Timing Analysis Based on Primitive Path Delay Fault Identification

Mukund Sivaraman* and Andrzej J. Strojwas
Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

We present a novel timing analysis mechanism which is based on identifying primitive path delay faults (primitive PDFs) in a circuit. We show that this approach gives the exact maximum delay of the circuit under the floating mode of operation assumption. Our timing analysis approach provides a framework where component delay correlations and signal correlations arising from fabrication process, signal propagation, and signal interaction effects can be handled very accurately. Under these effects, timing analysis using previously reported floating mode timing analyzers, e.g., viability, TrueD-F etc., is very pessimistic. Our timing analysis approach based on primitive PDF identification is also more efficient than conventional floating mode path sensitization analysis mechanisms in situations where critical paths need to be re-identified due to component delay speedup (e.g., post-layout delay optimization). We demonstrate the applicability of our timing analysis approach for a variety of benchmark circuits, and demonstrate the pessimism of conventional floating mode timing analysis approaches in accounting for signal propagation effects.

1. Introduction

Over the past ten years, the problem of determining the maximum delay of a combinational circuit accounting for path sensitization has been the focus of intense research. In this context, it has been proved in [6] that the maximum delay of a family of circuits (i.e., a given circuit netlist whose component delays vary within specified ranges) is bounded by the maximum delay, determined under a floating mode of operation assumption, of that circuit instance whose component delay values are set to their individual maximum values. Based on this, numerous floating mode timing analyzers have been reported [3][5] which assume component delays to be fixed at their individual maximum values and perform the path sensitization under a floating mode of operation assumption. However, the implicit assumption here is that all component delays can simultaneously be set to their individual maximum values. This may not be the case due to component delay correlations and signal correlations arising from fabrication process fluctuations, signal propagation effects (gate output transition time affects delay of not just immediate fanout gates, but also of succeeding downstream logic), and signal interaction effects (capacitive coupling, multiple gate input transitions). The assumption that all component delays can simultaneously be at their individual maximum values is a very pessimistic way of accounting for these effects, therefore these previously reported floating mode timing analyzers are very conservative in their delay estimates. Moreover, their sensitization mechanisms offer no way of accounting for these effects in a more realistic manner.

The floating mode timing analysis mechanism proposed in this paper forms a framework that can handle these fabrication process, signal propagation, and signal interaction effects very accurately. Our timing analysis mechanism is based on identifying primitive path delay faults (primitive PDFs [12]). It can be derived from the work in [12] that primitive PDFs are the path sets which determine circuit stabilization time. Our ***P**rimitive PDF Identification based **T**iming **A**nalysis (PITA)* mechanism can find the *realizable* maximum circuit delay under the floating mode of operation accurately. Moreover, it finds a set of paths, called *primitive anchor paths*, the maximum of whose delays gives the maximum circuit delay. The primitive anchor path set is a subset of all paths in the circuit which satisfy the $\text{SENV}_{\text{loose}}$ criterion [6], consequently there are fewer primitive anchor paths in a circuit than viable paths [3].

The primitive PDFs in a circuit do not depend on component delays. So if all 'long' primitive PDFs (i.e., those with delay greater than some lower bound value) are identified, they need not be re-identified if one or more components are speeded up. Thus, in a delay optimization scenario, our sensitization analysis needs to be done only once. In contrast, using previously reported floating mode sensitization analyses in this scenario would require iteratively identifying and speeding up critical paths, necessitating re-evaluation of the sensitizability of at least a

This research was supported by the Semiconductor Research Corporation under grant No. DC068-95.

* Presently with the Design Technology Center, Hewlett-Packard Co., Palo Alto, CA.

0-89791-993-9/97 $10.00

subset of the paths at every iteration.

In the next section, we explain primitive PDFs and introduce necessary terminology. In Section 3, we establish the relationship between primitive PDFs and the maximum delay of a circuit. Section 4 discusses our timing analysis mechanism. Section 5 compares our method to previously reported ones. Results for benchmark circuits are presented in Section 6. We present our conclusions in Section 7.

2. Background

2.1. Primitive path delay faults

Under the *path delay fault model* [8], a *path is said to have a delay fault* if its delay is greater than a given timing constraint. A single path delay fault (SPDF) corresponds to a delay fault on a single path. A non-robust SPDF corresponds to a path that is singly non-robustly testable [9][10]. The side-input conditions along the gates of such a path are the same as that for a statically sensitizable path [2], so such a path may also be called a *singly statically sensitizable path*. A multi path delay fault (MPDF) corresponds to the simultaneous presence of delay faults on more than one path. A non-robust MPDF corresponds to a jointly non-robustly testable (jointly statically sensitizable) path set. For a path set $\Pi = \{\pi_1, \pi_2, ...\}$ to be jointly statically sensitizable, only those gate side-inputs along each path π_i which do not lie on any other path $\in \Pi$ need to satisfy the static sensitization conditions.

