Gate-Level Timing Verification Using Waveform Narrowing

Eduard Cerny, Jindrich Zejda

Dép. IRO, Université de Montréal, C.P. 6128, Succ. Centre-Ville
Montréal (Québec), H3C 3J7 Canada

Abstract

We present a novel gate-level timing verification method that determines if a combinational circuit satisfies a maximal or a minimal required propagation delay. It is based on computing the greatest fixpoint over a set of equations derived from the gate forward and partial inverse functions, and their interconnections, in the domain of sets of (abstract) waveforms. False negative results can be obtained, however, these can be refined by limited case analysis (based on the analysis of reconvergent fanout). The resolution method is independent of the circuit delay model used, transition or floating mode; only the initial set of waveforms on the primary inputs distinguishes them. The method can accommodate interval gate delays, separate rise and fall transition delays, and component delay correlation.

1. Introduction

The function of a static timing verifier is to ascertain that timing constraints such as set-up and hold times on flip-flops are satisfied in the range of operating clock frequencies and delay variations. Many techniques, delay models and sensitization criteria have been developed [1,2,3,9,11, etc.] to deal primarily with the problem of false path elimination. Except for the most recent exact results such as [7], the methods make simplifying assumptions that under increased pessimism (false negative answers) make the problem tractable. Yet, all of them are complex and many subtleties may be difficult to understand and interpret by the implementors of the verifiers and circuit designers. The objective is thus to provide a method that can progressively trade accuracy for speed, is relatively easy to understand, and can be implemented with reasonable effort. An approach toward this goal is described in this paper.

Static timing verification methods can be divided into two classes: a) decision methods, and b) optimization methods. In (a) the clock frequency is fixed (e.g., by the application), and we verify if the circuit satisfies the timing constraints. In (b) we try to determine the fastest clock frequency by finding the longest (and the shortest) delay through the combinational part of the circuit. A decision procedure can be used in an optimization method based on searching.

We address the decision problem: Given a combinational circuit consisting of a loopless interconnection of logic gates and delay elements, we determine whether a transition can occur on the circuit's output after some limit value D^{max} (or before some value D^{min}) under the 2-vector or floating-mode circuit delay [7]. A *timing violation* occurs when the circuit delay exceeds the limit being verified. Our method can accommodate rise and fall interval delays, and component delay correlation due to the technological process. That is, the interval delays on all gates can have a common factor variation and a local variation of, e.g., 10%. Timing violations occurring only when delays on different gates tend to the opposite ends of the intervals are false.

Our method was inspired by the work on multiple-fault coverage analysis [5, 14, 15], and by constraint resolution using relational interval arithmetic [4,8,10,13]. It initially assigns all possible signal waveforms on the circuit lines, and then eliminates (sub) waveforms that *necessarily cannot cause* the timing violation. It proceeds as follows: The basic four waveforms {0x0, 0x1, 1x0, 1x1} are propagated through the circuit, where x represents a region of possible instability delimited by a time interval. Backward deduction is performed based on the subwaveforms on the primary outputs that violate the requirements (e.g., exceed D^{max}), eliminating all those waveforms on gates' inputs that necessarily could not contribute to the narrowed waveform sets on the outputs. Correlation of waveforms is enforced on fanout branches and stems, and a new forward / backward propagations are initiated if the waveform sets are reduced by the correlation. When no further changes are detected (i.e., the greatest fixed point of the system of non-linear interval equations of the gate network is found), and if the waveforms are all eliminated then the timing violation cannot occur. Otherwise, there is a possibility of a violation, i.e., due to the elimination of waveforms based only on necessary conditions false negative results can be obtained. These can be further refined, e.g., by limited

case analysis over subwaveforms on some circuit nodes identified by stem region (reconvergent fanout) analysis.

