A robust cell-level crosstalk delay change analysis

Igor Keller, Ken Tseng and Nishath Verghese
Cadence Design Systems
555 River Oaks Pkwy
San Jose, California, USA
{ikeller,kentseng,nashv}@cadence.com

ABSTRACT

In this paper we present a robust and efficient methodology for crosstalk-induced delay change analysis for ASIC design styles. The approach employs optimization methods to search for worst aggressor alignment, and it computes crosstalk induced delay change on each stage considering an impact on downstream logic. Computational efficiency is achieved using pre-characterized current models for drivers and compact macromodels for interconnect. The proposed methodology has been implemented in a commercial noise analysis tool. Experimental results obtained on industrial designs demonstate high accuracy and reduced pessimism of the proposed methodology.

1. INTRODUCTION

Recent advances in process technology scales the aspect ratio of wires to be taller and thinner to control wire resistance. A side effect of this scaling is that coupling capacitance between wires becomes the dominant portion of the total wire capacitance [1, 2]. At the same time the signal transition times become faster resulting in stronger aggressors on adjacent victim wires [2]. Another side effect of process advancement is the faster scaling of cell internal delay than the interconnect delay, making accurate analysis of interconnect delay important.

Effects of crosstalk on delay grow linearly, if not faster, with the latest technology advances due to (i) increase in coupling-to-total capacitance ratio, (ii) decrease in supply voltage resulting in a reduction of gate overdrive (iii) shortening of clock period causing transition waveforms to play a bigger role, and (iv) tighter margins requiring more accurate timing analysis and less overestimation of delay.

The problem of delay calculation in the presence of crosstalk can be formulated as finding the worst-case delay among all possible alignments and aggressor waveforms. However, the computation of delay in the presence of noise is a challenging problem due to the following factors: (i) delay is sensitive to aggressor/victim alignment, (ii) linear model for a switching driver and effective capacitance principle may become inadequate [2], and (iii) waveform becomes irregular in the presence of noise, making the conventional metric of delay measurement non-robust.

Due to the high sensitivity of delay on alignment and aggressor timing window constraints, a search for the worst-case (WC) alignment may be approached using constrained nonlinear optimization techniques. However, optimization in a multi-dimensional space of aggressor alignments with each iteration requiring simulation of a nonlinear circuit can be prohibitively expensive. The problem is further complicated by the unique waveform response of each receiver of the victim net to the same input transition, such that a WC alignment for one receiver may not be the same for another.

Another challenge with crosstalk induced delay calculation is distortion of transition waveforms. The distorted waveform often deviates from waveforms used in delay characterization causing inaccuracy in arrival times in the downstream logic cone. If the crosstalk is severe, the victim waveform may even become non-monotonic (bumpy), the effect of which is not properly modeled in existing gate delay systems.

Figure 1 clearly shows the problem associated with aggressor alignment and delay measurement based on a predefined waveform crossing threshold. The waveforms were computed in Spice using an exhaustive sweep of alignment parameters on a victim net in a 0.13 micron industrial design. The worst-case delay pushout when measured at the V_{ref} = 50% Vdd crossing is 724ps. This happens when the rising waveform has a falling noise bump that barely touches the threshold and then rises again. However, this bump on the waveform causes almost no delay change in the gate driven by the victim net, where the transition has only a slight bump at the end. It is evident from the figure that the V_{ref} crossing of the receiver output waveform occurs prior to that of the receiver input waveform on which the delay change is usually measured, causing the aggressor alignment to be sub-optimal. By monitoring the victim net's receiver output response during the aggressor alignment sweep, we determined that the actual worst-case alignment should be much earlier, with the resulting delay pushout being 537ps, which is 35% less pessimistic. Also, we observe that the receiver output waveform is smoothened owing to the fact that gates in digital design are low-pass filters.

In certain cases of high coupling/total capacitance ratio of the victim net, the response on receiver output may become non-motononic as well. Such cases are similar to amplification of propagated noise, and such an event should be caught and reported to the user as an indication of a potential noise and, possibly, delay problem.

Recently proposed methodologies for delay calculation in the presence of crosstalk include using a heuristic-based Miller factor [3] and analyzing coupled interconnect using linear(ized) and nonlinear models for drivers [2, 4, 5]. Suggested alignment methods for victim and aggressors transitions are based on the superposition principle along with usage of noise pulse width and height [2, 4, 5]. The usage of pre-characterized 4-D look-up tables representing alignment as a function of

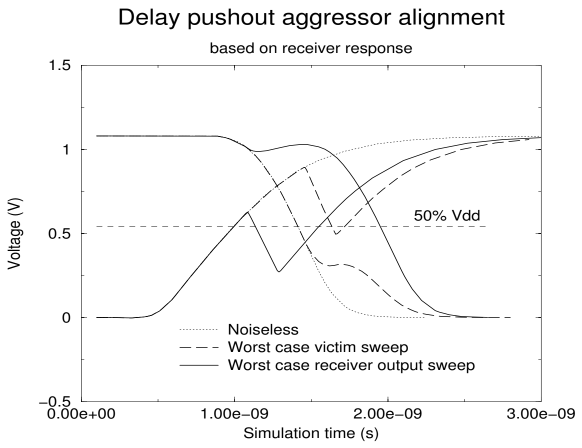

Figure 1: Responses on recevier input and output suggest different alignments and delays on the victim

nominal slew rate, height and width of a noise glitch on the victim net and output load of receiving gate has been proposed in [9]. Recent papers [10, 11] describe current-based models for gates that were successfully used in noiseless delay and noise analysis, respectively.

