Timing Macro-modeling of IP Blocks with Crosstalk*

Ruiming Chen and Hai Zhou
Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208

Abstract

With the increase of design complexities and the decrease of minimal feature sizes, IP reuse is becoming a common practice while crosstalk is becoming a critical issue that must be considered. This paper presents two macro-models for specifying the timing behaviors of combinational hard IP blocks with crosstalk effects. The gray-box model keeps a coupling graph and lists the conditions on relative input arrival time combinations for couplings not to take effect. The black-box model stores the output response windows for a basic set of relative input arrival time combinations, and computes the output arrival time for any given input arrival time combination through the union of some combinations in the basic set. Both macro-models are conservative, and can greatly reduce the pessimism existing in the conventional "pin-to-pin" model. This is the first work to deal with timing macro-modeling of combinational hard IP blocks with the consideration of crosstalk effects.

1 Introduction

With the progress of deep sub-micron technologies, 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 large as the sum of the area capacitance and the fringing capacitance, and the trends indicate that the role of coupling capacitance will be even more dominant in the future as feature sizes shrink. This makes crosstalk a major problem in IC design. Crosstalk introduces noise between adjacent wires, and even alters the functions of circuits. If an aggressor and a victim switch simultaneously on the same direction, the victim will speed up. Likewise, if an aggressor and a victim switch on the opposite directions, the victim will slow down.

With the growing complexity of VLSI systems, the intellectual property (IP) reuse is becoming a common practice. There are previous researches dealing with the timing macro-modeling of IP blocks, and all of them are based on the concept of path delay [1, 2, 3, 4, 5, 6]. The simplest of such models is the "pin-to-pin" delay model used to characterize standard cells and other complex combinational circuits. For example, given a simple combinational circuit shown in Figure 1(a), its "pin-to-pin" delay model is shown in Figure 1(b). The numbers shown on each arc give the minimal and maximal delays from one pin to another. In this case, if $a(x)$ and $A(x)$ are used to represent the earliest and latest arrival time of signal x, respectively, we have

$$a(x) = a(a) + 3; A(x) = A(a) + 5;$$

$$a(y) = min(a(b)+2, a(c)+3); A(y) = max(A(b)+4, A(c)+4).$$

To model a sequential circuit with memory elements, the "pin-to-pin" model is extended to include timing constraints from a clock pin to an input pin (to model setup and hold conditions) and delay arcs from a clock pin to an output pin (to model the latch output to circuit output delay) [1]. Functionality may also be used to reduce the pessimism in these models [5, 6].

*This work was supported by NSF under CCR-0238484.

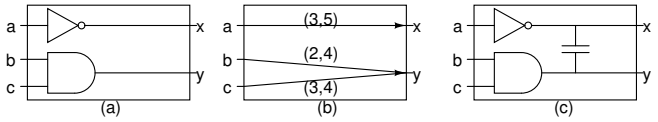

Figure 1: (a) A simple circuit; (b) Its "pin-to-pin" macro-model; (c) Coupling destroys path delay concept.

Unfortunately, coupling totally destroys the path delay concept. For example, as shown in Figure 1(c), if the wires x and y are coupled to each other, the arrival time on x and y cannot be decided through paths from the inputs. Instead, the relative switching time of a, b, c is important in deciding the arrival time on x and y. Suppose when

$$a(a) = A(a) = 0, a(b) = A(b) = 1, a(c) = A(c) = 0,$$

we have the maximal delay variation on the coupled signals x and y such that

$$(a(x), A(x)) = (2, 10), (a(y), A(y)) = (1, 11).$$

Then for any t such that

$$a(a) = A(a) = t, a(b) = A(b) = t + 1, a(c) = A(c) = t,$$

the delay variation will also be maximal and we have

$$(a(x), A(x)) = (t + 2, t + 10), (a(y), A(y)) = (t + 1, t + 11).$$

This means that the delays between pins are influenced by the relative arrival time of inputs. So the conventional "pin-to-pin" macro-model for IP blocks is pessimistic.

Many researches have been done on timing analysis or modeling considering crosstalk [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Most of them consider how to compute the delays of a set of coupled nets given their input time, so the resulting delays are accurate only for a specific input arrival time combination. Sasaki et al. [17, 18] proposed relative window methods, and Agarwal et al. [19] proposed an analytical method, to depict the delay change curves for noise-aware static timing analysis (STA). However, they can only handle the simple cases with one victim and multiple aggressors.