In the example shown in Figure 1 [11], the path set

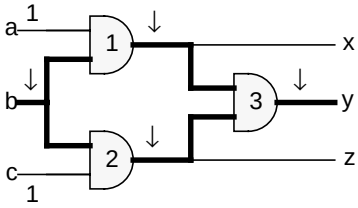

Figure 1. Illustration of an MPDF.

{↓*b-g1-g3-y*, ↓*b-g2-g3-y*} is jointly statically sensitizable. Moreover, for the test shown, the circuit output *y* is faulty only if the delays of both paths ↓*b-g1-g3-y* and ↓*b-g2-g3-y* are greater than the timing constraint T, since they both present final controlling values to gate *g3*. In other words, the delay of the jointly statically sensitized path set {↓*b-g1-g3-y*, ↓*b-g2-g3-y*} is determined by the shorter of the two paths.

In general, given a set of paths Π ($\Pi = \{\pi_1, \pi_2, ...\}$), a path delay fault (PDF) Π corresponds to the simultaneous presence of delay faults on all the paths $\in \Pi$. A jointly non-robust PDF corresponds to a jointly non-robustly testable (*jointly statically sensitizable*) path set. All occurrences of the word 'jointly' in this document should be interpreted to mean 'singly' when the cardinality of the path set is equal to 1. Also, we will use the terms 'PDF' and 'path set' interchangeably, and the terms 'non-robustly testable' and 'statically sensitizable' interchangeably in the remainder of the text. Note that the delay of a jointly statically sensitizable path set Π is determined by the shortest path $\in \Pi$. This can be easily inferred from the argument presented above in the context of Figure 1. More precisely, under a non-robust test for Π, along each of the paths $\in \Pi$, any side-input with a controlling final value has to belong to another path $\in \Pi$ (because Π is *jointly* statically sensitized), and so the shortest path $\in \Pi$ will present the earliest controlling value at each such gate.

Primitive PDFs have been formally defined in [12]. Informally, singly statically sensitizable paths belong to the set of primitive PDFs. Also, sets of paths Π, which are jointly statically sensitizable and *minimal* (i.e., no subset of Π is jointly statically sensitizable), belong to the set of primitive PDFs. Also note that since the single/joint static sensitization criterion (gate side-inputs must eventually settle to non-controlling values) is delay independent, the set of primitive PDFs in a circuit remains the same even if component delays change (i.e., the set of primitive PDFs is delay invariant).

Furthermore, the authors of [12] have stated and proved a theorem which we restate as follows:

Theorem 1: *The circuit will exhibit fault free timing behavior for all input vectors and for all timing constraints larger than some given value (say, T) if and only if it can be guaranteed that all the primitive path sets are free of delay faults for T.*

Note that a primitive path set is delay fault free if even one of its constituent paths has delay less or equal to the given timing constraint. This follows directly from the fact that the delay of a jointly statically sensitizable path set is determined by the shortest path belonging to the path set.

2.2. Terminology

A *family of circuits* corresponds to a given logic-level circuit netlist with associated component delay models. The delay of a component depends on many factors: input transition time, output loading, signal activity on neighboring components, internal component state (i.e., floating node voltage in MOS gates), fabrication process fluctuations, supply voltage and operating temperature. A circuit instance is normally defined as a circuit obtained by setting the fabrication process parameters to fixed values. For our purposes however, we define a *circuit instance* to be a circuit obtained by setting its component delays to fixed values within their respective component delay models. In the remainder of the text, we will use the term *circuit* to

imply a family of circuits, unless noted otherwise.

The *maximum circuit instance delay* under a given mode of operation assumption is the maximum delay of the circuit instance over all possible inputs under this mode of operation, accounting for path sensitization. The *maximum circuit delay* under a given mode of operation assumption and a given component delay model is the maximum delay of the circuit over all possible inputs under the given mode of operation and over all circuit instances possible under the given component delay model, again accounting for path sensitization. The circuit instance at which this maximum occurs is termed the *worst-case circuit instance*. The *maximum path delay* is the maximum delay a path can attain for a given component delay model.

3. Relating primitive PDFs to timing analysis

3.1. Maximum circuit instance delay

Say a given circuit has a set of primitive path sets $P = \{\Pi_{p1}, \Pi_{p2}, ..., \Pi_{pN}\}$. Let us denote the circuit instance by the variable $\underline{x}$, which represents all the parameters that determine the delay of each component in the circuit. From Theorem 1, we can bound the maximum delay of the circuit instance as the maximum of the primitive path set delays at that instance, i.e.:

$$\text{maxdelay}_{\underline{x}}(\text{ckt}) = \max\{\text{delay}_{\underline{x}}(\Pi_{p1}), ..., \text{delay}_{\underline{x}}(\Pi_{pN})\} \quad \text{(EQ1)}$$

We call this the *PITA maximum circuit instance delay.* Recall that the delay of a primitive path set (for that matter, any jointly statically sensitizable path set) Π, $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$, is determined by the shortest path $\in \Pi$, i.e.,

$$\text{delay}_{\underline{x}}(\Pi) = \min\{\text{delay}_{\underline{x}}(\pi_1), ..., \text{delay}_{\underline{x}}(\pi_n)\} \quad \text{(EQ2)}$$

The shortest path $\pi_{pi,j}$, which determines the delay of the primitive path set Π_{pi} at circuit instance $\underline{x}$, is called the *primitive anchor path* of Π_{pi}. Correspondingly, given an arbitrary jointly statically sensitizable path set $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$, the shortest path π_j, which determines the delay of Π at circuit instance $\underline{x}$, is called the *anchor path* of Π.

