Determination of Worst-Case Aggressor Alignment for Delay Calculation*

Paul D. Gross, Ravishankar Arunachalam**, Karthik Rajagopal, and Lawrence T. Pileggi
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

Increases in delay due to coupling can have a dramatic impact on IC performance for deep submicron technologies. To achieve maximum performance there is a need for analyzing logic stages with large complex coupled interconnects. In timing analysis, the worst-case delay of gates along a critical path must include the effect of noise due to switching of nearby aggressor gates. In this paper, we propose a new waveform iteration strategy to compute the delay in the presence of coupling and to align aggressor inputs to determine the worst-case victim delay. We demonstrate the application of our methodology at both the transistor-level and cell-level. In addition, we prove that the waveforms generated in our methodology converge under typical timing analysis conditions.

1. Introduction

Crosstalk between adjacent lines is becoming increasingly significant as IC dimensions enter the deep submicron (DSM) region. MOS feature size scaling in lateral dimensions causes the coupling capacitance between adjacent lines to become a significant, sometimes dominant portion of the total net capacitance. Consider the following prediction courtesy of the SIA roadmap[1]: For a 100 nm technology, the height of metal wires is expected to be 360 nm versus a spacing of only 130 nm. Gates driving parallel lines at these dimensions will strongly impact one another due to the coupling.

When coupling capacitance is dominant, fast switching in aggressor gates can induce a large amount of noise on the victim line. If an aggressor and victim switch simultaneously in the same direction, the victim will speed up[2][3]. Likewise, if the aggressor and victim switch in opposite directions, the victim will slow down[2][3].

The principal problem in computing the worst-case delay inclusive of coupling noise is ascertaining the relative switching times of coupled gates. Present timing analyzers must rely on trial-and-error methods to align the switching times, and this is impractical for highly coupled lines. In this paper we propose a waveform iteration methodology to obtain the aggressor alignment that induces worst-case delay on the victim.

Our waveform iteration methodology can be applied at both the transistor-level and cell-level. The methodology is designed to handle simultaneous switching of coupled lines with arbitrary start times for the aggressor inputs. We will prove that the waveforms generated at the driving and fanout nodes converge to the actual worst-case waveforms. We will also demonstrate that in practice it is not necessary to use the final (converged) waveforms while optimizing for the worst-case alignment of the aggressor input(s).

Traditional worst-case delay calculators that model coupling capacitance as twice the capacitance to ground cannot capture the coupling influence of aggressor ports. The result is that these uniport models tend to be overly pessimistic on average, yet can easily underestimate the worst-case delay when the victim driving strength is weak relative to the aggressors'[2][4]. We are able to explicitly capture the influence of coupling using reduced order N-port macromodels without losing a great deal of computational efficiency. A significant benefit of the methodology described in this paper is that there is no dependence on N-port macromodel passivity, which requires costly multi-input multi-output (MIMO) macromodels.

The rest of the paper is organized as follows: Formulating the alignment problem is the topic of Section 2. Section 3 discusses the waveform iteration methodology we employ to align aggressor inputs for worst-case victim delay. Section 4 describes *how we generate* the waveforms used in Section 3 at both the transistor-level and cell-level. Proof of waveform convergences is the topic of Section 5. Section 6 describes *how we use* the waveforms we generate for worst-case alignment. Results illustrating waveform convergence and worst-case alignment are provided in Section 7. Section 8 contains some closing remarks.

2. Problem Formulation

Computing the worst-case delays for a path in timing analysis corresponds to calculating the worst-case delays for each of the logic stages (dc coupled components, or gates and their associated interconnect) that comprise that path. In static timing analysis this is a difficult problem due to the potentially dominant impact of RC line coupling on delay. To effectively handle this problem, the coupled interconnect for each logic stage should be modeled in terms of a reduced order macromodel to control the overall circuit complexity. Various methods can be used for the model-order reduction: AWE[5], PRIMA[6], PVL[7], PACT[8] and Arnoldi method[9]. Importantly, our methodology is compatible with all these model-order reduction schemes and does not require macromodel *passivity*.

If SPICE were used to model the nonlinear transistors in a coupled stage, for example, the interconnect macromodel should be pas-

* This work was supported in part by the Semiconductor Research Corporation under contract 98-DC-068 and a grant from Intel Corporation.
** Supported by an IBM Fellowship.

ICCAD98, San Jose, CA, USA

sive. However, presently, only multi-input multi-output (MIMO) methods of model-order reduction can guarantee passivity for RLC interconnect[6][8]. The drawback of a MIMO approach is that the amount of information needed to accurately model the reduced interconnect grows as a function of the number of ports. Specifically, if q is the order of approximation for the reduced order model and N is the number of ports, then a total of $N.q$ variables are needed for each term of the reduced model. Importantly, it is expected that the analysis time of a logic stage will be dominated by the coupled interconnect complexity in most cases, therefore, efficient handling of the interconnect macromodel is of the utmost importance.

A more efficient means of model-order reduction is to use single-input multi-output (SIMO) techniques. Here the order of approximation, q, is sufficient to accurately represent each term used in the model-order reduction. However, for any SIMO-compatible scheme to remain passive, it must ensure that non-active models are used with the interconnect.

Once we have a workable interconnect macromodel, obtaining the worst-case delay is still a difficult problem. The brute force approach is to manually align the inputs in a trial-and-error fashion that is time-consuming and inefficient. The trial-and-error approach exists because it is not possible to solve for each gate's output waveform while concurrently taking into account the coupling noise due to the switching of other lines.