In this paper we suggest a novel approach to crosstalk delay change analysis which allows for an accurate and robust accounting of crosstalk effects on stage delay even in the presence of strong nonlinearity, high coupling/total capacitance ratios and irregular waveforms. The methodology is based on (i) using an accurate nonlinear current model for the gate, (ii) a smart partition of linear and nonlinear components of the stage allowing efficient search for WC alignment using methods of constrained optimization, (iii) altering the conventional notion of "delay" using ROP (receiver-output probing) metric, and (iv) using a detailed waveform (PWL) description instead of a linear saturated ramp.

2. STATEMENT OF THE PROBLEM

The crosstalk delay change calculation described in the curent paper is a part of the outer timing window iteration loop which also includes calculation of the noiseless stage delay and Static Timing Analysis (STA). These iterations are required for convergence of timing windows on each stage of a circuit, since they are affected by crosstalk, which in turn depends on timing windows of the neighboring nets.

In the current work we consider calculation of the delay change at a given iteration of the outer loop, using timing windows from the previous iteration.

First, we define a stage as a subcircuit of the design consisting of a victim net, its aggressor nets, and their corresponding drivers (see Figure 2). In the present paper we define a *driver* as the last Channel-Connected Component (CCC) of the driving cell. The practical assumptions used in industry is that for a multi-CCC cell, effects of crosstalk on the cell *internal* delay, defined as a delay from an input of the cell to an input of the last CCC, can be neglected. Since the present paper is devoted to computation of delay *change* on a stage due to crosstalk, one needs only consider the last CCC of the driving cell. All single-CCC cells (e.g. inverter or nand) are drivers by definition.

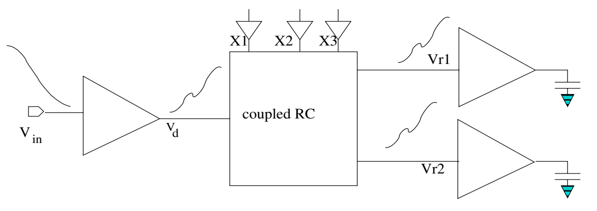

Figure 2: A stage includes coupled interconnect and drivers of the victim and aggressor nets.

It is known that the effect of a switching aggressor on delay on the victim is roughly proportional to the slew rate of transition on the aggressor. The slew rate on the aggressor, in turn, depends on the strength of drivers holding the aggressor's neighbors (second-order aggressors). This necessitates accounting for the second-order aggressor, that are also included into the stage but assumed to be "quiet" and therefore grounded using an appropriate resistance.

Here we consider the case when the victim and aggressor nets each have a single driver. The case of multiple drivers can be handled similarly.

We denote a vector of voltages on all interface nodes by X and enumerate them such that $X_0, X_1, ..., X_N$ denote voltages on the victim and aggressor driver outputs, respectively, and $X_{N+1}, ..., X_{N+M}$ - receiver inputs, where N, M are the number of aggressors and receivers, respectively (see example on Figure 2).

Let $J[V(t)]$ denote a functional over a space of transition waveforms used as a transition time metric associating a number (transition time) to a transition waveform. The conventional transition time metric, denoted by J_{ref}, is defined as the time when transition crosses V_{ref} (typically 50%Vdd).

We define a *nominal* output transition, $\bar{X}_{j+N}(t)$ as a rising transition occuring on an input of $j-$th receiver in the presence of "quiet" aggressors. Similarly, we define the *noisy* output transition, $\tilde{X}_{j+N}(t)$ as a transition probed on the input of the $j-$th receiver in the presence of switching aggressors.

We further define the *crosstalk delay change* on the $j-$th stage's output as the difference in transition times as measured using some functional J on *noisy* and *nominal* transitions on the stage's output caused by the same transition on one of the stage's inputs:

$$D_j = J[\tilde{X}_{j+N}(t)] - J[\bar{X}_{j+N}(t)]. \quad (1)$$

We assume that transitions on the victim and aggressor nets can occur within corresponding switching windows calculated during STA on the previous iteration of the timing windows loop. Let $W_k = [\tau_{ek}, \tau_{lk}], k = 0, ..., N$ denote switching windows (SW) on the victim and aggressor nets, with τ_{ek}, τ_{lk} the earliest and latest possible transition times with respect to the victim's clock. If clocks of an aggressor and the victim net are asynchronous, the aggressor's switching window is set to be infinitely wide. We then define *alignment vector* $\boldsymbol{\tau} = \{\tau_1, ..., \tau_N\}$ as the vector of aggressor transition times.