So when considering crosstalk effects, all the current macro-modeling methods cannot generate an *accurate* model for a *complex* IP block. We propose two input-dependent models for specifying the timing behaviors of complex IP blocks. To the best of our knowledge, it is the first work dealing with timing characterization of IP blocks with crosstalk effects.

The rest of the paper is organized as follows. Section 2 shows the requirements of a viable macro-model. Section 3 presents the extraction and application of SIMPMODEL. This fast gray-box macro-model computes and utilizes the conditions on the relative input arrival time combinations for couplings not to take effect. Section 4 present the extraction and application of UNIONMODEL. This accurate black-box model stores the output response windows for a basic set of relative input arrival time combinations, and computes the

output arrival time for any given input arrival time combination through the union of some stored output response combinations. Section 5 reports the experimental results on the proposed macro-models and their comparison with "pin-to-pin" model and STA results. Finally, the conclusion and future work are discussed in Section 6.

2 Macro-model requirements

A viable timing macro-model of combinational hard IP block should satisfy the following requirements:

- Hiding of implementation details. It is a requirement for protecting intellectual property.
- Model accuracy. Given an input arrival time combination, the arrival time window of any primary output generated by the timing macro-model should be as close as possible to the corresponding actual output arrival time window.
- Conservative modeling. Given an input arrival time combination, the actual time window of any primary output should be in the range of the corresponding output time window generated by the timing macro-model.

When crosstalk effects are considered, the delay between pins depends not only on the structural detail of IP blocks, but also on the relative arrival time of primary inputs, or *input pattern*s.

Definition 1 (input/output pattern) *An input/output pattern of an IP block is an arrival time window vector where each element is an arrival time window on one primary input/output of the IP block.*

So an *input-dependent* model may greatly improve the accuracy. As is said before, when considering crosstalk effects, all the current macro-modeling methods *cannot* generate an *accurate* model for a *complex* IP block. We introduce two input-dependent macro-models that satisfy all the requirements.

3 SIMPMODEL

3.1 Delay model

The conventional "pin-to-pin" model assumed that the coupling effects were always active, which led to a very pessimistic estimation. In order to get a more accurate model, we compute and utilize the conditions under which the couplings do not take effect.

Hassoun [20] presented a dynamically bounded delay model to represent the delay of a net. We use a modified dynamically bounded delay model. A directed graph $G = (V, E, C)$ represents the IP block. Each vertex in V represents a primary input, a primary output, a gate, or a net. A connection connecting inputs, outputs, gates or nets is represented by one edge in E. Let set C_v be the set of aggressor nodes connected via a coupling capacitor to victim node v, W_v be the timing window of node v, and

$$C = \cup_{v \in V}\{C_v\}$$

Each node v has a dynamically bounded delay model consisting of a fixed delay range $[\delta_v, \Delta_v]$, and, for each coupling capacitor attached to v from an aggressor node a, a predicate $\gamma_{v,a}$ indicating whether the coupling takes effect. If the coupling does not take effect, the minimal delay should be increased by $\delta_{v,a}$, and the maximal delay should be decreased by $\Delta_{v,a}$. So the minimal delay of node v is

$$\delta_v + \sum_{a \in C_v} \gamma_{v,a}\delta_{v,a},$$

and the maximal delay of node v is

$$\Delta_v - \sum_{a \in C_v} \gamma_{v,a}\Delta_{v,a},$$

where $\gamma_{v,a}$ is defined as

$$\gamma_{v,a} = \begin{cases} 0 & \text{if } W_v \text{ overlaps with } W_a, \\ 1 & \text{otherwise.} \end{cases}$$

3.2 SIMPMODEL details

First, the SIMPMODEL reduces the original complex directed acyclic graph (DAG) G to a DAG called *coupling graph* that contains only the input pins, the output pins, and the nodes with coupling. Each edge in this coupling graph represents that the end node is reachable from the start node through a path without extra coupling nodes, and the weights of each edge represent the minimal and the maximal delays from the start node to the end node. This task can be accurately done by any conventional STA.

