer and can be neglected.

The paper is organized in 5 sections as follows. In Section 2, we define the conventions used for measuring delay and show examples that illustrate the effect of various physical and temporal parameters on line delay. Section 3 on delay computation is presented in two parts, with the first part assuming no temporal proximity and the second accounting for it. In Section 4, we compare the results of our method with those obtained from circuit simulation. Finally, some concluding remarks along with directions for future research are given in Section 5.
2 Proximity effects on URC lines

We first establish some conventions for measuring the delay and transition times adopted in the paper. It has been shown in the case of logic gates that delay can be negative if measured at the popular 50% point of the voltage swing of the gate [1, 4]. Instead, if we choose the thresholds to be the unity differential gain points of the gate's DC transfer curve, \( V_{th} \) and \( V_{ih} \), we always have positive delay [4, 11], which corresponds to a physically meaningful notion of delay. In order to conform to the gate delay models, we use the \( V_{th}, V_{ih} \) values of the gate driving the interconnect for measuring delay on the line. Thus, for a rising input, we measure the propagation delay, \( \Delta \), from the time instant the input to the line crosses \( V_{th} \) to the time instant the output of the line crosses \( V_{th} \). The transition time, \( \tau \), is measured from the instant the signal crosses \( V_{th} \) to the instant it crosses \( V_{ih} \).

Since we are interested in developing URC delay models suitable for timing analysis, these models must be developed independently of the driver characteristics. The temporally relevant characteristic of the driver output waveform is captured by the signal transition time \( \tau \), much like the gate delay models. However, to obtain easily computable analytical delay expressions while at the same time accounting for the wave-shape effects, we find it convenient to model the input waveform as an exponential assuming:

\[
v_i(t) = V_{oh} - (V_{oh} - V_{oh})e^{-\frac{t}{\tau}}
\]  

such that it crosses the \( V_{th} \) and \( V_{ih} \) thresholds at the same time instants as the output waveform of the driver.

We now describe the experiments that show how the delay on the line of interest can be affected significantly by events occurring on adjacent lines. Consider the geometry of the two lines shown in the cross-sectional and longitudinal views in Figure 2.1. The physical parameters of interest are also shown in the figure. The table listed in [24] was used to guide the selection of the line widths and spacings. Both lines were excited by exponential inputs and were identically loaded. We modeled the load by a \( \pi \)-circuit to allow the possibility of an interconnect load. We are interested in the variaton of delay on line 2 as a result of the switching activities on line 1. We denote the delay on line 2 when line 1 is quiescent by \( \Delta^{(1)} \) and the delay when line 1 is switching by \( \Delta^{(2)} \). The transition time of input to line 1 is denoted by \( \tau \), and that of line 2 by \( \tau_2 \). The temporal separation between the inputs to line 2 and line 1, \( \tau_{21} \), is the time interval between the onset of the two exponentials. Thus, a negative value means that the input
Figure 2-2: Effect of some important parameters on delay

to line 1 arrives first and a positive value implies the input to line 2 arrives first. In all the graphs of Figure 2-2, we plot the normalized delay, $\Delta^{(2)}/\Delta^{(1)}$ as a function of the normalized temporal separation, $(s_{t21})/\Delta^{(1)}$. We can summarize these plots as follows:

- The delay on line 2 reaches a minimum value for a particular value of the temporal separation between the inputs before it starts to increase with increasing magnitude of temporal separation. The value of the temporal separation that causes the minimum delay is itself a function of line length, input transition time ratios, load, etc. (see Figure 2-2(a-c)).
- The closer the two lines are, greater the effect of line 1 on line 2 owing to the increased coupling capacitance between the lines (see Figure 2-2(a)).
- Faster input transitions on line 1 relative to the input transition on line 2 have greater effect on line 2 delay (see Figure 2-2(b)).
- Long coupled lines show greater proximity effect than short ones (see Figure 2-2(c)).
- When the transitions on line 1 and line 2 are in the same direction, the effective coupling capacitance decreases and hence the delay of line 2 due to the proximity effect decreases. However, for an opposite going transition on line 1, the effective coupling capacitance increases and hence the delay on line 2 increases when
Figure 2-3: Why only three lines are important

temporal proximity is present (see Figure 2-2(d)).