3. Waveform Iteration Methodology

The methodology we propose to align aggressor inputs for worst-case victim delay is delineated via the flowchart shown in Figure 1. Model-order reduction of the interconnect is carried out only once before the iterative procedure begins. The start times of the aggressor inputs are selected arbitrarily (it is logical to start them at the same time the victim starts switching). However, we assume two things: (i) the edge rate of the primary victim input is known a priori, and (ii) the edge rates of primary aggressor inputs are as fast as can be expected from the design. Computing the initial voltage waveforms at the driver and fanout nodes involves treating the other lines as non-switching. This is easily handled in terms of the interconnect macromodel equations.

Model-order reduction engenders two matrices: an admittance matrix (denoted the Y-parameter matrix) and a transfer-function matrix (denoted the H-parameter matrix). The Y-parameters are used to relate voltages at each driver port to the current at the driver port under scrutiny [10]. The H-parameters are used to relate voltages at each driving point to the voltage at the fanout node of interest.

We calculate the Y-parameters (in a SIMO manner[5]) by applying a unit-step voltage to one of the driving points, shorting the remaining ports, then determining the current "flowing" into the interconnect at each port. An example is shown in Figure 2(a) that computes all of the Y-parameters in the first column of the Y matrix.

The H-parameters are computed by applying a unit-impulse to each driving point while shorting the remaining ports. Then, we solve for the voltage at each fanout node. An example that computes all of the H-parameters in the first column of the H matrix is shown in Figure 2(b).

Model-order reduction of the interconnect permits us to generate approximate models for the interconnect admittances and transfer functions in terms of the dominant poles. The impulse response of a Y-parameter term can be written as:

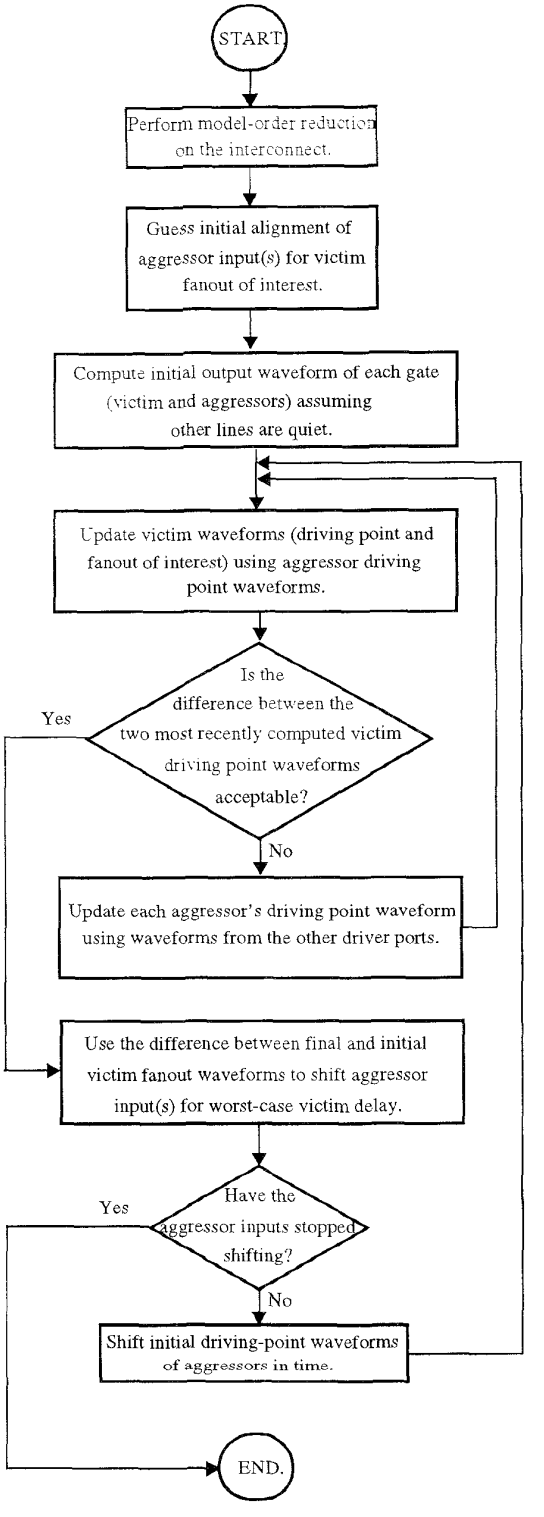

FIGURE 1: Waveform iteration methodology.

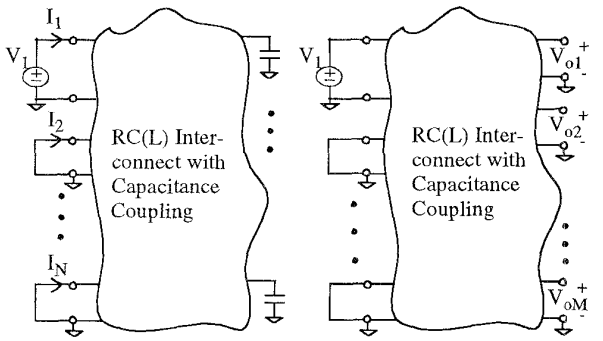

FIGURE 2: a) Solving for Y-parameters b) Solving for H-parameters.

$$Y_{ij}(s) = \sum_{m=1}^{n} \frac{sk_m}{s - p_m} \quad (1)$$

A q-th order dominant pole approximation is of the form:

$$\hat{Y}_{ij}(s) = \sum_{m=1}^{q} \frac{s\hat{k}_m}{s - \hat{p}_m} \quad (2)$$

where $q << n$ for most lossy interconnect problems.

The initial voltage waveforms at each driving point are computed by shorting the voltage waveforms at the other driver ports. Updating the victim (and aggressor) waveforms involves incorporating the switching effects of the other lines. The coupling effects are treated via the transadmittances (a subset of the Y-parameters).