Theorem 2: *The PITA maximum circuit instance delay is exactly equal to the maximum circuit instance delay under the floating mode of operation.*

Proof: Since every non-primitive path set (i.e., a jointly statically sensitizable path set that is not minimal) in the circuit is a superset of some primitive path set, every non-primitive path set's delay is upper bounded by some primitive path set's delay. Therefore, the PITA maximum circuit instance delay, which is the maximum of all the primitive path set delays (i.e., the maximum of all the primitive anchor path delays), is also equal to the maximum of the delays of all jointly statically sensitizable path sets (i.e., the maximum of all anchor path delays). The set of all anchor paths in a circuit are exactly those paths in the circuit that satisfy the $\text{SENV}_{\text{loose}}$ criterion. This follows directly from the gate side-input conditions along a path that satisfies the $\text{SENV}_{\text{loose}}$ criterion (at each gate along the path, the on-path input is the earliest arriving controlling value, otherwise all inputs to the gate settle to non-controlling values). Therefore, the PITA maximum circuit instance delay is exactly equal to the maximum delay of the circuit instance found using the $\text{SENV}_{\text{loose}}$ criterion, which is equal to the maximum circuit instance delay under the floating mode of operation assumption. ■

In proving Theorem 2, we notice that the set of all anchor paths in a circuit are exactly those paths in the circuit that satisfy the $\text{SENV}_{\text{loose}}$ criterion. We also know that the set of all primitive anchor paths is a subset of the set of all anchor paths. We therefore infer the following:

Corollary 2: *The number of primitive anchor paths in a circuit is less than the number of paths in the circuit that satisfy the $\text{SENV}_{\text{loose}}$ criterion, and consequently also less than the number of viable paths in the circuit.*

3.2. Maximum circuit delay

The maximum circuit delay is given as the maximum of the maximum circuit instance delay (Equation EQ1) over all possible circuit instances:

$$\begin{aligned}\text{maxdelay}(\text{ckt}) &= \max_{\forall \underline{x}} \text{maxdelay}_{\underline{x}}(\text{ckt}) \\ &= \max_{\forall \underline{x}} [\max\{\text{delay}_{\underline{x}}(\Pi_{p1}), ..., \text{delay}_{\underline{x}}(\Pi_{pN})\}] \\ &= \max\left\{\max_{\forall \underline{x}} \text{delay}_{\underline{x}}(\Pi_{p1}), ..., \max_{\forall \underline{x}} \text{delay}_{\underline{x}}(\Pi_{pN})\right\} \\ &= \max\{\text{maxdelay}(\Pi_{p1}), ..., \text{maxdelay}(\Pi_{pN})\}\end{aligned} \quad \text{(EQ3)}$$

We call this the *PITA maximum circuit delay.* The maximum delay of a primitive path set (in fact, any jointly statically sensitizable path set) Π, $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$, can be derived from Equation EQ2:

$$\begin{aligned}\text{maxdelay}(\Pi) &= \max_{\forall \underline{x}} \text{delay}_{\underline{x}}(\Pi) \\ &= \max_{\forall \underline{x}} \min\{\text{delay}_{\underline{x}}(\pi_1), ..., \text{delay}_{\underline{x}}(\pi_n)\}\end{aligned} \quad \text{(EQ4)}$$

When the component delays are completely uncorrelated, the above expression can be rewritten as:

$$\text{maxdelay}(\Pi) = \min\left\{\max_{\forall \underline{x}} \text{delay}_{\underline{x}}(\pi_1), ..., \max_{\forall \underline{x}} \text{delay}_{\underline{x}}(\pi_n)\right\} \quad \text{(EQ5)}$$

i.e.:

$$\text{maxdelay}(\Pi) = \min\{\text{maxdelay}(\pi_1), \ldots, \text{maxdelay}(\pi_n)\} \quad \text{(EQ6)}$$

This implies that under the assumption that the delay of each component varies independently of the others within the given bounds, the maximum delay of a jointly statically sensitizable path set is given by the minimum of the maximum delay of each path belonging to this set. Note that since the component delays vary mutually independently, the maximum delay of each path π_i in Equation EQ6 can be found by adding up the maximum possible delay values of the components along π_i. Under independently varying component delays, the PITA maximum circuit delay is given by Equation EQ3 and Equation EQ6.

