Fixed Points for Multi-Cycle Path Detection

Vijay D'Silva
Computing Laboratory
Oxford University, U.K.
Email: vijay.dsilva@comlab.ox.ac.uk

Daniel Kroening
Computing Laboratory
Oxford University, U.K.
Email: daniel.kroening@comlab.ox.ac.uk

***Abstract*—Accurate timing analysis is crucial for obtaining the optimal clock frequency, and for other design stages such as power analysis. Most methods for estimating propagation delay identify multi-cycle paths (MCPs), which allow timing to be relaxed, but ignore the set of reachable states, achieving scalability at the cost of a severe lack of precision. Even simple circuits contain paths affecting timing that can only be detected if the set of reachable states is considered. We examine the theoretical foundations of MCP identification and characterise the MCPs in a circuit by a fixed point equation. The optimal solution to this equation can be computed iteratively and yields the largest set of MCPs in a circuit. Further, we define conservative approximations of this set, show how different MCP identification methods in the literature compare in terms of precision, and show one method to be unsound. The practical application of these results is a new method to detect multi-cycle paths using techniques for computing invariants in a circuit. Our implementation performs well on several benchmarks, including an exponential improvement on circuits analysed in the literature.**

I. Introduction

A primary feature of high performance integrated circuits is their clock frequency. The maximal clock speed for a circuit is determined using timing analysers that operate on information culled from several design stages such as simulation, logic synthesis and layout. This paper demonstrates via theoretical and experimental results that invariants of a sequential circuit can improve the efficiency and accuracy of timing analysis, just as spatio-temporal correlations improve power-estimation [7].

The topological delay of a circuit is an efficiently computable but excessively pessimistic upper-bound on propagation delay. Considering false paths, which do not propagate signals, and multi-cycle paths (MCPs), which can take more than one clock cycle to propagate a value, allows timing constraints to be relaxed. Practical techniques for identifying multi-cycle paths must be efficient, accurate and sound. Delay-independent methods are particularly important because they can be be applied early in the design flow. An unsound analysis may result in *speed-paths* in the post-silicon phase: paths that do not meet physical timing requirements. Such errors are horrendously expensive to rectify.

Multi-cycle paths and their effect on timing optimisation were first studied in [1]. Multi-cycle path identification techniques can be classified along three dimensions: (1) the topological granularity; combinational paths or register pairs, (2) the criterion used; either sensitisation or state transitions and (3) reachability information considered. In early work, a delay-based sensitisation criterion and an explicit representation of the reachable states was used to identify individual paths [1], [4], [11]. The number of paths and states in a circuit are respectively exponential in the number of gates and registers – both already large values – so an explicit analysis is doomed.

This research is supported by the Semiconductor Research Corporation (SRC) under contract no. 2006-TJ-1539 and by a Microsoft European PhD Fellowship.

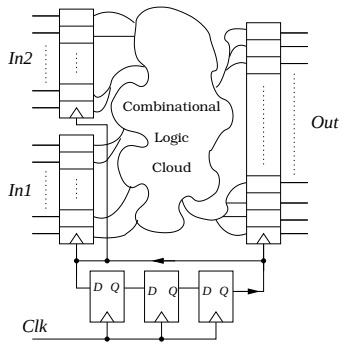


Fig. 1. Circuit with MCPs due to reachable states

To combat path explosion, pairs of registers, called multi-cycle pairs, have been analysed instead of individual paths [9]. State-explosion is overcome by considering all and not just reachable states, which is well suited to SAT-based [8], and ATPG-based [5] methods. The loss of precision from ignoring reachable states prompted heuristics to examine small state machines in the circuit and to use partial reachability information [6].

A. Limitations in Existing Work

We discuss on three limitations in existing work:

1) Lack of precision from ignoring the reachable states.
2) Lack of scalability in enumerative methods.
3) The connection between different criteria for MCP detection is unclear, making it difficult to combine them.

The problem of not considering reachable states is best illustrated with an example. Consider the circuit in Figure 1. There are two 8-bit input registers In_1 and In_2, and one 16-bit output register *Out*. If the shift register producing the clock for these is initialised to 100, all paths between the input and output registers are MCPs. *All* techniques that do not compute reachable states will be unable to identify these MCPs.