The problem of calculation of the WC crosstalk delay change can be formulated as determining the maximal delay change (either positive or negative) between a given input and output of the stage in the presence of switching aggressors, each occuring within the corresponding switching window: $\boldsymbol{\tau} \in \mathbf{W}$, where $\mathbf{W}$ is an N-dimensional cube spanning the switching windows.

3. PROPOSED METHODOLOGY

In the next five subsections we describe the major components of the proposed methodology of finding the WC delay change on a stage. First in (3.1) we describe a precharacterized current model (CM). Then, in (3.2) we explain how a non-linear circuit can be partitioned into linear and non-linear parts in order to efficiently calculate transitions on the stage output in the presence of crosstalk for multiple alignments between victim and aggressors. Last three subsections describe the flow and provide details of the proposed algorithm of finding WC alignment.

3.1 ViVo current model for gate

Each gate in the standard cell library is pre-characterized for a set of current models, called a ViVo (Voltage in, voltage out) model, generated for each interface CCC of the gate.

Since the characteristic relaxation time of devices in a CCC is much smaller than CCC's input or output transition slew rates, the current drawn by a gate during switching can be well represented by a voltage-controlled current source, which is a function of instantaneous voltages on the input and output and their time-derivatives:

$$I_d = I(V_i, V_o,) + C_i(V_i, V_o,)\dot{V_i} + C_o(V_i, V_o,)\dot{V_o}; \quad (2)$$

The coefficients of the two last terms in (2) describe current charging an effective capacitance between input and output pins (Miller effect), and an effective ground capacitance on the output pin, respectively. The corresponding Miller and ground capacitors are related to C_i, C_o as $C_M = C_i$, $Cg = C_o + C_i$.

Although the coefficients C_i, C_o are in general nonlinear functions of V_i, V_o a simplified version of (2) with constant C_i, C_o is mostly useful in practice.

An efficient 2-D lookup table is used to store values of $I(V_i, V_o)$ in (2).

Since the current table from ViVo is a function of instantaneous voltages on input and output and not dependent on transition history, its generation is done through a series of DC simulations using Spice. Each such simulation is performed for a pair of constant voltages on input/output pins, and generates an entry in the 2-D current table.

The capacitaces C_M and C_g are found through a series of transient simulations with voltage transtions applied on input and output nodes, during which a current through the output node is measured.

The ViVo CM is created per pair of input and output pins of the CCC and it models current drawn by the output pin of the CCC for various voltage values on the input and output pins.

Sensitization for DC simulations is determined from a logic function extracted from the CCC using binary-decision diagrams (BDD).

Generation of a complete set of ViVo CMs for a gate is very fast, taking less than 1 sec for a cell.

As part of the characterization carried out for multi-CCC cells, in addition to the ViVo CMs we characterize for the slew on the input of the last CCC as a function of slews on the cell's inputs.

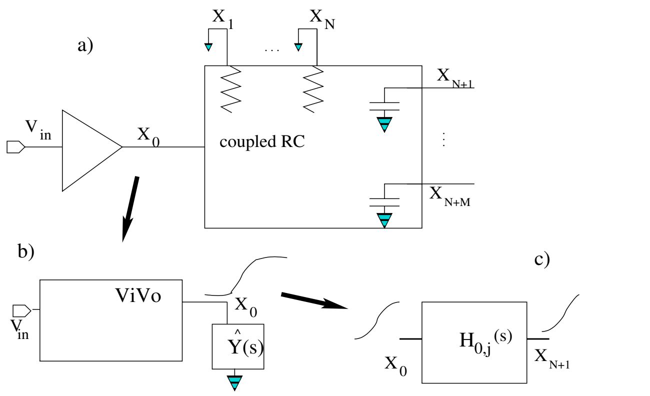

Figure 3: Calculation of nominal (noiseless) delay on a modified stage: victim driver is replaced with a current model, aggressor drivers - with holding resistors, and the linear portion of network - with a reduced-order macromodel.

3.2 VivoSim: dedicated simulation engine

Here we describe an efficient dedicated simulation engine, referred to below as VivoSim, for computation of noisy and noiseless transitions on the outputs of a stage. It is extensively used in the inner optimization loop described in (3.4) where the WC alignment is determined.

Computational efficiency is achieved via identification of an alignment-independent component of the solution and its usage throughout multiple alignment iterations. In addition, due to a specific form of the circuit model approximating the original stage, an efficient implementation of the dedicated simulation engine is possible allowing run times up to two orders of magnitude faster compared to Spice simulation with minor loss of accuracy.

We start from the MNA formulation for the stage typically consisting of large distributed RC-interconnect and a nonlinear elements (victim driver):

$$\mathbf{C}\dot{\mathbf{X}} + \mathbf{G}\mathbf{X} = \mathbf{B}\mathbf{U}, \quad (3)$$

Here C, G are capacitance and conductance matrices, and the excitation vector $\mathbf{U} = \{I_{drv}, U_1, ..., U_N, 0, ..., 0\}$ includes nonlinear driver current, I_{drv}, which is a function of voltages on its switching input and output, and voltage sources from Thevenin models of corresponding aggessor (see 3.3). We use a ViVo CM described in the previous subsection to model the current drawn by the victim driver.