Brand et al. [1] also performed backward deductions without branching, however, unlike their method we process waveforms and do not enumerate any paths or propagate logic conditions. Another method that considers intervals in waveforms is descibed in [12], except that the intervals identify constraints on the minimal duration of stable logic values, in a method based on static path sensitization (that has since been shown to compute neither a lower nor an upper bound on the actual delay). Also, their method requires path enumeration. Finally, our representation of input waveforms for the 2-vector delay calculation is similar to that used in [6] for current estimation: we also forward propagate sets of all possible transitions and ignore correlation due to fanout stems. The major difference is, however, that we distinguish 4 waveform classes and propagate narrowed waveform sets both forward toward the primary outputs and backward toward the inputs, within a framework of a global fixpoint calculation over the domain of sets of abstract waveforms. Backward deductions leading to fanout stems then introduce waveform correlation on these stems, and thus reduce the inherent pessimism of the method.

The paper is organized as follows: We introduce the waveforms and their forward propagation through the circuit in Section 2 and, in Section 3, we discuss the backward deductions. The method is illustrated using a simple example for the verification of both the two-vector-transition and the floating-mode circuit delays. We then discuss the treatment of interval, rise and fall gate delays, and outline handling of component delay correlation in Section 4. Our prototype implementation is briefly presented in Section 5, and preliminary experimental results are given in Section 6. Section 7 concludes the presentation.

2. Waveforms in tree circuits

We consider combinational circuits consisting of 2-input gates (and, or, nand, nor, xor), inverters, buffers and (interval) delay elements. Signal transition times are discrete, associated with integers $-\infty$, ..., -1, 0, 1, ..., ∞. We analyze the circuit delay under pairs of vectors [7]: The circuit is quiescent (the previous input vector was at $t = -\infty$) before applying a 0 -> 1 or a 1 -> 0 transition to some primary inputs at t=0. (Floating-mode circuit delay will be discussed in Section 3.)

If we observe the propagation of the transitions from the primary inputs (PI) to the primary outputs (PO), the waveforms on the circuit lines fall into 4 abstract waveform classes:

Class 00: The initial value is 0, it may undergo the earliest change to 1 at time t, then possibly keep changing between 0 and 1, and finally return to stable 0 at time T at the latest. Such a waveform is denoted 00[t, T] (<u>00</u>[t,T]) when it excludes (includes) the stable 0 waveform. <u>00</u>[-,-] represents stable 0. 00[t,t] represents a 1-glitch of width zero at time t; our method can either pass or filter such glitches.

Class 11: Complement of class 00, denoted 11[t,T] or <u>11</u>[t,T].

Class 01: From stable 0, it may undergo the earliest change to 1 at time t, followed by any number transitions, with the last one to stable 1 at time T at the latest. Denoted 01[t,T].

Class 10: Complement of class 01, denoted 10[t,T].

Waveform xy[$-\infty$,∞], x,y $\in \{0,1\}$, represents the *universal* xy waveform. A waveform w=xy[t,T] is a *subwaveform* of w'=x'y'[t',T'], $w \subseteq w'$, iff xy = x'y', $t \geq t'$ and $T \leq T'$, and the presence of stable xy in w implies the same for w'. A set W of (up to 4) waveforms (one per class) is a subset of W', $W \subseteq W'$, iff $w \in W \Rightarrow \exists w' \in W'$ and $w \subseteq w'$. In defining the classes, we do not care when and how many transitions occur between t and T, we are only interested in the earliest and latest changes, and the possibility of transitions in between.

We first consider tree circuits with fixed gate delays and then generalize to combinational circuits containing reconvergent fan-out and other delay models. The waveforms can be easily computed on the lines of a tree circuit by propagating the waveforms from PIs through the gates to POs. Since each input can undergo two possible transitions or remain stable, giving 4^n pairs of vectors where n is the number of PIs, we must reduce the number of possible combinations to consider. In a tree circuit, the values on all the inputs and internal lines that are not on the same path from a PI to a PO are independent, hence we can propagate all the 4^n possible changes at the same time, by propagating the entire set of 4 waveforms from each primary input simultaneously. We compute the 16 possible waveform combinations on (2-input) gates and then, on each gate output, we *merge* the resulting waveforms in each class into a single waveform. This waveform is an upper envelope over the contributing waveforms in that class. In this way, the number of waveforms to propagate further is again reduced to four only. When POs are reached, the waveforms are tight envelopes over the sets of real waveforms on the outputs, hence the latest (earliest) transition times are generally good upper (lower) bounds on the actual latest (earliest) transition times in each waveform class.