While these examples show that proximity can affect delay by as much as 35%, the effect can in fact be more pronounced when more than two adjacent lines experience near-simultaneous switching. While we have considered only two lines, it is natural to ask what the effect of having more than two lines would be. It turns out that for timing analysis purposes, for a given line of interest, it suffices to consider only its immediate neighbors; thus we have to deal with a system of at most 3 coupled URC lines. This is explained in Figure 2-3 where we depict a 5-line system. Assuming all 5 lines have the same thickness, width, length and height from the ground plane and are equally spaced, we consider delay on line 3 as a function of the switching activities on other lines. On simulating this arrangement with HSPICE [14], the values for $C_{23}$ and $C_{13}$ were found to be 57.09pF/m and 3.32pF/m. Note the relatively small value of $C_{13}$. Based on circuit simulation of several such 5-line systems, we found that switching activities on lines 1 and 5 had negligible impact on the delay of line 3. Therefore, in the remainder of the paper we will be concerned with 3-line systems only. We describe our technique of delay computation for such a system in the next section.

3 Delay computation using moments

In order to simplify the presentation, we will first describe the delay computation of a single line assuming its neighbors are quiescent. This will then pave the way to account for switching events on the adjacent lines. In both these approaches, we first compute the moments directly from the differential equations governing the voltages and currents on the line. A circuit simulation approach would approximate the waveform using these moments and then perform a transient simulation to compute the delay. However, using the poles and residues computed from the moments, we directly compute the delay without doing a transient simulation. We denote the delay on line 2, when lines 1 and 3 are quiescent by $\Delta^{(1)}$, when the input to any one of lines 1 and 3 is switching in temporal proximity to the input on line 2, by $\Delta^{(2)}$, and when all three inputs switch in temporal proximity, by $\Delta^{(3)}$.

3.1 Delay computation without temporal proximity

Consider a single line terminated with a load $Y_L$ as shown in Figure 3-1. For computing delay, we are interested in finding the output waveform at $x = l$, $v_o(t)$. The input to the line, $v_i(t)$, is an exponential given by (2.1). Since interconnects are linear circuits, it is more conve-
In order to operate in the complex frequency domain, we can express the output voltage in the $s$ domain by:

$$V_p(s) = V_i(s)H(s)$$  

(3.1)

where $V_p(s)$ is the Laplace transform of $v_p(t)$ and $V_i(s)$ is the Laplace transform of $v_i(t)$ and is given by:

$$V_i(s) = \frac{1}{s(1 + sr_f)}$$  

(3.2)

and $H(s)$ is the transfer function. We seek a simple expression for $H(s)$ so that the inverse transform to the time-domain can be easily found. A popular technique, due to [15], of finding a simple expression for $H(s)$ is to expand it in a Taylor series about $s = 0$ and by matching the first few moments of the circuit, a simple rational expression can be obtained by a Pade approximation [2]. An efficient way of computing moments for lumped, linear circuits was first shown in [15]. Subsequently, a technique for computing the moments for distributed RLG C lines, in the context of circuit simulation, was described in [3, 19]. We have adapted the method in [3] for distributed, coupled URC lines as follows. We start with the Telegrapher’s equations that govern the URC line in the complex frequency domain as:

$$\frac{d}{dx}V(x,s) = -R I(x,s)$$  

(3.3)

$$\frac{d}{dx}I(x,s) = -sCV(x,s)$$  

(3.4)

where $R$ is the per-unit-length resistance of the line and $C$ is the per-unit-length total capacitance of the line (in case lines 1 and 3 are adjacent but quiescent then $C$ for line 2 is $C = C_{22} + C_{12} + C_{23}$, where $C_{22}$ is the capacitance to ground). We can combine these two equations to get a second order differential equation involving $V(x,s)$:

$$\frac{d^2}{dx^2}V(x,s) = sRCV(x,s)$$  

(3.5)

Suppose $V(x,s)$ can be expanded about $s = 0$ as $V(x,s) = V_0(x) + sV_1(x) + s^2V_2(x) + \ldots$. By matching the like powers of $s$ we get the following set of differential equations:

$$\frac{d^2}{dx^2}V_0(x) = 0$$  

(3.6)

$$\frac{d^2}{dx^2}V_1(x) = RCV_0(x)$$  

(3.7)
\[
\frac{d^2}{dx^2} V_k(x) = R C V_{k-1}(x)
\] (3.8)

where \( V_k(x) \) is the \( k \)th voltage moment and is a function of position. Equations (3.6)-(3.8) provide a method for recursively computing the moments at any point \( x \) on the line. Solving (3.6)-(3.8) we have:

\[
V_0(x) = A_{01} x + A_{02}
\] (3.9)

\[
V_k(x) = V_{kp}(x) + A_{k1} x + A_{k2}
\] (3.10)

where \( V_{kp}(x) \) is the particular solution of (3.8) obtained by integrating \( V_{k-1}(x) \) twice. The constants \( A_{01}, A_{02}, A_{k1}, A_{k2} \) can be determined from the boundary conditions at \( x = 0 \) and \( x = l \).