The linear part of the circuit, described by the terms in left-hand side of Eq.(3), is modeled using a reduced-order macromodel. It is computed from the original RC-interconnect using a Krylov subspace method resulting in $Y_{k,0}(s), H_{j,k}(s), k = 0, ..., N, j = 1, ..., M$ that describe, respectively, admittances and transfer functions. The Krylov method [6] uses a projection of the original state space onto a state space of much lower dimension, while matching the frequency response to that of the original interconnect up to a frequency limit of 10^{10}Hz.

The reduced macromodel of interconnect enables an efficient and accurate propagation of responses over the linear portion of the network, and accounts for crosstalk effects of victim and aggressor wires.

In order to compute a nominal transition on the stage outputs each, aggressor driver is replaced with a holding resisitance while the second-order aggressors are decoupled (Figure 3a). As our experiments show, inclusion of wire resistances as well as modeling finite impedance of the aggressor net drivers is crucial for achieving a good accuracy in terms of stage delay. At the same time we found that second-order aggressors have much smaller effect on the nominal transition, and can be decoupled.

The calculation of nominal transitions on the stage outputs is performed in two steps. In the first step a driver output response, $\bar{X}_0$ is computed using ViVo current model for the victim driver and a 1-port reduced-order model of the load, $\hat{Y}$ (Figure 3b). The corresponing KCL equation for the driver output node, from which the nominal response is found using a numerical integration, reads:

$$I_0(t, X_0(t)) = \mathcal{L}^{-1}\{\hat{Y}\mathcal{L}[X_0(t)]\}. \tag{4}$$

Here $x_0(s) = \mathcal{L}[X_0(t)]$ and $\mathcal{L}[\cdot]$ is the Laplace transform. The load $\hat{Y}$ is a Padé approximation of the driving-point admittance $Y_{0,0}$.

In the second step, using the voltage substitution thorem [7] the driver is replaced with a voltage source modeling the driver output transition (see Figure 3c) and the receiver responses are found using the transfer functions $H_{0,k}$ as:

$$X_{N+k}(t) = \mathcal{L}^{-1}[H_{k,0}x_0(s)], \tag{5}$$

The computation of time-domain responses (5) is done using efficient recursive convolution [8].

Computation of noisy transitions on the stage outputs is performed on a modified stage circuit shown in Figure 4a. Using the voltage substitution theorem the aggressor drivers of the original stage are replaced with the voltage sources $X_k^*(t)$ that are pre-calculated using nonlinear driver models (see details in (3.3)).

The noisy transition are computed in the following three steps. In the first step we compute the alignment-independent components of the algorithm: current responses and voltage responses to voltage transitions on each aggressor driver output.

Replacing $k-$th aggressor driver with corresponding voltage source $X_k^*(t)$ and short-circuiting all other drivers we find

$$I_{0,k}(t) = \mathcal{L}^-1\{Y_{0,k}\mathcal{L}[X_k^*(t)]\}, \tag{6}$$
$$V_{j,k}(t) = \mathcal{L}^{-1}\{H_{k,k}\mathcal{L}[X_k^*(t)]\}. \tag{7}$$

Here $I_{0,k}$ is short-circuit current response at the victim driver output, and $V_{j,k}$ is open-circuit voltage response at the $j-$th receiver input due to k-th aggressor transition.

In the second and third steps we calculate noisy driver output and receiver input transitions, respectively, using the precomputed $I_{0,k}, V_{j,k}$. The calculation of noisy transitions is the alignment-dependent part of the algorithm, and is done for each alignment vector $\boldsymbol{\tau}$.

The driver output noisy transition is computed from a circuit shown in Figure 4b which is obained from the original one using a corresponding ViVo CM for the victim driver and replacing the linear part with its Norton equivalent circuit. The Norton equivalent circuit consists of the reduced-order model of load, $\hat{Y}$ and appropriately aligned set of current responses $I_{0,k}$:

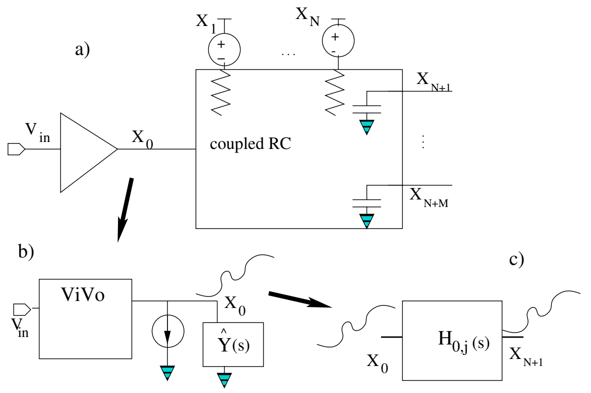

Figure 4: Calculation of noisy delay on a modified stage: victim driver is replaced with a current model, aggressor drivers - with voltage sources, and the linear portion of network - with a reduced-order macromodel.

$$I_0(t, X_0(t)) - \mathcal{L}^{-1}\{\hat{Y}\mathcal{L}[X_0(t)]\} = \sum_{k=1}^{N} I_{0,k}(t - \tau_k). \tag{8}$$