The remaining task is to determine the value of $\gamma_{v,a}$. When considering crosstalk effects, the determination of $\gamma_{v,a}$ becomes a chicken-and-egg problem, and the conventional way is to use iteration methods. With the coupling graph, the macro-model users can use STA tools to get the output pattern directly. Since the coupling relations are often complex, this kind of macro-model leaves much work to users. Instead, we design a fast conservative approximation method to determine the value of $\gamma_{v,a}$.

It is obvious that the value of $\gamma_{v,a}$ is determined by the relations among the time windows of primary inputs that can reach v or a. Let set I_v be the set of primary inputs that can reach node v, v_i be the ith input that can reach node v, b_{v_i} and w_{v_i} be the minimal and maximal delays from input v_i to v respectively, $[x_{v_i}, y_{v_i}]$ is the arrival time window of input v_i. Then we have these two rules:

$$\min_{v_i \in I_v}(x_{v_i} + b_{v_i}) > \max_{a_j \in I_a}(y_{a_j} + w_{a_j}) \Rightarrow \gamma_{v,a} = 1,$$

and

$$\max_{v_i \in I_v}(y_{v_i} + w_{v_i}) < \min_{a_j \in I_a}(x_{a_j} + b_{a_j}) \Rightarrow \gamma_{v,a} = 1.$$

After getting the coupling graph, SIMPMODEL assumes that $\gamma_{v,a} = 0$ for all the coupling nodes. Then do a PERT traversal on the coupling graph to calculate the minimal and maximal delays from each primary input to each node, and put the results into a list L. Till now, a model composed by a coupling graph and a result list is extracted successfully.

Then during the model application, once the input pattern is given, we can check the two rules to determine the value of each $\gamma_{v,a}$ according to the information in the list L. Once the delay of each node is determined, we traverse the coupling graph to get the desired output pattern.

We assume that $\gamma_{v,a} = 0$ for all the coupling nodes, so after checking these two rules, if $\gamma_{v,a} = 0$ for a pair of coupling nodes v and a, it maybe equal to 1 in reality, but if $\gamma_{v,a} = 1$ for a pair of coupling nodes v and a, it must be 1 in reality, so SIMPMODEL always achieves a conservative output pattern for a given input pattern.

4 UNIONMODEL

SIMPMODEL shows a method to fast estimate the output pattern, while its accuracy is sacrificed to get a high speed, and SIMPMODEL is a gray-box model, that is, it unveils part of the implementation details of IP blocks. However, the accuracy of the model and the hiding of implementation

details are strongly required in many situations. The following UNIONMODEL satisfies these two requirements, that is, UNIONMODEL is an accurate black-box timing macromodel.

Let A_i represent the time window of pattern A corresponding to pin i. First, we introduce some definitions.

Definition 2 (adjacent windows) *If two windows a and b overlap with each other at only one common point, then a and b are adjacent, denoted as $a \diamond b$.*

Definition 3 (combinable input patterns) *If the windows corresponding to the same pin in two patterns A and B are adjacent, and the other pairs of windows corresponding to the same pins are the same respectively, then A and B are combinable, denoted as $A \simeq B$*

Definition 4 (subwindow) *If window a is contained in window b, then a is a subwindow of b, denoted as $a \subseteq b$.*

Definition 5 (subpattern) *Let A and B be two patterns. If for any i, $A_i \subseteq B_i$, then A is a subpattern of B, denoted as $A \subseteq B$.*

Definition 6 (overlapped patterns) *If each window in pattern A overlaps with the corresponding window in pattern B, then A and B are overlapped patterns, denoted as $A \cap B \neq \phi$.*

Definition 7 (union of patterns) *The union of two overlapped patterns A and B, denoted as $A \cup B$, is a pattern where each window is the union of the two corresponding windows in A and B.*

For example, in Figure 2, time windows a, b, c, d are $[0, 10]$, $[10, 20]$, $[0, 10]$ and $[10, 20]$, respectively. Input patterns $A\{a, c\}$ and $B\{b, c\}$ are combinable input patterns that can be united into an input pattern $\{[0, 20], c\}$. But input pattern $\{a, c\}$ and $\{b, d\}$ are not combinable, because a and b are adjacent, and c and d are adjacent but not the same.