In order to find the moments of the transfer function, we find the voltage moments at \( x = l \) assuming an impulse voltage source at \( x = 0 \). The voltage moments at \( x = 0 \) are therefore, \( V_0(0) = 1, V_1(0) = 0, ..., V_k(0) = 0 \), which can be used to evaluate the constants \( A_{01}, A_{02}, A_{k1}, A_{k2} \).

The constants \( A_{01}, ... A_{k1} \) can be evaluated from the following equations at \( x = l \).

The voltage at \( x = l \) is related to the current by:

\[
I(l, s) = V(l, s) Y_L(s)
\] (3.11)

Expanding \( I(x, s) \) and \( Y_L(s) \) in powers of \( s \) and matching like powers of \( s \) we have:

\[
I_0(l) = V_0(l) Y_{L0}
\] (3.12)

\[
I_k(l) = \sum_{i=0}^{k} V_i(l) Y_{L(k-i)}
\] (3.13)

Note, however, that the voltage and current moments are related by (3.3). Thus we have

\[
I_k(l) = \frac{1}{R \frac{d}{dx}} \left. \frac{dV_k(x)}{dx} \right|_{x=l}
\] (3.14)

Using (3.13) and (3.14) at \( x = l \) and an impulse source at \( x = 0 \), we can solve (3.6)-(3.8) recursively to obtain the moments of the transfer function \( m_0, m_1, m_2, ... \).

For interests of accuracy, we match the first four moments to find a 2-pole, 1-zero approximation of the transfer function [9, 11]. While a Padé approximation could be used to find this, it is well known that this process sometimes yields spurious poles on the right-half of the complex \( s \) plane, which is not physically possible in the case of \( URC \) interconnects. However, recently, a stable technique was proposed in [23] that bypasses a Padé approximation and gives the poles and residues of the transfer function directly from the moments:

\[
p = \frac{-m_2}{m_3}
\] (3.15)
\[ p_2 = \begin{vmatrix} m_0 & m_1 \\ m_1 & m_2 \end{vmatrix} \]

\[ k_1 = \frac{m_0 + m_1 p_2}{p_1 - p_2} \]

\[ k_2 = \left( \frac{m_0 + m_1 p_1}{p_1 - p_2} \right) p_2^2 \]

where \( p_1, p_2 \) are the two poles, and \( k_1, k_2 \) are the two residues at the respective poles. Thus, we can write the output voltage as:

\[ V_o(t) = \frac{\left( 1 + x\tau_1 \right)}{(1 + x\tau_1)(1 + x\tau_2)} \]

(3.19)

where \( \tau_k = \frac{1}{p_k} \) for \( k = 1, 2 \), \( \tau_z = (k_1 + k_2) / k_1 p_2 + k_2 p_1 \), and \( V_i(t) \) is given by (3.2). The output voltage in the time-domain is then given by:

\[ v_o(t) = 1 + a_1 e^{-\frac{t}{\tau_1}} + a_2 e^{-\frac{t}{\tau_2}} + a_3 e^{-\frac{t}{\tau_z}} \]

(3.20)

where \( a_1, a_2, a_3 \) are functions of \( \tau_1, \tau_2, \tau_z, \tau_z \). Solving for delay would involve computing the time instant \( t_i \) at which the output waveform crosses the appropriate threshold voltage \( V_i \). We could accomplish this using the Newton-Raphson (NR) algorithm in an efficient way if a good initial guess is provided to the algorithm. Doing dimensional analysis [20], we note that \( t_i \) can be rewritten as [11]:

\[ \frac{t_i}{\tau_1} = f\left( \frac{\tau_1}{\tau_z}, \frac{\tau_2}{\tau_z}, \frac{V_i - V_{at}}{V_{ah} - V_{at}} \right) \]