Theorem 3: *When the component delays are assumed to vary independently of each other, the PITA maximum circuit delay is exactly equal to the maximum circuit delay under the floating mode of operation.*

Proof: It has been proved in [6] that assuming independently varying component delays and under the floating mode of operation assumption, the maximum circuit delay is equal to the maximum circuit instance delay of that circuit instance whose component delay values are set to their individual maximum values. For this worst-case circuit instance, we can infer from Theorem 2 that the PITA maximum circuit instance delay is exactly equal to the maximum circuit instance delay under the floating mode of operation. Hence proved.

■

If component delays are correlated (to a large or a small extent), it may not be possible to set the delay of all components simultaneously to their individual maximum values (i.e., the circuit instance corresponding to all component delays set simultaneously to their individual maximum values may be unrealizable). Therefore, previously reported floating mode timing analyzers which perform the sensitization analysis under a floating mode of operation with all component delays set to their individual maximum values will yield a pessimistic estimate of the realizable maximum circuit delay. By using Equation EQ3 and Equation EQ4 to find the maximum circuit delay under a general component delay model, the PITA maximum circuit delay will not be pessimistic in this regard.

Theorem 4: *For any component delay model, the PITA maximum circuit delay is exactly equal to the realizable maximum circuit delay under the floating mode of operation.*

Proof: We can infer from Theorem 2 that the PITA maximum circuit instance delay is exactly equal to the maximum circuit instance delay under the floating mode of operation, i.e, equations Equation EQ1 and Equation EQ2 determine exactly the maximum delay of a circuit instance. Maximizing these equations over all possible (i.e., realizable) circuit instances give equations Equation EQ3 and Equation EQ4, which is precisely the PITA maximum circuit delay for any given component delay model. Hence proved.

■

4. Primitive PDF identification based timing analysis - PITA

The PITA approach is shown in Figure 2. It takes as input the gate-level netlist, a lower bound value T on the circuit delay, the component delay models and their respective upper bound (maximum) delay values. PITA operates in two modes. In the ‘critical path’ mode, it outputs the maximum circuit delay and the associated critical primitive path set (i.e., the primitive path set whose primitive anchor path determines this delay). In the ‘all paths’ mode, it identifies all ‘long’ primitive path sets, i.e., those primitive path sets whose maximum delays are greater than the input lower bound threshold value T.

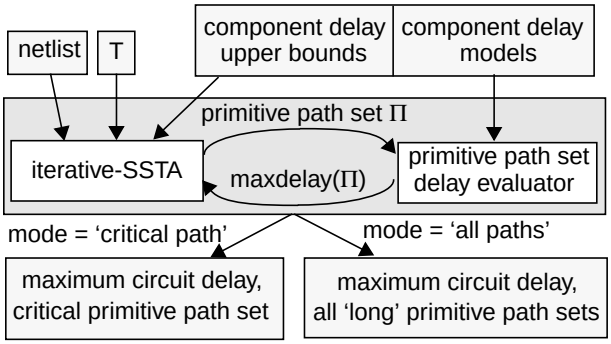

Figure 2. The PITA approach.

The core of the PITA methodology is based on the *iterative-SSTA* (*iterative S*ignal *S*tabilization *T*ime *A*nalysis) algorithm, which is used to identify primitive path sets whose delay upper bounds are greater than some lower bound threshold value. The iterative-SSTA procedure uses only the upper bounds on the component delays to identify these primitive path sets. Every time such a primitive path set is identified, the *primitive path set delay evaluator* is invoked which computes the maximum delay of the primitive path set using the user-input component delay models.

In the ‘all paths’ mode, the lower bound T value remains constant at the input value, so primitive path sets whose delay upper bounds are greater than the input T value are identified and reported as ‘long’ primitive path sets. In the ‘critical path’ mode however, as primitive path sets with delay upper bounds greater than T are identified, T is updated to the maximum of the maximum primitive path set delay found so far. The updated T value is then used as the new lower bound threshold for further identifying primitive path sets, and this process continues until no further primitive path sets are found. The resulting T value

is the PITA maximum circuit delay.

4.1. Iterative-SSTA

We have previously reported our primitive path set identification procedure [13] which is based on the iterative-SSTA algorithm. We only give a brief overview of the procedure here.

Consider a 4-valued logic system {S0, S1, U0, U1}, which represents the possible signal values under a 2-vector delay test $\bar{v} = \langle v_1, v_2 \rangle$. The symbols S0, S1, U0, and U1 represent the stable logic 0 value, the stable logic 1 value, possible transitions for the test but eventually settling to a logic 0 value, and possible transitions but eventually settling to a logic 1 value, respectively.

