Functional Timing Analysis Made Fast and General

Yi-Ting Chung and Jie-Hong R. Jiang, *Member, IEEE*

***Abstract*—Functional, in contrast to structural, timing analysis for circuit delay computation is accurate, but computationally expensive in refuting false critical paths. Despite recent progress on satisfiability-based functional timing analysis, the formulation generality and computation efficiency remain room for further improvement. This paper provides a unified view on different notions of timed characteristic functions and efficient transformation for satisfiability solving. Experimental results show functional timing analysis on industrial designs can be made up to several orders of magnitude faster and more generally applicable than prior methods.**

***Index Terms*—functional timing analysis, satisfiability solving, timed characteristic function.**

I. Introduction

In modern synthesis flow of very large scale integration (VLSI) design, timing analysis is essential in identifying timing critical regions for re-synthesis, determining operable clock frequencies, and avoiding wasteful over-optimization and thus accelerating design closure in meeting stringent timing constraints. As timing analysis often has to be repeatedly performed, how to make the computation efficient and accurate becomes a crucial task.

There are two main approaches to timing analysis. *Static timing analysis* (STA), based on pure structural (or topological) analysis, though fast with linear-time complexity, can be too pessimistic in estimating circuit delay due to the ignorance of false or nonsensitizable paths [3]. *Functional timing analysis* (FTA), on the other hand, provides accurate delay calculation, but is computationally intractable, i.e., NP-hard, in identifying false critical paths [10].

Many FTA algorithms, e.g., [1], [3], [5], [6], [9], [10], [13], [14], [16], [18], have been proposed. When delay-dependency is concerned, an FTA algorithm can be delay-independent [5] or delay-dependent [3]. The former (latter) identifies true and false paths without (with) respect to some timing library. Whereas the former is incomplete in that not every delay path can be concluded true or false regardless of arbitrary delay assignments, this paper focuses on the latter analysis.

When the underlying computation engine is concerned, an FTA algorithm can be equipped with an automatic test pattern generator, e.g., [1], [6], or by a satisfiability (SAT) solver, e.g., [9], [13], [18]. Since ATPG-based computation involves sophisticated circuit transformation and multi-fault testing, it is difficult to implement and scale. In contrast, SAT-based computation allows simple implementation due to its clean separation between timed characteristic function (TCF) construction [10] and SAT solving. Although recent advances in SAT solving techniques [7], [11], [12] make SAT-based FTA a viable approach, FTA for large industrial designs remains challenging due to the massive numbers of variables and clauses when translating a complex TCF into a conjunctive normal form (CNF) formula for SAT solving. On the other hand, prior SAT-based FTA methods focused only on maximum delay computation; other timing computation problems naturally addressable under the TCF formulation are largely ignored. Moreover, prior methods [9], [18] cannot handle arbitrary gate types. Although formulation for general gate types has been proposed in [13], its complex formulas make SAT solving inefficient.

The limitations of prior methods motivate this work towards the development of a general TCF framework supporting fast and scalable computation for various FTA problems and arbitrary gate types. The main results include

1) a generalized FTA framework supporting delay computation, and analysis for combinational cyclic circuits,
2) a unified implication-based CNF encoding for various types of TCFs for arbitrary complex gate types under both combined and separate rise/fall-time delay models,
3) a TCF reduction technique with an improved equivalence table, based on the list of essential arrival times,
4) a reusing strategy reducing CNF variables,
5) a model generation mechanism, which produces a true critical path along with its sensitization condition if the target delay is sensitizable, and
6) an algorithm to identify timing critical regions of a circuit for potential timing optimization.

Experimental results show that our method achieves substantial speedup over prior FTA methods and effective critical region identification.

The rest of this paper is organized as follows. Section II provides the background of circuit behavior model and sensitization criteria. TCFs and their effective CNF translations are then presented in Section III, and the reduction techniques of TCFs are showed in Section IV. Section V presents efficient and general FTA algorithms for delay computation and critical region identification. Experimental results are evaluated in Section VI. Finally, conclusions are given in Section VII.

II. Preliminaries

A. Propositional Satisfiability

In this paper we encode an FTA problem as a propositional formula in the *conjunctive normal form* (CNF) for SAT

Manuscript received September XX, 20XX; revised November XX, 20XX. A preliminary version of this paper appeared in [4]. This work was supported in part by the National Science Council under grants NSC 99-2221-E-002-214-MY3, 99-2923-E-002-005-MY3, and 101-2923-E-002-015-MY2.

Y.-T. Chung is with Synopsys, Inc., Taipei, Taiwan (E-mail: q82465@gmail.com).

J.-H. R. Jiang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan (E-mail: jhjiang@cc.ee.ntu.edu.tw).

solving. The dual representation of CNF is the *disjunctive normal form* (DNF). Precisely, a CNF (DNF) formula is a conjunction of clauses (disjunction of cubes); a *clause* (*cube*) is a disjunction (conjunction) of literals; a *literal* is a Boolean variable or its negation. The corresponding literal of variable x in a cube or clause c is denoted as $lit(x) \in c$. The *polarity* of literal l, denoted $pol(l)$, for variable x is defined to be 0 if $l = \neg x$ and 1 if $l = x$. The satisfiability problem asks whether there exists a satisfying assignment to the set of variables that makes a CNF formula true. The reader is referred to [12], [11], [7] for modern SAT solving techniques, and to [19], [15] for circuit-to-CNF conversion.

B. Circuit Model

A (combinational) circuit $C(N, E)$ is composed of nodes (or gates) N and directed edges $E \subseteq N \times N$. Two disjoint subsets of N are distinguished as primary inputs (PIs) and primary outputs (POs). Each node is associated with two attributes: function and delay. We assume the function can be arbitrary, from simple gate types, such as buffer, inverter, NAND, NOR, etc., to complex function units, such as XOR, multiplexer, AOI, etc. In the sequel, we sometimes do not distinguish a node from its function and its output variable when it is clear from the context. We assume the gate delay can vary from pin to pin and vary between rise and fall time. Without loss of generality, interconnect delays are assumed to be integrated into the gate delays under this timing model.

For a node f in a circuit, we let $FI(f)$ and $FO(f)$ denote the fanin and fanout nodes of f, respectively. For $g \in FI(f)$, we say g is of *controlling value* $v_c \in \mathbb{B} = \{0, 1\}$ of f if the output value of f can be completely determined by g with value v_c regardless of the truth assignments to other inputs. On the contrary, g is of *non-controlling value* $v_n \in \mathbb{B}$ of f if the output value of f cannot be completely determined by g with value v_n without referring to the truth assignments to other inputs. For example, any input of an AND gate is of controlling value 0. For a complex gate, such as XOR, its inputs may not have controlling values however.

The notion of controlling values can be generalized to *controlling cubes*. For a complex gate f, a truth assignment to a minimal (strict) subset $S \subset FI(f)$ that determines the output value of f independent of other fanins forms a controlling cube. Specifically, let P_1 and P_0 denote the sets of all prime implicants (or, simply, primes) of the onset and offset of f, respectively. The set of all controlling cubes can be obtained by the union of P_1 and P_0 excluding the minterm primes, that is, the primes whose literal counts equal the size of $FI(f)$. In addition, a literal in a controlling cube c is called a *controlling literal* of c. For example, the controlling cubes of the gate f with function $P_1 = \{xy, \neg x\neg y\neg z\}$ and $P_0 = \{x\neg y, \neg xy, \neg xz\}$ has a controlling cube set $C = \{xy, x\neg y, \neg xy, \neg xz\}$, and $\neg y$ is a controlling literal of cube $x\neg y$.

C. Sensitization Criteria

Among the various modes of circuit operation when functional timing analysis is concerned, *floating-mode operation*, which we adopt, is the most popular due to its simplicity and robustness [10]. Under this mode of operation, the signals of a circuit are of unknown initial values and steady to their final values induced by a set of truth assignments on the PIs. Therefore only one input vector is necessary for the timing analysis. In contrast, a two-vector based analysis, e.g., [16], does not respect the *monotone speedup property*, which maintains that the analyzed circuit delay should not increase due to the delay decrease of some constituent delay elements [10]. Moreover, since the two-vector analysis is with respect to fixed exact delays, it is improper in practice because often the delay of a delay element varies. In contrast, a delay in floating mode analysis is treated as an upper/lower bound. The analysis obeys the monotone speedup property and handles delay variation naturally. Therefore it is more desirable for logic synthesis than two-vector analysis.

Under the floating-mode operation, various path sensitization criteria can be defined.The *exact criterion* [3] and *viable criterion* [10] are two commonly studied criteria. When the truth and falsity of a specified path is concerned, the analysis of the former is exact whereas that of the latter is conservative [3]. Nevertheless, when the timing analysis is performed for all paths of a circuit without tracing a particular path, the viable criterion becomes exact as was shown in [14]. This paper is mainly concerned with computing the longest/shortest true delay among all paths.

III. Timed Characteristic Functions

A *timed characteristic function* of a node f characterizes a set of PI assignments that make the output value of f change from its initial unknown value to a final steady value and meet a specified timing requirement. Specifically, an *earlier TCF* $\chi^{f,<t}$ (respectively, a *later TCF* $\chi^{f,>t}$) is satisfiable if there exists some PI assignment making f steady earlier (respectively, later) than time t. Similarly, a *no-earlier TCF* $\chi^{f,\geq t}$ (respectively, a *no-later TCF* $\chi^{f,\leq t}$) is satisfiable if there exists some PI assignment making f steady no-earlier (respectively, no-later) than time t. Clearly $\chi^{f,<t} = \neg\chi^{f,\geq t}$ and $\chi^{f,>t} = \neg\chi^{f,\leq t}$. Moreover, a *0/1-specified TCF* $\chi^{f=v,\Box t}$, for $v \in \{0, 1\}$ and $\Box \in \{<, \leq, >, \geq\}$, further constrains the final steady value of f to logic value 0 or 1. For example, $\chi^{f=1,<t}$ characterizes the set of PI assignments that make f steady to value 1 earlier than time t. By the TCF definition, we know that $\chi^{f,\geq t_1} \rightarrow \chi^{f,\geq t_2}$ for $t_1 \geq t_2$, and such implication properties hold for other types of TCFs as well. Note also that $\chi^{f,<t} = \chi^{f=0,<t} \vee \chi^{f=1,<t}$. Although adding 0/1-specificity may double the TCF formula size, it allows distinction between rising and falling delays and thus permits more accurate timing analysis.

