Pessimism Reduction in Coupling-Aware Static Timing Analysis Using Timing and Logic Filtering

Debasish Das, Kip Killpack*, Chandramouli Kashyap*, Abhijit Jas†, Hai Zhou
EECS, Northwestern University, Evanston, IL 60208
*Strategic CAD Labs, Intel Corporation, Hillsboro, OR 97124
†Validation and Test Solutions, Intel Corporation, Austin, TX 78746

***Abstract*— With continued scaling of technology into nanometer regimes, the impact of coupling induced delay variations is significant. While several coupling-aware static timers have been proposed, the results are often pessimistic with many false failures. We present an integrated iterative timing filtering and logic filtering based approach to reduce pessimism. We use a realistic coupling model based on arrival times and slews and show that non-iterative pessimism reduction algorithms proposed in previous research may give potentially non-conservative timing results. On a functional block from an industrial 65nm microprocessor, our algorithm produced a maximum pessimism reduction of 11.18% of cycle time over converged timing filtering analysis that does not consider logic constraints.**

I. Introduction

With continuous scaling of technology, the aspect ratio of on-chip interconnect wires has continued to increase. As a result the coupling capacitance between adjacent wires on the same metal layer is a dominant component of the total wire capacitance [14]. In addition, modern process technologies have multiple metal layers and a detailed extraction algorithm extracts layer-to-layer capacitance as coupling cap rather than virtual ground cap. In Figure 1, we show the dominance of the coupling cap in the total capacitance of nets extracted from a functional block of an industrial high-performance microprocessor in 65nm technology. The x-axis shows the ratio of coupling capacitance to ground capacitance and the y-axis shows the percentage of nets that have a ratio less than or equal to the value on the x-axis.

Switching activity on the coupled wires induces interference on the wire of interest. Following convention, we refer to the wire of interest as victim and the coupling wires as aggressors. In particular, the switching of the aggressors around the same time as the victim transition causes changes in the delay of the victim signal. When the aggressors switch in the same direction as the victim net, the delay decreases; when the aggressors switch in the opposite direction, the delay increases.

Coupling-aware static timers have been introduced in the last few years to account for this effect. Such timers have to balance the need for being conservative with the need for being realistic and not showing too many false failures. Since victim delay is adversely impacted only when the aggressors switch in the temporal vicinity of the victim, timing windows are used on both victim and aggressor nets, and coupling is considered only when the windows overlap. While this reduces pessimism significantly, additional pessimism can be removed if one considers the logic interactions among the signals. For instance, even if the windows of the aggressors and the victims overlap, it may be logically impossible for all aggressors to fall when the victim rises. Thus, logical interactions along with timing filtering is essential for coupling-aware static timing tools to reduce the number of false failures and enhance designer productivity.

This paper presents a comprehensive framework for performing logic and timing filtering in an integrated manner. This framework is implemented in an industrial static timer and is shown to reduce the number of false failures significantly. The key contributions of our work are:

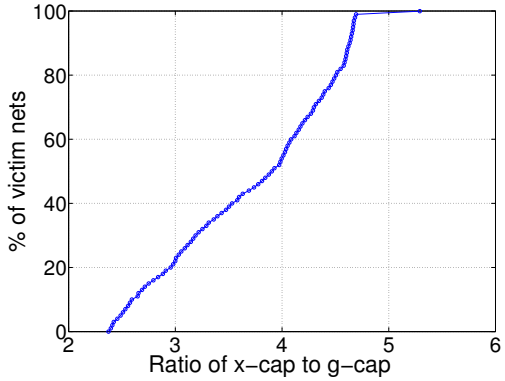

Fig. 1. CDF of ratio of x-cap to g-cap for each net

1) An integrated iterative timing and logic filtering algorithm: We show that iterations are required, since once logic filtering is done, some aggressors get filtered and thus change timing of the victim, necessitating further analysis.
2) A sensitivity-based method for selecting the subset of aggressors on which to apply logic analysis: Doing logic analysis on the entire set of aggressors is often not runtime efficient.
3) Insights into algorithm behavior (Section III) when the underlying coupling model is aware of victim/aggressor alignment and slews: We use a charge sharing model with a focus on fast evaluation and good fidelity to drive the algorithm in the right direction.

We next briefly review the existing literature relevant to this work. Since delay and crosstalk are inherently chicken-and-egg problem (victim delay depends on the amount of charge injected by the aggressors which in turn depends on the victim arrival times and slews), iterative methods are needed to perform coupling analysis [20], [8], [6], [3]. Zhou [20] established the theoretical foundation for the iterative analysis. [8] presented an iterative static timing analysis algorithm based on some initial switching windows (best case crosstalk delays as opposed to worst case crosstalk delays considered by [6]) and iteratively updating the timing information until convergence. Though [8] used a novel coupling model in timing analysis based on arrival and transition times, they did not explore the model dynamics which, as we show, turns out to be a key factor in coupling-aware static timing analysis using logic constraints. To exploit logic feasibility conditions, two recent researches proposed pessimism reduction [19], [4]. We show in this paper that under an alignment- and slew-aware coupling model, pruning by combining timing and logic is no longer a non-iterative step, contrary to [4], [19]. In fact, ignoring the interaction between timing and logic conditions can potentially lead to non-conservative timing results, as shown in Section IV-A.