The associative and commutative *merge* is defined as: $xy[t_1,T_1] \cup xy[t_2,T_2] = xy[\min(t_1, t_2), \max(T_1,T_2)]$; if either operand contains stable waveform, the result contains it too.

Consider the circuit in Fig.1. The two-vector transition delay waveform set to be applied to the PI's is W_2 = {00[-,-], 01[0,0], 10[0,0], 11[-,-]}, however, to simplify the example we use only {01[0,0], 10[0,0]} on a_1, a_2, a_3, and a_4.

Forward propagation of these waveforms is simple through buffers and inverters:

w: {01[2,2], 10[2,2]} b_1: {01[1,1], 10[1,1]}
b_2: {01[1,1], 10[1,1]} u: {01[3,3], 10[3,3]}
v: {01[2,2], 10[2,2]}

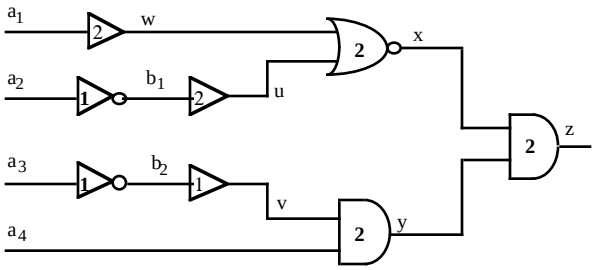

Fig.1: A tree circuit (numbers are fixed output delays).

For x we propagate all 4 combinations on the inputs u and w of the NOR gate, and add a delay of 2 units:

x: nor(01[3,3], 01[2,2]) = 10[4,4]
nor(01[3,3], 10[2,2]) = 00[4,5]
nor(10[3,3], 01[2,2]) = 00[-,-]
nor(10[3,3], 10[2,2]) = 01[5,5]

After merging: x: {00[4,5], 01[5,5], 10[4,4]}

Similarly, for y and z we get

y: {00[2,4], 01[4,4], 10[2,2]}
z: {00[4,7], 01[7,7], 10[4,4]}

The forward propagation of waveforms through a logic gate (the computation of the set of image waveforms) is trivial for the inverter, buffer and the fixed delay element. For the two-input AND gate (and similarly for other multiple-input gates) it consists of two steps:

1) The partial waveforms $xy[t_c, T_c]$ are computed (up to 16 of them) as shown in Table 1. For xy=00 or 11, if $t_c \le T_c$ does not hold then it is stable 00[-,-] or 11[-,-] (we set $t=+\infty$, $T=-\infty$).

2) A single waveform for each class is formed by *merging* the partial waveforms in each class: There are (Table 1) up to 9 partial waveforms of class 00, up to 3 for 01 and 10, and one for 11.

The image calculation for other types of gates can be derived similarly, or by replacing the gate by its equivalent network of 2-input AND gates and inverters (this may introduce more pessimism due to internal reconvergent fanout and waveform merging).

The simplistic forward waveform propagation is not sufficient when signal correlation due to reconvergent fanout is present. The true behavior is still contained in the forward propagated waveforms, but so are many impossible ones, producing as pessimistic results as methods based on the longest topological path. However, we can do better as shown next, yet without path enumeration and/or symbolic constraint propagation.

3. Backward deductions

Fig.1 was obtained from Fig.2 [9] by splitting stem a into 4 independent inputs, and stem b into b_1 and b_2. The forward propagated waveforms (shown in normal type) in Fig.2 are the same as in Fig.1, since we assumed signal independence. Yet, the circuit in Fig.2 is stable, z = 0.