(3.21)

where \( V_i = V_{ih} \) depending on whether we consider rising or falling transitions. For a given technology and driver size, we solved (3.20) numerically for a large number of circuits for rising and falling transitions and implemented the function as two 3-D tables for \( \frac{V_i - V_{at}}{V_{ah} - V_{at}} \) and \( \frac{V_{ih} - V_{at}}{V_{ah} - V_{at}} \). Therefore, for a given circuit, we interpolate into this precomputed table to get the initial guess for the NR algorithm to solve (3.20) for the delay, \( \Delta^{(1)} \). Since the initial guess is very close to the actual value, we found in practice that only one iteration of the NR algorithm suffices to get very accurate delay values and subsequent iterations do not improve the accuracy significantly. As we show in Section 4, for an exponential input, this method provides almost the same accuracy as a detailed circuit simulation.
Figure 3-2: A 3-line system

3.2 Delay computation in presence of temporal proximity

Now suppose we have a 3-line system (see Figure 3-2) and we are interested in the delay computation of the middle transmission line, line 2. In the s domain, we define the output voltage as:

\[ V_{o2}(s) = H_{12}(s)V_{i1}(s) + H_{22}(s)V_{i2}(s) + H_{32}(s)V_{i3}(s) \]  
(3.22)

where \( V_{i1}(s), V_{i2}(s) \) refer to the input and output of the \( j^{th} \) line. \( H_{12}(s) \) and \( H_{32}(s) \) are the transfer functions from lines 1 and 3 to line 2 respectively, which capture the effect of the switching activities on lines 1 and 3 on line 2. Since we are dealing with linear systems (we are dealing only with interconnects and not considering the non-linear drivers in this analysis), we use the superposition principle to compute each transfer function independently by assuming an impulse function for the input of interest with the other two inputs being zero. Moment computation, in principle, is the same as before though the algebra becomes a little more involved since we are dealing with vectors of voltages and currents. We denote vectors and matrices by **bold** font. Equations (3.3) and (3.4) become:

\[ \frac{d}{dx} V(x, s) = -RI(x, s) \]  
(3.23)

\[ \frac{d}{dx} I(x, s) = -sCV(x, s) \]  
(3.24)

where,

\[ R = \begin{bmatrix} r_{11} & 0 & 0 \\ 0 & r_{22} & 0 \\ 0 & 0 & r_{33} \end{bmatrix} \]

\[ C = \begin{bmatrix} c_{11} + c_{12} + c_{13} & -c_{12} & -c_{13} \\ -c_{12} & c_{22} + c_{12} + c_{23} & -c_{23} \\ -c_{13} & -c_{23} & c_{33} + c_{13} + c_{23} \end{bmatrix} \]

Combining these two equations we get:
\[
\frac{d^2}{dx} V(x, s) = sRCV(x, s) \tag{3.25}
\]

As before, we expand \( V(x, s) \) in powers of \( s \) to get:

\[
\frac{d^2}{dx} V_0(x) = 0 \tag{3.26}
\]

\[
\frac{d^2}{dx} V_k(x) = RCV_{k-1}(x) \tag{3.27}
\]

where \( V_k(s, x) \) is the vector of \( k \)th voltage moments. A standard technique to decouple the vector differential equations (3.26)-(3.27) is to use similarity transforms [13]. We find a non-singular matrix \( T \) such that

\[
V = TV_m \tag{3.28}
\]

and

\[
T^{-1}RCT = \text{diag}(\lambda_1, \lambda_2, \lambda_3) \tag{3.29}
\]

where \( V_m \) is the vector of mode voltages and the \( i \)th column vector of \( T \), \( T_{.i} \), is the eigenvector corresponding to \( \lambda_i \) of the matrix product \( RC \). Such a diagonalization of \( RC \) is always possible for URC lines since \( R \) and \( C \) are real, symmetric matrices [13]. Equations (3.26)-(3.27) can now be written as scalar differential equations:

\[
\frac{d^2}{dx} V_{m_i,0}(x) = 0 \tag{3.30}
\]

\[
\frac{d^2}{dx} V_{m_i,k}(x) = V_{m_i,k-1}(x) \tag{3.31}
\]

