Propagation of Last-Transition-Time Constraints in Gate-Level Timing Analysis

Maroun Kassab[1], Eduard Cerny[1], Sidi Aourid[1], Thomas Krodel[2]

[1]Laboratoire LASSO, Dép. IRO, Université de Montréal
C.P. 6128, Succ. Centre-Ville, Montréal (Québec) H3C 3J7 Canada
email: {kassab,cerny,aourid}@iro.umontreal.ca
[2]Nortel, PO Box 3511 Station C, Ottawa ON, K1Y 4H7 Canada
email: krodel@nortel.ca

Abstract

Waveform narrowing is an attractive framework for circuit delay verification as it can handle different delay models and component delay correlation efficiently. The method can give false negative results because it relies on local consistency techniques. We present two methods to reduce this pessimism: 1) global timing implications and necessary assignments, and 2) a case analysis procedure that finds a test vector that violates the timing check or proves that no violation is possible. Under floating-mode, global implications eliminate timing check violation without case analysis in the c1908 benchmark, while for a tighter requirement case analysis finds a test vector after only 5 backtracks.

1. Introduction

Verifying delays in gate-level circuits is more difficult as the details of the design are fading away from the view of the designers who must rely on tools that synthesize blocks from descriptions in high level languages and then connect them together, often manually. Due to circuit complexity, traditional manual verification or simulation is not suitable, and considering just the topological delay of the circuit is too conservative and may be costly in useless redesign effort. Unfortunately, computing the delay of a combinational circuit is an NP-Complete problem. Consequently, exact methods are exponential in nature, and more research into heuristics and user interfaces is required to satisfy industrial expectations from static timing verification tools.

The complexity of the problem is caused by the fact that in general not all signal paths in a circuit can propagate transitions (false paths). When the longest paths are false, the actual delay of the circuit is less than this value, i.e., less than the topological delay of the circuit. Hrapcenko [12] presented early an extended discussion on the subject, and proved that minimal circuits may have delays that are less than the topological delays. The circuit shown in Figure 1 illustrates this point [12]. Many techniques have been developed to deal with the problem of false path [1]-[9]. Path oriented timing verifiers suffer from poor performance as they may have to enumerate a very large number of paths, however, it is possible to improve the performance by memorizing inconsistencies between sub-paths [9], and thus reduce the search space. In [5] the authors reduced the problem of determining whether the delay of the circuit is greater than δ to an ATPG problem. In [6] an exact method based on timed Boolean functions and an OBDD representation was formulated, however, it may experience exponential space explosion for certain circuits. Our group has developed a method based on abstract waveform narrowing [2] inspired by constraint logic programming (CLP) using relational interval arithmetic [17]. It can efficiently handle component delay correlation [1] and adapt to different circuit-delay modes (two-vector transition or floating mode) by a simple change in the abstract waveforms applied to the inputs of the circuit.

No exact delay calculation algorithm performs efficiently on all circuits, and the time complexity is exponential on some circuits. A trade-off between tightness of the upper bound on the max. circuit delay and the efficiency of the method is thus necessary. The method of [1, 2] is an excellent candidate for such a trade-off. Exact answers can still be obtained by performing case analysis over the waveforms on certain circuit nets, guided by heuristics. In this paper we show how the addition of global timing implications and a new heuristic for the selection of nets for the case analysis can considerably improve the performance of the algorithm. When combined with delay correlation [1], the resulting algorithm is closer to what is needed in industrial applications. Presently we are integrating our engine with a timing verifier developed at Nortel, with the objective to test the method on industrial circuits.

The paper is organized as follows: Section 2 gives basic definitions and terminology. Section 3 formalizes the problem of whether a circuit has a floating delay greater

than or equal to δ as a constraint satisfaction problem. Sections 4 and 5 introduce global timing implications and the improved case analysis. Section 6 contains experimental results and Section 7 concludes the presentation.

2. Definitions and Terminology

A ***combinational logic circuit*** is represented by a directed acyclic graph $\xi = (gates, nets)$ where gates are the vertices and delayless nets are the edges.

A ***gate*** is one of: AND, NAND, OR, NOR, NOT, BUFFER, DELAY, XOR, XNOR. Delays are intervals $[d_{min}, d_{max}]$, although in the max. floating-mode delay calculation only the d_{max} bound is used.

A ***path*** is an alternating sequence of nets and gates (n_0, g_0, n_1, g_1,..., n_k, g_k, n_{k+1}). n_0 is a net connected to an input of g_0, n_{k+1} is a net connected to the output of gate g_k, and n_j connects the output of gate g_{j-1} to an input of g_j, j=1 to k. A ***side input*** of a path (n_0, g_0, n_1, g_1,..., n_k, g_k, n_{k+1}) is any input net of gate g_l, l = 0, ..., k, other than n_l. The ***path length*** d_{path} is the sum of the d_{max} delays of the DELAY elements lying on the path. The ***topological delay*** *top* of a circuit ξ is the length of its longest path. The topological delay top_n of a net *n* is the length of the longest path starting at a primary input and ending at *n*. The topological delay $top_{n1\text{-}n2}$ between two nets n_1 and n_2 is the length of the longest path starting at n_1 and ending at n_2.

A ***controlling value*** at a gate input uniquely determines the gate output value. A ***non controlling value*** at a gate input is a value that is not *controlling*.

The ***floating mode circuit delay*** is the minimum time after which all the outputs of the circuit settle to a final stable value for all possible input vectors applied at time 0, assuming that the initial state of all circuit nets is unknown (non-controlling).