The trade-off between scalability and precision made by ignoring the reachable states is evident in published benchmarks. For example, 279 seconds were required to detect 29 pairs of registers connected by multi-cycle paths in the `s953` ISCAS benchmark in Nakamura et al.'s 1998 paper [9]. This computation time was reduced to 19.9 seconds using BDD-based methods in 2000 [8], and to 0.5 seconds in 2002 by Higuchi's ATPG-based procedure [5]. However, though in 1998 [9], 4 seconds were required to detect 150 pairs of registers connected by multi-cycle paths, including the critical path, in the `s838` benchmark, *no* paths were found by the ATPG procedure published in 2002 [5], which terminated in 0.3 seconds. These two extreme examples demonstrate that while techniques for this problem have become more efficient, in some cases, the price is a proportional and practically unacceptable loss of precision.

The explosion due to enumerating multi-cycle pairs is also easy to see in Figure 1. Several existing techniques check that every pair of input and output registers is an MCP pair, though computing the set of reachable states of the shift register suffices to identify all MCPs. In other words, computing *invariants* about small parts of the circuit may be sufficient to identify a large number of MCP pairs.

Finally, different criteria exist in the literature on MCP detection. The relationship between these criteria is unclear and occludes combinations of methods based on their efficiency and precision.

B. Contributions and Organisation

This paper contains the following contributions.

1) Fixed point characterisation of the set of MCPs: This characterisation yields an iterative method for computing the largest set of multi-cycle paths in the circuit.
2) Relationship between approximate MCP criteria: We define a general notion of a sound MCP criterion. Thus we can identify methods that are sound and show one criterion to be unsound. Additionally, the relative precision of different methods can be stated mathematically.
3) MCP identification using invariants: We use our theoretical results to devise a method for MCP detection by computing invariants of small parts of the circuit, dramatically improving scalability.

Our fixed point characterisation is obtained by modelling the set of reachable states and paths in a circuit as a complete lattice. We define a function on this lattice that iteratively eliminates paths that violate the MCP criterion. The largest set of MCPs in the circuit is an extremal fixed point of this function. This mathematical setting makes it easy to identify sound criteria and relate them.

The paper is organised as follows: Section II establishes the mathematical framework and the main characterisation, Section III identifies and relates sound MCP criteria, Section IV presents a method for extracting invariants which is evaluated in Section V.

II. Multi-Cycle Paths as Fixed Points

A. Basic Definitions

Let $\mathbb{B} = \{0, 1\}$ be the set of Boolean values. For simplicity, we consider circuits that contain only AND gates and inverters.

Definition 1: A *netlist* C is a directed graph (V, E, τ), where V is a finite set of vertices, $E \subseteq V \times V$ is the set of directed edges and $\tau : V \to \{\text{AND}, \text{INV}, \text{REG}, \text{INPUT}\}$ maps a vertex to its type, where AND is an AND gate, INV is an inverter, REG is a register, and INPUT is an input. The in-degree of a vertex of type AND is at least one, of type INV and REG is exactly one and of type INPUT is zero.

We write $v_1 \to v_2$ iff $(v_1, v_2) \in E$ and $v_1 \rightsquigarrow v_2$ if there is a path from v_1 to v_2. A *combinational path* p is a sequence of vertices $v_0, \ldots, v_n$ such that $v_i \to v_{i+1}$ and $\tau(v_i) \neq \text{REG}$ for $0 \leq i \leq n$, and there exist $v, v' \in V$ with $v \to v_0$, $v_n \to v'$, and $\tau(v) = \tau(v') = \text{REG}$. The vertices in p are called *path-inputs* (or *on-path inputs*). A *side-input* is a vertex v such that $v \to v_i$ for some v_i in p and v is not a path-input. We require that all combinational paths are acyclic. Let $\mathit{Paths}(C)$ be the set of combinational paths in a netlist C.