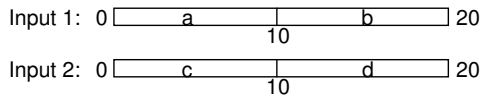

Figure 2: An example of input patterns.

It is obvious that combinable input patterns are also overlapped patterns. Combinable input patterns have another important property:

Lemma 1 *Suppose A and B are input patterns for an IP block, if $A \simeq B$, let input pattern $X = A \cup B$, then the output pattern for X is the union of the output patterns for A or B.*

For example, in Figure 2, the output window for input pattern {[0,20],[0,10]} is the same with the union of output patterns for input patterns {[0,10],[0,10]} and {[10,20],[0,10]}. UNIONMODEL uses this property to model the timing behavior of combinational IP blocks.

4.1 Model extraction

First, a wide range input pattern $\mathcal{P}$ is generated. Since sequential circuits dominate the reality, and combinational parts are embedded in sequential circuits, the arrival time of primary inputs of a combinational circuit is between 0 and T, where T is the clock period of the sequential circuit. We can make a reasonable assumption that T is upper-bounded by a value denoted as T_{max}, then we choose the arrival time window $[0, T_{max}]$ as $\mathcal{P}_i$, $i = 1, \ldots, n$, where n is the number of primary inputs.

Then, each window in $\mathcal{P}$ is evenly partitioned into k parts, where k is a positive integer. For each primary input, we pick one part from the corresponding k parts as an input window in the resulting small range input pattern, then we can construct k^n distinct input patterns, called *basic patterns*.

Performing STA or simulations on the IP block, we can get the output pattern for each basic pattern. The results are put into a table T: each entry corresponds to one basic pattern and its output pattern. This table T provides all the required information for UNIONMODEL application.

4.2 Model application

Given any input pattern I, the windows in I are shifted the same distance to make I a subpattern of $\mathcal{P}$. If failed, then this case cannot use UNIONMODEL, and the "pin-to-pin" model will be used. If succeeded, the output pattern can be easily calculated by union operations on output patterns in the table.

First, based on Lemma 1, if we can construct an input pattern P satisfying $I \subseteq P$ by union operations on basic patterns in the table T, then the output pattern constructed by uniting all the output patterns of these basic patterns is a conservative approximation of the output pattern of I. We designed a procedure called Pattern-Construction that can complete this construction successfully, which is shown in Figure 3. The subroutine Extr-CmbPattern(S, i, A, B) searches for a pair of combinable patterns A and B satisfying $A_i \diamond B_i$ in pattern set S. If this search succeeds, A and B are removed from S and *true* is returned. Procedure Pattern-Construction searches and puts all the basic patterns overlapping with I into pattern set S, then for each primary input i, it unites all the combinable patterns found by Extr-CmbPattern. After the ith outer loop, the following condition is satisfied:

$$\forall P \in S : I_i \subseteq P_i.$$

Thus, at the end of this procedure, S contains only one pattern, and I is the subpattern of this pattern.

Algorithm `Pattern-Construction`

```
Notations:
T:  a table storing all the basic patterns
I:  a given input pattern
n:  the number of windows in I

Procedure:
S = {P : (P ∈ T) ∧ (P ∩ I ≠ Φ)}
for i = 1 to n
   while(Extr-CmbPattern(S, i, A, B)=true)
      put X = A ∪ B into S;
return S
```

Figure 3: Input pattern construction.

For example, in Figure 2, suppose the input pattern is

$$I\{[2, 19], [3, 18]\}.$$

Initially, S contains patterns $A\{a, c\}$, $B\{a, d\}$, $C\{b, c\}$ and $D\{b, d\}$, then in the first outer iteration, $i = 1$, we first unite A and C to get pattern $E\{[0, 20], [0, 10]\}$, then unite B and D to get pattern $F\{[0, 20], [10, 20]\}$, so before the second outer iteration, S contains E and F, and I_1 is a subwindow of E_1 and F_1. Then in the second outer iteration, $i = 2$, we unite E and F to get a pattern $X\{[0, 20], [0, 20]\}$. Obviously I is a

subpattern of X. From this procedure, we can see that the resulting input pattern is the same with the union of all the basic patterns overlapping with I. So based on Lemma 1, we have the following theorem:

Theorem 1 *The output pattern for an input pattern I can be conservatively calculated by uniting the output patterns for basic patterns that overlap with I.*

Thus, it is not required to do the expensive *input* pattern unions. Instead, we search for all the basic patterns overlapping with I, then look up the table T to find and unite all the *output* patterns corresponding to these basic patterns to get the desired output pattern directly. And during this union step, we do not need to search for the combinable patterns and unite them pair by pair. Instead, for each primary output, we can simply select a window composed by the earliest and the latest arrival time among the corresponding windows of these output patterns as the corresponding output window of the desired output pattern. Based on this union procedure, we know that UNIONMODEL achieves the conservative output pattern for any given input pattern.

Because of the large number of table entries, the most expensive step is to search for the basic patterns that overlap with I in the table. We need to design a method to find them efficiently. We label the primary inputs by $1 \dots n$, and label the k parts in each window of $\mathcal{P}$ by $1 \dots k$ in the increasing order of the earliest arrival time. Then each basic pattern can be represented by a distinct key with radix k: $(a_n \dots a_1)_k$, where $0 \le a_i \le k-1$ for $i = 1 \dots n$, and the ith window of this pattern is the (a_i+1)th part in $\mathcal{P}_i$. Then we put the input patterns and their corresponding output patterns in a table in the increasing order of keys. For a given input pattern I, we can find the range of parts $[i_p, i_q]$ in $\mathcal{P}_i$ overlapping with I_i for $i = 1 \dots n$, then the keys of the basic patterns that overlap with I are all the distinct keys satisfying $i_p \le a_i \le i_q$ for $i = 1 \dots n$. Obviously this method can easily find the basic patterns that overlap with I in the table.

An important contribution of UNIONMODEL is that it provides a framework for timing macro-modeling of combinational hard IP blocks with the consideration of crosstalk effects, that is, the obtainment of output patterns for basic patterns is not restricted to STA. Instead, other timing analysis or simulation methods to calculate the output patterns can be embedded into this macro-model. Also we can easily incorporate functional information into this macro-model.

4.3 Speed-up techniques

The bottleneck of UNIONMODEL is the large number of basic patterns. However, from Section 1 we have known that if input pattern A can be constructed by shifting input pattern B, the output pattern of A can also be constructed by shifting the output pattern for B. We call that A and B are *relatively the same*. Obviously many basic patterns are relatively the same. For example, in Figure 2, the corresponding windows in input patterns {a,c} and {b,d} are shifted 10 unit time respectively, so these two patterns are relatively the same. So it is not necessary to perform timing analysis on all the basic patterns respectively. We only perform timing analysis on the basic patterns that are not relatively the same with each other. Our experiments show that this technique can reduce the total number of basic patterns nearly half.

The number of basic patterns in UNIONMODEL is k^n, where k is the number of parts of an input window in $\mathcal{P}$, and n is the number of inputs. So it is very efficient to speed-up the extraction of UNIONMODEL if we can reduce the number of inputs. In reality, it is possible to partition the input set into several disjoint sets such that the relative arrival time of any two inputs in different sets has no influence on the same coupling nodes. Then the number of basic patterns in UNIONMODEL is $\sum_{1 \le i \le u} k^{n_i}$, where u is the number of disjoint sets, and n_i is the number of inputs in the ith disjoint set. This correlated input set partitioning technique will divide a large size problem to many small size problems, which will speed-up the overall running much. Our experiments show that this method can reduce the number of basic patterns greatly for some cases.

When the range of input windows in $\mathcal{P}$ is large, we need to partition each input window in $\mathcal{P}$ into many parts to guarantee the accuracy, so the number of basic patterns increases greatly. In order to avoid this problem, we randomly select some input patterns, do STA or simulation to get the output patterns, then we can find the range of input patterns where delay changes frequently, so we can shrink the range of input windows in $\mathcal{P}$ to this range, and for the input patterns that cannot be shifted into $\mathcal{P}$, we use conventional "pin-to-pin" model. This range shrinking step can greatly reduce the number of basic patterns, and benefit the model extraction and application speed.

