Timing Analysis With Crosstalk is a Fixpoint on a Complete Lattice

Hai Zhou

***Abstract*—Increasing delay variation due to capacitive and inductive crosstalk has a dramatic impact on deep submicron technologies. It is now impossible to exclude crosstalk from timing analysis. However, timing analysis with crosstalk is a mutual dependence problem since the crosstalk effect in turn depends on the timing behavior of a circuit. In this paper, we establish a theoretical foundation for timing analysis with crosstalk. We show that solutions to the problem are fixpoints on a complete lattice. Based on that, we prove in general the convergence of any iterative approach. We also show that, starting from different initial solutions, an iterative approach will reach different fixpoints. The current prevailing practice, which starts from the worst case solution, will always reach the greatest fixpoint, which is the loosest solution. In order to reach the least fixpoint, we need to start from the best case solution. The convergence rates for both discrete and continuous models are discussed. Based on chaotic iteration and heterogeneous structures of coupled circuits, techniques to speed up iterations are also provided.**

***Index Terms*—Crosstalk, lattice theory, noise, static timing analysis.**

I. Introduction

With the progress of deep submicron technology, shrinking geometries have led to a reduction in self-capacitance of wires. Meanwhile, coupling capacitances have increased as wires have a larger aspect ratio and are brought closer together. For present day processes, the coupling capacitance can be as high as the sum of the area capacitance and the fringing capacitance, and trends indicate that the role of coupling capacitance will be even more dominant in the future as feature sizes shrink [18]. On the other hand, with the reduction of gate delays and increasing clock frequencies, the mutual coupling inductances are also greatly increased among the wires. These make crosstalk a major problem in IC designs. Crosstalk can affect the behavior of a circuit in two ways:

1) introducing noise among coupling wires;
2) altering the delay of a switching signal.

When coupling capacitances and inductances are big enough, signal switching on an aggressor wire can induce a large amount of noise on a victim line. If an aggressor and a victim switch simultaneously in the same direction, the victim will speed up. Likewise, if an aggressor and a victim switch in opposite directions, the victim will slow down.

Manuscript received March 8, 2002; revised February 27, 2003. This work was supported in part by the National Science Foundation under Grant CCR-0238484. This paper was recommended by Associate Editor S. Sapatnekar.

The author is with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 USA (e-mail: haizhou@ece.nwu.edu).

Digital Object Identifier 10.1109/TCAD.2003.816211

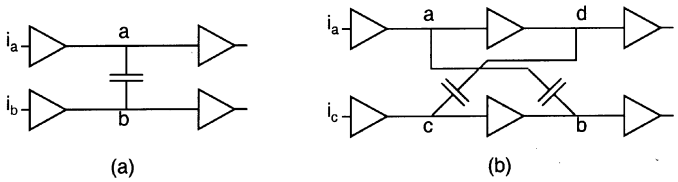

Fig. 1. Timing analysis with crosstalk is a mutual dependence problem. (a) Local problem. (b) Global problem.

Assuming that coupling capacitances and inductances dominate all other capacitances and inductances of a wire, failure to take crosstalk effect into timing analysis may produce results far off from the reality. However, timing and crosstalk effect are mutually dependent. This makes timing analysis with crosstalk a mutual dependence problem. Our work in this paper equally applies to crosstalk induced by capacitive and inductive couplings. However, to simplify the presentation, we only talk about capacitive couplings in the rest of the paper. For example, consider the two coupled nets in Fig. 1(a). The switching time on net a is dependent on the switching time on net b. However, the switching time on net b is not fixed, it is dependent on the switching time on net a.

One way to solve the mutual dependence in timing analysis with crosstalk is by iteration. First, a switching time on a is computed based on a fixed initial switching time on b. Then, the switching time on a is fixed and used to compute a new switching time on b. These two steps are iterated until we find a converged solution. By considering nets a and b simultaneously in one system, the relative window method of Sasaki and De Micheli [16] generates the switching time on a and b directly from the switching time on i_a and i_b. However, it is hard to say that their method avoids iterations since the simulation they use may implicitly use iterations. Furthermore, even in the ideal case, we can directly generate switching time on a and b from switching time on i_a and i_b, we still cannot avoid the mutual dependence problem. This is because there are two kinds of mutual dependence problems in timing analysis with crosstalk. We call the problem in Fig. 1(a), that is, the mutual dependence of a set of directly coupled nets, local mutual dependence problem. Besides that, a circuit structure may introduce cyclic dependences which form a global mutual dependence problem. For example, even though we may be able to generate switching time on a and b directly from those at i_a and c, the time on c (together with the time on d) is not available until we know the time on a.

The current practice to solve these mutual dependence problems relies on iterative approaches. Usually such approaches first assume a situation of crosstalk coupling (often a worst case situation). Then they compute the timing information and use

it to modify the crosstalk coupling situation. They generally do each pass on the whole circuit and iterate till the solution converges.

In this paper, we establish a theoretical foundation for timing analysis with crosstalk. We show that solutions to the problem are fixpoints on a complete lattice. Based on that, we prove in general the convergence of any iterative approach. We also show that, starting from different initial solutions, an iterative approach will reach different fixpoints. The current prevailing practice, which starts from the worst case solution, will always reach the greatest fixpoint, which is the loosest solution. In order to reach the least fixpoint, we need to start from the best case solution. The convergence rates for both discrete and continuous models are discussed. Based on chaotic iteration and heterogeneous structures of coupled circuits, techniques to speed up iterations are also provided.

The rest of the paper is organized as follows. In Section II, we formulate general timing analysis (either dynamic timing simulation or static timing analysis) as computing a fixpoint of a mathematical transformation. In Section III, it is shown that multiple fixpoints generally exist in such systems through an example on a simple coupling delay model. Section IV shows the existence of a fixpoint and the fact that the solution space and its fixpoints form two lattices. Section V proves the convergence of iterative approaches starting from the top and bottom elements, under both discrete and continuous models. In Section VI, the least fixpoint is shown to be the tightest upper bound and, thus, the optimal solution in timing analysis. Section VII presents speeding up techniques based on chaotic iteration and heterogeneous circuit structures. Section VIII discusses some implications of the paper and its relations with other works. Finally, Section IX concludes the paper.