A ***timing-check*** is a tuple $\sigma = (\xi, s, \delta)$ where ξ is a combinational circuit, *s* is a primary output of ξ, and δ is an (integer) delay value. It represents the following decision problem: *Does s of circuit ξ have a delay greater than or equal to δ ?*

3. Overview of Waveform-Narrowing

A *timing-check* $\sigma = (\xi, s, \delta)$ is transformed into a constraint system that is consistent iff the output *s* has a delay greater than or equal to δ. A constraint system is composed of a finite set of variables $\{X_1, X_2,..., X_n\}$ which take values from their respective domains $D_1, D_2,..., D_n$, and a set of relational constraints $\{C_1, C_2,..., C_m\}$, each specifying which values of the variables are mutually compatible. The variables and the relational constraints represent the signal values on circuit nets and the gates, respectively. The specific circuit-delay mode and the output timing constraint δ introduce further restrictions on the domains.

3.1 Domains

A real digital signal waveform is a mapping $f: R \rightarrow R$. Abstracted to a binary waveform in discrete time, it becomes a mapping $f: Z \rightarrow \{0,1\}$. The space of all binary waveforms is $\boldsymbol{BW} = \{ f: Z \rightarrow \{0, 1\} \}$.

3.1.1 Abstract Waveforms

Definition 1: An ***abstract waveform*** (AW) is a subset of $\boldsymbol{BW}$ defined as $w = v|_{lmin}^{max} = \{ f \in \boldsymbol{BW} \mid \forall t > max.\ f(t) = v \wedge \exists t' \in [lmin, max].\ f(t') \neq v \}$. The *abstract waveform* space is $\boldsymbol{AW} = \{ v|_{lmin}^{max} \mid v \in \{0, 1\}, lmin, max \in Z \}$.

$v|_{lmin}^{max}$ contains binary waveforms that are stable at value *v* after time *max* and undergo the last transition at or after time *lmin*. Obviously, not any subset of $\boldsymbol{BW}$ can be represented by an *abstract waveform*. This leads to some approximations when the union operation is defined on *abstract waveforms* as it is not equivalent to the corresponding set union. References to *v*, *lmin* and *max* of an *abstract waveform w* are denoted *w.v*, *w.lmin* and *w.max*, respectively. *w.v* is the ***class*** and [*w.lmin*, *w.max*] the ***last-transition interval*** of *w*. If *w.lmin* > *w.max* then [*w.lmin*, *w.max*] is empty and *w* itself is also empty, denoted by $w = \phi$.

Relations and Operations on AW having the same *class* are defined as follows:
Equality: $w_1 = w_2$ iff $w_2.max = w_1.max \wedge w_2.lmin = w_1.lmin$ or both are empty.
Narrowness: w_1 is said to be ***narrower*** than w_2, denoted $w_1 < w_2$ iff $(w_1.max \leq w_2.max \wedge w_1.lmin > w_2.lmin) \vee (w_1.max < w_2.max \wedge w_1.lmin \geq w_2.lmin)$. $w_1 \leq w_2$ iff $(w_1 < w_2) \vee (w_1 = w_2)$.

Narrowing an *abstract waveform* means changing its *lmin* and/or *max* to make it narrower than its previous value. An *abstract waveform* w_1 that is *narrower* than w_2 contains fewer *binary waveforms* than w_2.
Inclusion: $w_1 \subset w_2$ iff $w_1 \leq w_2$. **Intersection:** $w_1 \cap \phi = \phi$, $\phi \cap w_1 = \phi$, $(w_1 \neq \phi) \wedge (w_2 \neq \phi) \Rightarrow w = w_1 \cap w_2$ with $w.v = w_1.v = w_2.v$ and $w.lmin$ = maximun($w_1.lmin$, $w_2.lmin$) and $w.max$ = minimum($w_1.max$, $w_2.max$). **Union:** $w_1 \cup \phi = w_1$, $\phi \cup w_1 = w_1$, $(w_1 \neq \phi) \wedge (w_2 \neq \phi) \Rightarrow w = w_1 \cup w_2$ with $w.v = w_1.v = w_2.v$ and $w.lmin$ = minimum($w_1.lmin$, $w_2.lmin$) and $w.max$ = maximum($w_1.max$, $w_2.max$).

Lemma 1: If $(w_1 \neq \phi) \wedge (w_2 \neq \phi)$ then $(w_2.max + 1 \geq w_1.lmin) \wedge (w_1.max + 1 \geq w_2.lmin) \Leftrightarrow (w_1 \cup w_2 = \{ f \in \boldsymbol{BW} \mid (f \in w_1 \vee f \in w_2) \})$.

The result of performing $\boldsymbol{AW}$ union may in general include binary waveforms that were not originally included in the operands. $w = w_1 \cup w_2$ includes, beside w_1 and w_2, a minimal subset of $\boldsymbol{BW}$ that makes both w_1 and w_2 representable by a single AW, and no *w'* narrower than *w* contains both w_1 and w_2.

3.1.2 Abstract Signals

Definition 2: An *abstract signal* S is a pair of abstract waveforms $(\underline{w}, \overline{w})$: $\underline{w}.v = 0$ and $\overline{w}.v = 1$.
The components $\underline{w}$ and $\overline{w}$ of S are denoted $\underline{S}$ and $\overline{S}$, respectively. The space of all *abstract signals* is $\boldsymbol{AS} = \{w \in \boldsymbol{AW} | w.v = 0\} \times \{w \in \boldsymbol{AW} | w.v = 1\}$; it is the domain of the variables in the constraints in our method.