In the third step (see Figure 4c) the noisy transitions at receivers are calculated using the transfer functions $H_{j,0}$ and the pre-calculated (in stage one) and apppropriately aligned voltage responses:

$$X_{N+j} = \mathcal{L}^{-1}\{H_{j,0}\mathcal{L}[X_0]\} + \sum_{k=1}^{N} V_{j,k}(t - \tau_k), \tag{9}$$

As our experiments show, the reduced-order model for the load, $\hat{Y}$, needs to have just a few Padé terms to adequately approximate the load. In most cases only 1 Padé term is sufficient for the error to be within 2-3% of Spice. In this case the load becomes a Π-load. In a few cases where 1 pole is not sufficient, more Padé terms are preserved (up to 4).

Note that the described algorithm requires that $I_k(t), V_{j,k}(t)$ be computed once for a stage. Once they are computed, calculation of a noisy transition on the stage's output for a given alignement vector τ required one integration of a small set of nonlinear equations and one recursive convolution.

3.3 PWL transitions of aggressors

In order to calculate a delay change on a victim net, fast transitions on all its aggressor nets have be to known. Therefore the first step of the proposed flow is calculation of fast transitions on all nets that can be aggressors. In order to ensure conservative results the fast transitions are computed on each net and propagated to next stages during a initial breadth-first traversal of a design. On each net several driver output transitions are computed, one for each input pin of the last CCC of the driver. The calculation is done using the same method as described in the previous section for computing nominal response on the driver output. For single-CCC gates stimulus transition is created using a PWL transition waveform propagated from the previous stage. For a multi-CCC gates we use a pre-characterized 1-D slew propagation table in order to generate a transition stimulus on the input of the last CCC.

The calculated transition is stored compressed in the form of a 8-piece PWL wave on each gate output of the design. An efficient compression scheme allows us to store one such PWL wave in just 8 bytes.

In the case where slews from commercial STA tool are used at inputs to each driver and no propagation to the next stage is done, calculation of fast transitions on all nets is very fast even for big designs since it requires only two (one for rising and one for falling transitions) nonlinear simulations on the simple circuit shown in Figure 3b. For example, for a 500K-gate design the calculation of fast PWL transitions completes in less than 10min on a 2GHz Linux machine.

3.4 Delay change calculation

The proposed methodology assumes that the analysis is done in a topological order and, therefore waveforms on the inputs of all drivers are known.

Since timing windows on aggressor nets are taken from previous iteration of the outer STA convergence loop, the delay change calculation on a given stage can be viewed as independent on other stages. The delay change, as defined in Eq.(1) using a transition time metric J_{ref}, can be found using noisy and nominal transitions calculated on each output of a stage using the method described in detail in (3.2).

As explained in the Introduction, the metric J_{ref} used today in delay calculators, has its setbacks. In this paper we suggest a different metric, referred to below as ROP (Receiver-Output Probing) metric, J_{rop}, which is based on propagation of the noisy transition to the receiver output:

$$J_{rop}[X_{N+j}(t)] \equiv J_{ref}[Z_j(t)]. \tag{10}$$

Here $Z_j(t)$ denotes a transition on $j-$th receiver's output caused by a transition on its input, $X_{N+j}(t)$.

The ROP metric has the following advantages over the conventional metric: (i) it makes the noisy waveform more regular since the receiver gate is a low-pass filter, and (ii) it accounts for the impact of actual waveform on delay of the receiver, and therefore improves the accuracy of delay change measured on this net.

The computations done on each alignment iteration includes two numerical simulations of a small nonlinear circuit (each for victim driver and each of its receivers) and one linear analytical simulation (propagation of noisy transition from driver output to receivers).

As explained earlier, application of the conventional transition time metric J_{ref} to a non-monotonic waveform constitutes a cliff in the conventional methodology, which is not robust because small variations in parameters of the circuit (e.g. driver size, supply voltage, etc) or aggressor alignment is likely to result in a big change in transition times for downstream stages. The proposed ROP delay metric lacks such a setback of the traditional methodology. Also, it helps to uncover "marginal" violations such as a glitch on top of a transition tail, that are not caught by either conventional delay or noise analyses.

In addition, search for the WC delay change is done using techniques of constrained optimization which ensures that conservative results are found, something that is not guaranteed with any heuristic-based alignment methodology.

The described separation of alignment-independent components of the problem from those changing inside the alignment loop drastically reduces computational complexity of the proposed methodology. Owing to nested optimization loops in the proposed methodology, it is critical to employ an efficient computation framework for the analysis using fast similation of the circuit comprising the nonlinear current souce of the first/last CCC of the gates and their Padé-approximated loads. Furthermore, efficient computational techniques for reduced-order modeling of the interconnect must be employed.

The proposed flow of delay change calculation on a given stage is summarized in Figure 5. Note that it does not include the calculation of fast transitions on each aggressor's driver output which is done in a separate traversal of the design.

3.5 Constrained optimization of WC alignment

The innermost loop of the presented algorithm iterates over aggressor alignments in a search of the WC delay change based on the ROP metric. As noted before, this is done separately for each receiver, since the type of receiver and its load affect the WC alignment and delay change.