where \( V_{m_i,k}(x) \) is the \( k \)th moment of the mode voltage on line \( i \). The computation of the moments of the transfer function now is in principle the same as before. The boundary condition at \( x = 0 \) is the vector of source voltages (one of them impulse and the other two zero, depending on the transfer function being computed) and at \( x = l \) the voltage moments are related to the current moments as:

\[
I(l, s) = Y_T(s)V(l, s) \tag{3.32}
\]

where \( Y_T(s) \) is the row vector of the load admittances. We define vector moment relations similar to (3.12)-(3.13) and use (3.23) to relate the vector voltage and current moments. Thus, the moments of the desired transfer function are computed. From these moments, we get a 2-pole, 1-zero approximation of the transfer function using (3.15)-(3.18) and the output response in the \( s \) domain using (3.19). Since we are dealing with matrices that are at most \( 3 \times 3 \), the moment computation for coupled lines can still be done efficiently. In fact, these moments can be precomputed while extracting the electrical circuit from the layout before the timing analysis starts.

The actual output voltage on line 2 in the time domain can be obtained by finding the inverse Laplace transform of (3.22). Such an inversion can be formed symbolically since the forms of
the three transfer functions are quite simple and are similar to that of the no proximity case. The temporal separations, \( \Psi_{21} \) and \( \Psi_{02} \), can be handled simply by multiplying the corresponding transfer function by \( e^{-\Psi_{21}^2} \) or \( e^{-\Psi_{02}^2} \). The resulting \( V_{out}(t) \) will be a nonlinear equation that needs to be solved at the desired threshold voltage to compute delay. However, to attain convergence relatively quickly, we use the NR algorithm to solve for delay by using \( \Delta^{(1)} \) as the initial guess. Note that \( \Delta^{(1)} \) can be easily computed using the moments of \( H_{23}(s) \) which are required anyway to compute \( V_{out}(s) \). In general, if the proximity effect is not severe, only a few iterations of the NR algorithm will suffice to get accurate values of \( \Delta^{(3)} \). The cases that require more iterations are when the lines are physically very close and the combination of the temporal parameters is such that the proximity effect is severe. Thus, our approach for modeling proximity is much faster than a transient simulation and is suitable for use in a timing analyzer. In the following section, we compare the accuracy of our approach with that of a circuit simulator.

4 Experimental results

In order to compare the efficacy of our method, we performed 100 random simulations of a 3-line system similar to the one shown in Figure 3-2, using HSPICE. The parameters of interest that were varied are listed in Table 4-1. The physical separation between the two lines was fixed at 0.7\( \mu \). The output load at each line was modeled as a \( \pi \)-circuit to model the loading of other interconnect structures driven by these lines, as shown in Figure 4-1. All three lines were excited by exponential inputs. In all the error histograms in this section, the x-axis is the absolute error in picoseconds (absolute error was chosen rather than relative error to prevent the artificial inflation of error for small delay values) between our model and that obtained by simulation and the y-axis is the number of cases that fall within the corresponding absolute error. We first consider the case of no temporal proximity with a single rising transition on line 2 and lines 1 and 3 quiescent at 0V. In Figure 4-2(a), we show the error in delay computed by the method in Section 3.1 compared to a transient simulation. As can be seen from the figure, our method involving a table lookup and one iteration of the NR algorithm provides excellent accuracy, with most of the delay values within 3ps of the actual value. The mean error, \( \mu \), was -1.13ps and the standard deviation, \( \sigma \), was 0.41ps.

We next consider the case of two rising inputs applied to lines 1 and 3. The range of their temporal separations with respect to line 2 in the simulations are listed in Table 4-1. This range was chosen so that there was a marked effect of proximity on delay (see Figure 2-2). The error in delay computed by our method with respect to a detailed circuit simulation is shown in Figure 4-2(b). The relatively small value of the mean error shows that our method gives excellent results on average. Next, we applied falling inputs to lines 1 and 3 and the results are shown in Figure 4-2(c). Once again, the small value of the mean error shows that on average our method gives very accurate results. Table 4-2 shows how ignoring the proximity can severely underestimate or overestimate delay. We compare the minimum, maximum and mean error in delay if we ignore the transitions on neighboring lines and compute delay using the method of Section 3.1, versus taking proximity into account and computing delays using the method of Section 3.2. Both rising and falling transitions on the adjacent line are considered.