Relations and Operations on *AS*.
Equality: $S_1 = S_2$ iff $(\underline{S}_1 = \underline{S}_2) \wedge (\overline{S}_1 = \overline{S}_2)$.
Narrowness: $S_1 < S_2$ iff $((\underline{S}_1 < \underline{S}_2) \wedge (\overline{S}_1 \leq \overline{S}_2)) \vee ((\underline{S}_1 \leq \underline{S}_2) \wedge (\overline{S}_1 < \overline{S}_2))$; $S_1 \leq S_2$ iff $(\underline{S}_1 \leq \underline{S}_2) \wedge (\overline{S}_1 \leq \overline{S}_2)$.
Inclusion: $S_1 \subset S_2$ iff $S_1 \leq S_2$.
Intersection: $S_1 \cap S_2 = (\underline{S}_1 \cap \underline{S}_2, \overline{S}_1 \cap \overline{S}_2)$.
Union: $S_1 \cup S_2 = (\underline{S}_1 \cup \underline{S}_2, \overline{S}_1 \cup \overline{S}_2)$.

3.2 Gate Constraints

Gate constraints are derived from the Boolean gate functions. Let $g: \boldsymbol{BW} \times \boldsymbol{BW} \rightarrow \boldsymbol{BW}$ be the timed Boolean function [6] of a 2-input gate *G*. For example, in the case of a 2-input AND with fixed delay *d*, $g(I_1(t), I_2(t)) = I_1(t-d) \cdot I_2(t-d)$. Let D_i, D_j, and D_s be the domains in $\boldsymbol{AS}$ associated with the variables of the inputs X_i, X_j, and the output X_s, respectively. Let x_i, x_j, and x_s be subsets of $\boldsymbol{BW}$ such that:

$x_i = \{w \in \boldsymbol{BW} | w \in \underline{D}_i \vee w \in \overline{D}_i\}$,
$x_j = \{w \in \boldsymbol{BW} | w \in \underline{D}_j \vee w \in \overline{D}_j\}$,
$x_s = \{w \in \boldsymbol{BW} | w \in \underline{D}_s \vee w \in \overline{D}_s\}$,

The projections of D_i, D_j, and D_s to the terminals of G are then as follows:

$x'_i = \{w_1 \in x_i | \exists w_2 \in x_j, \exists w_3 \in x_s, g(w_1, w_2) = w_3\}$,
$x'_j = \{w_2 \in x_j | \exists w_1 \in x_i, \exists w_3 \in x_s, g(w_1, w_2) = w_3\}$,
$x'_s = \{w_3 \in x_s | \exists w_1 \in x_i, \exists w_2 \in x_j, g(w_1, w_2) = w_3\}$.

The constraint relation $C_g(X_i, X_j, X_s)$ derived from *g*, is an operator that changes the values of D_i, D_j, D_s as to become the *narrowests* possible to contain x'_i, x'_j and x'_s, respectively. The projections and the operators on $\boldsymbol{AS}$ are used to define a system of equations over $\boldsymbol{AS}$ which is solved by computing the greatest fixpoint [1, 2].

Example 1: let $D_i = (0|_{-\infty}^{33}, 1|_{50}^{100})$, $D_j = (0|_{25}^{75}, \phi)$ and $D_s = (0|_{35}^{125}, \phi)$. Applying the constraints of the 2-input AND gate with delay 0 to D_i, D_j, D_s (D_s at output) yields the following new values: $D'_i = (\phi, 1|_{50}^{100})$, $D'_j = (0|_{35}^{75}, \phi)$ and $D'_s = (0|_{35}^{75}, \phi)$.

3.3 Constraint System

Given a *timing-check* $\sigma = (\xi, s, \delta)$, the construction of the constraint system is straight forward following the circuit description. Let $\xi(\{Gate_1, Gate_2, ..., Gate_m\}, \{Net_1, Net_2, ..., Net_n\})$ be the circuit of *m* gates and *n* nets where each gate is connected to a subset of the nets. We build a constraint system composed of *n* variables $X_1, X_2, ..., X_n$ associated with the *n* domains $D_1, D_2, ..., D_n$, respectively, and *m* relational constraints $C_1, C_2, ..., C_m$, where C_i operates on the domains corresponding to the variables of the nets connected to $Gate_i$. The initial values for all the domains of the constraint system of σ are $(0|_{-\infty}^{+\infty}, 1|_{-\infty}^{+\infty})$ so as to contain any possible BW. For floating-mode delay calculation, we restrict the primary input domains to waveforms that are stable after time 0: $\boldsymbol{F} = (0|_{-\infty}^{0}, 1|_{-\infty}^{0})$. To verify if the output *s* has a delay greater than or equal to δ, we restrict the signal domain of *s* to the waveforms having transitions at or after time δ, i.e., $D_s = (0|_{\delta}^{+\infty}, 1|_{\delta}^{+\infty})$.

The constraint system is tightened (solved) by repeatedly applying the local projections of domains as induced by the gate constraints (Section 3.2) until no *narrowing* of any domain is possible, i.e., the (unique) greatest fixpoint of the system of equations is reached. We implemented this iterative computation efficiently using an event-driven scheduler. It also includes selective state saving needed for backtracking in case analysis.