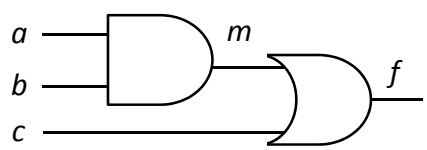

Fig. 1. Circuit under Timing Analysis

Example 1: Consider the circuit of Figure 1. Assume every gate is of 1 unit delay, and the signal arrival times of inputs a, b, and c are 0. Then the various TCFs of f with $t = 2$ can be computed as follows.

$$\begin{array}{lll} \chi^{f=0,>2} = 0 & \chi^{f=1,>2} = 0 & \chi^{f,>2} = 0 \\ \chi^{f=0,\leq 2} = (\neg a \vee \neg b)\neg c & \chi^{f=1,\leq 2} = ab \vee c & \chi^{f,\leq 2} = 1 \\ \chi^{f=0,<2} = 0 & \chi^{f=1,<2} = c & \chi^{f,<2} = c \\ \chi^{f=0,\geq 2} = (\neg a \vee \neg b)\neg c & \chi^{f=1,\geq 2} = ab\neg c & \chi^{f,\geq 2} = \neg c \end{array}$$

It can be verified that $\chi^{f,\Box t} = \chi^{f=0,\Box t} \vee \chi^{f=1,\Box t}$ for $\Box \in \{<, \leq, >, \geq\}$, and $\chi^{f,>t} = \neg\chi^{f,\leq t}$. Since the maximum topological delay of this circuit is 2, the later TCFs $\chi^{f=0,>2}$, $\chi^{f=1,>2}$, and $\chi^{f,>2}$ are all false, that is, no PI assignments can make f settle to value 0 or 1 later than time 2. In contrast, the no-later TCF $\chi^{f,\leq 2}$ is a tautology, but $\chi^{f=0,\leq 2}$ and $\chi^{f=1,\leq 2}$ are not tautologies due to the logic value constraints of f (that is, to make $f = 0$, both $a \wedge b$ and c must be 0, whereas to make $f = 1$, $a \wedge b$ or c must be 1). For the earlier TCF $\chi^{f,<2}$, it characterizes the set of PI assignments that make the value of f settle earlier than the topological maximum delay. By the timing constraint and the circuit structure, f can only be sensitized by the path from c to f, rather than by the paths from a to m to f and from b to m to f. Since c cannot sensitize f to value 0 before the value of m is determined, $\chi^{f=0,<2}$ is unsatisfiable. On the other hand, $\chi^{f=1,<2} = c$ because $f = 1$ can be sensitized by $c = 1$ at time 1. For the no-earlier TCF $\chi^{f,\geq 2}$, it characterizes the PI assignment $\neg c$ that sensitizes the structural critical paths.

Below we consider various timing analysis problems using TCF formulation. Different TCFs are then compared for their efficacy.

A. Maximum Delay for Signal Steadiness

Maximum delay computation is essential in detecting setup-time violation. To ensure that the delay of a circuit is always smaller than some threshold T, the satisfiability of formula

$$\bigvee_{p \in PO} \chi^{p,\geq T} \tag{1}$$

is checked. The formula is unsatisfiable if and only if the circuit has no timing violation with respect to upper bound T since there exists no PI assignment under which the value of some PO remains unknown at time T. So the maximum delay of a circuit can be computed by gradually reducing the threshold T from some trivial upper bound, e.g., computed by STA based on circuit topology, until Formula (1) becomes satisfiable. Therefore the delay calculation can be applied for checking setup time violation.

In implementation, the TCFs of Formula (1) can be constructed recursively from POs to PIs and further converted into a CNF formula for SAT solving as we detail below.

1) Prior Formulation: For TCF formulation without 0/1-specificity, prior work [18] reformulated the exact [3] and viable [10] sensitization criteria (with path tracing) for circuit maximum delay computation (without path tracing) with the following no-earlier TCFs

$$\begin{aligned} \chi^{f,\geq t} &= \bigvee_{g_i \in FI(f)} \{\chi^{g_i,\geq t-d_i} \wedge [(g_i = v_{c_i}) \wedge \\ &\quad \bigwedge_{g_j \in FI(f)} (\chi^{g_j,\geq t-d_j} \vee (g_j = v_{n_j})) \vee \\ &\quad \bigwedge_{g_j \in FI(f)} (g_j = v_{n_j})]\} \text{ and} \qquad (2) \\ \chi^{f,\geq t} &= \bigvee_{g_i \in FI(f)} \chi^{g_i,\geq t-d_i} \wedge \\ &\quad \bigwedge_{g_i \in FI(f)} (\chi^{g_i,\geq t-d_i} \vee (g_i = v_{n_i})) \qquad (3) \end{aligned}$$

respectively, where d_i is the pin-to-pin delay from g_i to f and v_{c_i} and v_{n_i} are the controlling and non-controlling values of g_i.[1] Equations (2) and (3) were considered in [18] as exact and approximative circuit delay computation, respectively, even though Equation (3) is in fact exact [14].

The recursive definition of $\chi^{f,\geq t}$ naturally translates to a combinational circuit. For a k-input simple gate f, Equations (2) and (3) result in $(k^2 + 13k + 2)$ and $(5k + 3)$ clauses with $(4k + 1)$ and $(k + 1)$ extra variables being introduced, respectively, by Tseitin's circuit-to-CNF conversion [19]. The satisfiability of such a TCF can be difficult to solve especially when the corresponding circuit is large. (Note that the increased number of nodes in the new circuit is bounded from above by the summation of the number of possible arrival times of every node in the original circuit.)

For TCF formulation with 0/1-specificity, prior work [9] intended to improve [18] by exploiting early TCF to simplify TCF circuits. The following equations were proposed.

$$\begin{aligned} \chi^{f,\geq t} &= \chi^{f=1,\geq t} \vee \chi^{f=0,\geq t} \\ &= (f \wedge \neg\chi^{f=1,<t}) \vee (\neg f \wedge \neg\chi^{f=0,<t}), \qquad (4) \end{aligned}$$

which is used in expressing Formula (1). In addition,

$$\begin{aligned} \chi^{f=1,<t} &= \begin{cases} \bigwedge_{g_i \in FI(f)} \chi^{g_i=1,<t-d_{r_i}}, \text{ for AND-gate } f \\ \bigvee_{g_i \in FI(f)} \chi^{g_i=1,<t-d_{r_i}}, \text{ for OR-gate } f \end{cases} \\ \chi^{f=0,<t} &= \begin{cases} \bigvee_{g_i \in FI(f)} \chi^{g_i=0,<t-d_{f_i}}, \text{ for AND-gate } f \\ \bigwedge_{g_i \in FI(f)} \chi^{g_i=0,<t-d_{f_i}}, \text{ for OR-gate } f \end{cases} \qquad (5) \end{aligned}$$

where d_{r_i} and d_{f_i} are the corresponding rising and falling pin-to-pin delays from g_i to f, respectively. In Equation (4), TCF $\chi^{f,\geq t}$ is obtained from two subcases $\chi^{f=1,<t}$ and $\chi^{f=0,<t}$. Note that $\chi^{f=v,\geq t} \neq \neg\chi^{f=v,<t}$, but rather $\chi^{f=v,\geq t} = (f \oplus \neg v) \wedge \neg\chi^{f=v,<t}$.

Observe that, since Equation (5) in circuit representation consists of a single gate, no internal variable needs to be introduced in conversion to CNF. On the other hand, because the equations work for simple gates only, timing analysis of circuits with complex gates cannot be handled directly in [9]. In fact, as proposed back in [13], a general formulation for

[1] Equation (2) looks different from the one in [18] as it was previously expressed by both exact and viable TCFs.

complex gates had been obtained with the following equations

$$\chi^{f=1,<t} = \bigvee_{c\in P_1} \bigwedge_{lit(g_i)\in c} \chi^{g_i=pol(lit(g_i)),<t-d_{r_i}} \text{ and}$$
$$\chi^{f=0,<t} = \bigvee_{c\in P_0} \bigwedge_{lit(g_i)\in c} \chi^{g_i=pol(lit(g_i)),<t-d_{f_i}}, \quad (6)$$

where recall that P_1 and P_0 are the sets of all prime implicants of f and $\neg f$, respectively, and $pol(lit(g_i)) = 0$ if $lit(g_i) = \neg g_i$ and $pol(lit(g_i)) = 1$ if $lit(g_i) = g_i$. When Equation (6) is translated to CNF by Tseitin's transformation, the corresponding TCF circuit has $(2|P_v| + 1 + \sum_{c\in P_v} |c|)$ clauses and $|P_v|$ extra variables for $\chi^{f=v,<t}$, $v \in \{0, 1\}$. (In this paper, the cardinality of a set S is denoted as $|S|$; a cube is treated as a set of literals.) Note that when AND and OR gates are concerned, Equation (6) reduces to Equation (5) with some proper circuit simplification.

Note that, although in the worst case the number of prime implicants of a Boolean function can be exponential in the number of variables, TCFs can still be effectively constructed because a primitive gate in a standard cell library cannot have many inputs (typically less than six).

2) Our Formulation: For TCF formulation without 0/1-specificity, a close examination of Equations (2) and (3) reveals that they are essentially equivalent in circuit delay computation. In fact, as shown back in [14], Equation (3) yields exact (rather than approximative, as interpreted in [18]) analysis when path tracing is not performed.

Building upon Equation (3), we propose a no-earlier TCF formula general for arbitrary complex gates and compact for CNF translation. By using implication, the formula below only requires linear time CNF conversion without building TCF circuits or introducing any extra variable.

Proposition 1: Consider the satisfiability of the no-earlier TCF of some node f in a circuit with respect to some timing requirement t. Let C be the set of controlling cubes of f. The no-earlier TCF of f without 0/1-specificity can be expressed as

$$\chi^{f,\geq t} \rightarrow \bigvee_{g_i\in FI(f)} \chi^{g_i,\geq t-d_i} \wedge \bigwedge_{c\in C} \bigvee_{lit(g_i)\in c} (\chi^{g_i,\geq t-d_i} \vee \neg lit(g_i)), \quad (7)$$

and be directly translated into the CNF formula

$$(\neg\chi^{f,\geq t} \vee \bigvee_{g_i\in FI(f)} \chi^{g_i,\geq t-d_i}) \wedge \bigwedge_{c\in C} (\neg\chi^{f,\geq t} \vee \bigvee_{lit(g_i)\in c} (\chi^{g_i,\geq t-d_i} \vee \neg lit(g_i)) \quad (8)$$

with $|C|+1$ clauses by the Plaisted-Greenbaum encoding [15].