A *state* of a netlist is a mapping of registers to Booleans. Let $\vec{r}$ be a vector of the registers r_1 to r_n in a netlist. An element $(b_1, \ldots, b_n)$ of $\mathbb{B}^n$ represents a state in which r_i has the value b_i. A netlist defines a finite state machine or transition system.

Definition 2: A netlist C with n registers and m inputs defines a *transition system* $M_C = (S, I, T, S_0)$, where $S = \mathbb{B}^n$ is the set of states, $I = \mathbb{B}^m$ is the set of inputs, $S_0 \subseteq S$ is the set of initial states, and $T \subseteq S \times I \times S$ is the transition relation. If a transition $(s, i, s') \in T$, then the result of providing the input i when the circuit C is in state s is the state s'.

A state $s \in S$ is *reachable* if there is a sequence of transitions that respect T from a state in S_0 to the state s. Let $\mathit{Reach}(C)$ denote the set of reachable states of a circuit C as defined by M_C. Let $r(s)$ denote the value of register r in state s.

B. Multi-Cycle Paths

We adapt the definition of MCPs in [1] to netlists. A *controlling value* of a logic gate determines the gate's output value irrespective of other inputs. If e is the controlling value of a gate, $\bar{e}$ is the *non-controlling* value. For example, the controlling value of an AND gate is 0 and of an OR gate is 1.

A multi-cycle path can take more than one cycle to stabilise, and a k-cycle MCP (k-MCP) can take at least k cycles to stabilise. Observe that a $(k + 1)$-MCP is also a k-MCP. Stabilisation is defined using dynamic sensitisation [2], [12].

Definition 3: Given a delay assignment, a combinational path p in C is *dynamically sensitisable* iff there exists a reachable state, and an input such that each pair of gates v_{i-1}, v_i in p satisfies two conditions:

1) if any input of v_i has a controlling value, v_{i-1} must have the controlling value and be the first to present it;
2) if all inputs to v_i have non-controlling values, v_{i-1} must be the last to present its value.

A dynamically sensitisable path is required to propagate at least one value. Formalising this condition with a logical predicate enables us to encode MCP identification as a satisfiability problem, and facilitates the characterisation we provide.

Let p be a combinational path from register r_1 to r_2. Let $del(v)$ denote the delay assigned to a gate v. If v_i is the path input to a gate v_{i+1}, let v_i' denote the side input. Define the predicate $\texttt{cont}(v_i)$ to be true if v_i has the controlling value for v_{i+1}. The predicate $\texttt{dsen}(p, s, i)$, defined below, is true for a path $p = v_1 \dots v_{n+1}$, state s, and input i if applying the input i in state s dynamically sensitises p. The parameters i and s are implicit in determining the values of v_i and v_i'.

$$\texttt{dsen}(p, s, i) \stackrel{\textbf{def}}{=} \bigwedge_{j=1}^{n} \texttt{cont}(v_j') \Rightarrow (\texttt{cont}(v_j) \wedge del(v_j) < del(v_j')) \\ \wedge \neg\texttt{cont}(v_j) \Rightarrow (\neg\texttt{cont}(v_j') \wedge del(v_j) > del(v_j'))$$

If the input to a path p changes and p cannot be dynamically sensitised for the next k cycles, then p is a k-MCP. The predicate mcp below is true if *every* change in the input to p (here r_1) is followed by k cycles in which p is not dynamically sensitised.

$$\texttt{mcp}(p, k) \stackrel{\textbf{def}}{=} \forall s_0 \in Reach(C), (s_0, i_1, s_1) \dots (s_{n-1}, i_n, s_n) \in T : \\ r_1(s_0) \neq r_1(s_1) \Rightarrow \bigwedge_{j=1}^{k} \neg\texttt{dsen}(p, s_j, i_j)$$

The paths in a circuit that satisfy $\texttt{mcp}(p, 2)$ are the set of MCPs. Such a definition of the set of MCPs provides no insight into how this set or conservative approximations thereof can be computed. Instead, we define the set of MCPs using fixed points.