Definition 3: Given a *timing-check* $\sigma = (\xi, s, \delta)$ and its corresponding constraint system composed of the variables $X_1, X_2, ..., X_n$, their respective domains $D_1, D_2, ..., D_n$, and the constraints $C_1, C_2, ..., C_m$, a *binary waveform* $w \in D_k$ is said to be σ-***compatible***, iff it is part of a solution, i.e., iff there is a waveform in each D_i, $i \neq k$, such that with *w* from D_k, the constraint system is satisfied. *w* is said to be σ-***incompatible*** if it is not σ-*compatible*.

Theorem 1: The fixpoint of the evaluation is reached in a finite number of steps.

Theorem 2: If $D_s = \{\phi, \phi\}$ then no transition is possible on output *s* at or after time δ.

Example 2: Consider the *timing-check* $\sigma = (\xi, s, 61)$ where ξ is the circuit of Figure 1 [12]. Assuming the max. delay of 10 on the output of each gate, *top* = 70 and the *floating-mode* delay is 60, because the path $\{n_1, g_2, n_2, g_3, n_3, g_4, n_4, g_6, n_6, g_7, n_7, g_8, s\}$ is false. We now illustrate our method on this example. Let $D_{e_1}, D_{e_2}, D_{e_3}, D_{e_4}, D_{e_5}, D_{e_6}, D_{e_7}, D_{n_1}, D_{n_2}, D_{n_3}, D_{n_4}, D_{n_5}, D_{n_6}, D_{n_7}, D_s$ be the domains associated with the variables of the corresponding nets. The initial values are: $D_{e_i} = (0|_{-\infty}^{0}, 1|_{-\infty}^{0})$, $i \in \{1, 2, 3, 4, 5, 6, 7\}$: the floating-mode inputs; $D_{n_i} = (0|_{-\infty}^{+\infty}, 1|_{-\infty}^{+\infty})$, $i \in \{1, 2, 3, 4, 5, 6, 7\}$: any possible waveform; $D_s = (0|_{61}^{+\infty}, 1|_{61}^{+\infty})$: only the waveforms that violate the timing check, i.e., transitions after time 60.
Waveforms propagation yields: $g_1 \Rightarrow D_{n_1} = (0|_{-\infty}^{10}, 1|_{-\infty}^{10})$ the maximal delay of g_1 is 10; therefore, no transition is possible on n_1 after time 10; $g_2 \Rightarrow D_{n_2} = (0|_{-\infty}^{20}, 1|_{-\infty}^{20})$;

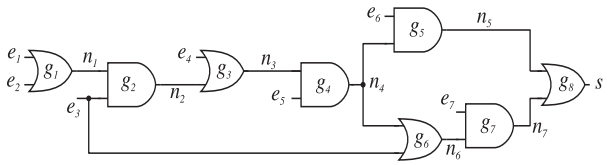

Figure 1 False path circuit

$g_3 \Rightarrow D_{n_3} = (0|_{-\infty}^{30}, 1|_{-\infty}^{30})$; $g_4 \Rightarrow D_{n_4} = (0|_{-\infty}^{40}, 1|_{-\infty}^{40})$; $g_5 \Rightarrow D_{n_5} = (0|_{-\infty}^{50}, 1|_{-\infty}^{50})$; $g_6 \Rightarrow D_{n_6} = (0|_{-\infty}^{50}, 1|_{-\infty}^{50})$; $g_7 \Rightarrow D_{n_7} = (0|_{-\infty}^{60}, 1|_{-\infty}^{60})$; $g_8 \Rightarrow D_s = (0|_{61}^{70}, 1|_{61}^{70})$ and $D_{n_5} = (0|_{-\infty}^{50}, \phi)$ and $D_{n_7} = (0|_{51}^{60}, 1|_{51}^{60})$: the *last-transition-interval* on *s* is propagated to n_7 and the controlling waveforms on n_5 are removed because they block the way on n_7; $g_7 \Rightarrow D_{n_6} = (0|_{41}^{50}, 1|_{41}^{50})$ and $D_{e_7} = (\phi, 1|_{-\infty}^{0})$; $g_6 \Rightarrow D_{n_4} = (0|_{31}^{40}, 1|_{31}^{40})$ and $D_{e_3} = (0|_{-\infty}^{0}, \phi)$; $g_4 \Rightarrow D_{n_3} = (0|_{21}^{30}, 1|_{21}^{30})$ and $D_{e_5} = (\phi, 1|_{-\infty}^{0})$; $g_3 \Rightarrow D_{n_2} = (0|_{11}^{20}, 1|_{11}^{20})$ and $D_{e_4} = (0|_{-\infty}^{0}, \phi)$; $g_2 \Rightarrow D_{n_1} = (0|_{1}^{10}, 1|_{1}^{10})$ and $D_{e_3} = (\phi, \phi)$ which then yields $D_s = (\phi, \phi)$: hence no transition is possible on *s* at or after t = 61.

4. Global Timing Implications

The method based on waveform narrowing uses local gate constraints, i.e., the global circuit function is not taken into account. By analyzing the circuit topology, however, we can deduce some of the functionality. *Static learning* [14] is used to identify some of the *class*-based implications. It is implemented in a pre-processing stage that determines tables of implications. When a *class* becomes empty in the domain of a net, *learning* tables are used to impose *class* restrictions on other domains. In addition we use the notion of *static* and *dynamic timing dominators* to identify global implications related to the existence of transitions at or after a certain time *lmin*, as described next.