The rest of the paper is organized as follows. We present our coupling model in Section II. Application of our coupling model

*This work is supported by a grant from Intel Corporation

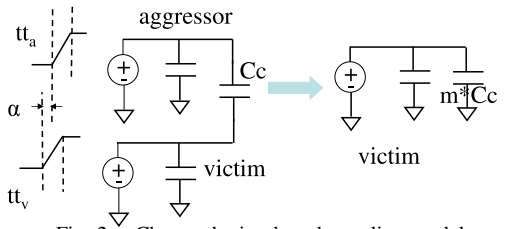

Fig. 2. Charge sharing based coupling model

in timing filtering is described in Section III. Logic filtering and our algorithm *LogicTimer* are presented in Section IV. Detailed results are presented in Section V. Finally we conclude our paper in Section VI.

II. Coupling Model

Our coupling model is based on a charge sharing model as presented in [12], [10]. We are aware that such a model is an abstraction of more complex simulation-based models [13], [7] but algorithmic contributions presented in this paper are orthogonal to model choice as long as the model is slew and alignment based. Our goal is to use a fast evaluation model that has high fidelity in order to drive the algorithm in the right direction as the model is evaluated many times in our algorithm. This is analagous to using the Elmore delay model in physical design. Additionally, the chosen model allows us to derive an analytical understanding of algorithm behavior in Section III.

The circuit model for Miller coupling factor (m) computation is shown in Figure 2. m is computed based on the victim and aggressor arrival times and respective slews. We use the following notations:

$$\text{Victim Arrival Time} = at_v$$
$$\text{Aggressor Arrival Time} = at_a$$
$$\text{Victim Slew} = tt_v$$
$$\text{Aggressor Slew} = tt_a$$
$$\text{Arrival Time Difference } \alpha = at_v - at_a$$

Note the difference of our notations from prevalent conventions. For the purpose of defining and evaluating the model, we consider at as 0% arrival time rather than 50% point. tt represents the time taken for the signal to transition between 0% and 100% of V_{dd}. This is computed by linearly extending a 20-80% slew from static timing analysis. The analytical derivations for the charge sharing model are omitted; interested readers are referred to [12], [10] for details.

We define the voltage percentage over which charge matching is applied as β. Based on different values of β, different bounds on the coupling factor can be obtained. In this work, we do charge matching over 0-100% voltage of the victim with aggressor (β = 1). Thus, the bounds for our coupling model can vary between 0 and 2 but there are other similar coupling models such as [12] where the bounds can be much wider than 0 and 2. The algorithmic details reported in this paper are independent of the chosen value of β. Should one choose to be more conservative, choosing β as 0.5 provides a practical upper bound of 3 to Miller coupling factor [5], [12] and thus our model can provide an upper bound similar to the Elmore delay model. As described in Section IV, using a model with high fidelity we can identify important aggressors. A more complex model can be employed for final signoff timing with these important aggressors.

Details of our coupling model are presented in Figure 3 and 4. Figure 3 presents the situation when victim and aggressor are switching in the same direction. The function ψ^{SS} is a piecewise linear function whose domain and range are α and m respectively. Figure 4 presents the situation when victim and aggressor are switching in opposite directions. We call this function ψ^{OS}.

case 1: $tt_a \geq tt_v$
- $-\infty < \alpha \leq -tt_a$, $m = 1.0$
- $-tt_a < \alpha \leq tt_v - tt_a$, $m = \frac{-\alpha}{tt_a}$
- $tt_v - tt_a < \alpha \leq 0$, $m = 1 - tt_v/tt_a$
- $0 < \alpha \leq tt_v$, $m = 1.0 + \frac{\alpha - tt_v}{tt_a}$
- $tt_v < \alpha \leq \infty$, $m = 1.0$

case 2: $tt_a < tt_v$
- $-\infty < \alpha \leq -tt_a$, $m = 1.0$
- $-tt_a < \alpha \leq 0$, $m = \frac{-\alpha}{tt_a}$
- $0 < \alpha \leq tt_v - tt_a$, $m = 0$
- $tt_v - tt_a < \alpha \leq tt_v$, $m = 1.0 + \frac{\alpha - tt_v}{tt_a}$
- $tt_v < \alpha \leq \infty$, $m = 1.0$

Fig. 3. Function ψ^{SS} for Coupling Factor (Similar Switching)

case 1: $tt_a \geq tt_v$
- $-\infty < \alpha \leq -tt_a$, $m = 1.0$
- $-tt_a < \alpha \leq tt_v - tt_a$, $m = 2.0 + \frac{\alpha}{tt_a}$
- $tt_v - tt_a < \alpha \leq 0$, $m = 1 + tt_v/tt_a$
- $0 < \alpha \leq tt_v$, $m = 1.0 + \frac{tt_v - \alpha}{tt_a}$
- $tt_v < \alpha \leq \infty$, $m = 1.0$

case 2: $tt_a < tt_v$
- $-\infty < \alpha \leq -tt_a$, $m = 1.0$
- $-tt_a < \alpha \leq 0$, $m = 2.0 + \frac{\alpha}{tt_a}$
- $0 < \alpha \leq tt_v - tt_a$, $m = 2.0$
- $tt_v - tt_a < \alpha \leq tt_v$, $m = 1.0 + \frac{tt_v - \alpha}{tt_a}$
- $tt_v < \alpha \leq \infty$, $m = 1.0$

Fig. 4. Function ψ^{OS} for Coupling Factor (Opposite Switching)

III. Coupling Model Application to Timing Filtering

We apply our coupling model to crosstalk-aware static timing analysis. Each net in the circuit has a driver port as source and several receiver ports as sink. During static analysis, timing events are propagated using a breadth-first-search beginning from input ports. Static timing is performed in both min and max mode to identify lower and upper bounds on arrival times and slews for each net. These bounds define arrival time and slew windows for each net. Both rise and fall windows exist on each net due to rise and fall timing events generated during static timing. The min and max modes are evaluated concurrently to enable timing window analysis.