Incorporating the coupling effects in subsequent waveform iterations is represented by the relaxation procedure portrayed as the inner loop of Figure 1. We can use the voltage waveforms from the previous iteration (Gauss-Jacobi), or the most up-to-date voltage waveforms (Gauss-Seidel). *In practice, it is reasonable to update only the victim waveforms once* (implying it is not necessary to iterate on the inner loop of Figure 1). This is much like the process of waveform relaxation, but is extremely efficient since it is applied only at the interconnect port nodes [11].

The driving point waveforms and fanout waveforms of interest can be computed at the transistor-level or cell-level. Determining the delay at the transistor-level is straightforward: simply look at the differences between input, driving point, and fanout waveforms. However, at the cell level it should be noted that we use the Thevenin-voltage model proposed in [12], hence, it is the Thevenin voltage parameters[1] that are part of the iteration process, rather than the driving point waveforms. More on this in Section 4.2.

4. Computation of Victim and Aggressor Waveforms

The voltage waveforms needed for our iterative methodology can effectively be computed at either the transistor-level or the cell-level. We distinguish between the two levels of abstraction in the following subsections.

[1] For the purpose of this paper, the reader can consider the Thevenin voltage waveforms as "identical" to the driving point waveforms.

4.1. Transistor-Level Computation

Analysis of a logic stage at the transistor-level requires decoupling of the transistors and the interconnect, clustering of the transistors into "gates", and efficient macromodeling of the interconnect (since it can be the dominant runtime bottleneck). The interconnect can be efficiently modeled in terms of the reduced-order Y-parameters. From the Y-parameters, each gate "sees" the driving point and transfer admittances, as shown in Figure 3 for the ith driver.

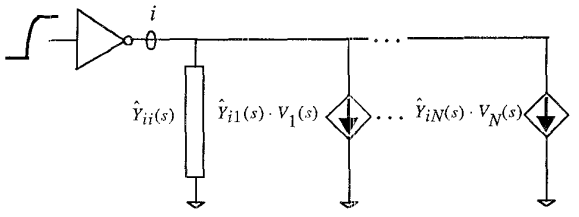

FIGURE 3: Modeling the Y-parameters of the ith driver.

The driving point admittance terms for an RC-coupled interconnect circuit, $\hat{Y}_{ii}(s)$, can be synthesized as a parallel combination of q resistor-capacitor chains in series (see Figure 4) [13][14]. Each transadmittance term $\hat{Y}_{ij}(s)$, j≠i, is treated as a voltage-controlled current-source. Assuming that the voltage waveforms at each driver port can be modeled as piecewise linear, we can treat the waveforms at the other ports as a sum of ramps for the entire window of simulation time. From the step response of $\hat{Y}_{ij}(s)$ ($\hat{Y}_{ij}(s)/s$), it is straightforward to write an analytical time domain expression for the unit ramp response. Now, each coupling current contribution at the *ith* driver node due to the *jth* port will be a sum of a number of scaled ramp responses. The total coupling current is just the sum of the individual contributions from each of the other ports.

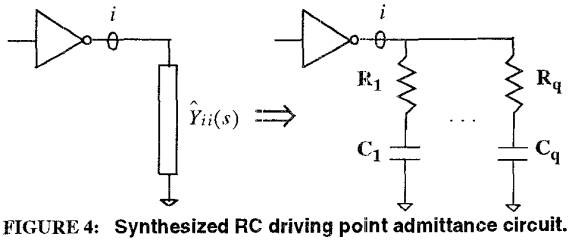

FIGURE 4: Synthesized RC driving point admittance circuit.

To calculate the voltages at the fanout points we use the transfer functions relating the waveforms at the driver nodes to the waveforms at the fanout points. For example, the transfer function of the *ith* fanout point and the *jth* driver node is $\hat{H}_{ij}(s)$. Assuming the driver voltages are treated as a sum of ramps, each $\hat{H}_{ij}(s)$ term is calculated as a sum of scaled ramp responses (as in the previous paragraph). By superposition, the total response at the fanout node is just the sum of all the individual responses.

The ith gate can be represented at the transistor-level with the following differential, nonlinear equations:

$$\mathbf{F}_i(\mathbf{x}'_i, \mathbf{x}_i, \mathbf{d}_i, \mathbf{u}_i) = \mathbf{0} \quad (3)$$

where $\boldsymbol{x}_i'$ is a vector containing derivatives of the unknown variables x_i with respect to time, $\boldsymbol{x}_i$ is a vector of the unknown variables in the circuit (usually node voltages), $\boldsymbol{d}_i$ is a vector containing the relaxed voltages at each of the other ports, and $\boldsymbol{u}_i$ is a vector of input sources to the gate. Equation (3) is solved specifically for the voltage waveform at the *ith* driver node.

When decoupled in this manner, macromodel passivity is not a factor, any transistor-level simulation engine (e.g. SPICE) can be used to solve for the gate voltages and currents as they interact with the interconnect ports. In fact, since the logic gates are partitioned into logic stages for timing analysis, most traditional timing simulation algorithms would not show much runtime advantage over SPICE for this problem, since these algorithms tend to demonstrate their greatest efficiency improvement for large designs.

We chose to solve (3) using TETA, a Transistor-Level Engine for Timing Analysis[15]. TETA uses Trapezoidal integration and successive chords to solve the differential, nonlinear equations. To speed up the current computation for each MOSFET in the gate, TETA employs a clever table lookup model. The advantage of TETA is that it provides a significant speedup over SPICE for small gate problems while providing comparable accuracy, and provides the ability for making further runtime vs. accuracy tradeoffs.

4.2.Cell-level Computation

When gate models are already available at a higher level of abstraction, we can use the Thevenin voltage model proposed in [12] to model the gate and interconnect interaction. Due to the nature of these gate models, the simulation algorithms must be chosen to conform with the worst-case delay calculation methodology outlined in Section 3.