The propagation of the *last-transition interval* is the main mechanism in proving that no violation is possible. In Example 2, only one path was the potential carrier of the violation. There was no ambiguity in deciding which net is its cause when the gate constraints were applied: at gate g_8 it was able to decide that net n_5 cannot be the cause of the violation because (largest of $\overline{D}_{n_5}.max$ and $\underline{D}_{n_5}.max$) + 10 < (smallest of $\overline{D}_s.lmin$ and $\underline{D}_s.lmin$). This is why $1|_{-\infty}^{50}$ was narrowed to ϕ in n_5 and the *last-transition interval* was propagated to n_7. In more complex circuits, e.g., the carry-skip adder in Figure 2, we may not be able to make such an unambiguous decision. Consider the output C_7. Suppose that the topological delay from C_2 to C_7 is 750 and that the timing constraint on C_7 is to require transitions at or after time 750, i.e., $\sigma = (\xi_1, C_7, 750)$.

The gate constraints are able to propagate the *last-transition interval* from C_7 to X only because no other subpath can be the carrier of the transitions. Figures 2 and 3 illustrate the situation at X. To simplify the presentation, assume that the NAND gate driving X is delayless. *Class* 0 of M (controlling) is removed because it blocks the way on N and P. The *last-transition interval* present at X propagates to *class* 0 on N and P, but not to class 1, because both inputs do not need to have transitions to propagate to X. We thus

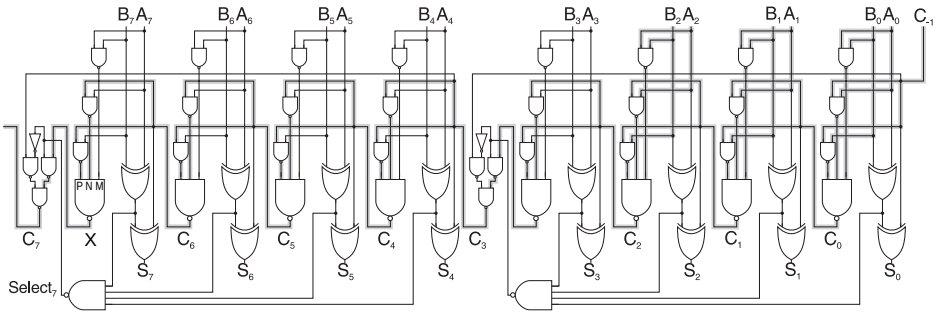

Figure 2 Carry-skip adder

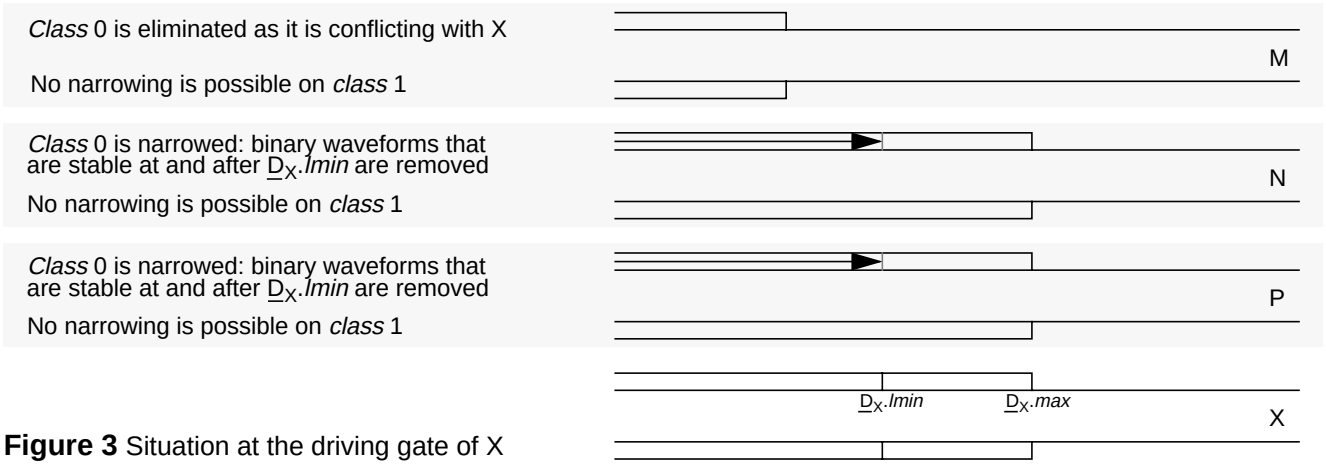

Figure 3 Situation at the driving gate of X

cannot decide which net is responsible for transition propagation, i.e., *lmin* can be increased, (the domain narrowed) only in the *controlling classes* of the NAND gate. However, the circuit topology implies that all paths to C_7 longer than 749 contain C_6. Therefore, we can restrict waveforms on C_6 to those having transitions at or after time ($minimum(\overline{D}_X.lmin, \underline{D}_X.lmin)$ - *max. topological delay from* C_6 to X).

4.1 Static Timing Dominators

Definition 4: A net x of ξ is a ***static carrier*** of $\sigma = (\xi, s, \delta)$ iff $\exists$ path in ξ containing x and s of length greater than or equal to δ.

Definition 5: The sub-circuit composed of the *static carriers of* $\sigma = (\xi, s, \delta)$ and their driving gates of ξ is the ***static-carrier circuit*** of σ.
For example, the *static-carrier circuit* of $\sigma = (\xi_1, C_7, 750)$ where ξ_1 is in Figure 2, is the sub-circuit of ξ_1 composed of the shaded nets and their driving gates.