Since straightforward optimization in the $N-$dimensional sub-space of aggressor alignments in the presence of timing constraints (switching windows) is not feasible, we use the following greedy algorithm consisting of two steps.

In the first step optimization is performed for a single variable defined as the average of the peak times ζ_k of each aggressor's voltage response, defined relative to the nominal transition time on the receiver input: $\zeta = \frac{1}{N}\sum_{k=1}^{N}\zeta_k$.

In each iteration of the optimization loop we first determine the so-called *suggested* averaged alignment, $\bar{\zeta}^{(i)}$ with which we try to satisfy timing constraints. If the voltage response from an aggressor cannot be peak-aligned with $\bar{\zeta}^{(i)}$, it is positioned as close as possible to it, while satisfying constraints. This means that the transition on the aggressor occurs on the corresponding boundary of its SW. This results in the so-called *actual* averaged alignment $\zeta^{(i)}$ which may be different from the *suggested* alignment due to timing constraints.

After a delay change computed on $i-$th alignment iteration using the method described in the previous section, a new *suggested* alignment, $\bar{\zeta}^{(i+1)}$, is found by searching for the maximum of a quadratic function interpolating the worst 3 *actual* alignment points and their corresponding delay changes. To begin the process, the first three value of ζ are chosen heuristically using the nominal transition and pre-calculated voltage and current responses.

Step 1 of the optimization process stops when one of the following events occur: (1) the process converges in terms of delay change, (2) the delay change values are oscillating, (3) three worst points lie on a straight line, or (4) the maximum number of iterations is exceeded.

Case 3 indicates a severe delay and noise problem on the net: the noise glitch induced by aggressors after the victim has fully transitioned propagates to the receiver output where it crosses V_{ref}.

If the transition on the receiver output correposonding to the WC alignment is non-monotonic, a warning is issued as it might indicate a potential functional failure or a high inaccuracy of the delay analysis. Consider an example of the WC alignment search for late rising transition on a victim net with 4 aggressors. The victim's noisy transition times computed using ROP metric and the aggresors' switching windows and peak times on each iteration are shown in Fig-

```
1 Compute the reduced-order model of the interconnect
2 Find victim's driver and receivers, switching windows and waveforms on aggressors and the victim's driver inputs
3 For each victim's driver do:
    3.1 Compute Padé-approximated model of load as seen by the driver
    3.2 For each input-output arc compute nominal transitions on the driver output using CM and the load
    3.3 Choose an arc for delay change calculation based on nominal delay and/or slack
    3.4 Propagate the nominal transition for the chosen arc to all receiver inputs
    3.5 Propagate the nominal transitions to each receiver output using CM for the receiver and its Padé-approximated load
    3.6 Compute current and voltage responses from each aggressor, $I_k(t), V_{j,k}(t)$
    3.7 For each receiver iterate until alignment converges:
        3.7.1 Select valid alignment and construct the combined current, $I(t) = \sum_{k=1}^{N} I_k(t - \tau_k)$
        3.7.2 Compute noisy transition on the driver output using CM, $I(t)$ and Padé-approximated load
        3.7.3 Propagate the noisy transition from driver output to the receiver input and superimpose with aligned voltage responses from aggressors $V_j(t) = \sum_{k=1}^{N} V_{j,k}(t - \tau_k)$
        3.7.4 Propagate the noisy transition further to the receiver output using the receiver's CM and Padé-approximated load
        3.7.5 Measure the delay change and pass it to the optimization engine for selection of next alignment
    3.8 Issue a warning if WC noisy transition on the receiver's output is non-monotonic
4 Store the delay change on the receiver's input for next iteration of STA loop
```

Figure 5: Methodology of crosstalk delay change calculation on a stage

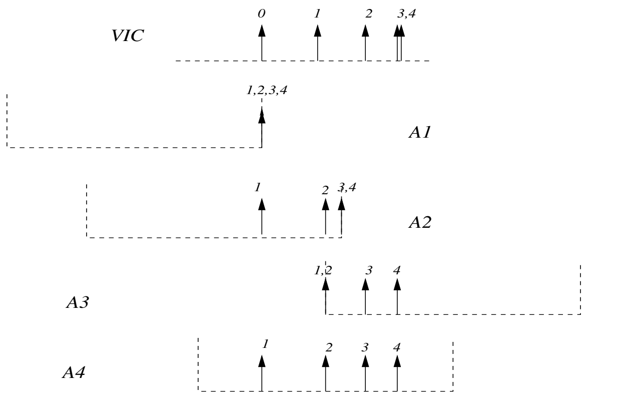

Figure 6: Optimization of WC delay change for a stage with victim and 4 aggressors

ure 6. Dotted vertical lines denote SW's on aggressors, and spikes denotes transition times on victim and aggressor nets during 4 alignment iterations. Numbers next to spikes denote the iteration number. In the example, the aggressor *A4* can be aligned with the victim transition on all iterations. In contrast, aggressors *A2, A3* can be shifted only on part of the iterations, while the aggressor *A1* has a fixed transition time throughout the iterations.