Proof: There are exactly two possible cases for the value of f being determined before time t. First, the value of every $g_i \in FI(f)$ is determined before time $(t - d_i)$. Second, every constituent input g_i of some controlling cube c is determined to its corresponding value $lit(g_i) \in c$ before time $(t - d_i)$. Since any of the above cases makes $\chi^{f,\geq t}$ false, the condition can be formally translated to

$$\neg\chi^{f,\geq t} = \bigwedge_{g_i\in FI(f)} \neg\chi^{g_i,\geq t-d_i} \vee \bigvee_{c\in C} \bigwedge_{lit(g_i)\in c} (\neg\chi^{g_i,\geq t-d_i} \wedge lit(g_i))),$$

whose negation equals

$$\chi^{f,\geq t} = \bigvee_{g_i\in FI(f)} \chi^{g_i,\geq t-d_i} \wedge \bigwedge_{c\in C} \bigvee_{lit(g_i)\in c} (\chi^{g_i,\geq t-d_i} \vee \neg lit(g_i))),$$

which reduces to Equation (3) for simple gates (with controlling values, in other words, with one-literal controlling cubes).

With the key observation that $\chi^{f,\geq t}$ is recursively defined in the above equation with the appearance only in the positive phase without any negation, implication suffices to express the TCF constraints and the above equation reduces to Formula (7). The validity can be shown as follows. Observe that the implication based translation is logically implied by the equation based translation. If the implication based translation yields no satisfying solution, then the equation based translation must yield no satisfying solution, either. Hence we only need to consider the satisfiable case.

We claim that any satisfying solution (in terms of PI assignments) to the recursive implication based translation of $\chi^{f,\geq t}$ must be an satisfying solution to the recursive equation based translation. Notice that, to verify the satisfiability of $\chi^{f,\geq t}$, a clause with a single literal $(\chi^{f,\geq t})$, i.e., a unit clause, will be added to the final CNF formula for SAT solving. Suppose a SAT solver returns a satisfying solution to the entire recursively constructed CNF formula. Consider this satisfying assignment restricted to the recursive translation of some node g' in the transitive fanin cone of f. Essentially there are three cases to analyze based on the valuation of the left-hand side (LHS) and right-hand side (RHS) of the equation $\chi^{g',\geq t'} = \bigvee_{g_i\in FI(g')} \chi^{g_i,\geq t'-d_i} \wedge \bigwedge_{c\in C'} \bigvee_{lit(g_i)\in c} (\chi^{g_i,\geq t'-d_i} \vee \neg lit(g_i)))$, where C' is the set of controlling cubes of g' and d_i is the pin-to-pin delay from g_i to g'. For the first case, when the LHS and RHS are both valuated to true, both equation and implication based translations agree and no discrepancy occurs between implication and equation based translations. For the second case, if the LHS were valuated to true and the RHS valuated to false, this valuation would be forbidden by both translations and thus is impossible. For the third case, when the LHS is valuated to false and RHS valuated to true, this valuation is allowed by the implication based translation, but disallowed by the equation based translation. However, since the RHS definition of $\chi^{g',\geq t'}$ is valuated to true under both translations, this valuation imposes the same logical constraints towards PIs. As all TCF variables in the recursive TCF definition appear in the same positive phase, the above argument holds recursively for any TCF variables assigned to true. As a consequence, any PI assignment satisfying the recursive implication based translation of $\chi^{f,\geq t}$ should also satisfy the recursive equation based translation. The proposition follows. ■

The advantage of using implication, instead of equation, is that we can apply Plaisted-Greenbaum encoding [15], instead of Tseitin encoding, in converting TCFs to CNF formulas. Specifically, Formula (7) can be directly translated into the CNF Formula (8). Hence, unlike prior methods, building TCF circuits is not necessary.

Note that, in converting the entire recursive definition of $\chi^{f,\geq t}$, Tseitin encoding is still needed for parts of the original circuit that are relevant to the literals $lit(g_i)$ in individual TCFs (since these literals may appear in both positive and negative phases). Nevertheless the conversion with Tseitin encoding is applied once on the original circuit and is shared by all individual TCFs.

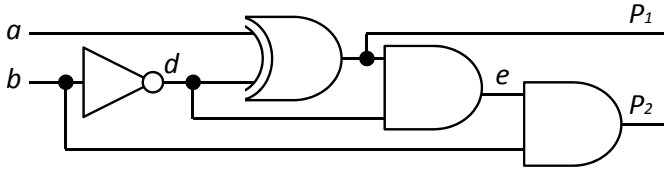

Fig. 2. Circuit under Timing Analysis

An example of TCF construction using Formula (7) is given below for illustration.

Example 2: Consider the circuit of Figure 2 with PIs a and b, and POs P_1 and P_2. Assume every gate has a unit delay and PIs have zero arrival time. For maximum delay computation, initially let $T = 4$ (the maximum topological delay) and check the satisfiability of $(\chi^{P_1,\geq T} \vee \chi^{P_2,\geq T})$ by constructing TCFs using Formula (7). By the recursive construction, we derive the following implications.

$$\chi^{P_1,\geq 4} \rightarrow (\chi^{a,\geq 3} \vee \chi^{d,\geq 3}) = (0 \vee \chi^{d,\geq 3}) = (\chi^{d,\geq 3})$$
$$\chi^{d,\geq 3} \rightarrow (\chi^{b,\geq 2}) = 0$$

That is, for P_1 to have its delay ≥ 4, some of its fanins must have delay ≥ 3. Moreover, since a has arrival time 0, $\chi^{a,\geq 3}$ must be false. Note that, since PIs steady at time 0, the no-earlier TCF of a PI with timing requirement larger than 0 (smaller than or equal to 0) equals constant 0 (constant 1). Hence the above two-literal clause is simplified to a unit clause, which contains only one literal $\chi^{d,\geq 3}$. By recursive TCF construction, for $\chi^{d,\geq 3}$ to be true, it implies $\chi^{b,\geq 2}$ must be true. However $\chi^{b,\geq 2}$ must be false because b has arrival time 0. So we can conclude P_1 cannot have delay ≥ 4. We proceed checking if P_2 can have delay ≥ 4, and the following implications can be derived.

$$\chi^{P_2,\geq 4} \rightarrow (\chi^{b,\geq 3} \vee \chi^{e,\geq 3})(\chi^{b,\geq 3} \vee b)(\chi^{e,\geq 3} \vee e) =$$
$$(0 \vee \chi^{e,\geq 3})(0 \vee b)(\chi^{e,\geq 3} \vee e) = (\chi^{e,\geq 3})(b)$$
$$\chi^{e,\geq 3} \rightarrow (\chi^{d,\geq 2} \vee \chi^{P_1,\geq 2})(\chi^{d,\geq 2} \vee d)(\chi^{P_1,\geq 2} \vee P_1)$$
$$\chi^{d,\geq 2} \rightarrow (\chi^{b,\geq 1}) = 0$$
$$\chi^{P_1,\geq 2} \rightarrow (\chi^{a,\geq 1} \vee \chi^{d,\geq 1}) = (\chi^{d,\geq 1})$$
$$\chi^{d,\geq 1} \rightarrow (\chi^{b,\geq 0}) = 1$$

Thereby the implication of $\chi^{e,\geq 3}$ can be simplified as follows.

$$\chi^{e,\geq 3} \rightarrow (\chi^{d,\geq 2} \vee \chi^{P_1,\geq 2})(\chi^{d,\geq 2} \vee d)(\chi^{P_1,\geq 2} \vee P_1) =$$
$$(0 \vee 1)(0 \vee d)(1 \vee P_1) = (d)$$

So the implication of $\chi^{P_2,\geq 4}$ can be further simplified as follows.

$$\chi^{P_2,\geq 4} \rightarrow (d)(b)$$

Because $\chi^{P_2,\geq 4}$ is unsatisfiable due to the fact that b and d cannot be both true, we conclude that the circuit delay must be smaller than 4. Similar analysis can be conducted with respect to a smaller T until the target TCF formula becomes satisfiable.

For TCF formulation with 0/1-specificity, observe that all earlier TCFs in Equation (4) have negative polarities whereas those in Equation (6) have positive ones. Therefore the Plaisted-Greenbaum encoding cannot be directly applied for CNF conversion. Fortunately, by negating Equation (6) its earlier TCFs are in negative polarities uniformly. Hence the Plaisted-Greenbaum encoding can be applied again as follows.

Proposition 2: Consider the satisfiabilities of $\neg\chi^{f=1,<t}$ and $\neg\chi^{f=0,<t}$ of some node f in a circuit with respect to some timing requirement t. Let P_1 and P_0 be the set of all prime implicants of f and $\neg f$, respectively. The earlier TCF with 0/1-specificity of f can be expressed as

$$\neg\chi^{f=1,<t} \rightarrow \bigwedge_{c\in P_1} \bigvee_{lit(g_i)\in c} \neg\chi^{g_i=pol(lit(g_i)),<t-d_{r_i}} \text{ and}$$
$$\neg\chi^{f=0,<t} \rightarrow \bigwedge_{c\in P_0} \bigvee_{lit(g_i)\in c} \neg\chi^{g_i=pol(lit(g_i)),<t-d_{f_i}}, \quad (9)$$

and be directly translated into the CNF formulas

$$\bigwedge_{c\in P_1} (\chi^{f=1,<t} \vee \bigvee_{lit(g_i)\in c} \neg\chi^{g_i=pol(lit(g_i)),<t-d_{r_i}}) \text{ and}$$
$$\bigwedge_{c\in P_0} (\chi^{f=0,<t} \vee \bigvee_{lit(g_i)\in c} \neg\chi^{g_i=pol(lit(g_i)),<t-d_{f_i}}), \quad (10)$$

respectively, with $|P_1|$ and $|P_0|$ clauses each.

Proof: Formula (9) is simply the implication version of the negated form of Equation (6). The reason why implication suffices can be shown by an argument similar to the proof of Proposition 1. ■

Although the implication-based Formula (9) simplifies CNF translation, Equation (4) requires conversion from earlier TCFs to a non-earlier TCF. We exploit an alternative formulation that requires no TCF type change as follows.