Using the model in [12], the gate is modeled by a time-varying piecewise linear Thevenin voltage source in series with a fixed resistance R_d, as shown in Figure 5. The parameters t0, Δt1, Δt2, etc. are stored as a function of input transition time, t_{in}, and capacitance load, C_L, similar to empirical delay modeling. We briefly review this

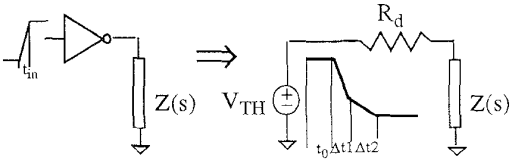

FIGURE 5: Gate driving a reduced order RC(L) load model.

model here, since unlike the transistor-level case, extensions to it are required to fit the proposed worst-case analysis methodology.

The model evaluation takes place using the effective capacitance (C_{eff}) principle. The value of C_{eff} is such that the average current drawn by C_{eff} is equal to the average current drawn by the actual load (refer to Figure 6). The steps involved in computing C_{eff} for a single line are as follows:

i) Assume initial guess for $C_{eff} = C_{total}$.

ii) Obtain the model parameters Δt1 and Δt2 (t0 is not required) from the empirical model, for $C_L=C_{eff}$.

iii) Calculate the current drawn by the load when driven by this Thevenin source in series with R_d, and integrate it over the time duration $\Delta t1+\Delta t2$ to obtain the charge, q_{load}.

iv) Compute q_{Ceff}, the charge transferred to C_{eff}, as a function of C_{eff}, and use the equation $q_{Ceff}=q_{load}$ to obtain a new value for C_{eff}.

v) Return to step (ii) and repeat (ii) to (iv) until C_{eff} converges.

vi) Compute final t0, Δt1, and Δt2 using the final C_{eff} value.

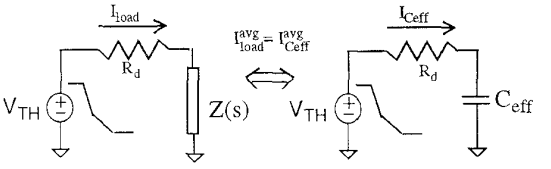

FIGURE 6: **Effective capacitance principle.**

Note that C_{eff} is only an intermediate value used to obtain the model parameters. Once the parameters of the Thevenin voltage are computed, the model is used to drive the *actual load* (or its reduced order model) to get the output waveforms.

With coupled interconnect, the iteration process that we employ involves decoupling the C_{eff} iterations for the aggressor and victim lines. The first C_{eff} iteration process is performed for each line assuming the other lines are quiet, implying that the Thevenin voltages at the other ports are shorted and the quiet lines are represented by R_d (the driver resistance). The initial guess for C_{effv} (C_{eff} for the victim line) is the sum of C_{total} (total capacitance to ground) for the victim line and all coupling capacitances connected to this victim. The parameters resulting from the first C_{eff} iteration process are sufficient to characterize victim and aggressor waveforms assuming all other lines are quiet.

When the victim is updated for coupling, the current delivered/absorbed by the Thevenin voltage of each aggressor is added to the load current. The "coupling" current is treated as another load for the victim, and the "coupling" charge transferred by each aggressor adds to the load charge. If the "coupling" charge is equated to the charge delivered to C_{eff} (note that equating the average current is the same as equating charge transferred), a modified C_{eff} equation is obtained for the victim. The equations for the C_{eff} iterations can now be derived similar to the single line switching case.

For ease of explanation, only a single ramp Thevenin voltage is considered here. Let Δt_v be the duration of the ramp for the victim gate. The charge transferred by the Thevenin voltage to the capacitance load C_{effv} during the time Δt_v is given by (4), where $p_v=-1/(R_{dv} \cdot C_{effv})$.

$$q_{Ceff} = Vdd\left(C_{effv} + \frac{C_{effv}}{\Delta t_v \cdot p_v}\cdot\left(1 - e^{p_v \Delta t_v}\right)\right) \quad (4)$$

The charge transferred to the interconnect load consists of two components, one due to the current drawn by $Y_{vv}(s)$, the driving point admittance at the output of the victim, and the other due to the current delivered (absorbed) by the aggressors. We refer to the first component as the "self-charge" q_s, and the second as the "coupled-charge" q_c. If $k_{vv}(j)$ is the jth residue and $p_{vv}(j)$ is the jth pole of $Y_{vv}(s)$ when expressed in pole-residue form, the self-charge can be expressed as

$$q_s = Vdd\left(-\sum_{j=1}^{q}\frac{k_{vv}(j)}{p_{vv}(j)} + \sum_{j=1}^{q}\frac{k_{vv}(j)}{\Delta t_v p_{vv}(j)^2}\left(1 - e^{p_{vv}(j)\Delta t_v}\right)\right) \quad (5)$$

where q is the order of the reduced order model.

The other component of the charge depends on the model parameters of the aggressor gates and the transadmittance $Y_{va}(s)$ for each aggressor.

$$q_c = Vdd\left(-\sum_{j=1}^{q}\frac{k_{va}(j)}{p_{va}(j)} + \sum_{j=1}^{q}\frac{k_{va}(j)}{\Delta t_a p_{va}(j)^2}\left(1 - e^{p_{va}(j)\Delta t_a}\right)\right) \quad (6)$$

In the case of multiple aggressors, the charge contributed by each is added to the coupled charge. The total charge transferred to the load by $V_{TH,v}$, is just the sum of q_s and q_c:

$$q_{load} = q_s + q_c \quad (7)$$

It follows that the effective capacitance iterations are performed by equating q_{load} obtained from (7) to q_{Ceff} obtained from (4). After one iteration is performed, we update the value of C_{effv}. If the updated effective capacitance of the victim is significantly close to the previously computed effective capacitance, we analyze and shift the aggressor Thevenin voltages, as described in Section 3. Otherwise, we carry out a new C_{eff} iteration for the aggressor gates.