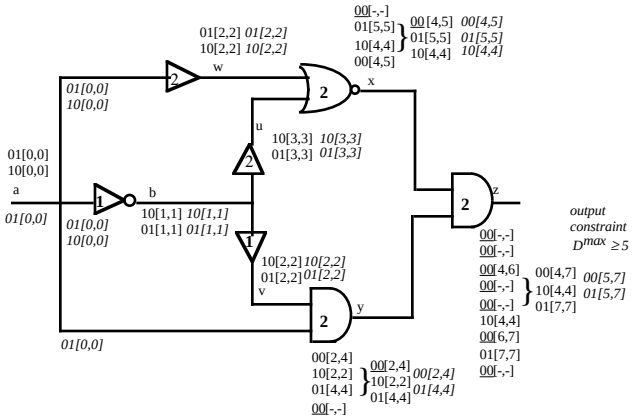

Fig.3: Circuit with reconvergent fanout.

Suppose, that the circuit is a part of a synchronous sequential circuit, such that any transition on z occurring at $D^{max} \ge 5$ would cause a timing violation. We can ask the following question: What waveforms on x and y could never produce a transition at or after time 5 on z? Such waveforms can be eliminated from the set of waveforms on these lines, i.e., the waveform sets are narrowed and become the source of backward deductions through predecessor gates. Since fanout branches and the fanout stem must carry the same signal, the deduced waveforms on the stem are an intersection of the waveform sets on the branches. The *intersection* operation is defined as follows:

$$xy[t_1, T_1] \cap x'y'[t_2, T_2] = \emptyset \quad \text{if } xy \ne x'y' \text{ or } t_1 > T_2 \text{ or } t_2 > T_1$$

$$= xy[\max(t_1,t_2), \min(T_1,T_2)] \quad \text{otherwise;}$$

the result contains a stable x if both operands contain it.

The backward deduced waveforms are shown in italic type in Fig.2. If the stem waveform set is narrowed by the intersection, e.g. **a** in Fig.2, a new forward propagation and backward deduction is initiated, etc., until no further narrowing can be achieved. The resulting waveforms on the PO answer the original question: Either a transition on the output remains possible in the violating interval in which case the violation may occur (but not necessarily), or the violation disappears (propagating 01[0,0] from a results in 00[-,-] on z), or during the backward deduction the branches of a fanout stem carry disjoint waveforms and the intersection is empty. The latter represents a contradiction in the waveform assignment; consequently, the timing violation cannot occur.

If we try to verify whether a transition is possible in the entire interval [4,7] in all waveforms on z, we cannot deduce anything different. This is because the set of waveforms on z is not narrowed and the deductions based on necessary conditions lead to the original waveforms from the forward propagation. However, if we consider subsets of waveforms at a time (a form of branching), we can deduce that no transition is possible on z. We have in fact verified earlier that no transition can occur on z for the waveforms 00 and 01. For the 10[4,4] waveform considered alone, the deductions lead to a contradiction on b, hence no 10 waveform is possible on z.

The backward deduction (or pre-image computation) of waveforms on gate input(s) from the waveforms on the output and the current waveforms on the input(s) is trivial for the inverter, buffer and the fixed delay element. For 2-input gates the pre-image calculation is more involved and must be done carefully so as to introduce no unwanted optimism. We compute the backward deduced input waveforms using partial inverse functions [8]. Let W_c', W_a', W_b' (resp. W_c, W_a, W_b) be the deduced (resp. current) sets of waveforms on the output c and the inputs a and b of a two-input gate (forward function) g, such that $W_c \subseteq g(W_a, W_b)$, $W_c' \subseteq W_c$. A possible conservative partial inverse function for input a is as follows:

$$W'_a = g_a^{-1}(W'_c, W_b) = \{w | w \in W_a \wedge g(\{w\}, W_b) \cap W'_c \neq \emptyset\}$$

The deduced waveforms replace the current waveforms on the gate's inputs and become the source for backward deductions through predecessor gates.

The following is an outline of a simple version of the verification algorithm, based on repeated forward and backward passes over a topologically ordered gate netlist. Forward propagation (backward deduction) follows this order (reverse order). It is easy to convert the algorithm to an efficient event-driven version.