Proposition 3: Consider the satisfiabilities of $\chi^{f=1,\geq t}$ and $\chi^{f=0,\geq t}$ of some node f in a circuit with respect to some timing requirement t. Let P_1 and P_0 be the set of all prime implicants of f and $\neg f$, respectively. The no-earlier TCF of

f with 0/1-specificity can be expressed as

$$\begin{aligned}\chi^{f=1,\geq t} &\rightarrow f \wedge \bigwedge_{c\in P_1}\bigvee_{lit(g_i)\in c}(\chi^{g_i=pol(lit(g_i)),\geq t-d_{r_i}} \vee \neg lit(g_i))\\ \chi^{f=0,\geq t} &\rightarrow \neg f \wedge \bigwedge_{c\in P_0}\bigvee_{lit(g_i)\in c}(\chi^{g_i=pol(lit(g_i)),\geq t-d_{f_i}} \vee \neg lit(g_i)),\end{aligned} \tag{11}$$

and be directly translated into the CNF formulas

$$\begin{aligned}&(\neg\chi^{f=1,\geq t} \vee f)\wedge\\ &\bigwedge_{c\in P_1}\bigvee_{lit(g_i)\in c}(\neg\chi^{f=1,\geq t} \vee \chi^{g_i=pol(lit(g_i)),\geq t-d_{r_i}} \vee \neg lit(g_i))\\ &(\neg\chi^{f=0,\geq t} \vee \neg f)\wedge\\ &\bigwedge_{c\in P_0}\bigvee_{lit(g_i)\in c}(\neg\chi^{f=0,\geq t} \vee \chi^{g_i=pol(lit(g_i)),\geq t-d_{f_i}} \vee \neg lit(g_i)).\end{aligned} \tag{12}$$

Proof: The validity of Formula (11) can be established by Equation (4) and a similar reasoning of Proposition 1. ■ Since $\chi^{f,\geq t}$ can be replaced by $\chi^{f=1,\geq t}\vee\chi^{f=0,\geq t}$, Formula (1) can be expressed recursively using Formula (11). The Plaisted-Greenbaum encoding can again be applied here.

B. Minimum Delay for Signal Steadiness

By duality, under the floating mode operation the minimum delay for signals starting to get ready can be formulated. The formulation may be useful in applications such as variable latency design [2], where the fraction of PI assignments that obey some required timing constraint is of concern. The minimum and maximum delays of signal steadiness provide a range in searching an optimal operating frequency in variable latency design. (Note however that such minimum delay cannot be used to determine the hold-time constraint since signal transition may happen before the computed minimum delay for signal steadiness.)

The minimum delay of signal steadiness can be obtained by computing the maximum T that makes the formula

$$\bigwedge_{p\in PO}\chi^{p,\geq T} \tag{13}$$

remain a tautology. (The formula is a tautology if and only if the circuit has no PI assignment under which the value of some PO remains unknown at time T.) Equivalently, it corresponds to computing the maximum T that makes the formula

$$\bigvee_{p\in PO}\chi^{p,<T} = \bigvee_{p\in PO}\neg\chi^{p,\geq T} \tag{14}$$

remain unsatisfiable. By the recursive TCF construction of Formula (8), we know that Formulas (13) and (14) are in CNF and DNF, respectively. Because tautology checking on CNF and satsifiability checking on DNF are computationally tractable, determining the minimum delay of a circuit's signal steadiness can be done in polynomial time.

Example 3: Consider the circuit of Figure 2 under the unit delay model. Assume all PIs have zero arrival time. Since the minimum topological delay is 1, $(\chi^{P_1,\geq T}\wedge\chi^{P_2,\geq T})$ is surely a tautology for $T=1$. For $T=2$,

$$\begin{aligned}&\chi^{P_1,\geq 2}\rightarrow(\chi^{a,\geq 1}\vee\chi^{d,\geq 1}) = (\chi^{d,\geq 1})\\ &\quad\chi^{d,\geq 1}\rightarrow(\chi^{b,\geq 0}) = 1\\ &\chi^{P_2,\geq 2}\rightarrow(\chi^{b,\geq 1}\vee\chi^{e,\geq 1})(\chi^{b,\geq 1}\vee b)(\chi^{e,\geq 1}\vee e) =\\ &\qquad(\chi^{e,\geq 1})(b)\\ &\quad\chi^{e,\geq 1}\rightarrow(\chi^{d,\geq 0}\vee\chi^{P_1,\geq 0})(\chi^{d,\geq 0}\vee d)(\chi^{P_1,\geq 0}\vee P_1) = 1\end{aligned}$$

From constructing $\chi^{P_2,\geq 2}$, we know under $b=0$ the formula cannot be a tautology. Since $\chi^{P_2,\geq 2}$ is not a tautology, the minimum topological delay for signal steadiness is 1 instead of 2.

In fact, some of the recursive calls in Examples 2 and 3 are redundant due to the equivalence of TCFs, and can be reduced by the techniques to be introduced in Section IV.

C. Cyclic Circuit Delay Computation

The TCF formulation can be naturally extended for timing analysis of combinational cyclic circuits [17].

Proposition 4: A cyclic circuit C is combinational if and only if

$$\bigvee_{p\in PO}\chi^{p,>maxD} \tag{15}$$

is unsatisfiable, where $maxD$ is the delay of the longest path (without gate recurrence) of C and TCF $\chi^{p,>maxD}$ can be expressed by Formula (7) with "$\geq$" replaced by "$>$."

Proof: ($\Longrightarrow$) If a cyclic circuit C is combinational, then Formula (15) must be unsatisfiable. Otherwise there must be some sensitizable path containing a loop making the delay value larger than $maxD$. That is, there must be some PI assignment that makes a PO sensitizable to the loop nodes, whose values are not readily determined by the PI assignment. However, in floating mode sensitization, all signals are assumed to be of the unknown initial value. If the value of a node on the loop cannot not be purely determined by other side fanin signals (not on the loop), the output of the node will remain unknown since the valuation of every loop node awaits the valuations of other loop nodes. Therefore, the cyclic circuit is not combinational because some PO cannot always steady to a final value within any finite time bound.

($\Longleftarrow$) If Formula (15) is unsatisfiable, then all POs steady to their deterministic final values within $maxD$ under every PI assignment. So any sensitized path in the circuit never form a cycle, and the circuit behaves combinationally. ■

Moreover, the TCF formulation of timing analysis for combinational cyclic circuits is the same as that for acyclic circuits, except that the computation of arrival times is different as to be detailed in Section IV-B.

D. Comparison on TCF Formulas

Table I compares our formulations with those of [18], [9], and [13] for maximum delay computation. For a k-input AND or OR gate, the number of extra variables and the number of

TABLE I
TCF COMPARISON

	TCF			PO	Generality	
	Eq	#Var	#Clause	Eq	CG	0/1
[18]	(2)	$k+1$	$5k+3$	(1)	No	No
	(3)	$4k+1$	$k^2+13k+2$	(1)	No	No
[9]	(5)	0	$2k+2$	(4)	No	Yes
[13]	(6)	$k+1$	$4k+4$	(4)	Yes	Yes
Ours	(7)	0	$k+1$	(1)	Yes	No
	(9)	0	$k+1$	(4)	Yes	Yes
	(11)	0	$k+3$	(1)	Yes	Yes

clauses corresponding to the TCF formulas in Column 2 are shown in Columns 3 and 4, respectively. The corresponding PO equations are showed in Column 5. Among the PO equations, only Equation (4) needs extra $2|\text{PO}|$ variables and $9|\text{PO}|$ clauses to convert the TCF type. The generality for each formulation in supporting complex gate types (denoted CG) and supporting rise/fall delays (denoted 0/1) are summarized in Columns 6 and 7, respectively.

TABLE II
TCF EQUIVALENCE CLASSES

	$[0, a_1]$	$(a_{i-1}, a_i]$	(a_m, ∞)
$\chi^{f,\geq t}$	1	$\chi^{f,\geq a_i}$	0
$\chi^{f,<t}$	0	$\chi^{f,<a_i}$	1
	$[0, a_1^1]$	$(a_{i-1}^1, a_i^1]$	(a_p^1, ∞)
$\chi^{f=1,\geq t}$	f	$\chi^{f=1,\geq a_i^1}$	0
$\chi^{f=1,<t}$	0	$\chi^{f=1,<a_i^1}$	f
	$[0, a_1^0]$	$(a_{i-1}^0, a_i^0]$	(a_n^0, ∞)
$\chi^{f=0,\geq t}$	$\neg f$	$\chi^{f=0,\geq a_i^0}$	0
$\chi^{f=0,<t}$	0	$\chi^{f=0,<a_i^0}$	$\neg f$
	$[0, a_1)$	$[a_{i-1}, a_i)$	$[a_m, \infty)$
$\chi^{f,>t}$	1	$\chi^{f,>a_{i-1}}$	0
$\chi^{f,\leq t}$	0	$\chi^{f,\leq a_{i-1}}$	1
	$[0, a_1^1)$	$[a_{i-1}^1, a_i^1)$	$[a_p^1, \infty)$
$\chi^{f=1,>t}$	f	$\chi^{f=1,>a_{i-1}^1}$	0
$\chi^{f=1,\leq t}$	0	$\chi^{f=1,\leq a_{i-1}^1}$	f
	$[0, a_1^0)$	$[a_{i-1}^0, a_i^0)$	$[a_n^0, \infty)$
$\chi^{f=0,>t}$	$\neg f$	$\chi^{f=0,>a_{i-1}^0}$	0
$\chi^{f=0,\leq t}$	0	$\chi^{f=0,\leq a_{i-1}^0}$	$\neg f$

IV. REDUCTION TECHNIQUES

A. Equivalence Reduction

Given a circuit with a node f, its TCFs $\chi^{f,\Box t}$ for all time t can be partitioned into equivalence classes. This equivalence relation can be exploited to simplify the recursive TCF construction.

In [13], [9], TCF equivalence based on arrival-time information is introduced. Assume that the possible arrival times $A(f)$ of node f are sorted in an ascending order as $\{a_1, a_2, \ldots, a_m\}$ with $a_{i-1} < a_i$. It was shown that $\chi^{f=v,<t} = \chi^{f=v,<a_i}$ if $a_{i-1} < t \leq a_i$. That is, two temporal conditions t_1 and t_2 of f are equivalent if they have the same next larger-or-equal arrival time in $A(f)$.

This paper further extends equivalence reduction for 12 types of TCFs as shown in Table II. Assume the TCFs of f, $f = 1$, and $f = 0$ have sorted arrival times $\{a_1, \ldots, a_m\}$, $\{a_1^1, \ldots, a_p^1\}$, and $\{a_1^0, \ldots, a_n^0\}$, respectively, (in ascending orders). For each type of TCF in the table, three equivalence time intervals are shown, including the first, i^{th}, and last intervals. For example, TCFs $\chi^{f,\geq t_1}$ and $\chi^{f,\geq t_2}$ belong to the same equivalence class as $\chi^{f,\geq a_i}$ if $a_{i-1} < t_1, t_2 \leq a_i$, i.e., both t_1 and t_2 are in the interval $(a_{i-1}, a_i]$. In this case, they can be expressed by the same representative TCF $\chi^{f,\geq a_i}$.

A table look-up approach is adopted for practical implementation. By consulting Table II, a representative TCF of the equivalence class corresponding to a given t can be retrieved to simplify the recursive construction. Our equivalence reduction method has several improvements over prior approaches [13], [9] in the following ways.