A couple of points are worth mentioning here: First, even though we separate the aggressor and victim C_{eff} equations, the effect of the coupling capacitances is seen in each individual equation even if the other lines are not switching. This is because the $Y_{vv}(s)$ computation is done with the coupling capacitances in place. Thus, it appears as though we decouple the system, while in effect we only decouple the switching effect of the other lines. Second, the impact of coupling depends on the relative directions of switching. If two lines switch in opposite directions, the C_{eff} value for each line increases (and hence the delay increases, since delay is proportional to load size). This part is automatically taken care of in the C_{eff} iteration process from the sign of the "coupling" charge transferred from one line to the other. The sign would change depending on the direction of switching, and hence this component would either increase or decrease the charge absorbed by the load.

5. Proof of Convergence for Waveforms

5.1.Transistor-Level

Convergence of our waveform iteration methodology at the transistor-level can be guaranteed following the proofs and theory for waveform relaxation in [11], [16], and [17]. For a multi-port network, we can write the system of equations required to solve for the driving point waveforms as:

$$\mathbf{F}(\mathbf{x}',\mathbf{x},\mathbf{d},\mathbf{u}) = \mathbf{0} \quad (8)$$

Using the terminology in (3), $\boldsymbol{x}$ is a vector of *all* unknown state variables in the system, namely the node voltages. The vector $\boldsymbol{d}$ represents all driving point voltages, and $\boldsymbol{u}$ is a vector of input sources to the system.

A necessary condition for convergence of the waveform relaxation method [11] is that capacitors to ground be present at every decoupled node. In the case of our waveform iteration approach, every driver node will have at least one capacitor to ground.

Consider the *ith* subcircuit shown in Figure 3. This decoupled circuit can be described by the system of differential, nonlinear equations in (3) including the additional constraint:

$$\mathbf{x}(0) = \mathbf{x}_0 \quad (9)$$

Equation (9) states that the computed solution of the unknown variables at the start time is equal to the actual initial values of these variables. This condition is immediately satisfied as we are assuming that the system has been in steady state until stimuli are provided. Specifically, all currents in the circuit will be zero and the node voltages will be their dc values.

Next, we note a subtle difference between (3) and its counterpart in [11] by carefully scrutinizing the **d** vector (called the decoupling vector in [11]). Referring to Figure 3, we do not directly use the relaxed voltages at the other ports as done in [11]. Instead, we apply the relaxed voltages as voltage-controlled current-sources using precomputed Y-parameters:

$$i_{ij}(t) = \hat{y}_{ij}(t) \bullet v_j(t) \quad (10)$$

Recall from Section 3 that $\hat{y}_{ij}(t)$ is the approximate unit ramp response relating the voltage at port j to the current at port i. Since the $\hat{y}_{ij}(t)$ terms can be precomputed at discrete simulation points in time, we can treat them as constant terms. Therefore, it is apparent that our decoupling vector is a simple linear transformation of the vector of the relaxed voltages.

It follows that equation (3) is consistent with the waveform relaxation model described in [11]. This is because we still treat the voltages in $\boldsymbol{d}$ as constant expressions, which is precisely how [11] expects them to be handled. From a circuit perspective, the relaxed voltages in (3) and their counterparts in [11] will both appear in the right-hand side of the nodal equations that describe these circuits.

With this in mind, (3) and (9) are consistent with the waveform relaxation model defined in [11]. Each unknown driver node voltage is assigned to a subcircuit where that port voltage becomes a state variable. Further, all driving point voltages of the logic block can be construed as a state vector used in both (3) and (9). Since we have a methodology conforming to the waveform relaxation model of [11], the technique described in this paper will converge for any initial guess of the driving point voltages provided that their initial values are correct. Note that in terms of efficiency, only the port waveforms are required for this waveform relaxation process.

5.2.Cell-Level

Proof of convergence of this waveform iteration methodology at the cell-level is described in [18]. Note that at this level of abstraction the iterations are extremely efficient since the entire waveshapes for all time are captured in terms of the two Thevenin voltage parameters. Due to space considerations in this paper, however, a more complete treatment of this work could not be included.

6. Using Victim and Aggressor Waveforms to Optimize for Alignment of Aggressor Inputs

It is our objective to find the starting point, in time, of each aggressor input such that the delay of the victim is maximized.

6.1.Definitions

We will denote the victim waveform by **V(t)**. We define an *ag-*

gressor as any gate which is not the victim. The *i*th aggressor waveform will be denoted by $\mathbf{A_i(t)}$. A coupled scenario is depicted in Figure 7 for a two line example. We will use superscripts to indicate the waveform iteration number. For example, $\mathbf{V^1(t)}$ is the *first* victim waveform generated with a quiet aggressor line.

We define the *noise on the victim line at the kth waveform iteration* as the difference between $\mathbf{V^k(t)}$ and $\mathbf{V^1(t)}$, where $k \in \{1, 2, \ldots\}$. Furthermore, we denote the maximum noise at the *kth* iteration as $\mathbf{V^k_N}$. Finally, we let x be the fraction of the supply voltage, V_{dd}, where we want to maximize the delay of the victim. For example, if $x = 0.5$ and $V_{dd} = 3V$, then we seek to maximize the delay of the victim at 1.5V.