In order to estimate the error due to an exponential and a ramp approximation of the driver output waveform, we performed the following experiment. The 3-line system was driven by a
Figure 4-1: The load circuit for the $l^{th}$ line

Table 4-1: The parameters used in the experimental validation

<table>
<thead>
<tr>
<th>name of parameter</th>
<th>parameter range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ht$</td>
<td>0.5 $\mu$m – 2.0 $\mu$m</td>
</tr>
<tr>
<td>$l$</td>
<td>1 mm – 5 mm</td>
</tr>
<tr>
<td>$w$</td>
<td>0.7 $\mu$m (fixed)</td>
</tr>
<tr>
<td>$sp_{12}, sp_{23}$</td>
<td>0.7 $\mu$m (fixed)</td>
</tr>
<tr>
<td>$th$</td>
<td>0.6 $\mu$m (fixed)</td>
</tr>
<tr>
<td>$\varepsilon_r$</td>
<td>3.9 (SiO$_2$)</td>
</tr>
<tr>
<td>$\rho$</td>
<td>28 x $10^{-9}$ $\Omega$m (Al)</td>
</tr>
<tr>
<td>$\tau_1, \tau_2, \tau_3$</td>
<td>50 $p$s – 500 $p$s</td>
</tr>
<tr>
<td>$(st_{21})/\Delta^{(1)}, (st_{23})/\Delta^{(1)}$</td>
<td>$-1$ – $0.5$</td>
</tr>
<tr>
<td>$R_{L11}, R_{L21}, R_{L31}$</td>
<td>50 $\Omega$ – 150 $\Omega$</td>
</tr>
<tr>
<td>$C_{L11}, C_{L21}, C_{L31}$</td>
<td>50 $fF$ – 250 $fF$</td>
</tr>
<tr>
<td>$C_{L12}, C_{L22}, C_{L32}$</td>
<td>50 $fF$ – 250 $fF$</td>
</tr>
</tbody>
</table>

* These are transition times measured from the usual 10%-90% points. They can be easily scaled to the $V_{ref}/V_{th}$ transition times using (2.1).

pair of inverters in a 0.6 $\mu$m CMOS technology as shown in Figure 4-3(a). We measured the driver delay from A to B, $\Delta$, and the transition time at A, $\tau_1$. The inverters were then replaced by an exponential and a ramp voltage source in turn such that they had the same transition time at A as $\tau_1$. The corresponding exponential and ramp delays, $\tilde{\Delta}$, were measured (see Figure 4-3(b)). This experiment was repeated a 100 times by randomly varying the line parameters according to Table 4-1 and the driver p- and n-channel transistor widths as shown in the figure. The resultant relative errors as a function of the driver delay normalized to the transition time at A are plotted in Figure 4-3(c)-(d). As can be seen, the exponential waveform is a better approximation compared to the ramp; further, the errors due to the exponential waveform are still within acceptable limits for timing analysis. To improve accuracy further, in fact, one can fit an approximate curve through the data points in Figure 4-3(c) for the exponential approximation and derive an error macromodel. Then, for a given $\tau_1$, the error macromodel can be used to correct the delay computed using the exponential approximation.
Table 4-2: Comparison of absolute errors

<table>
<thead>
<tr>
<th></th>
<th>No Proximity (abs err (ps))</th>
<th>Proximity model (abs err (ps))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>Rising</td>
<td>89.28ps</td>
<td>1.89ps</td>
</tr>
<tr>
<td>Falling</td>
<td>-1.11ps</td>
<td>-99.19ps</td>
</tr>
</tbody>
</table>

Figure 4-2: Error distribution using our model

5 Conclusions and future work

We have demonstrated the significance of the proximity effect on interconnect delay and presented a fast and accurate technique to account for this phenomenon. The delay model presented in the paper can be used with a timing analysis tool that operates on both the gate and interconnect domain while accounting for state-dependency. While we have used exponential waveforms for delay computation, we have given experimental results that show that an exponential waveform is a better approximation for a driver waveform than a ramp waveform. Further, the errors introduced by an exponential approximation of driver waveform are within
Figure 4-3: Comparison of errors due to exponential and ramp approximations

acceptable timing analysis error limits. Our future goals are to extend this model to account for inductive effects and consider non-uniform distributed lines. We are also investigating other waveshapes (such as two-time constant waveforms) for better approximating the driver output waveform. Another issue worth investigating is the problem of correctly modeling the input admittance of the coupled lines so that a suitable delay macromodel for the driver driving such a system can be derived.

References