1) Distinguished Arrival Times: For TCFs with 0/1-specificity, the set of arrival times of a node f is further distinguished into two sets $A^1(f) = \{a_1^1, a_2^1, ..., a_p^1\}$ and $A^0(f) = \{a_1^0, a_2^0, ..., a_n^0\}$ for those resulting in $f = 1$ and $f = 0$, respectively. This distinction reduces the number of arrival times and thus TCF equivalence classes.

2) Simplified Boundary Conditions: Under boundary conditions, the TCF of a node f is substituted with a constant 0, constant 1, literal f, or literal $\neg f$ for further reduction as shown in Table II. For example, if t is larger than the maximum arrival time a_m of a node f, then $\chi^{f,\geq t}$ is unsatisfiable since f always stabilizes before t. In this case, $\chi^{f,\geq t}$, $\chi^{f=1,\geq t}$ and $\chi^{f=0,\geq t}$ all equal Boolean constant 0. On the contrary, if t is not larger than the minimum arrival time a_1, then $\chi^{f,\geq t}$ is a tautology (but $\chi^{f=1,\geq t}$ and $\chi^{f=0,\geq t}$ are not necessarily tautologies). That is, $\chi^{f,\geq t}$ equals constant 1, and furthermore $\chi^{f=v,\geq t}$ can be simplified to $f \oplus \neg v$ by $\chi^{f=v,\geq t} = (f \oplus \neg v) \wedge \chi^{f,\geq t}$ for $v \in \{0, 1\}$.

Observe that, in Formula (11), let $v = pol(lit(g_i))$, $\chi^{g_i=v,\geq t-d_i}$ and $\neg lit(g_i)$ are always present together in a clause with $\neg lit(g_i) = g_i \oplus v$. When $\chi^{g_i,\geq t-d_i} = 1$, since $\chi^{g_i=v,\geq t-d_i} = g_i \oplus \neg v$, this clause must be satisfied due to $(\chi^{g_i=v,\geq t-d_i} \vee \neg lit(g_i)) = ((g_i \oplus \neg v) \vee (g_i \oplus v)) = 1$. Therefore, whenever $\chi^{g_i,\geq t-d_i} = 1$, substituting constant 1 for $\chi^{g_i=1,\geq t-d_i}$ and $\chi^{g_i=0,\geq t-d_i}$ is safe without altering the satisfiability of $\chi^{f,\geq t}$. (That is, the literals f and $\neg f$ in the second column of Table II can be replaced by 1 in constructing Formula (11).) As a result, our no-earlier TCFs without and with 0/1-specificity can be simplified with such constant substitution.

3) Reduced PI and PO TCFs: Our TCF equivalence reduction is applied to all nodes including PIs and POs. According to Table II, any TCF of a PI f is either a constant 0, constant 1, literal f, or literal $\neg f$ because any PI has only one arrival time. On the other hand, since the arrival times at POs are the only candidate circuit delays, this information is exploited to save unnecessary checking. More precisely, only PO arrival times are checked for circuit delay by Formula (1); once some candidate delay is falsified, this delay and other larger (or smaller) delays are removed from the arrival-time lists of all POs. For example, assume two POs p_1 and p_2 have arrival-time lists $\{4, 5, 7\}$ and $\{6, 7\}$, respectively. If $(\chi^{p_1,\geq 7} \vee \chi^{p_2,\geq 7})$ is

unsatisfiable, we remove 7 from the two lists. Then we check $(\chi^{p_1,\geq 6} \vee \chi^{p_2,\geq 6}) = (0 \vee \chi^{p_2,\geq 6})$. Note that this removal is crucial. If 7 were not removed from the list of p_1, then $\chi^{p_1,\geq 6}$ would equal $\chi^{p_1,\geq 7}$ instead of 0 and $\chi^{p_1,\geq 7}$ would be built again.

B. Arrival Time List Computation

For implementation allowing effective table look-up, an equivalence table is built based on Table II for a given circuit under timing analysis. To make table look-up more efficient, spurious candidate arrival times can be reduced as follows.

1) Lower Bound Tightening: Arrival times are commonly computed based on circuit topology from PIs to POs. In particular, the arrival time lists $A(f)$ for a 0/1 unspecified TCF, $A^1(f)$ for a 1-specified TCF, and $A^0(f)$ for a 0-specified TCF of node f are computed from fanin arrival times by

$$A(f) = \bigcup_{g_i \in FI(f)} \bigcup_{a_j \in A(g_i)} a_j + d_i, \tag{16}$$

$$A^1(f) = \bigcup_{c \in P_1} \bigcup_{lit(g_i) \in c} \bigcup_{a_j \in A^{pol(lit(g_i))}(g_i)} a_j + d_{r_i}, \text{ and}$$

$$A^0(f) = \bigcup_{c \in P_0} \bigcup_{lit(g_i) \in c} \bigcup_{a_j \in A^{pol(lit(g_i))}(g_i)} a_j + d_{f_i}, \tag{17}$$

respectively.

The above equations, however, may include some redundant arrival times. For example, the value of an XOR gate must be determined by its latest arriving input signal. If a fanin arrival time is smaller than the minimum arrival time of some other fanin, it cannot be the latest arriving input. Therefore, this fanin arrival time should not be propagated to the gate output. To remove such spurious arrival times, we propose the following computation.

Proposition 5: The arrival time lists of node f can be computed by

$$A(f) = \bigcup_{c \in P_1 \cup P_0} \bigcup_{lit(g_i) \in c} \bigcup_{(a_j \in A(g_i)) \wedge (a_j \geq M_c)} a_j + d_i \tag{18}$$

$$A^1(f) = \bigcup_{c \in P_1} \bigcup_{lit(g_i) \in c} \bigcup_{(a_j \in A^{pol(lit(g_i))}(g_i)) \wedge (a_j \geq M_c)} a_j + d_{r_i}$$

$$A^0(f) = \bigcup_{c \in P_0} \bigcup_{lit(g_i) \in c} \bigcup_{(a_j \in A^{pol(lit(g_i))}(g_i)) \wedge (a_j \geq M_c)} a_j + d_{f_i} \tag{19}$$

where $M_c = \max_{lit(g_i) \in c}\{\min A(g_i)\}$ for Equation (18) and $M_c = \max_{lit(g_i) \in c}\{\min A^{pol(lit(g_i))}(g_i)\}$ for Equation (19).

Example 4: Continuing Example 2, assume that the inverter is of $d_r = 1$ and $d_f = 2$, and all other gates are of $d_r = 5$ and $d_f = 7$. The arrival time lists A^1 and A^0 computed by Equation (19) are as follows.

$A^1(a) =$	$\{0\}$	$A^0(a) =$	$\{0\}$
$A^1(b) =$	$\{0\}$	$A^0(b) =$	$\{0\}$
$A^1(d) =$	$\{1\}$	$A^0(d) =$	$\{2\}$
$A^1(P_1) =$	$\{6,7\}$	$A^0(P_1) =$	$\{8,9\}$
$A^1(e) =$	$\{11,12\}$	$A^0(e) =$	$\{9,15,16\}$
$A^1(P_2) =$	$\{16,17\}$	$A^0(P_2) =$	$\{7,16,22,23\}$

In comparison, the arrival time lists of P_1 derived by Equation (17) are $A^1(P_1) = \{5,6,7\}$ and $A^0(P_1) = \{7,8,9\}$. As noted earlier, since the XOR gate P_1 must steady only after its two fanins are steady, the arrival times of a do not propagate and contribute to the arrival times of P_1 because they are smaller than those of d. On the other hand, $A^1(e) = \{6,11,12\}$ by Equation (17), but the AND gate e must steady to value 1 only after its two fanins are steady to value 1. So $A^1(d)$ does not propagate and contribute to $A^1(e)$.

For computing the arrival time lists of a cyclic circuit, the nodes not on loops are first visited without considering the delays from nodes on loops. (Effectively, we may treat a side fanin signal to a node on a loop and a side fanout signal from a node on a loop as a pseudo PO and PI, respectively, with empty arrival time lists.) After calculating the possible arrival times for nodes not on loops, we then iterate the arrival time list computation throughout the entire circuit until the iteration number reaches the length of the longest path without node recurrence.

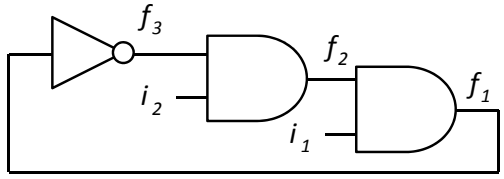

Fig. 3. Circuit under Timing Analysis

Example 5: Consider the cyclic circuit of Figure 3. Assume the arrival time lists for inputs are $A^0(i_1) = \{1,2\}$, $A^1(i_1) = \{3,4\}$, $A^0(i_2) = \{0,1\}$, and $A^1(i_2) = \{1,5\}$, and all gates are of $d_r = d_f = 1$. Based on Equation (19), the related arrival times are iteratively computed as follows.

Iteration 1:

$A^0(f_1) = \{2,3\}$	$A^1(f_1) = \{4,5\}$
$A^0(f_2) = \{1,2\}$	$A^1(f_2) = \{2,6\}$

Iteration 2:

$A^0(f_1) = \{2,3\}$	$A^1(f_1) = \{4,5,7\}$
$A^0(f_2) = \{1,2\}$	$A^1(f_2) = \{2,6\}$
$A^0(f_3) = \{5,6\}$	$A^1(f_3) = \{3,4\}$

Iteration 3:

$A^0(f_1) = \{2,3\}$	$A^1(f_1) = \{4,5,7\}$
$A^0(f_2) = \{1,2,6,7\}$	$A^1(f_2) = \{4,5,6\}$
$A^0(f_3) = \{5,6,8\}$	$A^1(f_3) = \{3,4\}$

Since for a cyclic circuit to be combinational, any loop should not be sensitized. For the circuit of Figure 3, three iterations suffice to derive candidate arrival times for checking whether or not the loop can be sensitized.