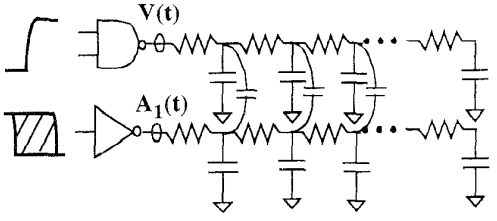

FIGURE 7: **Defining the *victim* and an *aggressor*.**

6.2.Description of the Optimal Alignment Strategy

To begin, we note that in practice it is reasonable to consider just the $\mathbf{V^2(t)}$ and $\mathbf{V^1(t)}$ waveforms for aligning the aggressor inputs. With this in mind, we use the time when the maximum noise between $\mathbf{V^2(t)}$ and $\mathbf{V^1(t)}$ occurs as an approximation to the time when the converged $\mathbf{V(t)}$ and $\mathbf{V^1(t)}$ attains its maximum noise. The steps to align the aggressor inputs follow that which was outlined in [2]:

1. We compute $\mathbf{V^2_N}$, the maximum difference between $\mathbf{V^2(t)}$ and $\mathbf{V^1(t)}$, separately *for each* aggressor (all other aggressors quiet). We denote the time when $\mathbf{V^2_N}$ occurs as t^1_N. Then, we determine when $\mathbf{V^1(t)} = x \cdot \mathrm{Vdd} - \mathbf{V^2_N}$ and designate this time as t^2_N. This is depicted in Figure 8 for the case of a falling victim waveform.

2. To align the noise such that $x \cdot \mathrm{Vdd} = \mathbf{V^1(t)} + \mathbf{V^2_N}$, we shift the input of the aggressor such that the maximum noise is shifted precisely by $t^2_N - t^1_N$ (in the case of Figure 8, this will result in a shift to the left). In the example of Figure 8, the waveshape of $\mathbf{A_1}^1\mathbf{(t)}$ remains constant irrespective of the start time of the aggressor's input, implying that the maximum noise induced by $\mathbf{A_1(t)}$ will remain the same (in a purely linear environment). As such, we directly shift the aggressor input by $t^2_N - t^1_N$.

A logical question to ask at this point is: *How does computing the maximum noise relate to computing the maximum delay?* In [2] it was shown that the maximum noise between $\mathbf{V^2(t)}$ and $\mathbf{V^1(t)}$ will correspond to the point in time where the delay between $\mathbf{V^2(t)}$ and the input to the victim is maximized, *if* $\mathbf{V^1(t)}$ is monotonic in the predominant region of switching. (A proof is provided in Appendix A).

Shifting the maximum noise by Δt will directly correspond to shifting the aggressor input by Δt because the aggressor waveform at the driving point is independent of the victim in the first iteration.

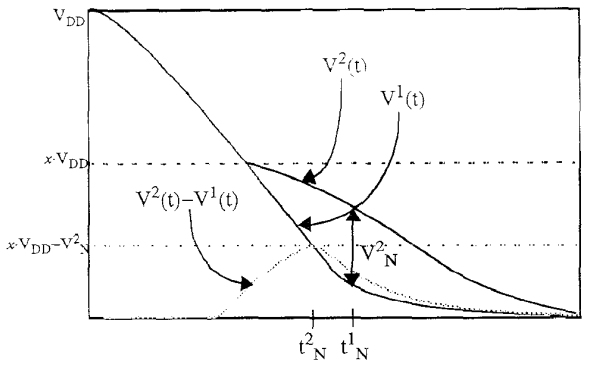

FIGURE 8: **Aligning maximum noise for worst-case delay.**

In otherwords, the waveshape of $\mathbf{A_1}^1\mathbf{(t)}$ remains the same (only displaced in time).

7. Results

7.1.Waveform Convergence

At the transistor-level, our algorithm was implemented in C using the transistor-level solver employed by TETA[15]. To demonstrate our methodology we used two simultaneously switching gates driving coupled bus lines from a 0.25μ commercial microprocessor design. The coupled RC load was modeled by distributed RC elements with 1056 resistors, 6461 capacitors to ground, and 667 coupling capacitors. A fourth order model was used to evaluate the interconnect. Interconnect ports were driven by a 2-input NAND and an inverter, as illustrated in Figure 7.

Figure 9 shows the intermediate and final waveforms of both the NAND and inverter at the driving point. Note that the first waveform of the NAND gate, designated "Nand Output without noise" represents the output assuming the inverter line is quiet. The driving point waveform of the NAND gate converged in 3 iterations. The inverter's waveform converged immediately. It can be inferred from Figure 9 that the driving strength of the inverter is greater than that of the NAND gate which, in part, explains why the NAND gate is influenced by the switching of the inverter.

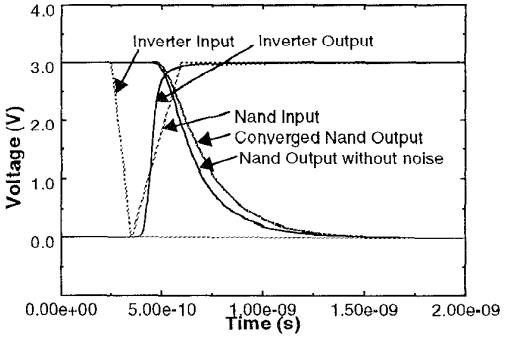

FIGURE 9: **Convergence at the gate driving points.**

We have observed that the converged waveform is usually close to the second iteration waveform. As a result, and as mentioned, only two iterations are typically necessary in practice. It is impor-

tant to realize that the traditional method of handling coupling capacitors via simultaneous switching (using *twice* the total coupling capacitance) will render an overly pessimistic driving point waveform for the NAND gate response in this example. Since TETA represents a circuit simulation, we did not compare the results with SPICE.

At the cell-level, however, we compared the coupled Ceff-model results with HSPICE simulation results. Figure 10 shows the noise and output waveforms for the two long coupled line example described previously. In this instance, the two bus lines were driven by inverting buffers. The noise and output waveforms at the far end (load point) obtained from our model are compared with those from HSPICE. As seen from Figure 10, the noise as well as the composite output waveform obtained from our model match the HSPICE waveforms very well. Only 3 C_{eff} iterations for each line were required for this example to obtain this level of accuracy.