Under this logic system, for a 2-input AND gate (with inputs *a* and *b*, and output *x*), there are 16 stabilization scenarios since each gate input can be one of S0, S1, U0, or U1. Each stabilization scenario can be derived from the excitation table of the AND gate enhanced with the new logic system and delay constraints, shown in Figure 3. We

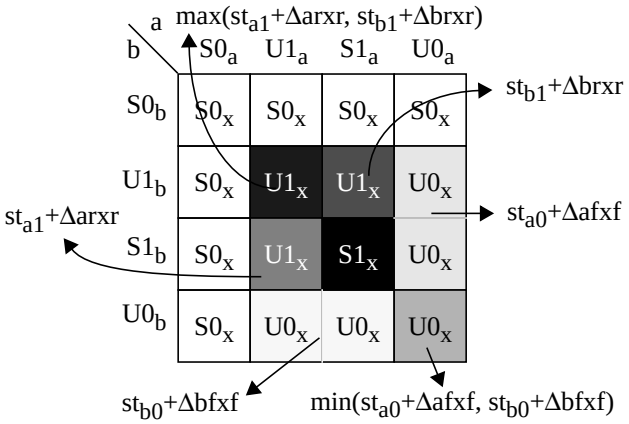

Figure 3. Stabilization table for an AND gate.

call this the *stabilization table* for the AND gate. Here, $U0_a$ means (a = U0), st_{x0} represents the time at which node *x* stabilizes to a logic 0 value, Δafxf represents the *a-to-x* gate delay for a falling input (and consequently a falling output), and so on. For instance, the bottom rightmost scenario is interpreted as follows: if both *a* and *b* can have transitions but eventually settle to a logic 0 value, then the gate output *x* stabilizes with possible hazards to a logic 0 value, and it is guaranteed to stabilize to its final logic value within the time $\min(st_{a0}+\Delta afxf, st_{b0}+\Delta bfxf)$.

From the stabilization table for a gate type, we can write out the SOP (Sum of Products) expression corresponding to all the stabilization scenarios. We call this the *stabilizing time expression propagation rule* (or, *ST-expression propagation rule*) for the gate type. The ST-expression propagation rule for the 2-input AND gate is:

$$
\begin{aligned}
st_{x0} > T_x \quad \Rightarrow\ & \\
& U0_a \wedge (S1_b \vee U1_b) \wedge \quad (st_{a0}+\Delta afxf > T_x) \\
\vee\ & (S1_a \vee U1_a) \wedge U0_b \wedge \quad (st_{b0}+\Delta bfxf > T_x) \\
\vee\ & U0_a \wedge U0_b \quad \wedge \quad (st_{a0}+\Delta afxf > T_x) \wedge (st_{b0}+\Delta bfxf > T_x) \\
st_{x1} > T_x \quad \Rightarrow\ & \\
& U1_a \wedge (S1_b \vee U1_b) \wedge \quad (st_{a1}+\Delta arxr > T_x) \\
\vee\ & (S1_a \vee U1_a) \wedge U1_b \wedge \quad (st_{b1}+\Delta brxr > T_x)
\end{aligned}
$$

Note here that $(S1_b \vee U1_b)$ and $(S1_a \vee U1_a)$ in the product terms can be interpreted as the appropriate side-input settling to a final non-controlling logic value under the floating mode of operation, while $U0_a$, $U0_b$, $U1_a$, and $U1_b$ in the product terms can be interpreted as the on-path input settling to a final 0/1 logic value under the floating mode of operation.

Starting from the primary inputs, we build *stabilizing time expressions* (*ST-expressions*) at each node in a levelized fashion by combining the ST-expressions at the gate inputs according to the ST-expression propagation rule at that gate. For a primary input *a*, the ST-expression is:

$st_{a0} > T_a \quad \Rightarrow \quad a(0\text{-}) \wedge \overline{a(0+)}, \quad st_{a1} > T_a \quad \Rightarrow \quad \overline{a(0\text{-})} \wedge a(0+)$

where a(0-) is the value of *a* for the first vector of the test and a(0+) is the value of *a* for the second vector of the test vector pair. The general form for the ST-expression at a node *n* is:

$$
\begin{aligned}
st_{n0} > T_n \quad \Rightarrow\ & Bcond_{n0}{}^{1} \wedge \{Cexpr_{n0}{}^{1} > T_n\} \\
\vee\ & \ldots \wedge \{\ldots\} \\
\vee\ & Bcond_{n0}{}^{N0} \wedge \{Cexpr_{n0}{}^{N0} > T_n\} \\
st_{n1} > T_n \quad \Rightarrow\ & Bcond_{n1}{}^{1} \wedge \{Cexpr_{n1}{}^{1} > T_n\} \\
\vee\ & \ldots \wedge \{\ldots\} \\
\vee\ & Bcond_{n1}{}^{N1} \wedge \{Cexpr_{n1}{}^{N1} > T_n\}
\end{aligned}
$$

where $Bcond_{n0}{}^{k}$ $(1 \le k \le N0)$, $Bcond_{n1}{}^{k}$ $(1 \le k \le N1)$ represent the Boolean conditions in terms of the primary input variables a(0-), a(0+), ..., such that node *n* may have transitions but stabilizes ultimately to a logic 0 and a logic 1 value respectively. $\{Cexpr_{n0}{}^{k} > T_n\}$ $(1 \le k \le N0)$, $\{Cexpr_{n1}{}^{k} > T_n\}$ $(1 \le k \le N1)$ are and-expressions of symbolic constraints on partial paths upto node *n*, and are of the form $\{(delay[path_i] > T_n) \wedge (delay[path_j] > T_n) \wedge \ldots\}$. It can be inferred that each Cexpr corresponds to a PDF (SPDF/MPDF) consisting of partial paths from the primary inputs to node *n*, and the corresponding Bcond represents its single/joint non-robust testability conditions (single/joint static sensitizability conditions).