We proceed to verify whether or not the cyclic circuit is combinational by justifying the satisfiability of $\chi^{f_1,>7}$. (Note that the 0/1-specified TCF is inappropriate for checking the combinationality of a cyclic circuit because any logical inconsistency due to the cyclic structures can make 0/1-specified

TCF unsatisfiable even under the presence of sensitizable paths.) We obtain the following implications.

$$\begin{aligned}\chi^{f_1,>7} \rightarrow (\chi^{f_2,>6} \vee \chi^{i_1,>6})(\chi^{f_2,>6} \vee f_2)(\chi^{i_1,>6} \vee i_1) = \\ (\chi^{f_2,>6} \vee 0)(\chi^{f_2,>6} \vee f_2)(0 \vee i_1) = (\chi^{f_2,>6})(i_1) \\ \chi^{f_2,>6} \rightarrow (\chi^{f_3,>5} \vee \chi^{i_2,>5})(\chi^{f_3,>5} \vee f_3)(\chi^{i_2,>5} \vee i_2) \\ = (\chi^{f_3,>5} \vee 0)(\chi^{f_3,>5} \vee f_3)(0 \vee i_2) = \\ (\chi^{f_3,>5})(i_2) \\ \chi^{f_3,>5} \rightarrow (\chi^{f_1,>4})\end{aligned}$$

We further derive the following implications.

$$\begin{aligned}\chi^{f_1,>4} \rightarrow (\chi^{f_2,>3} \vee \chi^{i_1,>3})(\chi^{f_2,>3} \vee f_2)(\chi^{i_1,>3} \vee i_1) = \\ (\chi^{f_2,>3} \vee 0)(\chi^{f_2,>3} \vee f_2)(0 \vee i_1) = (\chi^{f_2,>3})(i_1) \\ \chi^{f_2,>3} \rightarrow (\chi^{f_3,>2} \vee \chi^{i_2,>2})(\chi^{f_3,>2} \vee f_3)(\chi^{i_2,>2} \vee i_2) \\ = (1 \vee 0)(1 \vee f_3)(0 \vee i_2) = (i_2)\end{aligned}$$

(In the above derivation, we assume i_1 and i_2 are PIs. Recall that, for a PI i, its $\chi^{i,>t} = 1$ for $t < a_1$ and $\chi^{i,>t} = 0$ for $t \geq a_1$, where a_1 is the smallest arrival time of i.) Because $\chi^{f_1,>7}$ is satisfiable under $i_1 = i_2 = 1$, we conclude that the circuit is not combinational.

2) Subcircuit Timing Analysis: Timing analysis can be performed at an early stage by treating a (preferably timing-critical) intermediate node as a PO and computing its true delays. The spurious critical arrival times can then be detected and removed. This removal action may potentially propagate and make impossible some arrival times of nodes at the transitive fanout cone of this node. The reduction on the number of possible arrival times makes TCFs easier to solve.

C. CNF Variable Indexing

To convert TCF timing constraints into CNF formulas, variable index (VID) assignment is an important factor determining SAT solving speed since it affects the number of variables appearing in a CNF formula. To achieve effective VID assignment, the TCF variables (i.e., the variables representing TCFs) and the variables of the original circuit are assigned separately. For the former, after looking up the TCF equivalence table, some TCFs of a node may be equal if they are mapped to the same timing requirement. In addition, for a single-fanin node, such as the inverter and buffer, its fanout TCF is equal to its fanin TCF, which has a smaller timing requirement. In our algorithm, equivalent TCFs share one VID and are built only once. For the latter, Tseitin encoding is applied only for parts of the original circuit that are relevant to the literals appearing in TCF construction. Only a multi-fanin node in these parts will be given a distinct VID, while a single-fanin node reuses the same VID as its fanin node (for instance, the output of an inverter shares the same VID as its fanin, and has a negated literal). The conversion of circuit nodes is applied once and shared by all individual TCFs.

Example 6: Consider the circuit of Figure 2 under the unit delay model. By Equation (18), the arrival time list A of every gate is derived below.

$$\begin{aligned}A(a) &= \{0\} \\ A(b) &= \{0\} \\ A(d) &= \{1\} \\ A(P_1) &= \{2\} \\ A(e) &= \{2, 3\} \\ A(P_2) &= \{1, 3, 4\}\end{aligned}$$

To check if 4 is indeed the maximum circuit delay, we test whether $(\chi^{P_1,\geq 4} \vee \chi^{P_2,\geq 4})$ is satisfiable. By the TCF equivalence table, formula $(\chi^{P_1,\geq 4} \vee \chi^{P_2,\geq 4})$ simplifies to $(0 \vee \chi^{P_2,\geq 4})$. By Formula (7) and the shared VID assignment, the following TCFs are constructed.

$$\begin{aligned}\chi^{P_2,\geq 4} \rightarrow (\chi^{b,\geq 3} \vee \chi^{e,\geq 3})(\chi^{b,\geq 3} \vee b)(\chi^{e,\geq 3} \vee e) = \\ (\chi^{e,\geq 3})(b) \\ \chi^{e,\geq 3} \rightarrow (\chi^{d,\geq 2} \vee \chi^{P_1,\geq 2})(\chi^{d,\geq 2} \vee \neg b)(\chi^{P_1,\geq 2} \vee P_1) = \\ (0 \vee 1)(0 \vee \neg b)(1 \vee P_1) = (\neg b)\end{aligned}$$

In the above derivation, by equivalence table lookup we know $\chi^{P_1,\geq 2} = 1$. Also (d) is replaced by $(\neg b)$ due to VID sharing. The recursive construction is simpler than the previous one in Example 2, and the resultant formula becomes earlier to solve. Since the above formula is unsatisfiable, circuit delay 4 is unfeasible and will be removed from arrival time list of P_2.

V. Algorithm

A. Delay Computation

Figure 4 sketches a procedure for maximum delay computation without rise/fall time separation. It can be easily extended under a similar framework to the computation with rise/fall time separation, which is omitted for brevity. To avoid confusion between a TCF and its output variable, in the pseudo code $x^{f,t}$ represents the output variable of TCF $\chi^{f,\geq t}$.

While the code is self-explanatory, it should be noted that different delay search strategies can be applied depending on how functions *GetDelayList*, *GetNextDelay*, and *UpdateDelayList* are implemented. For instance, linear or binary search can be deployed with or without adaptive step-size adjustment. Counterintuitively empirical experience suggests that linear search in general works much better than binary search. Investigation reveals that, although linear search requires more SAT solving iterations than binary search, it allows the second improvement technique of Section IV-A more applicable and thus making the CNF formula at each iteration easier to solve.

Upon termination (line 14 of *ComputeDelay*), Formula (1) must be satisfiable for $D = \text{lowerDelay}$. That is, there exists a PI assignment to sensitize some true path achieving this delay value. By applying the assignment values to PIs, we can simulate and trace one true critical path based on the exact sensitization criterion [3].

ComputeDelay(C ***)*** `//compute maximum true-path delay of circuit` C
begin
01 L := *GetDelayList*(C);
02 (lowerDelay, upperDelay) := *MinMaxTopologicalDelay*(C);
03 **do**
04 D := *GetNextDelay*(L);
05 $\Phi := (\bigvee_{p \in PO} x^{p,D})$;
06 **for** every PO p
07 $\Phi := \Phi \wedge$ *BuildTcf*(p, D);
08 **if** *IsSat*(Φ)
09 lowerDelay := D;
10 **else**
11 upperDelay := D;
12 *UpdateDelayList*(C, L, lowerDelay, upperDelay);
13 **while** L non-empty;
14 **return** lowerDelay and its corresponding true path;
end

BuildTcf(*f,t****)*** `//derive` $\chi^{f,t}$ `in CNF`
begin
01 t := *GetNextLargerOrEqualArrivalTime*(f);
02 **if** $\chi^{f,t}$ has been built
03 **return** 1;
04 **if** $t > f.a_m$ `//largest arrival time of` f
05 **return** $(\neg x^{f,t})$;
06 **if** $t \leq f.a_1$ `//smallest arrival time of` f
07 **return** $(x^{f,t})$;
08 **if** f has only one fanin g_i
09 **return** *BuildTcf*(g_i, $t - d_i$) with $x^{g_i, t-d_i}$ replaced by $x^{f,t}$;
10 $\Phi := (\neg x^{f,t} \vee \bigvee_{g_i \in FI(f)} x^{g_i, t-d_i})$;
11 **for** each controlling cube c of f
12 $\Phi := \Phi \wedge (\neg x^{f,t} \vee \bigvee_{lit(g_i) \in c} (x^{g_i, t-d_i} \vee \neg lit(g_i)))$;
13 **for** each $g_i \in FI(f)$
14 $\Phi := \Phi \wedge$ *BuildTcf*(g_i, $t - d_i$);
15 **if** g_i's circuit CNF has not been built
16 $\Phi := \Phi \wedge$ *BuildCktCnf*(g_i);
17 **return** Φ;
end

Fig. 4. Algorithm: Delay Computation

B. Critical Region Identification

Our delay computation algorithm can be applied to identify true timing critical regions for delay optimization. Given a target required time of a circuit, topological timing critical regions (with small slacks) can be identified by conventional static timing analysis. Topological timing critical regions can be treated as an over-approximation of functional true critical regions. The approximation however can be very crude, and potentially many false critical gates and paths can be trimmed away. The true critical regions can be pinpointed by removing false arrival times with the improvement technique of Section IV-B2. In other words, a gate is not truly timing critical if and only if all of its possible arrival times greater than or equal to the required time are spurious. Note that the TCF of a non-critical gate equals constant 0 due to the boundary condition (the required time is greater than the largest possible arrival time) of TCF equivalence reduction. Effectively the computation considers only the timing critical sub-circuit, which can be much smaller than the entire circuit.

VI. Experimental Results

Our functional timing analysis tool SWIFT was implemented in the C++ language using MiniSat version 2.20 [7] as the underlying SAT solver. All experiments were conducted on a Linux machine with a Xeon 3.4 GHz CPU and 32 GB RAM. Large ISCAS, ITC, and other industrial benchmark circuits, whose profiles are shown in Table III, were selected for experiments. For the sake of comparison with prior work [9], which handles only simple gate types, all circuits are technology mapped using only buffers, inverters, AND-gates, OR-gates, NAND-gates, and NOR-gates. It should be noted, however, that our computation is not restricted to these simple gate types and is applicable to general complex gates.

TABLE III
PROFILES OF BENCHMARK CIRCUITS

Circuit	#Gate	#PI	#PO
b05	1022	35	60
b18	117941	3357	3343
b19	237959	6666	6669
c6288	2480	32	32
leon2	1119384	149507	149577
leon3	1272597	185134	185196
leon3mp	824294	109016	109089
netcard	983683	97899	97888
ray	235526	17443	23648
s35932	19876	1763	2048
uoft_raytracer	218671	17443	17112

A. Circuit Delay Computation

The results of circuit delay computation are reported in Tables IV and V. Four delay computation methods are compared under four delay models. The four studied delay models, in order, include the unit gate delay model, fanout delay model (by calculating a gate delay as $1 + 0.2 \times$ fanout number), TSMC $0.18\mu m$ library model with combined rise/fall times (by calculating a gate delay as $\max\{$rise delay, fall delay$\}$), and TSMC $0.18\mu m$ library model with separate rise/fall times. Under the first three timing models, the four compared methods include our re-implementation of prior work [9] (referred to in the tables as [9]), our re-implementation of [9] with the new reduction techniques of Section IV (referred to as [9]*), our proposed method with Formula (10) for TCF construction (referred to as SWIFT-01), and our method with Formula (8) for TCF construction (referred to as SWIFT-c). For the last timing model, the four methods compared include [9], [9]*, SWIFT-01, and our method with Formula (12) for TCF construction (referred to as SWIFT-01*).[2] Note that [9] and [9]* were re-implemented under the same delay search strategy as ours for fair comparison. Also since prior approach [18] was shown not as efficient as [9], we did not compare with it.