II. Timing Analysis as Fixpoints

In this section, we will formulate general timing analysis—whether it is static timing analysis or dynamic timing simulation—as a fixpoint computation. We are given a combinational circuit that is composed of a set of gates and their interconnections. A set of interconnect wires in the circuit are selected where timing information needs to be computed. They include the primary inputs, primary outputs, and the inputs and outputs of all the gates. Depending on the purpose of timing analysis, the timing information on each of these wires could be a delay, a slew, a switching window, the whole waveform, or any combination of them. In terms of mathematics, it could be a scalar value, a vector, or even a function of time (representing the whole waveform). In general, we use a variable x_i to represent such timing information on a wire i. Furthermore we use X to represent the vector $(x_1, x_2, \ldots, x_n)$, that is, the timing information for the whole circuit.

Depending on the actual delay model, the timing information on wire i depends directly only on a subset of other wires $i_1, i_2, \ldots, i_k$. These wires include all inputs of the gate fanning out to i and the coupled wires with i. It means that we can compute x_i by

$$x_i = t_i(x_{i_1}, \ldots, x_{i_k})$$

where function t_i is decided by the physical configuration of the circuit and the delay model used to compute the timing information on a wire. Such a function for computing the timing information on a wire is called a local transformation. Putting all local transformation together, we get a transformation for the whole circuit which can be written as

$$X = T(X). \tag{1}$$

A solution of timing analysis must be an X satisfying (1). Such a solution is also called a fixpoint of T.

In traditional static timing analysis without crosstalk, each x_i is composed of the minimum and maximum switching time, and only depends on the timing of its fanins. Since the fanin relations form a partial order in a combinational circuit, the timing information on all wires could be computed by one traversal from the primary inputs to the primary outputs. Furthermore, since the minimum time (maximum time) only depends on the minimum time (maximum time) of its fanins, the minimum time analysis could be separated from the maximum time analysis. This also means that different timing information on primary inputs will give different fixpoint. However, the timing information on primary inputs does not appear on the left-hand side of (1) and will be treated as given constants. Therefore, there is only one fixpoint in the traditional static timing analysis without crosstalk.

When crosstalk effects are included in timing analysis, besides the fanins, the timing information on wire i also depends on the timing information on its coupled wires. For the simplest coupling case shown in Fig. 1(a), we have

$$\begin{aligned} x_a &= t_a(x_{i_a}, x_b), \\ x_b &= t_b(x_{i_b}, x_a). \end{aligned}$$

As we can see, a cycle is formed here because of the mutual dependence of x_a and x_b. Thus, the transformation T becomes very complex in the presence of crosstalk.

For a complex transformation T, the iterative method is perhaps the only possible way to find its fixpoint. It works as follows. First, an initial solution X_0 is guessed, then new solutions are iteratively computed from previous solutions $X_1 = T(X_0)$, $X_2 = T(X_1) = T^2(X_0), \ldots$ until we find a fixpoint, that is $X_n = T^n(X_0)$ such that $T(X_n) = X_n$. To the best of our knowledge, all previous works in the literature use iterative methods to solve timing analysis with crosstalk. Most of them [3], [7], [12], [15]–[17], [21], [22], deal only with the local mutual dependence problem, that is, they consider how to compute the delays of a set of coupled nets given their input time. Specifically, Dartu and Pileggi [7] propose to use an effective capacitance gate delay model to model the gates with dominant coupling capacitance. Then, a set of equations are solved by an iterative approach to give the parameters. Gross *et al.* [12] design a waveform iteration approach to explicitly solve the equations in [7]. Sasaki *et al.* [16], [17] propose a relative window method which relates the delays of two coupled nets to their input time. However, their process to compute such relations by simulation may involve iterations implicitly. Iterative method is also used in [3] to compute the Miller factor. Other works [1], [4], [19]

analyze the whole circuit and, thus, considers the global mutual dependence problem.

In order to use an iterative method to find a fixpoint of a transformation T, two basic questions need to be answered.

1) Is T a convergent transformation (at least on a subset A of the domain)? In other words, does there exist a finite n such that $T^{n-1}(x) = T^n(x)\ \forall x \in A$.
2) Does T have a unique fixpoint?

Most previous works provide convergence proofs for their approaches. Gross *et al.* [12] show the convergence based on the approach's similarity to waveform relaxation [14]. Both Sapatnekar [15] and Arunachalam *et al.* [1] base their convergence arguments on the monotonic shrinking of the switching windows. However, none of them study the uniqueness of their solutions. As we will show in the next section, uniqueness is not guaranteed by convergence and simply finding one fixpoint is not enough.

III. Multiple Fixpoints

As will be seen in Section IV, theoretical study of an abstract transformation T can characterize the solution space and the structure of its fixpoints, and establish the convergence of iterative methods. However, it is almost impossible to establish the uniqueness of a fixpoint for an abstract transformation. This is because all theorems on uniqueness of fixpoints in mathematics give only sufficient conditions and these conditions are generally only true for very limited transformations. Banach's fixpoint theorem [2] is well known among such theorems.

Theorem 1 (Banach): Let P be a metric space and $d : P \times P \rightarrow \mathbf{R}$ be its distance function. If function $T : P \rightarrow P$ is a contraction, i.e., there is a constant $k < 1$ such that for any $x, y \in P$

$$d(T(x), T(y)) \leq kd(x, y)$$

then T has a unique fixpoint.

In order to use this theorem, a distance function (metric) has to be defined on the solution space, which may not be an easy task on the vector space in timing analysis. Furthermore, the condition in the theorem is so strong that it guarantees the convergence of an iteration no matter what initial solution is used. However, since it is a sufficient condition, violation of the condition in the theorem does not imply multiple fixpoints.

Generally speaking, unless it is mathematically proved, we cannot assume that a solution to the timing analysis is unique under any given delay model. We will exemplify this through a discrete coupling delay model [9] used by Sapatnekar [15], which is the simplest among existing works. An example in this model shows that there are more than one fixpoint in timing analysis, and starting from different initial solutions, different fixpoints may be reached.