Algorithm check_network_delay:

input: topologically ordered netlist; D^{min} or D^{max};
output: "NO violation" or "Potential violation";

```
begin
    for each PO do begin
        initialize all lines to universal waveforms;
        apply input sets of wavefroms to all PIs;
        forward propagation till POs reached;
        reduce PO waveforms to those
                violating requirements;
        if empty then NO violation on this PO; next PO;
        repeat
                backward deduction till PIs reached;
                forward propagation till POs reached;
        until no change in waveforms on any line;
        if all waveforms are empty or stable on PO then
                                NO violation occurs on this PO;
                        else    Potential violation on this PO;
end if; end do; end.
```

The algorithm terminates (the iteration reaches a fixpoint), because the image and pre-image computations produce monotonically non-increasing waveform sets, and there is a finite number of steps to reach a "bottom" value that is the empty set of waveforms. In our experiments with some of the ISCAS-85 benchmarks, there were at most 4 iterations in any one fixpoint calculation, suggesting complexity in O(n) where n is the number of gates. More experimental work is needed on real circuits and in real timing situations to determine the "practical" complexity of the method, especially when case analysis is used to reduce the pessimism of the method.

To conclude this section, we outline the verification of timing violations under the floating-mode circuit delay. The resolution method remains the same as with the two-vector transition delay, however, similarly as in [7], we change the waveform sets on the primary inputs: Instead of W_2 defined earlier, we use the set $W_{\omega-}$ = {01[-∞,0], 10[-∞,0]} on a. Here, the state of every line becomes undefined until the propagation of a definite 0 or 1. When we apply these waveforms to the circuit in Fig.2, the forward propagation yields the set {00[-∞,7], 01[-∞,7], 10[-∞,4]} on z. Backward deductions from these waveforms, one at-a-time, under $D^{max} \geq 7$ yields a contradiction, while if we choose $D^{max} \geq 5$, then only the 01 and 10 waveforms can be eliminated. Effectively, the waveform 00 on z may have a transition at time 6 when the delay on buffer u is 1 (which is within the bounded delay interval [0,2] implicitly considered in the floating-mode circuit delay assumptions).

4. Interval, rise and fall delays

As in other methods, we separate delays from gates and treat them as independent circuit elements that can be inserted anywhere on the interconnections. For simplicity we explained the basic method using fixed delays, and in this section we consider interval delays, rise and fall delays, and discuss an approach to dealing with component delay correlation.

4.1 Interval delays

Consider a delay element whose delay is in the uncertainty interval [d, D]:

Forward propagation: Given the input waveform $xy[t_i,T_i]$, the output waveform becomes

$$xy[t_o,T_o] = xy[t_i+d, T_i+D].$$

Backward deduction: Let $xy[t'_o,T'_o]$ be the waveform deduced on the output of the delay element, $xy[t'_o,T'_o] \subseteq xy[t_o,T_o]$. The new input waveform is

$$xy[t'_i,T'_i] = xy[\max(t_i, t'_o-D), \min(T_i, T'_o-d)].$$

4.2 Rise and fall delays

Each of the four waveform classes is propagated independently through a delay element, hence the introduced delay can be different for all four waveforms. For example, a distinct delay can be applied to the latest and earliest transitions of the 01 and the 10 waveforms. The 00 (11) waveform can have the earliest transition delayed by the min. rising (falling) delay and the latest transition by the max. falling (rising) delay. When the delay applied to the earliest transition of 00 and 11 waveforms projects this transition in the future of the delayed latest transition (non-causal behavior), a stable waveform of its class would be produced instead.

4.3 Component delay correlation

Due to the variations in the manufacturing process, the delays in a circuit can vary from one batch to another, but within a batch there is a correlation in the gate delays, i.e., all tend to the larger or to the smaller values. This can be expressed using a common factor in the interval [m, M], and a local variation [v, V] of delay values in each gate, usually within 10%. The overall effect can be modeled as a virtual "delay" signal carrying the common factor, connected to all so correlated delay elements; each delay element introduces its local variation and then uses the result [d,D] = [v*m, V*M] as its delay interval [10, 4].