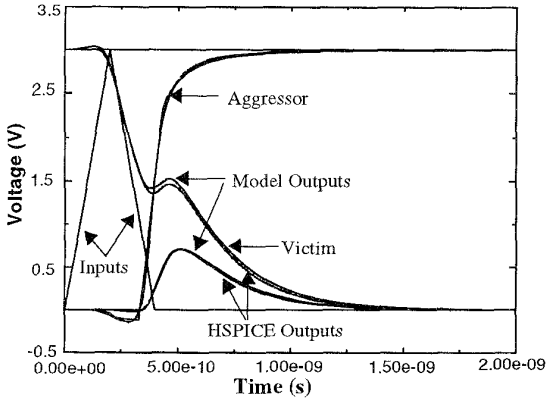

FIGURE 10: Noise and output waveform for two long coupled RC lines.

7.2.Optimal Aggressor Alignment Results

Using the two oppositely switching gates for the coupled bus-lines in Figure 7, we applied our optimization strategy at the transistor-level at both the driver node and a fanout node of the victim. In our experiments, the NAND gate played the victim while the inverter was the aggressor. The aggressor input had a fall-time of 100 ps, and we chose to maximize the 50% delay point[2]. In the results that follow we compare delays from our final optimized waveforms to the optimal waveforms obtained by trial and error (that is, sweeping the aggressor input over a wide range of switching times and gauging the worst-case delay).

In the first example, we maximize the delay at the victim's driving point. Here, we begin the aggressor input at 700 ps which culminates in the "Unshifted Victim Waveform" in Figure 11. The delay is computed to be 165 ps. Applying one iteration of our optimization strategy suggests that the aggressor input should be shifted left to 340 ps. The resulting waveform at the driver shown in Figure 11 ("Shifted Victim Waveform") has a delay of 215 ps (an increase of 30%). This agrees with the worst-case delay obtained by trial and error, hence this aggressor input alignment is correct.

Using the same coupled bus-lines and drivers, we applied the

[2]When the victim falls below 1.5V and does not rise above it again.

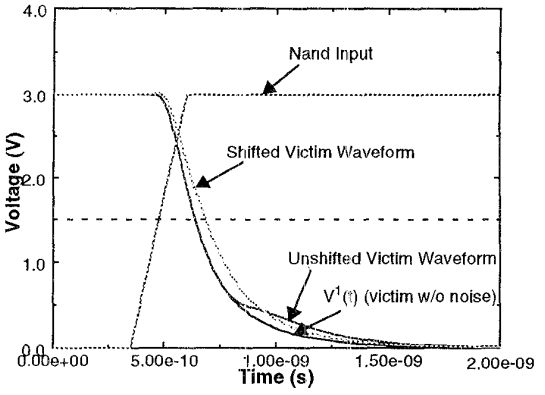

FIGURE 11: Determination of worst-case aggressor alignment at a driver node.

worst-case optimization strategy at a fanout node of the victim. The methodology is the same; we just have to remember to translate the first and second driving point waveforms to the fanout node of the victim using the transfer function parameters. Figure 12 illustrates the results. We begin with the aggressor input at 340 ps, which was optimal for the worst-case delay at the driving point. This results in a delay of 435 ps for the "Unshifted Victim Waveform". Following the steps of our methodology, we shift the aggressor input ahead by 470 ps to 810 ps. The resulting "Shifted Victim Waveform" produces a delay of 555 ps (an increase of 27.6%), which corresponds to the maximal delay that was calculated via exhaustive trial and error experiments.

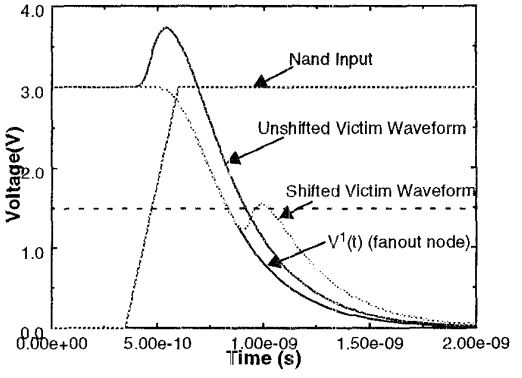

FIGURE 12: Determination of worst-case aggressor alignment at a fanout node.

Referring to Figure 12, note the impact coupling has on the delay. In Figure 11, the maximum delay was found by starting the aggressor at 340 ps. In Figure 12, the maximum delay was found by starting the aggressor at 810 ps. Observe that aggressor alignment for worst-case delay at the drivers does not correspond to aggressor alignment for worst-case delay at the fanout nodes. Still, we have illustrated that our algorithm easily handles optimization for worst-case alignment at either the driver or any other fanout node in the interconnect.

The strategy to align aggressor inputs exemplified at the transistor-level can also be used analogously at the cell-level.

8. Conclusions

Determining the worst-case alignment of fast aggressor signals for the maximum delay experienced by a victim gate is a crucial problem to solve in timing analysis. We have developed an interconnect-centric, transistor- and cell-level methodology based on waveform iterations that is suitable for solving this extremely difficult problem. We have shown that our methodology converges for both levels of abstraction. We have also illustrated its efficacy at the transistor-level in solving for the worst-case alignment of aggressor signals for industrial examples.

Appendix A: Proving that Worst-Case Delay Coincides with Maximum Noise

Here we refer the reader to Figure 13 (assuming the victim waveform is falling) for an example to follow throughout this proof. First we note that our goal has been to find t_N such that $t_N - t_I$ is a maximum. Here t_N is the time at which the x·Vdd point of the victim is maximized, and t_I is the x·Vdd point of the victim's input. t_D is just the x·Vdd point of $\mathbf{V^1(t)}$. This problem is clearly equivalent to maximizing $t_N - t_D$ since t_I and t_D are fixed over all waveform iterations, and t_D is always greater than t_I.