In the described alignment method we do not filter aggressors based on a criterion of non-overlapping of its SW with that of the victim, which is used in other methodologies. Such filtering can potentially engender a cliff in the analysis, especially when the SW's of the victim and aggressors slightly overlap.

In contrast to the mentioned approach, we can include all the aggressors that can impact the victim's transition into the analysis, allowing the simulation to determine the magnitude of each aggressor's contribution to the delay change. Aggressors whose switching windows are very far from the victim's transition time can be filtered. Note that retaining most of the aggressors does not increase computational cost of the analysis significantly, since the current and voltage responses due to each aggressor transition are computed once for all alignments.

The second step of the optimization procedure is a refinement of the WC alignment vector determined in the first step using the average alignment variable. This is achieved by employing the steepest descent method, where each aggressor's alignment is varied separately from others in search of the true WC result. Although the second step can be expensive, especially for large number of aggressors, it is rarely used since the ROP-based delay change as a function of aggressor alignment is usually flat near its maximum and the first step usually suffices.

4. RESULTS

First we demonstrate accuracy of the described fast simulation method on a simple though realistic circuit consisting of a chain of inverters driving nets with significant coupling capacitance. The worst-case alignments of aggressors on each of the noisy victim nets are determined using the described optimization procedure based on the ROP technique. The resulting worst-case alignment is then used in a Spice simulation on the same circuit.

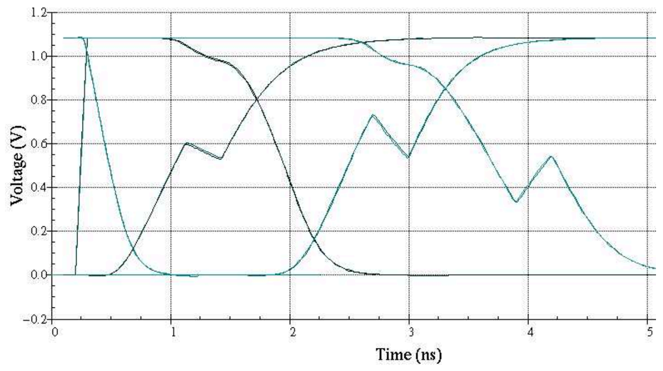

Figure 7: Waveforms computed in Spice and using the proposed fast simulator on a simple circuit

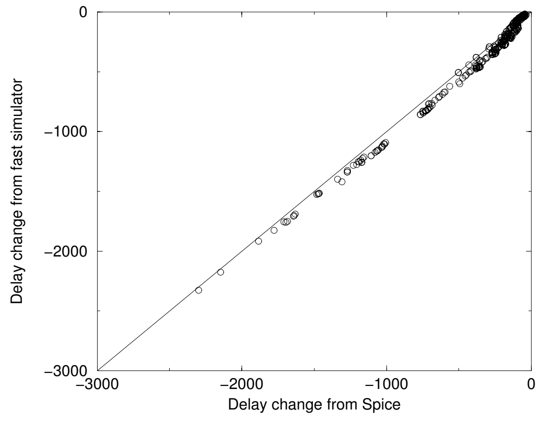

Figure 8: Delay speedup for falling transition computed by fast simuation Spice on a commercial circuit

The transition waves from VivoSim and Spice are shown in Figure 7 and are barely distinguishable. Similar comparisons made for multi-inputs and multi-CCC gates show similar correspondence.

Figure 8 demonstrates accuracy of the fast simulator and the described optimization method on a 14K-cell industrial design. The worst-case aligment and the corresponding delay change (pushout) are compared to those computed in Spice simulations using alignment sweep. The sweep step was chosen to be 5% of the nominal transition slew to ensure the worst alignment is determined.

Table 1: Runtime, memory growth and pessimism reduction for two industrial designs on a 2Ghz Opteron Linux machine

Design	Size (inst)	Runtime (sec)	Memory growth	Pessimism reduction	
				Average	*Worst*
A	18K	30sec	90Mb	10%	51%
B	150K	240sec	670Mb	14%	37%

We further present analysis results on two commercial designs. Design A is an 18K instance, 0.30 micron cell-based signal processor. Design B is a 150K instance, 0.13 micron cell-based network processor. Analysis results for the two designs are shown in Table 1. Delay change results

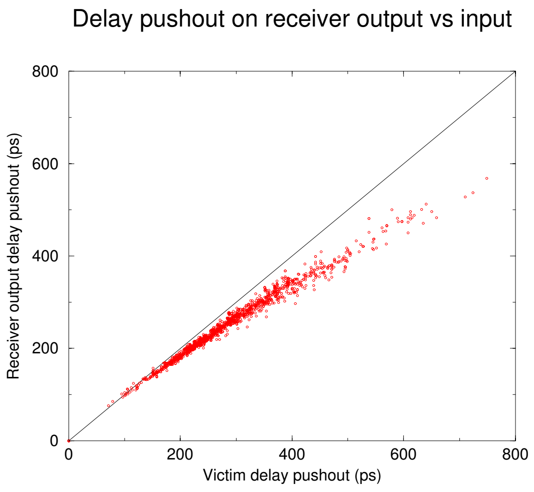

Figure 9: Delay change comparison for late rising transition

as measured on the receiver output are compared against the delay change results as measured on the receiver input. The amount of pessimism reduction is reported in the "Pessimism Reduction" column. Average and worst-case pessimism reductions are reported under the "Average" and "Worst" sub-columns. Also note that the runtime and memory growth scales almost linearly with size of design.