In Table IV, Column 1 lists the circuits, Column 2 lists the delays (including the longest topological delay and the computed longest true-path delay), Column 3 lists the numbers of satisfiability testings on attempted target delays, Columns 4, 6, 8, and 10 list the total numbers of TCF variables introduced for all SAT solving iterations, and Columns 5, 7, 9, and 11 list the total numbers of TCF clauses constructed for all SAT solving iterations.

[2] SWIFT-c is only applicable to the first three timing models (without separating rise and fall delays), and instead SWIFT-01* is applied in the fourth timing model for comparison.

Table IV reveals that topological delay may be far pessimistic compared to true circuit delay, e.g., circuits b05 and b19 under the unit and fanout delay models. It suggests the importance of accurate functional timing analysis and its application on identifying true critical region for timing optimization. Also, comparing [9] and [9]* in Table IV, we see that the reduction techniques of Section IV achieved (on average for each timing model) about 1% to 5% reduction on the numbers of variables and clauses. Comparing [9] and SWIFT-01, the formula sizes are reduced by about 3% to 10% in terms of variables, and about 49% to 53% in terms of clauses. Comparing [9] and SWIFT-c for the first three timing models, the formula sizes are reduced by about 52% to 58% in terms of variables, and about 53% to 58% in terms of clauses. On the other hand, for the fourth timing model, comparing [9] and SWIFT-01*, the formula sizes are reduced by about 14% in terms of variables, and about 29% in terms of clauses. The benefit of formula size reduction is to be further justified in term of SAT solving time in Table V.

Table V, under the same experiment as Table IV, compares mainly the CPU times in SAT solving. Column 1 lists the circuits, Column 2 lists the CPU times for STA, Columns 3, 5, 7, and 9 list the CPU times for SAT solving using the MiniSat core solver, denoted "CoreSlvr," and Columns 4, 6, 8, and 10 list the CPU times for SAT solving using the MiniSat extended solver, denoted "SimpSlvr," with simplification capabilities.[3]

Several observations can be made from Table V. Firstly, the runtimes of STA are negligible. Secondly, SSolver is effective in reducing the runtimes for the testcases that took CSolver more than hundreds of seconds to solve. It, however, incurred noticeable overheads especially for the formulas constructed by SWIFT-c, which were very effectively solved by CSolver. Thirdly, when [9] and [9]* are compared, the benefit of applying the reduction techniques of Section IV on [9] is not clear. [9]* sometimes improves [9] and sometimes not. Fourthly, SWIFT-c, SWIFT-01, and SWIFT-01* are clearly superior to [9] and [9]*. Fifthly, whenever SWIFT-c is applicable (for the first three timing models), it certainly outperforms all other methods. Finally, when SWIFT-01 and SWIFT-01* are compared (in the fourth timing model), they yielded similar performance gains over [9] and [9]*.

Notice that the reported CPU times of STA in Table V are the runtime for basic topological circuit traversal. For STA in an industrial setting, several extensions are necessary. Among them, one of the major complications arises due to the handling of user-specified path exceptions [8]. It incurs sophisticated arrival time information to propagate, and imposes substantial computation overhead. In fact, FTA may support path exceptions more naturally. For example, for path exceptions arising due to different operation modes (e.g., for testing, normal operation, and other modes) of a circuit, FTA can be easily constrained (through cofactoring) to the desired operation mode. FTA has potential as a practical alternative to STA and applicable to industrial designs, especially because whose logic depths cannot be large and TCF formulas are relatively easy to solve (as the TCF for each PO can be solved separately and the cone of influence reduction can be significant).

We did not experiment with the SAT-based two-vector analysis proposed in [16] since our focus is on floating mode analysis as discussed in Section II-C. However we would expect that the excessive number of event and value variables, which need to be created for every potential event propagation time instant, in the formula of [16] might lead to inefficient SAT solving.

B. Critical Region Identification

TABLE VI
CRITICAL REGION IDENTIFICATION

Circuit	#G	Topological		Functional		Time
		#G	#Path	#G	#Path	(s)
b05	1022	322	7435427	186	8669	0.04
b17	33741	1637	5585965	79	232	0.61
b18	117941	1101	77585298	465	20839024	18.67
c3540	1741	270	1054	91	100	0.02
c5315	25585	213	832	76	60	0.02
c7552	3827	304	97	61	8	0.01
i10	2724	452	127483	338	3071	0.08
s15850	11067	408	73984	389	22016	0.16
s38417	2608	230	476	112	10	0.06

Table VI evaluates the applicability of SWIFT on identifying timing critical regions under the unit delay model. For a circuit, its true delay is set to be the required time at its POs, and the gates and paths with non-positive slack values are declared critical. Column 2 shows the total number of gates of a circuit; Columns 3 and 4 (respectively Columns 5 and 6) show the numbers of critical gates and paths, respectively, with respect to topological arrival times (respectively functional true arrival times); Column 7 shows the runtime in identifying true critical regions.

The results suggest that SWIFT effectively removed spurious critical gates and paths. As a matter of fact, true critical regions can be much smaller than topological critical regions. By taking circuit b17 as an example, SWIFT detected, in 0.61 seconds (the time spent in SAT solving), that only 79 out of its 1637 topological critical gates are true critical gates, and at least 5585733 out of its 5585965 topological critical paths are false critical paths. Pinpointing true critical regions efficiently can be beneficial to timing optimization. The computation allows a logic synthesis tool to focus on the true critical regions for timing optimization through, for example, gate/wire sizing, buffer insertion, and other techniques.

One concern about FTA might be that temperature, aging, or other changes can make false paths become true. However, notice that FTA under the floating mode assumption is a conservative analysis and satisfies the property that $\chi^{f,\geq t1} \rightarrow \chi^{f,\geq t2}$ if $t1 \geq t2$. Therefore, by increasing a delay in a circuit, the true paths identified by FTA before this delay change will remain true paths. As long as proper gate/wire delay margins are maintained, FTA can be safely applied similar to STA. In

[3]The reported runtime excludes the preprocessing time for arrival time list computation as both prior and our methods were preprocessed in a similar way. The prior method may take slightly longer time because of converting circuits to CNF formulas.

contrast, a two-vector based functional timing analysis may be unsafe due to delay uncertainties.

VII. Conclusions

This paper has shown that functional timing analysis can be made fast and general compared with sate-of-the-art methods. Based on implication relation and other technical improvements, compact CNF encoding for TCFs without and with 0/1-specificity has been devised. Thereby the power of modern SAT solvers can be fully utilized. Experiments on large designs have demonstrated promising results on delay computation and critical region identification. Practical applications of functional timing analysis are anticipated. For future work, logic synthesis for functional timing optimization awaits further investigation.

Acknowledgments

The authors are grateful to Yuji Kukimoto for valuable discussions.

References

[1] P. Ashar and S. Malik. Functional timing analysis using ATPG. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(8): 1025-1030, Aug. 1995.

[2] L. Benini, E. Macii, M. Poncino, G. De Micheli. Telescopic units: A new paradigm for performance optimization of VLSI designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(3): 220-232, 1998.

[3] H.-C. Chen and D. Du. Path sensitization in critical path problem. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(2): 196-207, Feb. 1993.

[4] Y.-T. Chung and J.-H. R. Jiang. Functional timing analysis made fast and general. In *Proc. Design Automation Conference* (DAC), pp. 1055-1060, 2012.

[5] O. Coudert. An efficient algorithm to verify generalized false paths. In *Proc. Design Automation Conference* (DAC), pp. 188-193, 2010.

[6] S. Devadas, K. Keutzer, and S. Malik. Computation of floating mode delay in combinational circuits: Theory and algorithms. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(12): 1913-1923, Dec. 1993.

[7] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. Int'l Conf. on Theory and Applications of Satisfiability Testing* (SAT), pp. 502-518, 2003.

[8] Y. Kukimoto, M. Berkelaar, and K. Sakallah. Static Timing Analysis. In *Logic Synthesis and Verification*, S. Hassoun and T. Sasao (Eds.), Kluwer Academic Publishers, pp. 373-401, 2002.

[9] Y.-M. Kuo, Y.-L. Chang, and S.-C. Chang. Efficient Boolean characteristic function for timed automatic test pattern generation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 28(3): 417-425, March 2009.

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

[11] M. Moskewicz, C. Madigan, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. Design Automation Conference* (DAC), pp. 530-535, 2001.

[12] J. Marques-Silva and K. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. on Computers*, vol. 48, no. 5, pp. 506-521, May 1999.

[13] P. McGeer, A. Saldanha, R. Brayton, and A. Sangiovanni-Vincentelli. Delay Models and Exact Timing Analysis. In *Logic Synthesis and Optimization*, T. Sasao (Eds.), Kluwer Academic Publishers, pp. 167-189, 1993.

[14] P. C. McGeer, A. Saldanha, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vicentelli. Timing analysis and delay-fault test generation using path-recursive functions. In *Proc. Int. Conf. on Computer-Aided Design* (ICCAD), pages 180-183, 1991.

[15] D. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *J. Symbolic Computation*, 2:293-304, 1986.

[16] S. Roy, P. P. Chakrabarti, and P. Dasgupta. Event propagation for accurate circuit delay calculation using SAT. *ACM Trans. Design Autom. Electron. Syst.*, 12(3), Aug. 2007.

[17] M. Riedel. *Cyclic Combinational Circuits*. Ph.D. thesis, California Institute of Technology, 2003.

[18] L. Silva, J. Marques-Silva, L. Silveira, and K. Sakallah. Satisfiability models and algorithms for circuit delay computation. *ACM Trans. on Design Automation of Electronic Systems*, 7(1): 137-158, Jan. 2002.

[19] G. Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic*, pp. 466-483, 1970.

PLACE PHOTO HERE

Yi-Ting Chung received the B.S. and M.S. degrees in Electronics Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2010, and from National Taiwan University, Taipei, Taiwan, in 2012, respectively. She is currently an engineer in Synopsys, Inc. Her current research interests include timing analysis, formal verification, and logic synthesis.