Sapatnekar [15] considered the delay computation in the presence of crosstalk for a set of wires within a routing channel. For each driver, a switching window $[T_{\min}, T_{\max}]$ signifying the range of switching time at the input of the driver, and a source resistance R_d are specified. The intrinsic and coupling capacitances of a wire are computed from the routing. Then, coupling capacitances are modeled by effective capacitances to the ground and delays are computed by the Elmore delay model. The value of an effective capacitance is dependent on the switching time of the two coupling wires. Given a coupling capacitance C_c between two wires, if they switch at the same time and in the opposite direction, then an effective capacitance of $2C_c$ is used; if they switch at the same time and in the same direction, then an effective capacitance of 0 is used; if they do not switch at the same time, then an effective capacitance of C_c is used. However, in static timing analysis, a range of switching time is computed. Thus, the worst case analysis is used, which assumes that any signal switching within the range is possible. The algorithm to compute the wire delays works as follows. First, initialize a switching window on each wire such that the minimum and maximum time are computed by using 0 and C_c, respectively, as effective capacitances. Then, the maximum time on each wire is updated using effective capacitance of C_c or $2C_c$ based on whether switching windows are overlapping. Similarly, the minimum time on each wire is updated using 0 or C_c. These two kinds of updates are repeated in alternate order until there is no further change.

We now use an example to show that multiple fixpoints exist for this transformation. In the example, we have only two nets a and b, as shown in Fig. 1(a). The wires have the same length that gives a load capacitance of 0.5 pF. They also couple with each other with a capacitance of 0.5 pF. We assume that both their drivers have a resistance of 1 $K\Omega$ and the loading gates have input capacitance of 1 **fF**. Suppose the minimum and maximum arrival time of signal i_a, that is $T_{\min}(i_a)$ and $T_{\max}(i_a)$, be 0 and 100 ps, respectively. Similarly, let $T_{\min}(i_b) = 500$ ps and $T_{\max}(i_b) = 600$ ps.

According to the algorithm, the initial switching windows of wires are computed by using effective capacitance of 0 for the minimum time and that of C_c for the maximum time. That is, $[T_{\min}(a), T_{\max}(a)] = [501, 1101$ ps$]$ and $[T_{\min}(b), T_{\max}(b)] = [1001, 1601$ ps$]$. Now, since their switching windows overlap with each other, updates are needed. By using $2C_c$ as the effective capacitance for the maximum time, we get $[T_{\min}(a), T_{\max}(a)] = [501, 1601$ ps$]$ and $[T_{\min}(b), T_{\max}(b) = [1001, 2101$ ps$]$. Since no more update is needed, it is the converged solution.

However, if we assume that there is no switching window overlap at the beginning, we can use C_c as effective capacitance for both the minimum and maximum time. In this case, we have $[T_{\min}(a), T_{\max}(a)] = [1001, 1101$ ps$]$ and $[T_{\min}(b), T_{\max}(b)] = [1501, 1601$ ps$]$ as initial solution. Then, we find that there is no update needed, thus, it is also a converged solution.

IV. Structure of Fixpoints

From previous section, we know that a timing transformation T could have multiple fixpoints. In this section, we will study whether a fixpoint does exist for any given timing transformation and, if so, the structure of the fixpoints.

Since complete information is not used (functionality is not used), uncertainty is unavoidable in static timing analysis. This means that the timing information computed for each wire is a set representing all possible signal switchings, instead of a

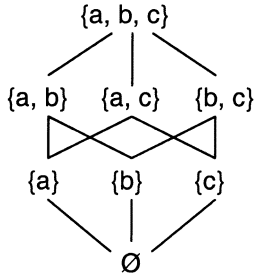

Fig. 2. Subsets of a given set form a complete lattice.

single switching. The timing analysis result guarantees that any physical signal switching is within the set of signal switchings. A set of signal switchings can be represented by a switching window and a range of slew rates such that any signal switching falling within the window and having a slew in the range is in the set. However, other representations are also possible.

Now, consider the family of all sets of signal switchings on a wire. The inclusion relation (that is $\subseteq$) forms a partial order on the family, which is:

- reflexive: $A \subseteq A$;
- antisymmetric: $A \subseteq B \wedge B \subseteq A \rightarrow A = B$;
- transitive: $A \subseteq B \wedge B \subseteq C \rightarrow A \subseteq C$.

Actually, the sets of signal switchings on a wire are subsets of the whole set that consists of all possible signal switchings. According to the lattice theory [8], a partially ordered set forms a *complete lattice* if any subset has a least upper bound and a greatest lower bound of its members. In fact, the family of all subsets of a given set with inclusion relation forms a complete lattice. Given a set $S = \{a, b, c\}$, the partial order of inclusion on its subsets can be represented by a Hasse diagram shown in Fig. 2. Here, two sets are connected by an edge if one is a subset of the other, and the subset is placed below the superset. Timing information of a circuit is a vector of timing information on all wires. The partial order on each wire can be extended point-wise to get a partial order on vectors: two vectors $A = (A_1, A_2, \ldots, A_n)$ and $B = (B_1, B_2, \ldots, B_n)$ satisfy $A \subseteq B$ if and only if $A_i \subseteq B_i$ for all $1 \leq i \leq n$. It can be shown that the vectors with such a partial order also forms a complete lattice.

Now, consider a transformation T. In static timing analysis, it works on a complete lattice we defined previously, that is, it transforms a vector of sets of signal switchings to another vector of sets of signal switchings. We say that T is a *monotonic* (or *order-preserving*) transformation when, for any vector of subsets X and Y, if $X \subseteq Y$ then $T(X) \subseteq T(Y)$. The monotonicity of transformation T can be proved based on the monotonicity of its member transformations $t_1, t_2, \ldots, t_n$. If a transformation t_i is not monotonic, it means that less possible switchings are produced under the condition of the same or more possible switchings on the fanins and coupling wires. This contradicts with the causality of physical effects.

Given a subset S of elements in a complete lattice L, we use $\bigvee S$ and $\bigwedge S$ to represent the least upper bound and the greatest lower bound of elements in S, respectively. The existence of a fixpoint in our system is guaranteed by the following theorem due to Knaster and Tarski [8].

Theorem 2 (Knaster–Tarski): Let L be a complete lattice and $T : L \rightarrow L$ an order-preserving map. Then

$$\bigvee \{x \in L | x \subseteq T(x)\} \in \text{fix}(T)$$

where $\text{fix}(T)$ represents the set of fixpoints of T.

Besides existence, the fixpoints of T—if there are more than one of them—are well-organized. This is stated by the following theorem [8].

Theorem 3: If L is a complete lattice, and $T : L \rightarrow L$ is an order-preserving map, then $\text{fix}(T)$ is a complete lattice.

It means that the greatest fixpoint is the union of all fixpoints and the least fixpoint is the intersection of them.