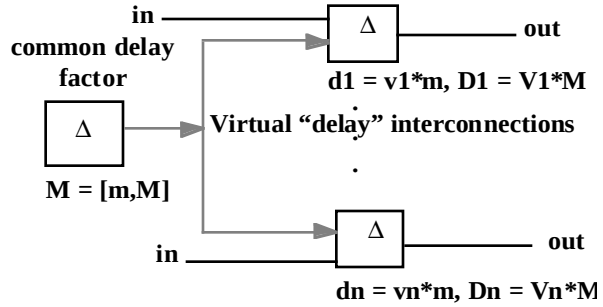

Fig.4: Virtual net for component delay correlation.

During backward deduction, the uncertainty interval of the narrowed waveform on the output of the delay element can in some cases reduce the uncertainty interval of the delay. This is propagated through the multiplication by the common factor to the fanout stem of the virtual delay interconnection (Fig.4) where the delay intervals are intersected. If the intersection is empty then the timing violation is impossible due to delay correlation. Otherwise, a reduced common factor interval is propagated to the delay elements, and new forward propagation and backward deduction phases through the network are initiated, etc.

5. Prototype implementation

The objectives of our prototype implementation were to minimize the programming effort, the parsing of netlist definitions, and to have as much flexibility and user interaction as possible. The candidate for the implementation was rather unorthodox, but turned out to satisfy best our needs: We implemented the algorithm in the hardware description language VHDL and used the simulator to perform the fixpoint calculations. It supports for the moment any circuit built from gates (currently constructed by interconnecting 2-input AND gates, inverters and delays), and consists of about 1500 lines of VHDL, including the various type definitions. It accepts gate-level (structural) VHDL description of the circuit to be analyzed, and the only changes that must be made to the entity and architecture definitions are in the type and mode of ports (to be inout) and in signal types. The components (gates) have a generic delay parameter (min and max); we provide a new architecture for them.

The VHDL signal (and port) type is resolved and consists of an array of 4 waveform definitions, one per waveform class. Each waveform is a record composed of a Boolean indicating whether the waveform is not empty, another Boolean indicating the possible presence of a stable 00 and 11 waveform, and two integers to hold the value of t and T. The intersection operation on stems is implemented by the signal resolution function. Each gate architecture consists of a process that is sensitive to events on the input and output ports of the gate. If an event occurs on an input port, the process drives the output port using the g function with the VHDL delta delay, while if an event occurs on the output port the process drives the input ports using the g^{-1} functions in one delta delay. The fixpoint resolution occurs in delta time. When no further events are scheduled in delta time (no further waveform narrowing is possible), the next real simulation time is reached (e.g., 1 fs later) indicating that the fixpoint calculation terminated, and at that time the PI and PO entities examine the resulting signal values. If these signals represent empty sets of waveforms, then the system of constraints is inconsistent.

6. Experiments

For the moment we analyzed the more difficult circuit c1908 from the ISCAS-85 testability benchmarks. With the max. gate delay of 10 the topological delay of the circuit on output 57 is 400. An upper bound of 240 on the actual delay under both the transition and the floating-mode delay models is obtained by performing a case analysis over the waveforms on that output and the primary input 902 (this input has one of the largest transitive fanouts through buffers and inverters). A summary of the analysis is in Table 2.

For example, for the transition delay, the number of input and output waveforms is 4 each, giving the total of 16 cases to assess. The CPU time is the sum of the CPU times of each such fixpoint calculation (using our experimental and inefficient VHDL implementation). Since floating-mode delay involves only two waveforms, the calculations take considerably less time.