C. MCPs as Fixed Points

We first introduce lattices and fixed points. A partial order on a set D is a binary relation $\sqsubseteq$ that is

1) *reflexive*: $d \sqsubseteq d$ for $d \in D$,
2) *anti-symmetric*: $d_1 \sqsubseteq d_2 \wedge d_2 \sqsubseteq d_1 \Rightarrow d_1 = d_2$, and
3) *transitive*: $d_1 \sqsubseteq d_2 \wedge d_2 \sqsubseteq d_3 \Rightarrow d_1 \sqsubseteq d_3$.

An upper bound of an element d is any d' such that $d \sqsubseteq d'$. The least upper bound of d_1, d_2, denoted $d_1 \sqcup d_2$, is the smallest upper bound of d_1 and d_2 with respect to $\sqsubseteq$. Lower bounds and greatest lower bounds, denoted $\sqcap$, are similarly defined. A *lattice* is a set with a partial order $\sqsubseteq$ and the operators $\sqcup$ and $\sqcap$. A lattice is *complete* if any subset of D has a least upper bound in D. A complete lattice, denoted $(D, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$, has a least element, denoted $\bot$ and a greatest element denoted $\top$.

Example 1: Let A be a finite set and $D = \wp(A)$ be the power-set of A. We have the complete lattice $(D, \subseteq, \cup, \cap, \emptyset, A)$, where the partial order is subset inclusion, the least upper bound is union, and the greatest lower bound is intersection, the least element is $\emptyset$, and the greatest element is A.

Let $F : D \to D$ be a function on a complete lattice. F is *monotone* if whenever $d_1 \sqsubseteq d_2$, it holds that $F(d_1) \sqsubseteq F(d_2)$. A fixed point of F is any value d such that $F(d) = d$. At this juncture, we recall two classic results in lattice theory, the Knaster-Tarski and Kleene fixed point theorems. The Kleene theorem requires that a function is *continuous* on the lattice, a stronger condition than monotonicity. Though we do not discuss continuity further, the functions we introduce satisfy this condition, allowing using the theorem.

Theorem 1: Let F be a continuous and hence monotone function on a complete lattice with least element $\bot$.

- (Knaster-Tarski) The set of fixed points of F forms a complete lattice.
- (Kleene) If F is continuous, the least upper bound of the set $\{F^n(\bot) | n \geq 0\}$ is the least fixed point of F.

These results have two implications. If we formalise the set of MCPs as a fixed point of a monotone function on a complete lattice, by the Knaster-Tarski theorem, such a fixed point is guaranteed to exist. If the lattice is finite as in our case, the Kleene fixed point also allows us to compute the least fixed point by iteratively applying F. Let $lfp(F)$ denote the least fixed point of F.

We now introduce the lattice of *approximate MCPs*. An element of the lattice is a set of states and combinational paths (Q, P), representing that the states in Q have been analysed and the paths in P are considered potential MCPs. We can order these elements, denoted $(Q, P) \sqsubseteq (Q', P')$, by the information they contain about MCPs. If $Q \subseteq Q'$, then (Q', P') contains more states from which a path may be sensitised and provides more precise information. Similarly, if $P' \supseteq P$, then some paths have been eliminated from P because they may be sensitisable, hence P' is a more conservative approximation. This lattice is defined formally below.

Definition 4: Let C be a circuit with set of states S and combinational paths $Paths(C)$. The *MCP lattice* is defined as $(D, \sqsubseteq, \sqcup, \sqcap, (Q_\bot, P_\bot), (Q_\top, P_\top))$, where

1) $D = \wp(S) \times \wp(Paths(C))$, contains sets of states and paths,
2) $(Q, P) \sqsubseteq (Q', P')$ if $Q \subseteq Q'$ and $P \supseteq P'$,
3) $(Q, P) \sqcup (Q', P') = (Q \cup Q', P \cap P')$,
4) $(Q, P) \sqcap (Q', P') = (Q \cap Q', P \cup P')$, and
5) $(Q_\bot, P_\bot) = (\emptyset, Paths(C))$, and $(Q_\top, P_\top) = (S, \emptyset)$ are the bottom and top elements respectively.

Observe that the set of all states is considered and not just the set of reachable states. The bottom element contains no states and all paths, modelling the situation of least information in which no states are known and all paths may be MCPs. The top element similarly models the most conservative result in which all states are considered and no MCPs are detected.