V. Searching a Fixpoint

Even though the Knaster–Tarski Theorem gives a fixpoint constructively, it cannot be used to compute a fixpoint since it is not feasible to compute the set $\{x \in L | x \subseteq T(x)\}$. The usual method to find a fixpoint is an iterative approach, which is also called *successive approximation*. At the very beginning, a guess of the fixpoint X_0 is used as the initial solution. Then, the mapping T is iteratively applied to the solutions $X_1 = T(X_0), X_2 = T(X_1), \ldots$ in the hope of finding an X_n such that $X_n = T(X_n)$. However, this hope cannot be fulfilled by starting from a random initial solution. For example, it is very possible that there exist $X \neq Y \in L$ such that $T(X) = Y$ and $T(Y) = X$. Therefore, if we unfortunately select X or Y as the initial solution, the iterations will be kept forever.

Fortunately, based on the monotonicity of the timing transformation T, the top and bottom elements are very good candidates for initial solutions. Following a tradition in lattice theory, we use $\bot$ and $\top$ to represent the bottom and the top elements of the complete lattice, respectively. Therefore, we have $\bot = (\emptyset, \emptyset, \ldots, \emptyset)$ and $\top = (P_1, P_2, \ldots, P_n)$ where P_i is the set of all possible signal switchings on wire i. Since $T(\top) \subseteq \top$, based on the monotonicity of T, we have

$$\begin{aligned} T(\top) &\subseteq \top \\ T^2(\top) &\subseteq T(\top) \\ T^3(\top) &\subseteq T^2(\top) \\ &\cdots. \end{aligned}$$

Therefore, we have a descending chain $\top \supseteq T(\top) \supseteq T^2(\top) \supseteq \cdots$. Similarly, since $\bot \subseteq T(\bot)$, based on the monotonicity of T

$$\begin{aligned} \bot &\subseteq T(\bot) \\ &\subseteq T^2(\bot) \\ &\subseteq T^3(\bot) \\ &\cdots \end{aligned}$$

Therefore, we have an ascending chain $\bot \subseteq T(\bot) \subseteq T^2(\bot) \subseteq \cdots$. The discussion of whether and how these chains converge to a fixpoint will be conducted based on the delay models used.

A. Convergence in Discrete Model

A discrete delay model means one that has only a finite number of delay values on each wire. For example, Sapatnekar [15] uses only three possible effective capacitance (0, C, $2C$) for a coupling capacitance, which means that the delay of any

wire can only have a finite number of values. Thus, the model is a discrete model.

In a discrete model, the number of possible switching windows on a wire is finite. This makes the solution space of the whole circuit also finite. Therefore, any chain in the space has only finite elements, which means that an iterative process will always reach a fixpoint within finite steps. In fact, the only properties we used about $\top$ and $\bot$ are $T(\top) \subseteq \top$ and $\bot \subseteq T(\bot)$. Therefore, any solution X_0 such that either $X_0 \subseteq T(X_0)$ or $X_0 \supseteq T(X_0)$ can be used as an initial solution to reach a fixpoint. If $X_0 \subseteq T(X_0)$, we get an ascending chain; if $X_0 \supseteq T(X_0)$, we get a descending chain.

The number of iterations to reach a fixpoint is upper bounded by the length of the longest chain in the lattice. Since the lattice is a vector space of the switching windows of all wires in the circuit, its longest chain length can be computed from the longest chain lengths of its elements.

Lemma 1: If a solution is a vector $X = (x_1, x_2, \ldots, x_n)$ and the length of the longest chain for x_i is l_i, then the length of the longest chain for X is $l = \sum_{i=1}^{n} l_i$.

Proof: Let

$$\begin{aligned} X_1 =& (x_{1,1}, x_{2,1}, \ldots, x_{n,1}) \\ \subset X_2 =& (x_{1,2}, x_{2,2}, \ldots, x_{n,2}) \subset \cdots \subset X_l = (x_{1,l}, x_{2,l}, \ldots, x_{n,l}) \end{aligned}$$

to be the longest chain. We claim that any consecutive X_i, X_{i+1} in the chain have difference only in one element, that is, $x_{k,i} \neq x_{k,i+1}$ if and only if $x_{j,i} = x_{j,i+1}$ for all $1 \leq j \leq n$ and $j \neq k$. If this is not true, then we have $x_{j,i} \subset x_{j,i+1}$ and $x_{k,i} \subset x_{k,i+1}$. Therefore, the chain length can be increased by inserting $X' = (x_{1,i}, \ldots, x_{j,i+1}, \ldots, x_{k,i}, \ldots, x_{n,i})$ between X_i and X_{i+1}. This is a contradiction. We also claim that the number of differences in each x_j in the chain is l_j. If this is not true, we can substitute the longest chain on x_j into the vectors. It will also increase the chain length. Based on these two claims, we know that $l = \sum_{i=1}^{n} l_i$. ∎

This Lemma shows that the iteration number is additive in terms of the number of changes on one wire. In Sapatnekar [15], since the number of changes on one wire is upper bounded by the number of coupled wires, the total number of iterations is upper bounded by the total number of couplings.

B. Convergence in Continuous Model

A continuous delay model means one that has continuous delay values on each wire. In this case, it is very possible that the chains generated in the iterative process have infinite elements. If this happens, an exact fixpoint cannot be generated in finite steps. However, if the iteration converges to a fixpoint, an approximation of the fixpoint can be found. In order for an iteration to converge to a fixpoint, stronger properties than the monotonicity are needed on the transformation T.

Definition 1: A function $T : L \to L$ is *or-continuous* if for any chain C, $T(\bigvee C) = \bigvee\{T(c) | c \in C\}$, or equivalently $T(\bigvee C) = \bigvee T(C)$. If $T(\bigwedge C) = \bigwedge T(C)$, T is called *and-continuous*.

It is easy to show that an or-continuous function is order-preserving: if $x \subseteq y$, then $x \bigvee y = y$, thus $T(x) \subseteq T(x) \bigvee T(y) = T(x \bigvee y) = T(y)$. Similarly, it is easy to show that an and-continuous function is order-preserving: if $x \subseteq y$, then $x \bigwedge y = x$, thus $T(x) = T(x \bigwedge y) = T(x) \bigwedge T(y) \subseteq T(y)$.