While combining the ST-expressions at the gate inputs according to the ST-expression propagation rule at that gate, the terms corresponding to those symbolic partial path delay constraints which cannot be satisfied for the given component delay upper bound values are pruned away. Thus, the ST-expression at a node represents in SOP form the 'long' non-robust partial PDFs and their respec-

tive non-robust tests.

By propagating the ST-expressions through the levels of logic in a circuit, we can obtain all non-robust PDFs (i.e., jointly statically sensitizable path sets) at each circuit output. However, given a path set of cardinality m, there may be many path sets of cardinality n (> m) which are supersets of this path set, and hence are not primitive. Computation of these non-primitive path sets adds to the memory overhead, and therefore is not computationally efficient. To overcome this problem, we do not compute the complete ST-expression at each node in one shot. Instead, we do this iteratively, where in each iteration i, we compute the ST-expression at each node only for a range $[l_i, u_i]$ of path set cardinalities. At the end of the i^{th} iteration, we identify the primitive path sets in the range of cardinalities $[l_i, u_i]$. This information is then stored on the edges of the circuit, and is used for pruning path sets of higher cardinalities during the ST-expression propagation in succeeding iterations.

4.2. Primitive path set delay evaluation

The delay evaluator is invoked every time the iterative-SSTA algorithm finds a primitive path set whose delay upper bound (i.e., maximum delay using component delay upper bounds) is greater than T. The delay evaluator finds the maximum delay of the primitive path set for the given component delay model. The delay evaluator is specific to the component delay model. For component delay models that are mutually independent (fixed, monotone speedup [3], and bounded delay models [7]), Equation EQ6 is used to evaluate this delay. If component delays are correlated, then Equation EQ4 must be used.

5. Comparisons

The basis for our primitive path set identification procedure is the stabilization tables for individual gate types (e.g., refer to Figure 3 for the stabilization table of a 2-input AND gate). These stabilization tables correspond to the SENV sensitization criterion, which is the set of necessary and sufficient conditions for a path to be sensitizable under the floating mode of operation assumption. The sensitization criterion proposed in [4] also corresponds to the SENV criterion. However, the manner in which the sensitization criterion is applied for timing analysis is different between the two. In [5] (which is a companion paper of [4]), an approach has been presented to answer the question “Is there a true path of delay ≥ T?”. This is based on simultaneously determining the sensitizability of all paths whose delays are greater than or equal to T, and it works by finding a test for a single stuck-fault at the circuit output using a timed D-calculus that is derived from the SENV criterion. To find the maximum circuit instance delay, the algorithm may need to be run multiple times for different T values. In PITA, only gate functionality is used to identify primitive paths sets. The component delay upper bounds are used only to prune jointly statically sensitized partial path sets whose delay upper bounds are less than some threshold value, and are not used to determine which gate input signal dominates the gate output at each gate. In other words, instead of performing a delay-dependent resolution at each gate, the partial path delay constraints are stored symbolically in the ST-expression at the gate. The component delay model itself is used only by the delay evaluator to compute the maximum delay of the primitive path sets. The advantage of doing so is in the ability to handle fabrication process effects, signal propagation effects, and signal interaction effects.

5.1. General component delays

The significance of the PITA approach is underscored in dealing with fabrication process, signal propagation, and signal interaction effects. Under these effects, not all components along a path may exhibit their maximum delay values simultaneously, so adding up component delay maxima (or upper bounds) along paths will result in a very pessimistic upper bound of the realizable maximum path delay. It may be argued that using a conventional floating mode timing analyzer, the maximum circuit delay accounting for these effects may be found as a two step process: firstly, determine the critical path using the floating mode timing analyzer with the component delays set at their individual upper bounds, and then find the maximum delay of the critical path accounting for these effects. This analysis is incorrect since, for the component delay value combination where the delay of the critical path is maximized under these effects, the path may be false. Moreover, a path whose delay upper bound is less than that of this critical path may in fact have a larger maximum delay. Essentially, without creating symbolic partial path delay expressions during the path sensitization analysis, the best the previously reported floating mode timing analyzers can do is to report as maximum circuit delay, the delay found by performing path sensitization analysis with all component delays simultaneously set to their maximum values.

In PITA, the iterative-SSTA algorithm uses the component delay upper bounds (i.e., component delay maxima) to identify primitive path sets whose delay upper bounds exceed the current lower bound threshold value, and the delay evaluator uses Equation EQ4 to determine the maximum primitive path set delay using component delay models that account for these fabrication process, signal propagation, and signal interaction effects. This

procedure will result in the realizable maximum circuit delay. Note that the maximum primitive path set delay evaluation may not be a trivial task: a non-linear numerical optimization may need to be performed if the component delays are expressed as functions of fabrication process fluctuations, and a signal hazard analysis may need to be performed to determine the worst-case scenario in the case of signal interaction effects.