Definition 6: Let Ψ be the *static-carrier circuit* of $\sigma = (\xi, s, \delta)$. Let Ψ' be a DAG derived from Ψ as follows: each net in Ψ corresponds to a vertex in Ψ'; each gate in Ψ with k inputs $x_1, x_2, ..., x_k$ and one output x_0 corresponds to k edges, from the vertex corresponding to x_0 to those corresponding to x_i, i=1 to k. Add a terminal vertex **T** to Ψ', and an edge to **T** from each vertex of an input of Ψ. Ψ' is a DAG with one source vertex **S** (corresponding to *s*) and one sink vertex **T**. The nets of Ψ corresponding to the dominators [15] of **T** (vertices lying on every path from **S** to **T**) are the ***static timing dominators*** of σ.

For example, for $\sigma = (\xi_1, C_7, 750)$ where ξ_1 is in Figure 3, C_7, X, C_6, C_5 are *static timing dominators* of σ.

Lemma 3: Let *d* be a *static timing dominator* of $\sigma = (\xi, s, \delta)$. Waveforms on *d* that are stable at and after time (δ - $top_{d\text{-}s}$) are σ–*incompatible*.
Proof: Follows from Lemma 6.1 in [10].

4.2 Dynamic Timing Dominators

The propagation of the *last-transition interval* of the output to the *static dominators* of the circuit represents global necessary assignments. Additional global implications can be determined by analyzing the contents of the abstract signal domains.

Definition 7: Let $\sigma = (\xi, s, \delta)$ be a *timing-check*, and *C* its constraint system. Let D_s be the domain associated with output *s*. If $D_s \neq (\phi,\phi)$ then *s* is said to be a ***0-dynamic carrier*** of σ. If net *y* is a *k-dynamic-carrier* and it is the output of gate *g* with max. delay d_{max}, then an input net *x* of gate *g* is a *k'-dynamic-carrier* of σ where $k' = (k + d_{max})$, provided that the domain D_x satisfies $D_x \cap (1|^{+\infty}_{\delta - k'}, 0|^{+\infty}_{\delta - k'}) \neq (\phi, \phi)$. A net x is a ***dynamic carrier*** of σ iff $\exists\, k \geq 0$ such that x is a *k-dynamic carrier* of σ.

Definition 8: Let Ψ be the circuit composed of the *dynamic carriers of* $\sigma = (\xi, s, \delta)$ and their driving gates. Ψ is the ***dynamic-carrier circuit*** of σ, and the *topological delay* $top_{x\text{-}s}$ between *x* and *s* of Ψ is the ***dynamic distance*** of *x*.

Intuitively, the *dynamic distance* of *x* is the maximum time a transition at *x* takes to reach *s*, and is equal to the largest integer *k* such that *x* is a *k-dynamic carrier* of σ. In fact, the concept of *dynamic carriers* is formulated by necessary conditions for a net to be the cause of a violation of the timing check, and the domain of a net that is not a *dynamic carrier* of $\sigma = (\xi, s, \delta)$ does not contain transitions that propagate to the *last-transition interval* of *s*.

Definition 9: The definition of ***dynamic timing dominators*** is obtained by replacing "static" with "dynamic" in Def. 6.

Theorem 3: For a *timing-check* $\sigma = (\xi, s, \delta)$ and a *dynamic timing dominator d*, let *k* be the largest integer such that *d* is *k-dynamic carrier* of σ. The waveforms on *d* that are stable at and after time (δ - *k*) are σ–*incompatible*.
Proof: Theorem 3 is a direct consequence of the fact that any net $x \notin \Psi$ cannot be the cause of a timing violation, i.e., D_x does not contain transitions that propagate to within the *last-transition interval* on *s*. This can be proven by contradiction: Suppose that there is a path $p = (x, g_{k_0}, n_{k_1}, \ldots, n_{k_p}, g_{k_p}, s)$ such that the domain of *x* contains transitions that propagate along *p* to the *last-transition interval* on *s*. This implies that the same property is true for all the nets of *p*. Then n_{k_p} has transitions at or after time (δ - max. delay of g_{k_p}) and consequently n_{k_p} is (δ - max. delay of g_{k_p})-*dynamic carrier* of σ. Similarly *x* is (δ - length of *p*)-*dynamic carrier* of σ. $x \in \Psi$ contradicts the original assumption.

Corollary 1: Let *d* be a *dynamic dominator* of $\sigma = (\xi, s, \delta)$ and *k* the *dynamic distance* of *d*. Narrowing the domain of *d* by intersecting it with $(1|^{+\infty}_{\delta - (\textit{dynamic-distance of } d)}, 0|^{+\infty}_{\delta - (\textit{dynamic-distance of } d)})$ maintains all the solutions of the original system.

The proof follows from Theorem 3.

Figure 4 exhibits the timing verification algorithm making use of Corollary 1.