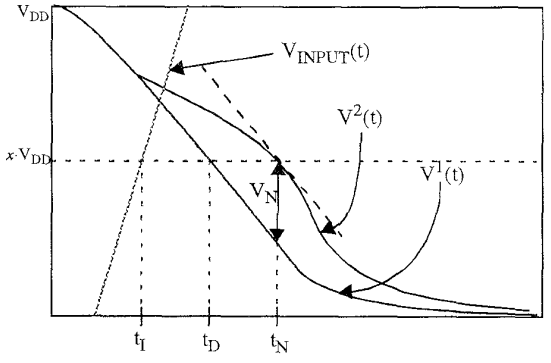

FIGURE 13: Coinciding maximum noise with maximum delay.

We want to show that $t_N - t_D$ is a maximum when the absolute value of $\mathbf{V^2(t_N)} - \mathbf{V^1(t_N)}$ is a maximum *and* $\mathbf{V^2(t_N)} = \mathbf{V^1(t_D)} = x$·Vdd.

Let $\mathbf{V_N} = |\mathbf{V^2(t_N)} - \mathbf{V^1(t_N)}|$. Now assume that there exists some point, t_1, for which $t_1 \geq t_N$ such that $\mathbf{V^2(t_1)} = \mathbf{V^1(t_D)}$. We assume that $\mathbf{V^1(t)}$ is strictly monotonic for any reasonable selection of x. This assumption is valid since all aggressor lines are quiet. Then it is clear that $|\mathbf{V^2(t_1)} - \mathbf{V^1(t_1)}| \geq |\mathbf{V^2(t_N)} - \mathbf{V^1(t_N)}|$ and is equal only when $t_1 = t_N$. This implies that $|\mathbf{V^2(t_1)} - \mathbf{V^1(t_1)}| \geq \mathbf{V_N}$.

However by definition, $\mathbf{V_N}$ is the maximum possible difference (in absolute value terms), so we must write $|\mathbf{V^2(t_1)} - \mathbf{V^1(t_1)}| = \mathbf{V_N}$. Hence, it follows that $t_1 = t_N$. In this manner, the worst-case delay is maximized.

Bibliography

[1]Semiconductor Industry Association (SIA), "The National Technology Roadmap for Semiconductors," 1994.

[2]Florentin Dartu and Lawrence T. Pileggi, "Calculating Worst-Case Gate Delays Due to Dominant Capacitance Coupling," Proceedings of the 34th ACM/IEEE Design Automation Conference, June 1997.

[3]G. Yee, R. Chandra, V. Ganesan and C. Sechen, "Wire Delay in the Presence of Crosstalk," Proceedings of TAU 97, the IEEE meeting on Timing Issues in Digital Systems, December 1997.

[4]L. Pileggi, "Coping with RC(L) Interconnect Induced Headaches," Proceedings of the International Conference on Computer-Aided Design, 1995.

[5]L.T. Pillage and R.A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," IEEE Trans. Computer-Aided Design, April 1990.

[6]Altan Odabasioglu, Mustafa Celik and Lawrence T. Pileggi, "PRIMA — Passive Reduced Order Interconnect Macromodeling Algorithm," Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 1997.

[7]Peter Feldmann and Roland W. Freund, "Efficient Linear Circuit Analysis by Pade Approximation via the Lanczos Process," IEEE Trans. Computer-Aided Design, May 1995.

[8]Kevin J. Kerns, Ivan L. Wemple and Andrew T. Yang, "Stable and Efficient Reduction of Substrate Model Networks Using Congruence Transforms," Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 1995.

[9]L. M. Silveira, M. Kamon and J. White, "Efficient reduced-order modeling of frequency-dependent coupling inductances associated with 3-D interconnect structures," IEEE/ACM Proc. DAC, pp. 376-380, June 1995.

[10]Frank Liu, Lawrence Pileggi and Andrzej Strojwas, "ftd: An Exact Frequency to Time Domain Conversion for Reduced Order RLC Interconnect Models," Proceedings of the Design Automation Conference, 1998.

[11]E.Lelarasmee, A.E.Ruehli and A.L. Sangiovanni-Vincentelli, "The Waveform Relaxation Method for Time-domain Analysis of Large Scale Integrated Circuits and Systems," IEEE Trans. Computer-Aided Design, July 1982.

[12]R. Arunachalam, F. Dartu and L.T. Pileggi, "CMOS Gate Delay Models for General RLC Loading," Proceedings of the International Conference of Computer-Aided Design, November 1997.

[13]M.E. Van Valkenburg, "Introduction to Modern Network Synthesis," John Wiley & Sons, Inc., 1960.

[14]Florentin Dartu, "Gate- and Transistor-Level Waveform Calculation for Timing Analysis," Ph.D. Dissertation, Carnegie Mellon University, August 1997.

[15]Florentin Dartu and Lawrence T. Pileggi, "TETA: Transistor-Level Engine for Timing Analysis," Proceedings of the 35th ACM/IEEE Design Automation Conference, June 1998.

[16]G.D. Gristede, A.E. Ruehli and C.A. Zukowski, "Convergence Properties of Waveform Relaxation Circuit Simulation Methods," Technical Report, IBM T.J. Watson Research Center, 1996.

[17]U. Miekkala, O. Nevanlinna and A.E. Ruehli, "Convergence and Circuit Partitioning Aspects for Waveform Relaxation," Technical Report, IBM T.J. Watson Research Center, 1990.

[18]Ravishankar Arunachalam, "CMOS Gate-Delay Models for Coupled RC(L) Interconnect Loads," M.S. Thesis, Carnegie Mellon University, May 1998.