5.2. Post-layout delay optimization

In the post-layout delay optimization scenario, given a circuit layout which has sensitizable paths whose delays exceed some given timing constraint, the objective is to speed up the circuit such that all sensitizable paths have delay less than the timing constraint. In this scenario, it follows from Theorem 1 that it is necessary and sufficient to speed up all the 'long' primitive anchor paths (i.e., those primitive anchor paths whose maximum delay is greater than the given timing constraint). PITA can be executed in 'all paths' mode to identify all 'long' primitive path sets, which can then be speeded up using transistor resizing, wire resizing, interconnect re-routing, buffer insertion etc. Here, the sensitization analysis to identify primitive path sets needs to be done only once.

If a conventional floating mode timing analyzer were to be used in this scenario, the path sensitization analysis may have to be performed multiple times. We justify this in the following argument. In the post-layout delay optimization scenario, say all the 'long' sensitizable paths (i.e., those whose delays exceed the given timing constraint) have been identified using some floating mode timing analyzer. It can be shown that it is neither necessary, nor sufficient to speed up all these 'long' sensitizable paths. Essentially, when a 'long' sensitizable path is speeded up, it may result in other 'long' sensitizable paths becoming false and also result in some 'long' false paths becoming sensitizable. Therefore, the path sensitizabilities would have to be re-evaluated every time a 'long' sensitizable path is speeded up, resulting in a situation where critical paths are identified and speeded up one at a time in an iterative manner. This is very inefficient compared to the manner in which PITA needs to be used just once.

6. Results

Results obtained from using PITA to determine the maximum circuit delay of some large ISCAS'85, ISCAS'89 and Logic synthesis'91 benchmark circuits are shown in Table 1 for two component delay models: the *unit component delay model* where every component (gate, wire) in the circuits is assumed to have unit delay, and a *k-factor delay model* [14] where gate delays are expressed as empirically obtained functions of input transition time and output load capacitance.

Table 1: PITA results.

Circuit	No. gates	No. paths	Unit delay		K-factor delays		
			PERT	PITA	PERT	PITA (no correl)	PITA (correl)
c880	402	17284	100	100	100	100	93.08
c432	251	583652	100	100	100	97.65	94.19*
c499	502	795776	100	100	100	100	99.47
c1355	598	8.34 e6	100	100	100	-	-
c1908	664	1.46 e6	100	93.10	100	97.19	95.32
c3540	1366	5.74 e7	100	-	100	95.61	94.69
c5315	2328	2.68 e6	100	98.10	100	98.38	96.51
dalu	2061	121636	100	100	100	94.44	81.02*
k2	2748	20808	100	100	100	100	76.87
i10	3194	1.38 e7	100	96.26	100	93.57	83.20*
i8	3590	105242	100	94.29	100	93.37	90.06
s13207.1	8592	2.69 e6	100	-	100	99.67	99.12
s1423	674	89452	100	100	100	99.18	94.09
s15850.1	10197	3.29 e8	100	-	100	96.17	94.92
s38584.1	21308	2.16 e6	100	100	100	100	97.87
s5378	3208	27046	100	100	100	100	99.30
s713	452	43624	100	92.45	100	93.67	90.64
s9234.1	5944	489708	100	100	100	99.55	95.02

In the results for each circuit in the unit component delay model, the PITA maximum circuit delay is shown normalized relative to the maximum topological delay (PERT delay [1]). Fairly large circuits (~21K gates, ~14 million paths) were handled by PITA for the unit delay model. However, the program ran out of memory for c3540, s13207.1 and s15850.1 before giving any result.

Results for the k-factor delay model are shown in the last three columns of the result table. The k-factor delay models for the gate library were obtained from SGS-Thomson Microelectronics, Agrate, Italy. Since we did not use circuit layouts, the load capacitance at each gate was assumed to be proportional to its fanout and the interconnect delays were assumed to be zero. Under the k-factor delay model, the upper bounds on the delay from a gate input to the gate output were calculated by evaluating the appropriate k-factor function under the maximum possible input transition time that can be seen by that gate input. The PERT delay shown is found by using these component delay upper bounds. Also, the transition time at a gate input affects the delay of the gate and also the transition time of the gate output, which in turn affects the delay of the fanout gates. Hence, component delays are correlated under the k-factor delay model due to signal propagation effects. Previously reported floating mode timing analyzers cannot account for these correlations since they use partial path delay upper bounds (derived by adding component delay maxima) during the path sensitization analysis to determine which signal dominates at a gate input. Results obtained by ignoring these correlations are shown

in the 'PITA (no correl)' column, which is the delay that would be reported by these floating mode timing analyzers, while the PITA *realizable* maximum circuit delay determined by incorporating these correlations is shown in the final column. Once again, the circuit delay numbers are shown normalized with respect to the corresponding PERT delay. It is observed for many circuits that not accounting for component delay correlations during the path sensitization analysis is as pessimistic as (or in some cases, even more pessimistic than) not accounting for path sensitization. For e.g., for circuit k2, path sensitization analysis without accounting for component delay correlation does not decrease the maximum circuit delay estimate at all, while PITA path sensitization analysis accounting for component delay correlation tightens the maximum circuit delay estimate by 23.13% (=100– 76.87).