For the two static timing modes, we must compute both min and max coupling induced delay push-out. Since delay monotonically increases with output load, we compute max (min) delay push-out using the max (min) coupling factor m. Max m results from opposite switching victim and aggressor transitions and min m results from like switching victim and aggressor transitions as shown by the coupling model in Figure 4 and Figure 3.

Slew selection for coupling induced push-out computation follows from the coupling model. Consider the equations in Figure 4 where tt_v is always found in the numerator and tt_a is always found in the denominator, both with positive sign. Thus to maximize m we choose maximum tt_v and minimum tt_a from the respective slew windows. A similar method is used to choose slews to minimize m. During static analysis of victim v, four coupling factors are computed: m for the maximum rise event, m for the maximum fall event, m for the minimum rise event and m for the minimum fall event. Note that we get different values of the four coupling factors when we analyze net a as a victim and v becomes its aggressor.

In rest of the paper, for each victim net we focus our attention on the max arrival time calculation (rise, fall). For each max event

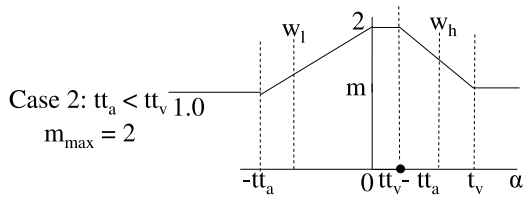

Fig. 5. m as a function of α for opposite switching

calculation we assume the aggressor can transition anytime between its earliest and latest arrival time. Min arrival time computation follows by symmetry. We begin our crosstalk-aware static timing analysis with the bounds of 0 and 2 on m. Timing windows are then iteratively refined. These iterations decrease the maximum bound and increase the minimum bound on m. Thus our iterations shrink the delay windows and reduce the pessimism. We refer to such an analysis as timing filtering. We discuss behavior of our coupling model under timing filtering next. Refer to the situation discussed in Section II. We are computing maximum m due to the maximum rise event on v and the fall window on aggressor a. We have been given (at_v, tt_v) for victim v and (at_a^{min}, tt_a), (at_a^{max}, tt_a) for the aggressor a. $\alpha, m \in R$ are defined in Section II.

We map the given aggressor window (at_a^{min}, at_a^{max}) to the reference of victim arrival time at_v. We look at all possible alignments between victim and aggressor arrival times and choose the alignment that maximizes or minimizes m depending on timing mode. Mapping the aggressor window to the victim reference will give the window $(at_a^{min} - at_v, at_a^{max} - at_v)$ as all feasible cases for α. We call this modified window as W=(w_l, w_h). According to the slew values we use one of the two cases presented in Figure 4. Maximum coupling factor computation can be formally presented as

$$\begin{aligned} m = \text{Maximize } & \psi^{OS}(w) \\ \text{subject to } & w_l \leq w \leq w_h \end{aligned} \tag{1}$$

A visual representation of the maximum coupling factor computation is shown in Figure 5. w_l and w_h are the bounds on α. As evident from the figure, if $tt_a < tt_v$, m is 2.0. Consider iterations i and j. Respective W for these iterations are W^i and W^j. As described in Section III subsequent iterations strive to reduce pessimism. We define relation $\sqsubseteq$ (Pessimism Reduction) over the set of W as follows

$$W^j \sqsubseteq W^i \quad if \quad w_l^j \geq w_l^i \quad \text{and} \quad w_h^j \leq w_h^i$$

Consider m^i and m^j as the respective coupling factors at iterations i and j for same aggressor net. The following theorem gives interesting insight into the proposed coupling model.

Theorem 1: $W^j \sqsubseteq W^i$ does not imply $m^j \leq m^i$

Proof: The proof is by contradiction. We will assume $W^j \sqsubseteq W^i$ and present a situation where $m^j \geq m^i$. Without any loss of generality consider $j = i + 1$ and $tt_a < tt_v$. ΔI represents change of iteration.

$$\begin{aligned} W^i &= (w_l^i, w_h^i) \\ tt_v - tt_a < w_l^i \leq tt_v &\Rightarrow \psi^{OS}(w_l^i) = 1.0 + \frac{tt_v - \alpha}{tt_a} \\ tt_v < w_h^i \leq \infty &\Rightarrow \psi^{OS}(w_h^i) = 1.0 \\ (\frac{\Delta w_l}{\Delta I})^{ij} = \frac{w_l^i - w_l^j}{i-j} & \qquad (\frac{\Delta w_h}{\Delta I})^{ij} = \frac{w_h^i - w_h^j}{i-j} \\ W^i \sqsubseteq W^j \Rightarrow (\frac{\Delta w_l}{\Delta I})^{ij} \geq 0 & \qquad W^i \sqsubseteq W^j \Rightarrow (\frac{\Delta w_h}{\Delta I})^{ij} \leq 0 \end{aligned}$$