We recall the standard next-state function of a transition relation. For any set of states $Q \subseteq S$ of a transition system with transition relation T, let $next(Q) = \{s' \in S \mid s \in Q, (s, i, s') \in T$ for some input i and some state $s \in Q\}$. An element of the MCP lattice can be refined using next states. Let $M_C = (S, I, T, S_0)$ be a transition system. The refinement function using dynamic sensitisation is

$$F_{\texttt{dsen}}((Q, P)) = \begin{cases} (S_0, Paths(C)) & \text{if } (Q, P) = (Q_\bot, P_\bot) \\ (Q \cup next(Q), P \setminus P') & \text{otherwise} \end{cases}$$

where $P' = \{p | s_1 \in Q, (s_1, i_1, s_2), (s_2, i_2, s_3) \in T, r \to p \to r', r(s_1) \neq r(s_2)$, and $\texttt{dsen}(p, s_2, i_2)\}$ is the set of paths dynamically sensitised in a transition from $s_2 \in Q$.

Applying $F_{\texttt{dsen}}$ to (Q, P) does not decrease Q or increase P, thus $F_{\texttt{dsen}}$ is a monotone function. Consider the sequence $F(\bot), F(F(\bot)) = F^2(\bot), \ldots, F^n(\bot)$, where $\bot = (Q_\bot, P_\bot)$. The first element of the sequence contains the initial states and the set of all combinational paths. Subsequent elements iteratively refine the set of MCPs while considering more reachable states. The lattice is finite hence complete, so by the Kleene fixed point theorem, the sequence converges to $lfp(F_{\texttt{dsen}})$. The first result of this paper is that this fixed point is exactly the set of MCPs in a circuit.

Theorem 2: Let C be a circuit, and $(Q, P) = lfp(F_{\texttt{dsen}})$. For any path $p \in Paths(C)$, $\texttt{mcp}(p, 2)$ is true iff $p \in P$.

Proof: ($\Rightarrow$) If $p \in P$, for all states $s \in Q$, and transitions $(s_1, i_1, s_2) \in T$, it must be that $r(s) = r(s')$ or that for all inputs i_2, $\neg\texttt{dsen}(p, s_2, i_2)$ holds. The set Q in the fixed point is precisely $Reach(C)$, so $\texttt{mcp}(p, 2)$ holds.
($\Leftarrow$) If $\texttt{mcp}(p, 2)$ holds, then by definition, from every reachable state s_0 and transition to states s_1 and input i_1, if $r(s_0) \neq r(s_1)$, then $\neg\texttt{dsen}(p, s_1, i_1)$ holds. One can show by induction that if p is in the set of paths in one iteration, then p is not removed by applying $F_{\texttt{dsen}}$. ∎

This characterisation provides an algorithm for iteratively computing the reachable states and simultaneously refining the set of MCPs. Thus it is guaranteed to contain the largest set of MCPs that can be identified using delay and reachability information. However, this set is in general expensive to compute, so we aim for conservative approximations.

III. Approximate MCP Criteria

We now focus on approximate criteria for MCP identification. An error in the result of an MCP analysis can have dire consequences leading to design errors that may only be detected in silicon. Approximate solutions must therefore be *sound*. That is, the solution must be a subset of the exact set of MCPs. Lemma 1 allows us to derive sound results by approximating the function $F_{\texttt{dsen}}$ instead of the result of the analysis. Such an approach is common in program analysis and verification [3]. If for all (Q, P) in this lattice, $F(Q, P) \sqsubseteq G(Q, P)$, we say that F is *more precise* than G.

Lemma 1: Let G be a monotone function on the MCP lattice. If $F_{\texttt{dsen}}$ is more precise than G, $lfp(F_{\texttt{dsen}}) \sqsubseteq lfp(G)$.

The lemma holds because $F_{\texttt{dsen}}(\bot) \sqsubseteq G(\bot)$ by assumption. Further, every application of the two functions satisfies this condition. If $lfp(F_{\texttt{dsen}}) = (Q, P)$ and $lfp(G) = (Q', P')$, then $(Q, P) \sqsubseteq (Q', P')$ hence $P' \subseteq P$ as required for soundness.