For circuits c432, dalu and i10, the program ran out of memory while calculating primitive path sets of high cardinalities. Nevertheless, the PITA implementation successfully identified 'long' primitive path sets of lower cardinalities, therefore, the numbers shown (marked with an asterisk) are lower bounds on the maximum circuit delay. Also note that the program ran out of memory for c1355 without giving any result. It is interesting to observe that the benchmark circuits for which the program ran out of memory for the two component delay models (i.e., the unit component delay model and the k-factor delay model) are different. This is because the effectiveness of ST-expression pruning in the iterative-SSTA algorithm depends on the component delay upper bounds, which are different for the two models.

For both component delay models, most circuits took only a few seconds of CPU time on an IBM/RS6000 58H PowerPC computer with 256 Mbytes of real memory. Some of the larger ones took a few minutes of CPU time, but note that the CPU time is very dependent on the choice of the user-input lower bound T value, and hence has not been reported here.

7. Conclusions

We have developed a new method for timing analysis (PITA) based on identifying primitive path sets in a circuit using component delay upper bounds and evaluating their maximum delay for user-input component delay models. We have defined the PITA maximum circuit instance delay as the maximum of the delays of the primitive path sets in the circuit, and proved that it is exactly equal to the maximum circuit instance delay under the floating mode of operation. Furthermore, the PITA maximum circuit delay is also shown to determine exactly the maximum circuit delay under the floating mode of operation. Previously published floating mode timing analyzers are capable of determining the maximum circuit delay accurately only for independently varying component delays, and are pessimistic in the presence of fabrication process, signal propagation, and signal interaction effects. The PITA approach provides a framework where these effects can be taken into account very accurately since the iterative-SSTA algorithm outputs symbolic primitive path set information rather than performing a delay-dependent resolution at each gate. We have demonstrated this for benchmark circuits for a k-factor delay model. It is also argued that post-layout delay optimization can be more efficiently performed with PITA than with conventional floating mode timing analyzers because the set of primitive path sets is invariant with respect to delay.

References

[1] T. Kirkpatrick and N. Clark, "PERT as an aid to logic design," IBM Tech. Rep., 1966.

[2] J. Benkoski, E. Vanden Meersch, L. J. M. Claesen, H. De Man, "Timing verification using statically sensitizable paths," *IEEE Trans. Computer-Aided Design*, vol. 9, Oct. 1990, pp. 1073-1084.

[3] P. McGeer and R. Brayton, "Efficient algorithms for computing the longest viable path in a combinational network," *Proc. 26th Design Automation Conf.*, 1989, pp. 561-567.

[4] S. Devadas, K. Keutzer, and S. Malik, "Computation of floating mode delay in combinational logic circuits: Theory and algorithms," *IEEE Trans. Computer-Aided Design*, vol. 12, Dec. 1993, pp. 1913-1922.

[5] S. Devadas, K. Keutzer, S. Malik, and A. Wang, "Computation of floating mode delay in combinational logic circuits: Practice and implementation," *IEEE Trans. Computer-Aided Design*, vol. 12, Dec. 1993, pp. 1923-1936.

[6] H. Chen and D. Du, "Path sensitization in critical path problem," *Proc. TAU'90.* ACM-SIGDA, 1990.

[7] S. Devadas, K. Keutzer, S. Malik, and A. Wang, "Certified timing verification and the transition delay of a logic circuit," *IEEE Trans. VLSI Systems*, vol. 2, Sept. 1994, pp. 333-342.

[8] G. L. Smith, "Model for delay faults based upon paths," *Proc. Int'l Test Conference*, Nov. 1985, pp. 324-349.

[9] C. J. Lin and S. M. Reddy, "On delay fault testing in logic circuits," *IEEE Trans. Computer-Aided Design*, vol. 6-5, Sept. 1987, pp. 694-703.

[10] E. S. Park and M. R. Mercer, "Robust and nonrobust tests for path delay faults in a combinational circuit," *Proc. Int'l Test Conf.*, 1987, pp. 1027-1034.

[11] S. M. Reddy, "Delay faults: Modeling, fault simulation, and test generation," *Int'l Test Conf.*, 1995, Tutorial 8.

[12] W. Ke and P. R. Menon, "Synthesis of delay-verifiable combinational circuits," *IEEE Trans. Computers*, vol. 44-2, Feb. 1995, pp. 213-222.

[13] M. Sivaraman and A. J. Strojwas, "Primitive path delay fault identification," *Proc. 10th Int'l Conf. on VLSI Design*, 1997, pp. 95-100.

[14] N. H. E. Weste and K. Eshraghian, "Principles of CMOS VLSI design," *2nd edition, Addison-Wesley Publishing Company*, 1993.