Applying Equation 1, m in this situation will be given by $\psi_{OS}(w_l)$. We represent $m = f(\alpha, tt_v, tt_a)$. Using a first order Taylor expansion around m^i we approximate m^{i+1} as

$$m^{i+1} = m^i + \frac{\partial f}{\partial tt_v} \times \Delta tt_v + \frac{\partial f}{\partial tt_a} \times \Delta tt_a + \frac{\partial f}{\partial \alpha} \times \Delta \alpha$$

Computing the partial derivatives of $\psi_{OS}(w_l)$, at iteration i we can approximate m^{i+1} as

$$m^{i+1} = m^i + \frac{1}{tt_a^i} \times \Delta tt_v - \frac{tt_v^i}{(tt_a^i)^2} \times \Delta tt_a - \frac{1}{tt_a^i} \times \Delta \alpha \tag{2}$$

$\Delta\alpha$ in Equation 2 is $(\frac{\Delta w_l}{\Delta I})^{ij}$ which is positive. Assuming pessimism reduction on both aggressor and victim slews, Δtt_a and Δtt_v can be considered negative without any loss of generality. It is possible to choose slews on iteration i to find $m^{i+1} > m^i$. Thus we get a contradiction. ■

Corollary 1: If Δtt_v and Δtt_a are insignificant compared to $\Delta\alpha$ then $W^j \sqsubseteq W^i \Rightarrow m^j \leq m^i$

Proof follows directly by doing order analysis on tt and α in Equation 2. Corollary 1 gives more insights into our model. Arrival time difference turns out to be the primary effect that changes m and thus helps in pessimism reduction. Slew differences are second order effects but they invalidate the monotonicity of m.

Monotonic changing of timing windows was essential for the proof of convergence of iterative techniques as presented in [20]. But in a realistic coupling model as we present in Section II, we have situations when monotonicity of m over the iterations does not hold and therefore may cause problems in convergence. The following observation presents insights into the timing convergence in presence of a realistic coupling model. Consider iterations i, $i+1$ and $i+2$.

Observation 1:

$$W^{i+1} \sqsubseteq W^i \wedge W^{i+1} \sqsubseteq W^{i+2} \Rightarrow W^{i+2} \sqsubseteq W^i$$

We represent change in m over iterations as Δm. $\Delta m^i = |m^{i+1} - m^i|$. For most cases we found that Δm decreases monotonically converging to a fixpoint. Thus $\Delta m^{i+1} \leq \Delta m^i$. This implies that $W^{i+2} \sqsubseteq W^i$.

Observation 2: Timing analysis converges if W^i is a strictly bigger interval than W^{i+2}.

Observation 3: Timing analysis produces oscillations if W^i is same as W^{i+2} and $W^i \neq W^{i+1}$.

Our experiments verify Observations 2 and 3.

IV. Logic Filtering

Timing filtering as presented in Section III assumes state transitions on victims and aggressors to generate maximum m. Logic filtering describes a process by which certain aggressor and victim state transition combinations are eliminated from consideration because such conditions are logically impossible. This can be explained with an example shown in Figure 6. Let us assume that the logic driving the two aggressors A1 and A2 are as shown in Figure 6. Both the latches are active high and driven by the clock C. Let N(t) denote the value of signal N in the circuit at time t. N(t) is commonly referred to as the current state for signal N and N(t+1) is the next state for signal N. The next-state functions for the aggressors A1 and A2 for the above circuit can be represented by the following equations.

$$\begin{aligned} A1(t+1) = \; & (C(t+1) = 0) * A1(t) + \\ & (C(t+1) = 1) * (\tilde{}M(t+1)) \end{aligned} \tag{3}$$

$$\begin{aligned} A2(t+1) = \; & (C(t+1) = 0) * A2(t) + \\ & (C(t+1) = 1) * M(t+1) * N(t+1) \end{aligned} \tag{4}$$

In Equations 3-4, $*$ denotes logical AND, $+$ denotes logical OR and$\tilde{}$ denotes logical NOT conditions. Note that the above equations assume a 0-delay model and do not take into consideration glitches.

To find out if $A1 = R$ and $A2 = R$ is logically feasible, we check if the Boolean condition $(A1(t) = 0) * (A2(t) = 0) * (A1(t+1) =$

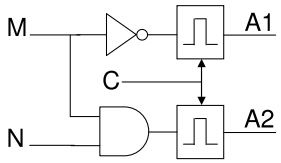

Fig. 6. Logic filtering example

V	A1	A2	A3	Feasibility
R	R	R	R	Infeasible
R	F	S	S	Feasible
R	F	F	F	Infeasible
R	S	F	S	Feasible
R	F	R	F	Feasible
.	.	.	.	.

Fig. 7. Logic conditions on a victim cluster

$1) * (A2(t+1) = 1)$ based on the above equations is satisfiable (i.e., has a solution). It can be easily verified that this condition is not satisfiable for this example. This indicates that the condition $A1 = R$ and $A2 = R$ is logically impossible.

In our current work we derive similar equations for aggressor and victim state transition combinations by analyzing the logic and use search engines like Satisfiability (SAT) [16] and Automatic Test Pattern Generation (ATPG) [18] to find out if certain combinations are impossible. Detailed discussion on formulating SAT clauses from circuit structures can be found in [15]. ATPG based techniques are used by [2] for pessimism reduction but unlike our work they did not consider the interaction between timing and logic filtering.