We can now classify the relative precision of different techniques in the literature by viewing them as functions over the MCP lattice. Recall that static co-sensitisation is a delay-independent sensitisation condition [2].

Definition 5: A path p is *statically co-sensitisable* iff there exists a reachable state, and an input to p such that, if a side-input to a gate g in p has a controlling value, the path input to g has the controlling value.

As before, we can encode this condition as a predicate (which is directly applicable in SAT-based methods). The co-sensitisation predicate for a path $p = v_1 \ldots v_{n+1}$, input i and state s is defined as

$$\texttt{csen}(p, s, i) \stackrel{\text{def}}{=} \bigwedge_{j=1}^{n} \left(\texttt{cont}(v'_j) \Rightarrow \texttt{cont}(v_i)\right) \wedge \left(\neg\texttt{cont}(v_i) \Rightarrow \neg\texttt{cont}(v'_i)\right) .$$

For any p, s, i as above, observe that $\texttt{dsen}(p, s, i) \Rightarrow \texttt{csen}(p, s, i)$, because co-sensitisation does not require delay constraints to be met. Thus, it is easier to co-sensitise a path in a circuit. Define the function $F_{\texttt{csen}}$ similar to $F_{\texttt{dsen}}$ using $\texttt{csen}$ instead of $\texttt{dsen}$. It holds that $lfp(F_{\texttt{dsen}}) \sqsubseteq lfp(F_{\texttt{csen}})$.

Relaxing delay constraints is one method of sound approximation. Another is to consider all states instead of reachable states – an approach that works well with SAT solvers. Let F be $F_{\texttt{dsen}}$ or $F_{\texttt{csen}}$. The function $\widehat{F}$, which ignores reachable states, is obtained by defining $\widehat{F}(Q_\bot, P_\bot)$ to be $(S, Paths(C))$ instead of $(S_0, Paths(C))$, where S includes all states. Then, $lfp(\widehat{F})$ will be (S, P) for some P because S cannot grow larger.

A third approximate method considers pairs of registers instead of combinational paths. This method can be modelled in our setting by viewing a register pair as all combinational paths between the pair. If one path between the registers is sensitisable, *all paths* are removed from the set of possible MCPs. Let $Paths(r_1, r_2)$ denote the paths between r_1 and r_2 and r_1^p, r_2^p denote the source and sink register for a path p. Let $G_{\texttt{dsen}}$ be defined as $F_{\texttt{dsen}}$ except that $G_{\texttt{dsen}}(Q, P) = (Q \cup next(Q), P \setminus P'')$, where $P'' = \cup_{p \in P'} Paths(r_1^p, r_2^p)$, where P' is as before. Observe that $P' \subseteq P''$, so for any (Q, P), $F_{\texttt{dsen}}(Q, P) \sqsubseteq G_{\texttt{dsen}}(Q, P)$. It follows that the least fixed point of $G_{\texttt{dsen}}$ is also a sound approximation.

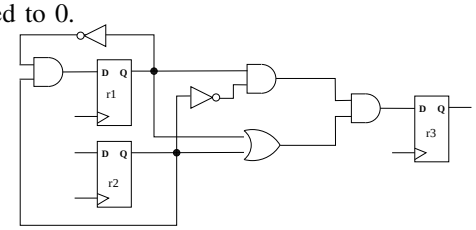
Approximation methods can further be combined. For example, [13] uses $\texttt{csen}$ with paths, but no reachable states. These functions are summarised in Table I with references indicating where they appear in the literature. The $\sim$ symbol indicates that partial reachability information is considered. Some work uses a condition $\texttt{tran}$, which we discuss next.

For a path p between r_1 and r_2, the transition condition is:

$$\texttt{tran}(p) \stackrel{\text{def}}{=} \forall s_0 \in Reach(C), (s_0, i_1, s_1), (s_1, i_2, s_2) \in T : r_1(s_0) \neq r_1(s_1) \Rightarrow r_2(s_1) \neq r_2(s_2)$$

The next example shows that this condition is not sound.