Figure 9 shows the scatter plot for delay pushout on Design B. Even though the average pessimism reduction is 14%, which is quite significant in itself, the large delay pushouts show even more pessimism reduction. Using the proposed method of noise-on-delay analysis, the reduction in noise-on-delay pessimism can be significant on long noisy nets.

Finally we present setup slack data computed on a 130nm industrial circuit with 500K instances. This circuit took 1068 sec to run on a 2GHz Opteron machine. The slacks computed using ROP technique and those computed using the conventional receiver-input probing (RIP) approach are compared in Fig. 10. Note the shift towards more positive slack with the ROP technique, and also the most critical path found via the conventional RIP method is improved using the ROP technique by 230ps. These results show that considerable pessimism reduction can be achieved through the proposed ROP method.

4.1 Conclusions

We have presented a methodology for crosstalk induced noise-on-delay analysis which is robust and less pessimistic. Our methodology employs optimization methods with a dedicated fast simulator to search for the optimal aggressor alignment while considering the circuit response of downstream logic. We have also described an efficient, accurate and automated current model characterization scheme. The accuracy of the current model matches transistor level Spice simulations to within 2-4 percent. The dedicated fast simulation engine using pre-characterized current models and reduced interconnect models have been described and their effectiveness and performance have been proven on large industrial designs.

Our noise-on-delay analysis engine has been tested on several 130nm industrial designs with good performance. Re-

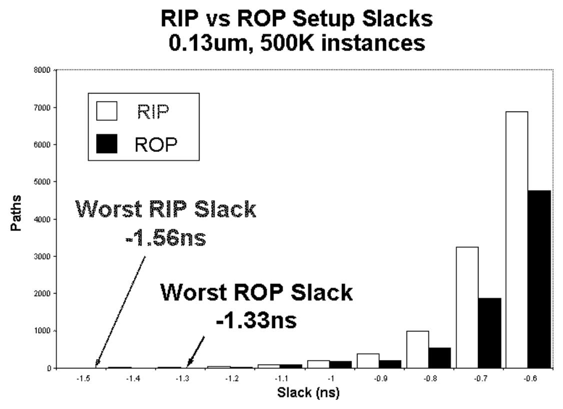

Figure 10: rip vs rop

sults obtained on 180K- and 500K-instance industial designs showed a significant reduction in delay pessimism over existing aggressor alignment and delay measurment techniques, and impoved delay accuracy in general.

5. REFERENCES

[1] K.L. Shepard. Design methodologies for noise in digital integrated circuits In *Proceedings of the Design Automation Conference*, pages 94–99, June 1998.

[2] F. Dartu, L. Pileggi. Calculating worst-case gate delays due to dominant capacitance coupling. In *Proceedings of the Design Automation Conference*, pages 46–51, June 1997.

[3] P. Chen,D.A.Kirkpatrick,and K.Keutzer Miller factor for gate-level coupling delay calculation In *Proceedings of the Design Automation Conference*, pages 68–75, June 2000.

[4] R. Arunachalam, K. Rajagopal, and L. Pileggi. Taco: timing analysis with coupling. In *Proceedings of the Design Automation Conference*, pages 266–269, June 2000.

[5] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. Pileggi. Determination of worst-case aggressor alignment for delay calculation. In *Proceedings of the ICCAD*, pages 212–219, November 1998.

[6] A. Odabasioglu, M. Celik and L. T. Pileggi, PRIMA: Passive Reduced-Order Interconnect Macromodeling Algorithm, In *IEEE Transactions on Computer-Aided Design*, September 1997.

[7] C.A. Desoer and E.S. Kuh, *Basic Circuit Theory*, McGraw-Hill, New York, 1969,

[8] V. Raghavan, J. E. Bracken and R. A. Rohre AWESpice: a general tool for the accurate and efficient simulation of interconnect problems al tool for the accurate and efficient simulation of interconnect problems In *Proceedings of the Design Automation Conference*, pages 87–92, June 1992.

[9] S. Sirichotiyakul, D. Blaauw, C. Oh, R. Levy, V. Zolotov, and J. Zuo Driver modeling and alignment for worst-case delay noise In *Proceedings of the Design Automation Conference*, pages 720–725, June 2001.

[10] J.F. Croix, and D.F. Wong Blade and Razor: Cell and Interconnect Delay Analysis Using Current-Based Models, In *Proceedings of the Design Automation Conference*, 386–391, June 2003.

[11] V. Zolotov, D. Blaauw, S. Sirichotiyakul, M. Becer, C. Oh, R. Panda, A. Grinshpon and R. Levy Noise propagation and failure criterian for VLSI designs In *Proceedings of the ICCAD*, pages 587–594, November 2002.