Consider a victim cluster $< V, A1, A2, A3 >$. We define a logic pattern as a combination of $Rise$(R), $Fall$(F) or $Stable$(S) conditions on $A1, A2, A3$ where possible victim logic conditions are $Rise$ or $Fall$. A collection of such patterns as shown in Figure 7, forms a *Logic Table* for the victim cluster. In Figure 8 we show the infeasible pattern counts for all victim nets in a functional block. For each victim, we check logic patterns for the top 3 aggressors when the victim is rising(R). The pattern RFFF is logically infeasible for 7481 out of 21681 possible victim nets. Thus, assuming worse-case logic conditions is overly pessimistic.

Given a logic table for a victim cluster, each pattern has a coupling induced delay push-out associated with it. We call this delay push-out the pattern rank. The logic filtering problem statement is as follows:*Find the pattern that produces the worst rank.*

A. Logic Filtering Preliminaries

Computation of coupling induced delay push-out for logic patterns extends the m computation as presented in Section III. Suppose we are computing the maximum m for a victim rise event. Static timing would assume aggressors fall, but in reality the aggressors can fall, rise, or remain stable. Therefore we must compute additional

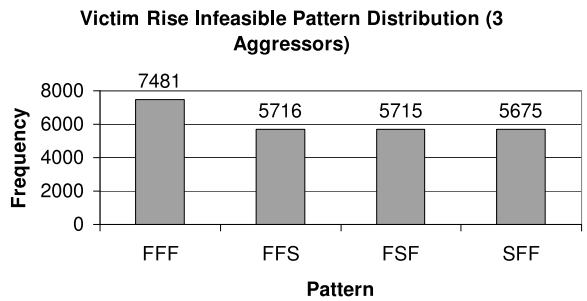

Fig. 8. Victim Rise Infeasible Pattern Distribution

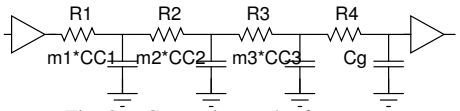

Fig. 9. Generating rank of a pattern

values of m : $m[v_R, a_R]$, $m[v_R, a_F]$ and $m[v_R, a_S]$ based on the logic conditions of the aggressor. Similar values of m are computed when victim is falling. $m[v_R, a_S]$ = 1.0 is a base case of the charge matching model where the aggressor is not switching. Computation of $m[v_R, a_F]$ remains the same as Equation 1. To compute $m[v_R, a_R]$, we take the rise timing window of the aggressor and map it to the max rise event of the victim to generate $W^R = (w_l^R, w_h^R)$ as described in Section III. The solution is similar to Equation 1 except with ψ^{OS} replaced with ψ^{SS}

$$m = \text{Maximize } \psi^{SS}(w) \quad (5)$$
$$\text{subject to } w_l \leq w \leq w_h$$

We give an example of rank generation for pattern RFSS from Figure 7. We generate m^{A1}, m^{A2} and m^{A3} using the extended formulations described above. The circuit model of this victim cluster is shown in Figure 9. $m1$, $m2$ and $m3$ are respectively $m^{A1}[v_R, a_F]$, $m^{A2}[v_R, a_S]$ and $m^{A3}[v_R, a_S]$. The rank E_d for the pattern under consideration is computed using Elmore delay [9]. Due to its high fidelity, E_d provides us with a fast and accurate estimate for the ranks.

$$\begin{aligned} E_d \quad = \quad & m1 * CC1 * (R1) + m2 * CC2 * (R1 + R2) \\ & + m3 * CC3 * (R1 + R2 + R3) \\ & + C_g * (R1 + R2 + R3 + R4) \end{aligned}$$

We show that logic filtering turns out to be a chicken-egg problem. Pattern ranking depends on m. m is a function of α, tt_v, tt_a, which are in turn functions of m. Thus change in m can potentially make a new row have the worst rank. We illustrate this with an example. Let the coupling capacitances for the victim cluster $< V, A1, A2, A3 >$ be $C_c^{A1} = 3.0$, $C_c^{A2} = 2.5$ and $C_c^{A3} = 2.0$. Suppose $m^{A1}[v_R, a_F] = 1.8$, $m^{A1}[v_R, a_R] = 1.0$, $m^{A1}[v_R, a_S] = 1.0$ and $m^{A2}[v_R, a_F] = 2.0$, $m^{A2}[v_R, a_R] = 1.0$, $m^{A2}[v_R, a_S] = 1.0$ and $m^{A3}[v_R, a_F] = 1.7$, $m^{A3}[v_R, a_R] = 1.0$, $m^{A3}[v_R, a_S] = 1.0$. We get RFRF as the worst ranked pattern. But even if $m[v_R, a_F]$ reduces for all aggressors, such as $m^{A1}[v_R, a_F] = 1.2$, and $m^{A2}[v_R, a_F] = 1.9$, and $m^{A3}[v_R, a_F] = 1.6$ in the next iteration, we get RSFS as the worst ranked pattern. It is also interesting to note that the worst ranked pattern for max victim rise can have an aggressor falling. To maintain conservatism patterns with like transitions should not be ignored during max analysis.