5. Case Analysis

When the net domains remain non-empty after the fixpoint calculation we cannot definitely conclude that a violation is possible. We adapted the FAN algorithm [13, 14]

```
function verify(ξ, s, δ) {
    construct the constraint system CS associated with (ξ, s, δ);
    set all domains, except inputs and s, to (0|_{-∞}^{+∞}, 1|_{-∞}^{+∞});
    set domains of inputs to (0|_{-∞}^{0}, 1|_{-∞}^{0});
    set domain of s to (0|_{δ}^{0}, 1|_{δ}^{0});
    schedule all constraints operating on inputs and s on EventQueue;
    return evaluate(CS, EventQueue);
}

function evaluate(CS, EventQueue) {
    if EventQueue is empty return PossibleViolation;
    if (reach_fixpoint(CS, EventQueue) = = NoViolation)
        return NoViolation;
    determine dynamic dominators;
    for each dominator d do {
        intersect domain of d with
        (0|_{δ – dynamic-distance of d}^{+∞}, 1|_{δ – dynamic-distance of d}^{+∞})
        if the domain of d changed then schedule all constraints
        operating on it on EventQueue;
    }
    return evaluate(CS, EventQueue);
}

function reach_fixpoint(CS, EventQueue) {
    while EventQueue not empty {
        take a constraint out of EventQueue;
        apply constraint on the domains;
        schedule all constraints operating on the
        domains that were modified on EventQueue;
    }
    if the domain of s is empty return NoViolation;
    else return PossibleViolation;
}
```

When the function **verify**(ξ, s, δ) returns NoViolation, the output s cannot have transitions at or after time δ.

Figure 4 Algorithm of the method

to perform case analysis by waveform splitting on nets, i.e., by restricting their domains to one *class* at a time with the objective of finding a test vector or proving that no violation is possible. We used SCOAP [16] *controllability* to guide the algorithm.

The main idea is to compute the *initial objectives* so as to set those nets which are inputs of gates in the *dynamic-carrier circuit* Ψ of σ that are not *dynamic carriers* to a *non-controlling* value regarding the gates they feed in Ψ. This is justified by the following reasoning: the timing violation at output s is originating in Ψ, hence we need to sensitize the paths in Ψ. To favor the longest paths, we established the *objectives* to be a triplet (k, $n_0(k)$, $n_1(k)$) as in [13], but the semantics are different: a path to s of delay n_0 (n_1) is potentially enabled by setting the net k to 0 (1). The backtrace procedures are identical to the ones in [13] and [14], except that at fanouts, n_0 (n_1) receives the largest incoming n_0 (n_1) instead of their sum. In the context of ATPG, backtrace is performed a minimal number of times. In our case such a strategy resulted in poor performance, because decisions on nets may have profound effect on Ψ, the source of the violation. The backtrace is initiated each time the size of the decision stack changes as a result of backtracks. Moreover, decisions are performed in 3 phases, following stem correlation pre-processing stage:

Stem correlation: We perform partial correlation on all reconvergent fanout stems that are *dynamic carriers*. This is done, for a stem Y, by computing the domain D_X of each variable X of the constraint system as follows: $D_X = D_{X0} \cup D_{X1}$ where D_{X0} and D_{X1} are the values of D_X when D_Y is intersected with $(0|_{-\infty}^{+\infty}, \phi)$ and $(\phi, 1|_{-\infty}^{+\infty})$, respectively. This has the effect of removing some of the incompatible waveforms from the domains of the variables and no decision is yet taken.

Phase 1: Let $d_0, d_1, \ldots, d_k$ be the consecutive *dynamic-dominators* of (ξ, s, δ) computed before any decision is taken, $(d_0 = s)$. Let $\xi_{d_i, d_{i+1}}$ be the sub-circuit of ξ composed of the fan-in cone of d_i excluding d_{i+1}. We fix the *class* value of nets in $\xi_{d_i, d_{i+1}}$, i = 0 to k-1, using the modified FAN algorithm. Then, we fix the *class* of nets in the fan-in cone of d_k.

Phase 2: We perform decisions on the whole circuit using the modified FAN algorithm.

Phase 3: We perform decisions on s, and then on the primary inputs after complete backtrace from *unjustified* nets. An output of a gate G is *unjustified* iff its domain is restricted to one *class* and if we can intersect the domain on each input with $(0|_{-\infty}^{+\infty}, \phi)$ or $(\phi, 1|_{-\infty}^{+\infty})$ to get non empty input domains that are inconsistent with the gate constraint.

6. Experimental Results

Experiments were executed on a Sun SPARCstation 10. The basic constraint system evaluation without global implications on *timing dominators* was able to eliminate timing check violation in the c5315 and c7552 of the NOR-gate implementations of the ISCAS'85 benchmarks [11] with delays of 10 on the outputs of all gates. The use of *timing dominators* eliminated timing violations from c1908 and c3540. Stem correlation eliminated timing-check violation from c2670 and c6288. The case analysis found test vectors for all circuits except c6288. Table 1 contains the results. Note that the value of δ for which a test vector is found represents the exact floating-mode delay of the circuit when the constraint system is inconsistent for (δ + 1) on all outputs. The columns of Table 1 contain, from left to right, the following information: 1) the circuit name, 2) the max. topological delay of the circuit, 3) the timing constraint δ, 4) the result of the first evaluation of the constraint system before the use of *timing dominators*, 5) the result after the use of *timing dominators*, 6) the result after stem correlation, 7) the number of backtracks in the case analysis, 8) the result of case analysis, and 9) the total CPU time.