Generally speaking, a delay model in timing analysis defines a response waveform as a function of switching waveforms on the fanins and coupling wires. However, in static timing analysis, the model needs to be expanded to compute a set of responses as a function of sets of waveforms on the fanins and coupling wires. A general way to expand a function from a set to its power set is by natural lifting.

Definition 2: Given $f : S \to S$, a natural lifting $F : 2^S \to 2^S$ is

$$F(A) = \bigcup_{a \in A} f(a), \qquad \forall A \in 2^S.$$

A timing transformation defined by a natural lifting is shown to be or-continuous, thus also order-preserving, but not necessarily and-continuous.

Lemma 2: If F is defined by a natural lifting, then it is or-continuous.

Proof: Given any chain $C = \{c_1, c_2, \ldots\}$ on the power set, we need to prove that

$$F\left(\bigcup c_i\right) = \bigcup F(c_i).$$

This can be done by a ping-pong argument. Given any $x \in F(\bigcup c_i)$, there must exist $y \in \bigcup c_i$ such that $f(y) = x$. However, $y \in \bigcup c_i$ means that there exists $j > 0$ such that $y \in c_j$. Therefore, $x = f(y) \in F(c_j)$, which implies that $x \in \bigcup F(c_j)$. On the other hand, given any $x \in \bigcup F(c_i)$, that is $x \in F(c_j)$ for some j, there must be $y \in c_j$ such that $x = f(y)$. Obviously $y \in \bigcup c_i$, thus $x = f(y) \in F(\bigcup c_i)$. ∎

To see why a function defined by the natural lifting may not be and-continuous, consider the following example. Let $S = (0, 1)$, an open interval between 0 and 1, and $f(x) = 1$ for all $x \in S$. For any $A \subseteq S$, if $A \neq \emptyset$ then $F(A) = \{1\}$. However, $F(\emptyset) = \emptyset$. Now, consider a decreasing chain $c_i = (0, 1/i)$ for all $i > 0$. We have $F(c_i) = \{1\}$ for all $i > 0$, thus $\bigcap F(c_i) = \{1\}$. However, since $\bigcap c_i = \emptyset$, we have $F(\bigcap c_i) = \emptyset$. Therefore $F(\bigcap c_i) \neq \bigcap F(c_i)$.

The following theorem shows that we need to use different approaches to find fixpoints for or-continuous and and-continuous functions.

Theorem 4: If T is or-continuous, then $\bigvee_{n \geq 0} T^n(\bot)$ is a fixpoint; if T is and-continuous, then $\bigwedge_{n \geq 0} T^n(\top)$ is a fixpoint.

Proof: Since $T^n(\bot)$ and $T^n(\top)$ for $n \geq 0$ are two chains, based on the definition of or-continuity and and-continuity, we have

$$\begin{aligned} T\left(\bigvee_{n \geq 0} T^n(\bot)\right) &= \bigvee_{n \geq 0} T^{n+1}(\bot) \\ &= \bigvee_{n > 0} T^n(\bot) \\ &= \bot \vee \bigvee_{n > 0} T^n(\bot) \end{aligned}$$

$$\begin{aligned} T\left(\bigwedge_{n \geq 0} T^n(\top)\right) &= \bigwedge_{n > 0} T^n(\top) \\ &= \top \wedge \bigwedge_{n > 0} T^n(\top). \end{aligned}$$

The properties that $\perp \bigvee x = x$ and $\top \bigwedge x = x$ for any x are used. ■

From Theorem 4 and the fact that natural lifting may not be and-continuous, it is very possible that a fixpoint may not be reached from the top element for a function defined by the natural lifting. Since the natural lifting is a popular way to extend a function from points to subsets, this provides one strong reason why timing analysis iterations should start from the bottom elements. The other reason is provided in the next section.

VI. Optimal Fixpoint

The following theorem shows that if fixpoints are found by the iterative method from the bottom and top elements, they must be the least and the greatest fixpoints.

Theorem 5: Let L be a complete lattice and $T : L \to L$ an order-preserving map. Define $\alpha := \bigvee_{n\geq 0} T^n(\perp)$ and $\beta := \bigwedge_{n\geq 0} T^n(\top)$.

1) If $\alpha \in \text{fix}(T)$, then α is the least fixpoint;
2) If $\beta \in \text{fix}(T)$, then β is the greatest fixpoint.

Proof: For any $x \in \text{fix}(T)$, we have $\perp \subseteq x$. Based on the monotonicity of T, we have $T^n(\perp) \subseteq T^n(x) = x$ for any $n \geq 0$. Therefore, $\bigvee_{n\geq 0} T^n(\perp) \subseteq \bigvee_{n\geq 0} x = x$. Similarly, we have $x \subseteq \top$ and thus $x = T^n(x) \subseteq T^n(\top)$ for any $n \geq 0$. Therefore, $x = \bigwedge_{n\geq 0} x \subseteq \bigwedge_{n\geq 0} T^n(\top)$. ■

From Theorem 3, the greatest fixpoint is the union of all fixpoints and the least one is the intersection of them. In terms of switching window, it means that a fixpoint with the largest switching windows will result from an initial assumption that all switching windows overlap with each other, and a fixpoint with the smallest windows will result from an initial assumption that no window overlaps with the other. Since the worst case scenario is used in delay models, any fixpoint in the timing analysis is an upper bound of the physical switching time. That is, if all the input signals switch within the given input windows, all other signals are guaranteed to switch within the windows given by the fixpoint. This means that the least fixpoint is the tightest upper bound and the greatest fixpoint is the loosest upper bound. *Therefore, in order to find the optimal fixpoint—the tightest upper bound or the solution with the minimum uncertainty—we should start with an initial solution where no window overlaps with the other.*

Now, consider the practice of changing delay model during iterations. One possible scenario is starting with a coarse estimation and gradually changing into more and more accurate models. We must be very careful with this practice, since the transformation is now changing with iterations and the convergence may not be guaranteed. Suppose the transformation in the ith iteration is T_i. If we start with $\perp$, we have $\perp \subseteq T_1(\perp), T_2(\perp) \subseteq T_2(T_1(\perp)), \ldots$, but this only gives us a chain if we also have $T_i(X) \subseteq T_{i+1}(X)$, which means a later model should not be more accurate than a previous one. So the practice of using finer and finer model can not be used in any iterative approach with increasing windows, or you take the risk of looping infinitely. On the other hand, it can be safely used in approaches with decreasing windows since we always have a chain

$$\top \supseteq T_1(\top) \supseteq T_2(\top) \supseteq T_2(T_1(\top)) \supseteq \cdots.$$

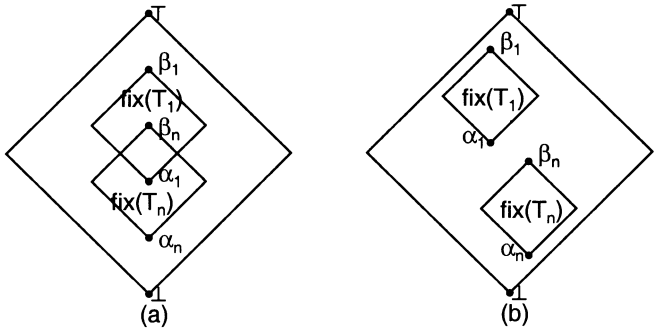

Fig. 3. Structures of fixpoints.