Corresponding ranks for pattern RFRF and RSFS are 9.3 and 9.75. But in the next iteration due to reason explained by Theorem 1, m can increase. Suppose we obtain the new values of $m[v_R, a_F]$ as $m^{A1}[v_R, a_F] = 1.2$, and $m^{A2}[v_R, a_F] = 2.0$, and $m^{A3}[v_R, a_F] = 1.7$. In this iteration the worst ranked pattern is still RSFS but the ranks for pattern RFRF and RSFS become 9.5 and 10.0 respectively. The worst rank of 9.75 obtained in the previous iteration is non-conservative and fix-point iterations as suggested by Zhou [20] are required to get conservative bounds on worst ranks. Also iterations are necessary since the pattern that generates worst rank can change due to m as explained earlier. Thus logic filtering with a realistic coupling model is an iterative approach as opposed to a single step process as suggested by previous researches [4], [19].

B. LogicTimer

We present our algorithm in Figure 10. Our algorithm has the flexibility to run in timing filtering or timing and logic filtering modes.

Algorithm: LogicTimer

```
Initialization: m_0 ← (0,2)
          Events_0 ← f(m_0)
while( |Events_i - Events_{i-1}| > ε)
 if(TIMING AND LOGIC FILTERING)
   Compute m_i for
        selected aggressors(Section IV-A)
   R ← Identify worse ranked pattern
        using m_i
   m_i ← g(Events_{i-1}, R)
 else if (TIMING FILTERING)
   m_i ← h(Events_{i-1})
 Events_i ← f(m_i)
```

Fig. 10. Logic and timing filtering static timing analysis

The algorithm is similar to Gauss-Jacobi [11] iteration technique. It initializes by assuming worst and best bounds on the aggressors as provided by our coupling model in Section II. Based on m, timing events on each node of the timing graph for iteration 0 are generated. The subscript with $Events$ refer to the iteration number. We continue our algorithm in *TIMING FILTERING* until we are close to convergence on all events. The converged timing windows are used to identify important aggressors for the logic filtering algorithm. The aggressors that do not align with the victim are not considered. Details of important aggressor identification are presented in Section IV-C. We generate logic tables for the selected aggressors and then change the mode of our algorithm to *TIMING AND LOGIC FILTERING*. At each iteration we compute coupling factors and use them to identify the worst possible pattern for each victim. We use the worst pattern to determine new m's. Finally using these updated m's, new events are generated on each node of the timing graph in a topological order. Once the coupling factors are generated, timing analysis approach is similar to the algorithm proposed by [3].

Convergence of the algorithm comes from Observation 1. For complexity analysis, suppose the number of nodes in a circuit are N. I^{TF} iterations are needed to converge using timing filtering. We are not considering the time for aggressor selection and logic table generation since it is called once. Although in any logic filtering algorithm, logic table generation has the highest overhead in terms of runtime. Suppose we took I^{LF} iterations to converge on logic tables. Consider the number of victim nets in the circuit as E. Generation of m in timing and logic filtering modes can be done in O(E) time. The overhead in pattern ranking is constant time in terms of number of patterns for each victim cluster. Once m's are generated, the timing events can be updated in O(N). Hence each iteration of our algorithm takes O($N + E$). Total complexity of our algorithm is given by O($(I^{LF} + I^{TF})(N + E)$).

C. Aggressor Selection based on sensitivities

We present a metric S to select important aggressors after the timing filtering iterations converge. Consider a victim cluster as described in Section IV. A victim net can be represented as an RC tree [1] with nodes as v^0,...,v^N. Nodes v^0,...,v^N can be further partitioned into 3 sets $Source$, $Internal$ and $Sink$. Consider v^0 in $Source$. Nodes in $Sink$ are connected to inputs of logic gates. Let C_g^i be the ground capacitance at node v^i where $0 \leq v^i \leq N$. Denote $p(i)$ as the predecessor or parent of node i in the RC tree and R^i as the resistance between nodes $p(i)$ and i. Denote R^{ki} as the total resistance of the portion of the unique path from v^0 to v^i that overlaps with v^0 to v^k. We are interested in computing delay from v^0 to $v^i : i \in Sink$.

Assume node v^i is coupled to some aggressor net $j \in (1..M)$ through the coupling capacitance C_c^{ij}. Without any loss of generality C_c^{ij} can be 0 when there is no physically extracted coupling from layout. Once the timing iterations converge, we can modify Elmore delay formulation in presence of coupling as

$$ED_c^i = \sum_{k=1}^{N} (R^{ki} \times (C_g^k + \sum_{j=1}^{M} \text{Couple(k,j)} \times m_{nom}^j \times C_c^{kj}))$$

where m_{nom}^j is the converged value of the coupling factor from timing iterations. Couple in Equation 6 is a predicate which evaluates to 1 if node v^i on victim net V is coupled with aggressor net A^j. In fact we can represent ED_c^i as

$$ED_c^i = g(m^1, m^2, ..., m^M)$$

Using a first order Taylor expansion on ED_c^i we obtain the following equation

$$g(m^1, m^2, ..., m^M) = g(m_{nom}^1, m_{nom}^2, ..., m_{nom}^M) + \frac{\partial g}{\partial m^1} \times (m^1 - m_{nom}^1) + ... + \frac{\partial g}{\partial m^M} \times (m^M - m_{nom}^M) \quad (6)$$