7. Conclusions

We have presented a novel method for verifying whether a combinational circuit satisfies a specific delay requirement. The method is deductive using necessary conditions over sets of at most four abstract waveforms on each circuit line, and may produce false negative results. We are currently implementing a fully automatic timing analysis system and model generators for a typical cell library using a C++ circuit object library and a Verilog parser. Potentially false negative answers are refined using limited case analysis, by imposing constraints over waveforms on critical nodes in the circuit. The nodes are identified using a topological analysis of the circuit (stem region analysis).

References

[1] D.Brand, V.Iyengar, "Timing Analysis Using Functional Analysis", IEEE Trans. Comp., Oct. 1988.

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

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

[4] T.Kamel, "Design and Implementation of a Static Timing Analyzer using CLP(BNR)", CRL Report, Bell Northern Research, Aug.1993.

[5] Y.Karkouri, E.M.Aboulhamid, E.Cerny, A.Verreault, "Use of Fault Dropping for Multiple Fault Analysis", IEEE Trans. Comp., 43 (1), Jan.1994, 98-103.

[6] H. Kriplani, F. Najm, I. Hajj, “Maximum Current Estimation in CMOS Circuits”, 29th DAC, June 1992, 2-7.

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

[8] O.Lhomme, “Consistency Techniques for Numeric CSPs”, 13th IJCAI , 1993.

[9] P.C.McGeer, R.K.Brayton, *Integrating Functional and Temporal Domains in Logic Design*, Kluwer Publ., 1991.

[10] W.J.Older, "Delay Networks and Intervals", CRL, Bell Northern Research Ltd., Internal Report, June 1991.

[11] J.M.Silva, K.A.Sakallah, "An Analysis of Path Sensitization Criteria", ICCD-93, Oct.1993.

[12] R.Stewart, J.Benkoski, “Static Timing Analysis Using Interval Constraints”, ICCAD-91, Nov. 1991, pp.308-311.

[13] A.Vellino, W.Older, "Constraint Arithmetic on Real Intervals", Constraint Logic Programming: Selected Research, ed. A.Colmerauer, F.Benhamou, MIT Press, 1993.

[14] A.Verreault, E.M.Aboulhamid, Y.Karkouri, "Multiple Fault Analysis Using a Fault Dropping Technique", 21st FTCS, June 1991.

[15] H.Cox, J.Rajski, "A method of Fault Analysis for Test Generation and Fault Diagnosis", IEEE Trans.CAD, 7(7), 1988.

Acknowledgments: The work was supported by an NSERC Canada Grant OGP0003879. Experiments were done on workstations from the Canadian Microelectronics Corp. We thank W. Older and J.-L. Martineau for helpful comments.

partial $xy[t_c, T_c]$	$00[t_a, T_a]$	$01[t_a, T_a]$	$10[t_a, T_a]$	$11[t_a, T_a]$
$00[t_b, T_b]$	00 $[\max(t_a, t_b), \min(T_a, T_b)]$	00 $[\max(t_a, t_b), T_b]$	00 $[t_b, \min(T_a, T_b)]$	00 $[t_b, T_b]$
$01[t_b, T_b)$	00 $[\max(t_a, t_b), T_a]$	01 $[\max(t_a, t_b), \max(T_a, T_b)]$	00 $[t_b, T_a]$	01 $[t_b, \max(T_a, T_b)]$
$10[t_b, T_b]$	00 $[t_a, \min(T_a, T_b)]$	00 $[t_a, T_b]$	10 $[\min(t_a, t_b), \min(T_a, T_b)]$	10 $[\min(t_a, t_b), T_b]$
$11[t_b, T_b]$	00 $[t_a, T_a]$	01 $[t_a, \max(T_a, T_b)]$	10 $[\min(t_a, t_b), T_a]$	11 $[\min(t_a, t_b), \max(T_a, T_b)]$

Table 1: Partial waveforms in forward propagation through a 2-input AND gate.

Gate Delays	Delay Mode	Output transitions	CPU time [s]	# of cases analyzed
(10,10)	Transition	[30,240]	175	16
(10,10)	Floating	[0,240]	60	4
(0,10)	Transition	[0,240]	165	16
(7,10)	Transition	[21,240]	174	16

Table 2: c1908 results.