However, this does not imply that an approach with decreasing windows is more efficient than that with increasing windows. Let α_1 and β_1 to be the least and greatest fixpoints of T_1 respectively. Similarly, let α_n and β_n to be the least and greatest fixpoints of T_n. For any $x \in \text{fix}(T_1)$ we have $x = T_1(x) \supseteq T_n(x)$ which means a fixpoint of T_n can be found if we start with x. Using a diamond to represent a complete lattice, the relation between $\text{fix}(T_1)$ and $\text{fix}(T_n)$ can be shown in Fig. 3. Depending on whether $\alpha_1 = T_n(\alpha_1)$, we have two cases. If $\alpha_1 = T_n(\alpha_1)$, then α_1 is a fixpoint of T_n, thus $\alpha_1 \subseteq \beta_n$. Since we also know that $\beta_n \subseteq \beta_1$, we can prove that $\beta_n = T_1(\beta_n)$. This is shown in Fig. 3(a) where the fixpoint sets of T_1 and T_n overlap. In this case, using finer and finer models from $\top$ will find β_n, but keeping with the coarse model from $\perp$ will give us a better solution α_1. In the case shown in Fig. 3(b), α_1 is not a fixpoint of T_n, but since $\alpha_1 \supseteq T_n(\alpha_1)$, we can iterate using T_n from α_1 to get β_n. That means we can keep using the coarse model until we find a fixpoint, then change to a finer model. However, in both cases, in order to find the best fixpoint α_n, the only way is to use T_n from $\perp$, which means that the solution using the coarse model has to be discarded.

VII. Speeding up Iterations

In a strict sense, applying the transformation T to a solution X_i in an iteration, i.e., computing $X_{i+1} = T(X_i)$, must use only the previous values to compute the new values even when some of the new ones are available. This is similar to Jacobi's method in solving matrix equations [11]. The usual practice (e.g., Chen *et al.* [4]), however, already deviates from this strict sense: the updating is done in a topological order of the circuit and a new value is always used if it is available. This is much like Seidel's method to solve matrix equations. Many useless updates are, thus, trimmed off. However, without further exploiting both circuit and coupling structures and their interaction, many updates are still wasted. For example, in Fig. 4, if updates are processed according to a topological order of the circuit, any update at d must be propagated to e, f, g, and h. However, if the update at d is not permanent, those propagations are wasted since they will be overwritten later.

Before designing a good update order, we need to establish its theoretical validity. That is, no matter what order is used, the process will always converge to the same fixpoint. This is called the scheme of *chaotic iteration* [6]. Here, a transformation T is composed of a set of partial transformations $t_1, t_2, \ldots, t_n$.

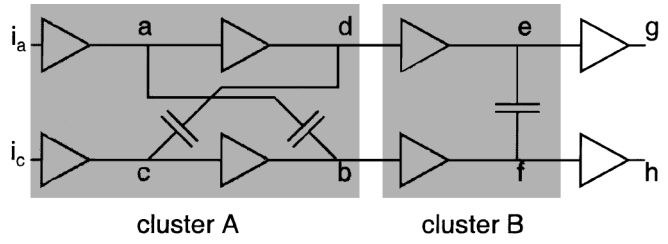

Fig. 4. Exploit circuit structure by clustering.

In each step, one or more partial transformations are applied to update timing information on one or more wires. All timing information on other wires is kept the same. We will use T_S to represent such a partial transformation done in one step, where S represents the set of wires where timing information is updated.

Lemma 3: Given a subset S of wires, if $X \subseteq T(X)$, then $X \subseteq T_S(X) \subseteq T(X)$; if $X \supseteq T(X)$, then $X \supseteq T_S(X) \supseteq T(X)$.

Proof: Let $X = (x_1, x_2, \ldots, x_n)$, $T(X) = (y_1, y_2, \ldots, y_n)$, and $T_S(X) = (z_1, z_2, \ldots, z_n)$. Then, for any wire i, if $i \in S$ then $z_i = y_i$. Otherwise, $z_i = x_i$. Therefore, if $X \subseteq T(X)$, that is, $x_i \subseteq y_i$ for all $1 \leq i \leq n$, then $x_i \subseteq z_i \subseteq y_i$ for all i. On the other hand, if $X \supseteq T(X)$, that is, $x_i \supseteq y_i$ for all $1 \leq i \leq n$, then $x_i \supseteq z_i \supseteq y_i$ for all i. ■

This lemma states that no matter what evaluation order is used, the generated sequence is monotonic in the same direction and it will not over-shoot the fixpoint generated by T. Furthermore, if the evaluation order is fair, that is, a partial transformation will always be applied if its inputs and outputs are not consistent, then the chaotic iteration will always reach the same fixpoint as T.

The structure of a circuit gives fan-in relations on the wires that can be represented by a direct acyclic graph (DAG). Since each coupling capacitor introduce a bidirectional edge on the pair of wires, we have a general directed graph. On this graph, each of the strongly connected components is identified and called a cluster. The timing analysis is then done in a topological order of the clusters, iterating within one cluster to convergence before moving to the next one. For example, in Fig. 4, two clusters A and B are identified. Not processing cluster B until having a stable cluster A means that previous mentioned useless updates will be trimmed off. Notice that clusters A and B have different structures. Actually, cluster B is simpler and corresponding to the local mutual dependence problem, so it is called a *local cluster*. Cluster A includes some local clusters forming a feedback loop and is corresponding to the global mutual dependence problem, so it is called a *global cluster*.