5 Experimental results

SIMPMODEL and UNIONMODEL have been implemented in C++ and tested on ISCAS85 benchmark circuits. For each circuit, we randomly designated the coupling between wires, and input patterns are also randomly generated. All experiments were run on a Linux PC with a 2.4G Hz Xeon CPU and 2.0 GB memory.

To verify the results of our methods, we designed a STA tool based on an iterative switch factor method that works as follows. First, it uses the modified dynamically bounded delay model to model the delay of coupling nodes, then iteratively does PERT-traversal on the graph representing circuit, and after each iteration updates all the $\gamma_{v,a}$. It does not stop until there is no change on all $\gamma_{v,a}$, and the output pattern in last PERT-traversal is the desired. We also implemented a "pin-to-pin" model for comparison, which assumed that all couplings take effect. For each case, we compare the results from our model applications with the results from the "pin-to-pin" model. Suppose for the same primary input, $\{[d_1, d_2]\}$ is the output pattern from our models, $\{[a_1, a_2]\}$ is the output pattern from STA, then the error of our models for this primary output is defined as

$$[(d_2 - d_1) - (a_2 - a_1)]/(a_2 - a_1).$$

The average error is the summation of the errors for all the primary outputs divided by the number of primary outputs. The maximal error is the maximum of the errors for the primary outputs.

A comparison of results among STA, SIMPMODEL and "pin-to-pin" model is shown in Table 1, where ETime is the time of model extraction, ATime is the time of model application, MaxE is the maximal error, and AveE is the average error. We can see that the results of SIMPMODEL are much more accurate than the "pin-to-pin" model, and the running time of the application of SIMPMODEL is always less than 1 second, much faster than STA in large cases. The errors are always non-negative, which confirms that SIMPMODEL is conservative.

Since UNIONMODEL can only deal with cases with a small number of correlated primary inputs, for the cases with a large number of correlated primary inputs, we designated the arrival time windows of most primary inputs to be invalid windows $[\infty, -\infty]$ and modeled only the timing behavior of the circuit stimulated by the remaining primary inputs that are denoted by RPI in our test. We calculated the output

Table 1: Comparison results of STA, SIMPMODEL and "pin-to-pin" model

circuit				STA	SIMPMODEL				"pin-to-pin" model	
name	#inputs	#outputs	#gates	time(s)	ETime(s)	ATime(s)	MaxE(%)	AveE(%)	MaxE(%)	AveE(%)
c17	5	2	6	0.00	0.00	0.00	0.00	0.00	0.06	0.03
c432	36	7	160	0.04	0.23	0.05	69.21	11.18	72.23	12.29
c499	41	32	202	0.05	0.42	0.11	35.00	1.62	46.21	6.72
c880	60	26	383	0.22	1.33	0.39	1309.11	141.15	8848.00	629.36
c1355	41	32	546	0.29	1.02	0.21	20.97	3.02	26.10	6.43
c1908	33	25	880	0.68	1.39	0.07	28.60	1.82	32.73	10.67
c3540	50	22	1669	2.30	3.82	0.16	212.72	11.49	212.72	20.54
c5315	178	123	2307	4.71	9.76	0.13	168.08	1.68	12458.40	107.48
c6288	32	32	2416	5.08	6.38	0.12	0.00	0.00	8.01	0.87
c7552	207	108	3513	10.2	16.26	0.12	32.06	0.41	13073.70	122.99

Table 2: Comparison results of STA and UNIONMODEL

circuit		STA	UNIONMODEL			
name	RPI #	time (s)	# basic patterns	ETime (s)	ATime (s)	MaxE (%)
c17	5	0.00	27,000	1.99	8.94	0.00
c432	5	0.03	3,200,000	2133.00	204.63	0.01
c499	5	0.06	1,048,576	1374.00	245.58	0.01
c1355	4	0.33	65,536	129.24	14.80	0.00
c5315	5	4.65	256	5.52	0.65	0.00

pattern for each basic pattern by STA. A comparison of the results from UNIONMODEL with STA is shown in Table 2. We can see that the results of UNIONMODEL are almost the same with the results of STA, that is, UNIONMODEL is an accurate model. From Table 2 we can also see that although the numbers of RPI in c499 and c5315 are the same, the number of basic patterns in c5315 is much less than in c499, which is the effect of less number of correlated inputs in c5315 than in c499. This confirms that our correlated input set partition improves the model extraction and application speed.