We then predict the gradients of Equation 6 as follows

$$\frac{\partial g}{\partial m^1} = \frac{g(\hat{m^1}, ..., m_{nom}^M) - g(m_{nom}^1, ..., m_{nom}^M)}{\hat{m^1} - m_{nom}^1} \quad (7)$$

Rearranging Equation 6 we can obtain the % change in delay due to respective change in m^j. On the basis of above discussion we present our sensitivity based metric for aggressor j as follows

$$S^j = \frac{\frac{\partial g}{\partial m^j}}{g(m_{nom}^1, ..., m_{nom}^M)} \quad (8)$$

$\hat{m^j}$ is required to compute the sensitivity S^j. $\hat{m^j}$ results from the underlying coupling model (Refer Section II) in the associated timing analysis flow. While doing max analysis in logic filtering we need to evaluate ψ^{SS} as shown in Equation 5. Upper bound of m in Figure 3 is 1.0. Therefore choosing $\hat{m^j}$ as 1.0 predicts the gradient value in right direction. Though we derived our metric using Elmore formulation, it is straightforward to use our metric in conjunction with RICE [17] engine to compute accurate delays and thus trade off on runtime to generate S^j.

V. Results

Our experiments are performed on a functional block from a 65nm industrial microprocessor. Logic table generation requires selection of important aggressors. We run *LogicTimer* in *TIMING FILTERING* mode till convergence and then select important aggressors. We compute the slacks on end nodes (flip-flop and latch data pins) and normalize the slacks with respect to the cycle time of the microprocessor.

We present our results on the setup slacks corresponding to maximum edges of timing windows. Hold slacks or minimum edge analysis is analogous. We compare normalized slacks of the first iteration (with coupling factors of 0 and 2) with slacks of the iteration when *LogicTimer* converges on all nets in *TIMING FILTERING* mode (Iteration 8). Maximum pessimism reduction relative to the cycle time of the microprocessor is 22.69% and median pessimism reduction is 3.72%. Note that most of the nodes have converged after just 3 iterations.

We selected the top N aggressors for logic table generation where aggressors are ranked on the basis of $m \times C_c$. Due to complexity of SAT, logic table generation with N = 9 aggressors for all victims was not feasible (O($E \times 3^9$)). Hence we divided the aggressors

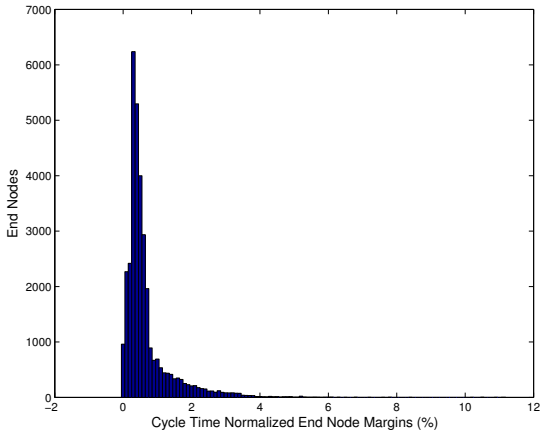

Fig. 11. Pessimism reduction (Logic Filtering on top of Converged Timing Filtering)

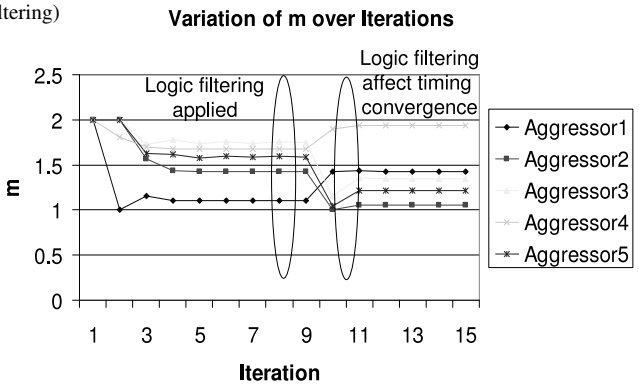

Fig. 12. Convergence of m

into groups of 3 and generated logic tables for victim clusters of size 4 (O($E \times 3 \times 3^3$)). We compute the worst rank of each group independently and do a superimposition to come up with the final rank. If logic table generation runtime is improved, larger tables can be generated and logic filtering results will also improve. Results of running our algorithm in *TIMING AND LOGIC FILTERING* mode are presented in Figure 11. We compare normalized slacks with the slacks of iteration 8 (converged timing filtering results). We obtain a maximum pessimism reduction of 11.18% relative to the cycle time of microprocessor. Median pessimism reduction is 0.44%. We also identified circuit nodes where arrival times are not monotonically decreasing over the iterations. Theoretical reasons behind such behavior comes from the behavior of our coupling model as explained by Theorem 1 where m is not monotonically decreasing. We show the variation of m over iterations on one such node for a few aggressors in Figure 12. We also found nodes which show oscillations in m as mentioned in Observation 3. Such nodes were few (around 1.17%) and in such cases the timing analysis algorithm should take conservative decisions to deal with the oscillations.

Results of running our algorithm on the basis of aggressor selection metric presented in Section IV-C is shown in Figure 13. Using $m \times C_c$ 3999 nodes reduced pessimism by 0.50% while using our metric S, about 5362 nodes reduced a similar amount of pessimism. This trend continues as our metric was able to reduce 0.60% of pessimism in 5362 nodes as compared to 2937 nodes using $m \times C_c$ metric. In fact as evident from Figure 13, the node distribution using S is more skewed toward increased values of pessimism reduction as compared to metric $m \times C_c$.