PLACE PHOTO HERE

Jie-Hong R. Jiang (S'03-M'04) received the B.S. and M.S. degrees in Electronics Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1996 and 1998, respectively, and the Ph.D. degree in Electrical Engineering and Computer Sciences from the University of California, Berkeley, in 2004. During his compulsory military service, from 1998 to 2000, he was a Second Lieutenant with the Air Force, R.O.C. In 2005, he joined the Department of Electrical Engineering of National Taiwan University, where he is an Associate Professor. His research interests include logic synthesis, formal verification, and computation models of physical and biological systems.

Dr. Jiang is a member of Phi Tau Phi and the Association for Computing Machinery (ACM).

TABLE IV
CIRCUIT DELAY COMPUTATION: FORMULA SIZE COMPARISON

Unit Delay										
Circuit	Delay	#SAT	[9]		[9]*		SWIFT-01		SWIFT-c	
			#Var	#Clause	#Var	#Clause	#Var	#Clause	#Var	#Clause
b05	54 → 42	8	15672	51104	14673	48048	14623	24054	7786	25030
b18	164 → 159	5	18746	58335	17881	55676	17729	27799	9221	28300
b19	168 → 158	7	84174	262549	80844	252329	80354	126061	41597	128850
c6288	124 → 123	3	4248	12747	4169	12465	4163	6209	2118	6279
leon2	42 → 42	1	514	1703	513	1700	506	844	250	829
leon3	44 → 44	1	354	1171	341	1132	337	566	173	560
leon3mp	40 → 38	3	433380	1408779	433363	1408726	374203	645212	157530	526881
netcard	29 → 29	1	144	503	120	425	118	218	70	226
ray	178 → 178	1	10338	35173	10319	35116	10201	17448	5051	17205
s35932	29 → 26	4	100608	301828	80736	242212	79584	121252	49152	138244
uoft_raytracer	178 → 178	1	11476	39015	11446	38923	11326	19353	5618	19109
Fanout Delay										
Circuit	Delay	#SAT	[9]		[9]*		SWIFT-01		SWIFT-c	
			#Var	#Clause	#Var	#Clause	#Var	#Clause	#Var	#Clause
b05	80.6→64.0	44	291364	945194	261188	851515	260910	425922	145404	469541
b18	242.8→238.0	12	20140	62692	19989	62235	19780	30969	9861	30677
b19	244.4→234.4	25	528358	1652131	520394	1627688	518621	812975	262406	820125
c6288	176.4→174.8	3	3222	9669	3184	9510	3179	4731	1606	4753
leon2	2070.0→2070.0	1	41544	149437	39598	139713	33454	65810	14628	55764
leon3	6854.4→6854.4	1	198674	599655	198673	599650	165905	267059	66569	201522
leon3mp	3093.2→3093.2	1	37263	112190	37263	112190	31119	49953	12488	37664
netcard	16390.2→16390.2	1	393228	1179685	393222	1179667	327686	524301	131078	393231
ray	383.6→383.6	1	796	2561	795	2558	731	1216	334	1087
s35932	42.8→39.0	4	86784	260356	83040	249124	81888	124420	42240	122692
uoft_raytracer	383.6→383.6	1	796	2561	795	2558	731	1216	334	1087
TSMC 0.18μm Cell Library with Merged Rise/Fall Time										
Circuit	Delay	#SAT	[9]		[9]*		SWIFT-01		SWIFT-c	
			#Var	#Clause	#Var	#Clause	#Var	#Clause	#Var	#Clause
b05	5.67→4.45	62	484714	1571927	448452	1459216	448054	729616	241959	782062
b18	13.49→13.43	3	2326	7415	2304	7346	2224	3607	1083	3457
b19	14.09→13.80	15	114446	352033	113504	349133	113024	174195	56743	174462
c6288	13.95→13.79	5	8358	25004	8358	25004	8349	12465	3450	10315
leon2	13.16→13.16	1	9356	28285	9354	28278	7818	12605	3142	9532
leon3	14.28→14.16	8	62946	190674	62873	190419	52917	85287	21517	65419
leon3mp	14.58→14.27	17	169690	511361	169655	511239	141943	227942	57133	172517
netcard	8.61→8.42	3	13180	39583	13176	39567	10990	17603	4404	13227
ray	31.84→31.75	5	10310	34683	10305	34668	10149	17183	4999	16866
s35932	2.83→2.64	5	85888	257669	82016	246053	80832	122885	41760	121125
uoft_raytracer	31.98→31.98	1	804	2711	803	2708	787	1339	386	1306
TSMC 0.18μm Cell Library with Separate Rise/Fall Time										
Circuit	Delay	#SAT	[9]		[9]*		SWIFT-01		SWIFT-01*	
			#Var	#Clause	#Var	#Clause	#Var	#Clause	#Var	#Clause
b05	4.67→3.52	59	433885	1407065	395960	1288509	395599	644785	395223	1035824
b18	11.08→10.98	7	9278	28949	9212	28743	9212	14506	9068	23260
b19	11.67→11.42	14	135078	415139	131787	405016	131531	201863	131083	331954
c6288	10.13→10.02	5	4068	12206	4039	12044	4038	5981	4031	9978
leon2	11.07→11.07	1	4678	14143	4676	14136	4676	7916	3140	7983
leon3	12.81→12.68	9	26375	79963	23556	71440	23542	39931	16214	41433
leon3mp	12.39→12.03	4	95668	324471	95660	324443	93612	192986	64012	197794
netcard	7.67→6.87	27	2915186	9076031	2915111	9075731	2874407	4917606	2334791	6173065
ray	26.77→26.43	18	55045	183570	55000	183422	54672	87265	54141	140306
s35932	2.45→2.35	5	40768	122308	38528	115588	38528	60836	37376	92548
uoft_raytracer	26.40→25.82	31	213908	743738	203693	708322	203693	339113	202964	540281

TABLE V
CIRCUIT DELAY COMPUTATION: RUNTIME COMPARISON

Unit Delay									
Circuit	STA (s)	[9]		[9]*		SWIFT-01		SWIFT-c	
		CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)
b05	0.00	0.03	0.11	0.04	0.10	0.01	0.07	0.00	0.03
b18	0.01	1.72	1.44	1.82	1.38	1.13	1.04	0.18	0.53
b19	0.02	11.50	8.12	11.74	6.88	6.91	6.06	0.52	1.88
c6288	0.00	0.19	0.15	0.39	0.22	0.13	0.17	0.15	0.06
leon2	0.09	0.21	0.10	0.39	0.10	0.00	0.03	0.00	0.02
leon3	0.10	0.06	0.12	0.33	0.10	0.00	0.02	0.00	0.02
leon3mp	0.07	12229.60	259.94	21576.08	222.79	7232.13	51.49	0.38	19.07
netcard	0.08	0.09	0.05	0.04	0.05	0.00	0.00	0.00	0.00
ray	0.02	2.01	0.84	2.83	1.04	0.37	0.69	0.39	0.67
s35932	0.00	6.95	2.49	3.67	1.83	1.00	0.56	0.02	0.70
uoft_raytracer	0.02	1.08	0.92	1.68	0.91	0.80	0.69	0.38	0.71

Fanout Delay									
Circuit	STA (s)	[9]		[9]*		SWIFT-01		SWIFT-c	
		CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)
b05	0.00	0.54	2.37	0.64	2.09	0.25	1.53	0.08	0.72
b18	0.01	1.23	1.07	0.98	0.88	0.77	0.75	0.14	0.50
b19	0.02	77.60	33.49	70.85	31.53	34.34	0.10	3.12	6.92
c6288	0.00	0.12	0.14	0.35	0.18	0.10	0.14	0.06	0.04
leon2	0.09	241.61	2.25	297.88	2.15	72.59	0.58	0.02	0.53
leon3	0.10	1172.05	54.23	1396.66	52.96	173.39	9.09	0.06	7.41
leon3mp	0.07	150.63	2.54	188.92	2.53	62.93	0.45	0.01	0.31
netcard	0.08	0.76	130.42	0.43	144.87	0.37	47.73	4.11	37.63
ray	0.02	0.13	0.19	0.39	0.23	0.08	0.17	0.03	0.18
s35932	0.00	8.92	2.32	5.76	1.83	1.13	0.63	0.03	0.66
uoft_raytracer	0.02	0.06	0.19	0.28	0.23	0.11	0.18	0.04	0.18

TSMC 0.18μm Cell Library with Merged Rise/Fall Time									
Circuit	STA (s)	[9]		[9]*		SWIFT-01		SWIFT-c	
		CSolver(s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)
b05	0.00	0.89	4.17	1.07	3.93	0.43	2.63	0.09	1.11
b18	0.01	0.16	0.21	0.08	0.16	0.08	0.13	0.02	0.09
b19	0.02	1.53	3.32	1.26	2.88	0.51	2.07	0.16	1.73
c6288	0.00	2.12	0.51	0.54	0.42	0.42	0.44	0.07	0.10
leon2	0.09	38.36	2.09	54.04	2.56	25.32	2.22	0.23	2.18
leon3	0.10	169.67	4.08	104.34	3.81	16.63	1.35	0.05	0.91
leon3mp	0.07	876.46	10.83	698.01	10.83	137.37	4.04	0.15	3.20
netcard	0.08	88.08	4.87	51.79	5.03	11.01	1.96	0.06	1.58
ray	0.02	0.82	0.77	0.70	0.79	0.19	0.47	0.05	0.25
s35932	0.00	6.06	2.87	4.28	2.06	1.53	0.61	0.01	0.64
uoft_raytracer	0.02	0.11	0.16	0.31	0.13	0.06	0.10	0.06	0.10

TSMC 0.18μm Cell Library with Separate Rise/Fall Time									
Circuit	STA (s)	[9]		[9]*		SWIFT-01		SWIFT-01*	
		CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)	CSolver (s)	SSolver (s)
b05	0.00	0.83	3.24	0.93	3.45	0.36	2.37	0.55	3.23
b18	0.01	0.58	0.71	0.47	0.58	0.51	0.52	0.45	0.50
b19	0.03	1.24	3.41	1.31	2.84	0.58	2.07	0.72	3.27
c6288	0.00	0.59	0.21	0.38	0.28	0.55	0.13	0.39	0.10
leon2	0.11	35.55	2.99	47.09	2.30	1.86	1.40	6.82	1.38
leon3	0.12	130.69	3.01	81.84	2.41	11.18	1.50	16.71	1.45
leon3mp	0.08	603.09	8.11	503.58	8.38	106.86	6.35	35.19	7.10
netcard	0.10	92964.40	2643.65	83647.58	3135.06	99380.81	582.49	83803.08	700.78
ray	0.02	0.58	3.27	1.01	3.12	0.93	2.01	0.97	2.44
s35932	0.00	2.91	1.31	3.69	0.98	2.04	0.52	1.40	0.87
uoft_raytracer	0.02	7.02	11.72	11.47	9.26	3.73	6.86	2.96	8.23