Table 1 Results for ISCAS'85 circuits

CIRCUIT	CIRCUIT MAX. TOP.	δ	BEFORE G.I.T.D.	AFTER G.I.T.D.	AFTER STEM C.	C.A. #BTRCK	C.A. RESULT	CPU (s)
c17	50	50^{E}	P	P	P	0	V	0.05
c432	190	190^{E}	P	P	P	1	V	18.82
c499	250	250^{E}	P	P	P	5	V	7.10
c880	200	200^{E}	P	P	P	0	V	3.06
c1355	270	270^{E}	P	P	P	1	V	8.17
c1908	340	311	P	N	-	-	-	0.90
c1908	340	310^{E}	P	P	P	5	V	11.58
c2670	250	241	P	P	N	0	N	3.67
c2670	250	240^{E}	P	P	P	7	V	17.07
c3540	410	391	P	N	-	-	-	5.12
c3540	410	390^{E}	P	P	P	3	V	56.00
c5315	460	451	N	-	-	-	-	1.56
c5315	460	450^{E}	P	P	P	16	V	21.97
c6288	1230	1221	P	P	N	0	N	56.36
c6288	1230	1220^{U}	P	P	P	A	A	A
c7552	380	371	N	-	-	-	-	0.72
c7552	380	370^{E}	P	P	P	1	V	8.34

Legend: (**G.I.T.D.** stands for Global Implications on Timing Dominators.)
P: Possible violation of the timing-check constraint. **N**: No violation of the timing-check constraint on any circuit output is possible. **V**: Test vector found. - (dash): Procedure not used (was not necessary). **A**: Abandoned due to excessive number of backtracks. (^{E}): Value represents exact floating-mode delay. (^{U}): Value represents upper bound on the maximal floating-mode delay.

Not included in Table 1 is the timing check performed on a 16 bit carry-skip adder, partly shown in Figure 2. The adder has a topological delay of 2000 and a floating-mode delay of 1000. This was determined in 25 seconds of CPU time after a total of 1636 backtracks. For $\delta = 1001$ the case analysis proved that the constraint system is inconsistent on all outputs, and for $\delta = 1000$ found a test vector.

The use of *timing dominators* was very effective on the traditionally difficult c1908 circuit. It proved that output 57_912 (topological delay of 340) cannot have a delay greater than 200 in 0.76 seconds. This particular case has 5 *timing dominators*, and no narrowing was performed on 3 of them by the original method.

7. Conclusions

We showed in this paper how global timing implications enhance the performance of the timing verification method based on waveform narrowing. Further refinements were achieved by enforcing correlation on reconvergent stems and new heuristics included in the case analysis. We are developing constraint models for complex gates (MUX, etc...), and we process SDF backannotation to test our method on industrial circuits.

References

[1] S. M. Aourid and E. Cerny, "CLP-Based Gate-Level Timing Verification with Delay Correlation," *IWLS'97*, Granlibakken Resort, Tahoe City, CA May 18 -21, 1997.

[2] E. Cerny and J. Zejda, "Gate-Level Timing Verification Using Waveform Narrowing," *Proc. EuroDAC*, Grenoble, Sept. 1994, 374-379

[3] D. Brand and V. Iyengar, "Timing Analysis Using Functional Analysis," *IEEE Trans. Comp*., Oct. 1988, C-37(10):1309-1314.

[4] S. Devadas, K. Keutzer and S. Malik, "Delay Computation in Combinational Circuits," *ICCAD-91*, Nov. 1991.

[5] S. Devadas, K. Keutzer and S. Malik, "Computation of Floating Mode Delay in Combinational Logic Circuit: Theory and Algorithms," *IEEE Trans. Computer-Aided Design*, vol. 12, no. 12, Dec. 1993, 1913-1923.

[6] W. K. C. Lam, R. K. Brayton and A. L. Sangiovani-Vincentelli, "Circuit Delay Models and their Exact Computation Using Timed Boolean Functions," *30th DAC*, June 1993, 128-134.

[7] S. Devadas, et al., "Certified Timing Verification and the Transition Delay of a Logic Circuit", *26th DAC*, June 1992.

[8] P. C. McGeer and R. K.Brayton, "Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network," *26th ACM/IEEE Design Automation Conference* 1989, 561-567.

[9] C. Oh and R. Mercer, "Efficient Logic-Level Timing Analysis Using Constraint-Guided Critical Path Search," *IEEE Trans. VLSI Systems*, vol. 4, no. 3, Sept. 1996, 346-354.

[10] S. Devadas, et al., "Event Suppression: Improving the Efficiency of Timing Simulation for Synchronous Digital Circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, June 1994, 814-822.

[11] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," *Proc. Int. Symp. Circuits and Systems, Special Session on ATPG and Fault Simulation*, Kyoto, Japan, June 1985.

[12] V. M. Hrapcenko. "Depth and Delay in a Network," *Soviet Math. Dokl*., 1978.

[13] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. Computers*, vol. C-32, Dec. 1983, 1137-1144.

[14] M. H. Schulz, E. Trischler and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. Computer-Aided Design*, vol. CAD-7, Jan. 1988, 126-137.

[15] R. E. Tarjan, "Finding Dominators in a Directed Graph," *SIAM Journal on Computing*, vol. 3, 1974, 62-89.

[16] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program," *Proc. 17th Design Automation Conf*., June 1980, 190-196.

[17] F. Benhamou and W. J. Older, "Applying Interval Arithmetic to Real, Integer and Boolean Constraints," *Journal of Logic Programming*," vol. 32, no. 1, 1997, 1-24.

Acknowledgments: This work was supported by NSERC Canada - Nortel North America Cooperative R&D Grant No. 169880. The experiments were done on a workstation on loan from the Canadian Microelectronics Corp.