There are many different ways to arrange iterations in a global cluster. If we always compute local clusters together, as implicitly suggested by [12], [1], the structure of a global cluster can be viewed as in Fig. 5, where each block represents a local cluster. To facilitate iterations, a set of gate inputs are selected as feedback edges whose removal makes the structure acyclic. Given initial values on feedback edges, timing analysis can be done in the acyclic part (perhaps with a complicated computation on each local cluster) to give new values on feedback edges. If these values become stable (i.e., new values are the same as old values), then a fixpoint is reached. Otherwise, next iteration will start with the new values. Given feedback edges, iterations in each global cluster can be processed in two ways. The first approach, called iterative approach, recomputes the whole cluster based on new values and repeats this until all values become stable. The second approach, called recursive approach, only recomputes the wires on the outer cycle after all inner cycles become stable. Although there is no direct relation between the number of iterations and the number of feedback edges, fewer feedback edges may give fewer possible value changes. However, finding the smallest number of feedbacks is NP-hard on a general graph [10].

One drawback of the approach is that, since all the feedbacks are gate inputs, value changes on them always need to be propagated. Studying the interactions in our system, we find that our system is a heterogeneous system. That is, there are two kinds of interaction relations: fanin relation is simple but strong; coupling relation is complex but weak. Fanin relation is simple because it is unidirectional, and it is strong because signal switching on input always influences signal switching on output. On the other hand, coupling relation is complex because it is bidirectional, and it is weak because signal switching on one net may not always influence the switching on the other. Based on these observations, it is good to use the following principle: *always treat coupling edges as feedback edges.* It works as follows. Initially, assume all switching windows are empty, that is, no coupling net switches at the same time. Thus, the first iteration is just a traditional timing analysis. After this iteration, a switching window is given on each wire. Then for each pair of coupling wires a and b, their timing proximity can be defined as

$$P(a,b) = \min(T_{\max}(a), T_{\max}(b)) - \max(T_{\min}(a), T_{\min}(b)).$$

In fact, it defines the overlap length of the two switching windows; when there is no overlap, it is negative and its absolute value defines the distance between the two windows. A set of coupling edges with the smallest timing proximities are selected as feedback edges whose removal makes the circuit acyclic (in the sense of treating each local cluster as a block). That is, the circuit will be viewed in the structure shown in Fig. 6. In the sequel iterations, the feedback edges are not physically broken up, but the timing computation on the two wires connected by each of them is separated. For example, when computing the time on wire a in Fig. 6, its coupling with wire b is considered but time on wire b is assumed to be fixed. Later, when computing time on wire b, its coupling with wire a is considered, but time on wire a is assumed to be fixed. Therefore, if the switching windows on a and b do not overlap during the iterations, time change on one wire does not influence the other wire. Our selection of feedback edges based on timing proximity intends to make this happen as frequently as possible.

VIII. Implications and Discussions

Except for the example in Section III, our paper does not depend on any specific delay or coupling models. This is intended to achieve generality and the "separation of concerns"—we want to separate the iteration mechanism from the actual coupling delay modeling. It makes all results proved in

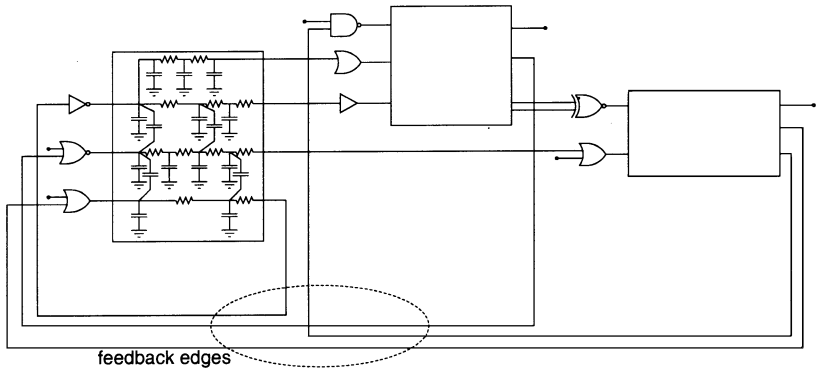

Fig. 5. Use gate fanins as feedback edges.

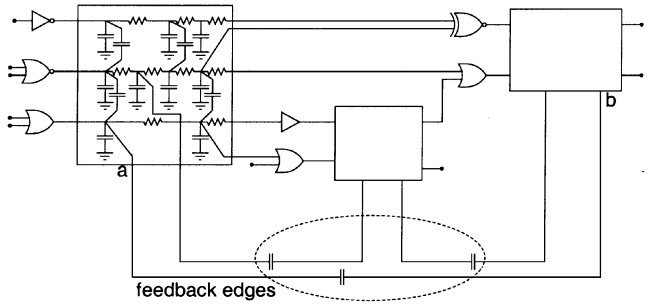

Fig. 6. Use coupling edges as feedback edges.

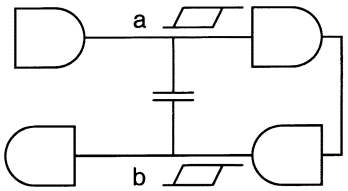

Fig. 7. Coupling between a signal and its transitive fanout.

this paper applicable to any specific model. However, there are some caveats worth of discussion.

First, using a specific coupling model, the example in Section III only shows that multiple fixpoints exist in that model. However, it also shows that *the possibility of multiple fixpoints cannot be excluded except so proved in any model*. In this aspect, it is interesting to note that Chen *et al.* [4] reported that their experiments reach the same result starting from different initial solutions. Since their coupling delay model is continuous, Kirkpatrick [13] simply conjectured that a continuous model will have a unique fixpoint. Following up our work [23], Chen *et al.* [5] used numerical fixpoint theory to explain many results in continuous models. However, their effort to prove the uniqueness of fixpoints in continuous models did not succeed. It is still an open question whether a continuous model, or specifically Chen's model [4], always has a unique fixpoint.