6 Conclusion and future work

We present two macro-models to characterize the timing behavior of combinational hard IP block with the consideration of crosstalk effects. The first model, SIMPMODEL, keeps a coupling graph and lists the conditions on input patterns for couplings not to take effect. It can fast estimate the output pattern for a given input pattern with the sacrifice of accuracy. The second model, UNIONMODEL, performs STA or simulations on basic input patterns, and based on these results constructs the output pattern for a given input pattern accurately. Both macro-models are conservative, and can greatly reduce the pessimism existing in the traditional "pin-to-pin" model. Since the number of basic patterns is large when the size of correlated input set is large, the extraction and application of UNIONMODEL become prohibitive. So an important future task is to find a better way to reduce the number of basic patterns. To the best of our knowledge, this is the first work to deal with timing macro-modeling problem with the consideration of crosstalk effects.

References

[1] A. J. Daga, L. Mize, S. Sripada, C. Wolff, and Q. Wu. Automated timing model generation. In *DAC*, pages 146–151, 2002.

[2] C. W. Moon, H. Kriplani, and K. P. Belkhale. Timing model extraction of hierarchical blocks by graph reduction. In *DAC*, pages 152–157, 2002.

[3] M. Foltin, B. Foutz, and S. Tyler. Efficient stimulus independent timing abstraction model based on a new concept of circuit block transparency. In *DAC*, pages 158–163, 2002.

[4] S. V. Venkatesh, R. Palermo, M. Mortazavi, and K. Sakallah. Timing abstraction of intellectual property blocks. In *CICC*, pages 99–102, 1997.

[5] H. Yalcin, M. Mortazavi, R. Palermo, C. Bamji, and K. Sakallah. Functional timing analysis for IP characterization. In *DAC*, pages 731–736, 1999.

[6] H. Yalcin, R. Palermo, M. Mortazavi, C. Bamji, and K. Sakallah. An advanced timing characterization method using mode dependency. In *DAC*, pages 657–660, 2001.

[7] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi. Determination of worst-case aggressor alighment for delay calculation. In *ICCAD*, pages 212–219, San Jose, CA, November 1998.

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

[9] S. S. Sapatnekar. A timing model incorporating the effect of crosstalk on delay and its application to optimal channel routing. *IEEE TCAD*, 2000.

[10] 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, November 2000.

[11] T. Xiao, C.-W. Chang, and M. Marek-Sadowska. Efficient static timing analysis in presence of crosstalk. In *Proceedings of 13th Annual IEEE International ASIC/SOC Conference*, pages 335–339, 2000.

[12] P. F. Tehrani, S. W. Chyou, and U. Ekambaram. Deep submicron static timing analysis in presence of crosstalk. In *International Symposium on Quality Electronic Design*, pages 505–512, 2000.

[13] T. Xiao and M. Marek-Sadowska. Functional correlation analysis in crosstalk induced critical paths identification. In *DAC*, pages 653–656, 2001.

[14] H. Zhou, N. Shenoy, and W. Nicholls. Timing analysis with crosstalk as fixpoints on a complete lattice. In *DAC*, pages 714–719, 2001.

[15] P. Chen, Y. Kukimoto, C.-C. Teng, and K. Keutzer. On covergence of switching windows computation in presence of crosstalk noise. In *ISPD*, pages 84–89, 2002.

[16] B. Thudi and D. Blaauw. Non-iterative switching window computation for delay noise. In *DAC*, pages 390–395, 2003.

[17] Y. Sasaki and G. De Micheli. Crosstalk delay analysis using relative window method. In *ASIC/SoC Conference*, 1999.

[18] Y. Sasaki and K. Yano. Multi-aggressor relative window method for timing analysis including crosstalk delay degradation. In *Custom Integrated Circuit Conference*, pages 495–498, 2000.

[19] K. Agarwal, Y. Cao, T. Sato, D. Sylvester, and C. Hu. Efficient generation of delay change curves for noise-aware static timing analysis. In *Proceedings of 15th International Conference on VLSI Design*, pages 77–84, 2002.

[20] S. Hassoun. Critical path analysis uing a dynamically bounded delay model. In *DAC*, pages 260–265, Los Angeles, CA, June 2000.