VI. Conclusions

In this paper we present a novel algorithm to do coupling-aware static timing analysis with logic constraints to remove pessimism.

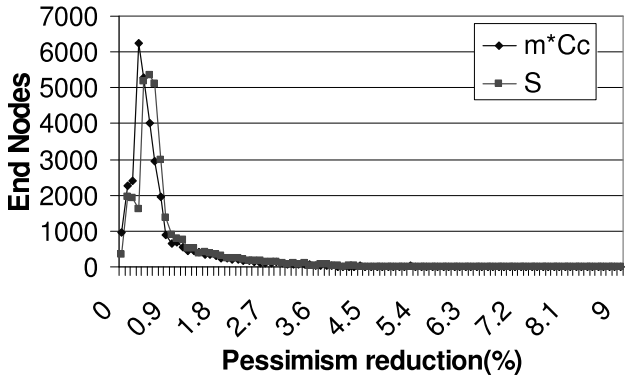

Fig. 13. Metric comparison ($m \times C_c$ vs S)

We also present the dynamics of a complex coupling model under timing analysis and its effect on logic filtering. Timing analysis with logic constraints must be iterative to give conservative timing data. On a functional block of a 65nm industrial microprocessor, our algorithm showed a maximum pessimism reduction of 11.18% on top of Timing Filtering Analysis. Our future work includes extending such a coupling model to statistical timing analysis and optimization.

References

[1] C. J. Alpert, A. Devgan, and C. V. Kashyap. RC Delay Metrices for Performance Optimization. In *IEEE Trans. on CAD of Integrated Circuits and Systems 20(5)*, pages 571–582, 2001.

[2] R. Arunachalam, R. D. Blanton, and L. T. Pileggi. False coupling interactions in static timing analysis. In *DAC*, pages 726–731, 2001.

[3] R. Arunachalam, K. Rajagopal, and L. T. Pilleggi. Taco: Timing analysis with coupling. In *DAC*, pages 266–269, Los Angeles, CA, June 2000.

[4] D. Chai, A. Kondratyev, Y. Ran, K. Tseng, Y. Watanbe, and M. Sadowska. Temporofunctional crosstalk noise analysis. In *DAC*, pages 860–863, 2003.

[5] P. Chen, D. A. Kirkpatrick, and K. Keutzer. Miller factor for gate-level coupling delay calculation. In *ICCAD*, San Jose, CA, Nov. 2000.

[6] P. Chen, D. A. Kirkpatrick, and K. Keutzer. Switching window computation for static timing analysis in presence of crosstalk noise. In *ICCAD*, San Jose, CA, Nov. 2000.

[7] D. Das, A. Shebaita, Y. Ismail, H. Zhou, and K. Killpack. Nostra-XTalk: A Predictive Framework for Accurate Static Timing Analysis in UDSM VLSI Circuits. In *Proc. ACM Great Lakes Symposium on VLSI*, pages 25–30, 2007.

[8] D. Das, A. Shebaita, H. Zhou, Y. Ismail, and K. Killpack. FA-STAC: A framework for fast and accurate static timing analysis with coupling. In *ICCD*, 2006.

[9] W. C. Elmore. The transient response of dampled linear networks with particular regard to wide-band amplifiers. *Journal of Applied Physics*, 19(1):55–63, Jan. 1948.

[10] M. Ghoneima and Y. Ismail. Accurate decoupling of coupled on-chip buses. In *ISCAS*, pages 4146–4149, 2005.

[11] M. T. Heath. *Scientific computing: an introductory survey*. McGraw-Hill Companies, Inc., 2002.

[12] A. Kahng, S. Muddu, and E. Sarto. On switch factor based analysis of coupled RC interconnects. In *DAC*, pages 79–84, 2000.

[13] I. Keller, K. Tseng, and N. K. Verghese. A robust cell-level crosstalk delay change analysis. In *ICCAD*, pages 147–154, 2004.

[14] R. Kumar. Interconnect and noise immunity design for the Pentium 4 processor. In *DAC*, pages 938–943, 2003.

[15] T. Larrabee. Test Pattern Generation Using Boolean Satisfiability. *IEEE Trans. Comput.-Aided Design Integrated Circuits*, 11(1):6–22, Dec. 1992.

[16] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *DAC*, pages 530–535, 2001.

[17] C. L. Ratzlaff, N. Gopal, and L. T. Pillage. RICE: Rapid Interconnect Circuit Evaluator. In *DAC*, pages 555–560, 1991.

[18] M. H. Schultz, E. Trishchler, and T. M. Sarfert. SOCRATES: A Highly Efficient Automatic Test Pattern Generation System. *IEEE Trans. Comput.-Aided Design Integrated Circuits*, pages 126–136, Dec. 1988.

[19] K. Tseng and M. Horowitz. False coupling exploration in timing analysis. In *IEEE Trans. on CAD of Integrated Circuits and Systems 24(11)*, pages 1795–1805, 2005.

[20] H. Zhou. Timing analysis with crosstalk is a fixpoint on a complete lattice. In *IEEE Transactions on Computer-Aided Design, September 2003*, pages 1261–1269, 2003.