Second, even though our framework is applicable to any coupling delay model, the accuracy of timing analysis is heavily dependent on the model. For example, it is very common for a signal to couple with its transitive fanout signals in a circuit. One such situation is shown in Fig. 7, where a signal a is coupled with its transitive fanout b. In one class of coupling delay models, the coupling effect is calculated by updating the envelope waveform of the victim's switching window under aggressor attacks [4]. In Fig. 7, assume signal a's right envelop overlaps with signal b's switching window. Then, an updating will push the envelop of a to the right, which then will enlarge the switching window of b. This results in an infinite loop and pushes the envelops to the infinity. To solve this problem, Chen *et al.* [4] used a causality based iteration scheme to reach a finite result if the coupling is between signals along a path. In terms of our framework, this scheme only defines a different coupling model and works within our framework. To make this clear, consider another coupling delay model where the switching window of a signal depends on the windows of its fanin signals plus an adjustment based on couplings[1]. In this model, the right envelop of signal a has an upper bound and will converge within finite steps.

Finally, a critical point in our framework is to understand a switching window as a set of switching waveforms. This enables representations other than switching windows. It is also a point to distinguish between the iteration mechanism and the coupling modeling: anything to relate a waveform to other waveforms belongs to modeling; anything based on this to establish relations among subsets of waveforms belongs to iteration mechanism. The monotonicity of timing transformation function proved in the paper is based on this view of waveform sets. Therefore, correct interpretation from a representation to the set it represents is very important in a timing analysis system. For example, a switching window generally represents all switching waveforms within the window[2]. However, using only envelop waveforms of a switching window, as is done in [4], [5], may not correctly model this interpretation. The nonconvergence problem discussed in Chen *et al.* [5] is caused by this problem. The nonmonotonicity they discussed belongs to the coupling delay model and is different from the monotonicity we proved in this paper.

Based on our discovery, Thudi and Blaauw [20] conducted Sapatnekar's iterative approach [15] from both the worst case and the best case initial solutions. Their results showed that the differences between the window sizes are ranged from 0.1% to 6%.

[1]For example, use $0\times$, $2\times$, or $1\times$ as effective capacitance based on whether they have coupling attack or not.

[2]Deciding what are the waveforms within a window is not an easy task, especially when slew rates need to be decided.

IX. Conclusion

In this paper, we established a solid theoretical foundation for timing analysis in the presence of crosstalk. It is first shown that timing analysis with crosstalk is to seek a fixpoint on a global timing transformation function. A specific delay model is used as an example to demonstrate that multiple fixpoints are very common for those transformations. The solution space is shown to form a complete lattice. Based on this and the monotonicity of the transformation function, fixpoints always exist and form another complete lattice. The convergences of iterative approaches starting from the top and bottom elements are firmly established. Since all the fixpoints are upper bounds of physical timing events, the least fixpoint is the optimal solution (tightest upper bound) which can be computed starting from the no-window-overlap assumption. Techniques exploiting the structures of the circuit and the couplings are also developed to speed up the iterations.

References

[1] R. Arunachalam, K. Rajagopal, and L. T. Pilleggi, "Taco: Timing analysis with coupling," in *Proc. Design Automation Conf.*, Los Angeles, CA, June 2000, pp. 266–269.

[2] B. Bollobas, *Linear Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1990.

[3] P. Chen, D. A. Kirkpatrick, and K. Keutzer, "Miller factor for gate-level coupling delay calculation," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2000.

[4] ——, "Switching window computation for static timing analysis in presence of crosstalk noise," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2000.

[5] P. Chen, Y. Kukimoto, C.-C. Teng, and K. Keutzer, "On covergence of switching windows computation in presence of crosstalk noise," in *Proc. Int. Symp. Physical Design*, 2002, pp. 84–89.

[6] P. Cousot and R. Cousot, "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *Proc. ACM Symp. Principles of Programming Languages*, Los Angeles, CA, Jan. 1977, pp. 238–252.

[7] F. Dartu and L. T. Pileggi, "Calculating worst-case gate delays due to dominant capacitance coupling," in *Proc. Design Automation Conf.*, Anaheim, CA, June 1997, pp. 46–51.

[8] B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order*. Cambridge: Cambridge Univ. Press, 1990.

[9] L. Gal, "On-chip cross talk – The new signal integrity challenge," in *Proc. Custom Integrated Circuits Conf.*, 1995, pp. 251–254.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA: Freeman, 1979.

[11] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[12] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi, "Determination of worst-case aggressor alignment for delay calculation," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1998, pp. 212–219.

[13] D. A. Kirkpatrick, private communication, 2002.

[14] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuites and systems," *IEEE Trans. Computer-Aided Design*, vol. 1, pp. 131–145, July 1982.

[15] S. S. Sapatnekar, "A timing model incorporating the effect of crosstalk on delay and its application to optimal channel routing," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 550–559, 2000.

[16] Y. Sasaki and G. De Micheli, "Crosstalk delay analysis using relative window method," in *Proc. ASIC/SoC Conf.*, 1999, pp. 9–13.

[17] Y. Sasaki and K. Yano, "Multi-aggressor relative window method for timing analysis including crosstalk delay degradation," in *Proc. Custom Integrated Circuit Conf.*, 2000, pp. 495–498.

[18] (2001) International technology roadmap for semiconductors. Semiconduct. Ind. Assoc. [Online]. Available: http://public.itrs.net.

[19] P. F. Tehrani, S. W. Chyou, and U. Ekambaram, "Deep sub-micron static timing analysis in presence of crosstalk," in *Proc. Int. Symp. Quality Electronic Design*, 2000, pp. 505–512.

[20] B. Thudi and D. Blaauw, "Non-iterative switching window computation for delay noise," IEEE Trans. Computer-Aided Design, 2002, submitted for publication.

[21] A. Vittal, L. H. Chen, M. Marek-Sadowska, K.-P. Wang, and S. Yang, "Crosstalk in VLSI interconnections," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 1817–1824, Dec. 1999.

[22] T. Xiao, C.-W. Chang, and M. Marek-Sadowska, "Efficient static timing analysis in presence of crosstalk," *Proc. 13th Annu. IEEE Int. ASIC/SOC Conf.*, pp. 335–339, 2000.

[23] H. Zhou, N. Shenoy, and W. Nicholls, "Timing analysis with crosstalk as fixpoints on a complete lattice," in *Proc. Design Automation Conf.*, 2001, pp. 714–719.

Hai Zhou received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1992 and 1994, respectively, and the Ph.D. degree in computer sciences from the University of Texas at Austin in 1999.

He is currently an Assistant Professor in electrical and computer engineering at Northwestern University, Evanston, IL. Before he joined the faculty of Northwestern, he was with the Advanced Technology Group, Synopsys, Inc., Mountain View, CA. His research interests include VLSI computer-aided design and the design and analysis of algorithms.