Example 2: Consider the circuit below with r_1 and r_2 initialised to 0.



The set of reachable states, written as (r_1, r_2, r_3) is $\{(0,0,0), (0,1,0), (1,1,0)\}$. The transition criterion is satisfied for the pair (r_1, r_3). Consider the delay assignment of $5, 10$

Function	Ref.	Topology	Condition	Reachability
$F_{\texttt{dsen}}$	[1]	Paths	`dsen`	✓
$F_{\texttt{csen}}$	[4]	Paths	`csen`	✓
$\widehat{F}_{\texttt{dsen}}$		Paths	`dsen`	×
$\widehat{F}_{\texttt{csen}}$	[13]	Paths	`csen`	×
$G_{\texttt{dsen}}$	[10]	Pairs	`dsen`	✓
$G_{\texttt{tran}}$	[9]	Pairs	`tran`	✓
$G'_{\texttt{tran}}$	[6]	Pairs	`tran`	∼
$\widehat{G}_{\texttt{tran}}$	[8]	Pairs	`tran`	×
$\widehat{G}_{\texttt{csen}}$	[13]	Pairs	`csen`	×

TABLE I
DIFFERENT APPROXIMATE MCP REFINEMENT FUNCTIONS

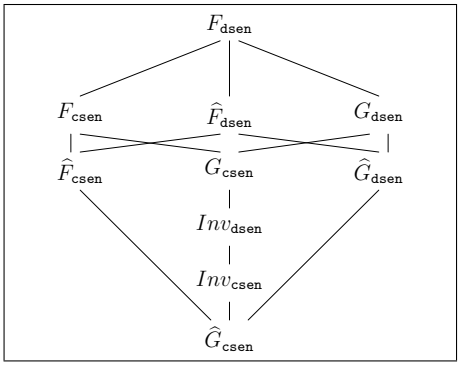


Fig. 2. Lattice of MCP refinement functions

and 15 nanoseconds for NOT, AND, and OR gates. If the state of (r_1, r_2) changes from $(1,1)$ to $(0,0)$, the 0 from r_1 arrives at the AND gates before the value from r_2 sensitising the path. ◁

Though the `tran` condition is unsound, methods using this condition are easy to remedy, by simply using `csen` instead. We conclude this section with a theorem relating different approximate methods (shown pictorially to save space).

Theorem 3: If a function F refines G, both defined over the MCP lattice, then F is drawn above G with a line between the two. The refinement relations between different MCP functions in the literature are shown in Figure 2.

Observe that the MCP refinement functions themselves form a lattice. Any function with an F of $\widehat{F}$ faces path-explosion independent of the criterion used. Functions considering reachable states (those besides $\widehat{G}, \widehat{F}$) face state explosion. Only the $\widehat{G}$ functions appear feasible, but ignoring reachable states results in a severe loss of precision. The best trade-off can be made by computing invariants of small parts of the circuit (shown by the *Inv* functions). Such a method is suggested in [6] using the `tran` predicate. We use sound criteria instead.

IV. Improving Precision with Approximate Reachability Information

In this section, we briefly outline how MCP detection can be improved with partial reachability information. A similar idea appears in [6]. The difference here is that we use static co-sensitisation instead of state transitions, and use simple statistical analysis of the circuit's topology to identify which invariants to compute. Further, we use state-of-the-art model checking technology such as Bounded Model Checking (BMC) with completeness threshold estimation to compute these invariants.

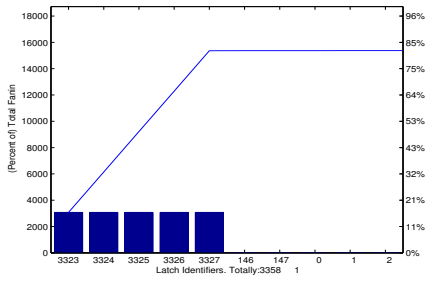


Fig. 3. Pareto plot of the fan-in of latches in the aeMB microprocessor core. Note that 5 latches contribute to over 80% of the total fan-in in the circuit

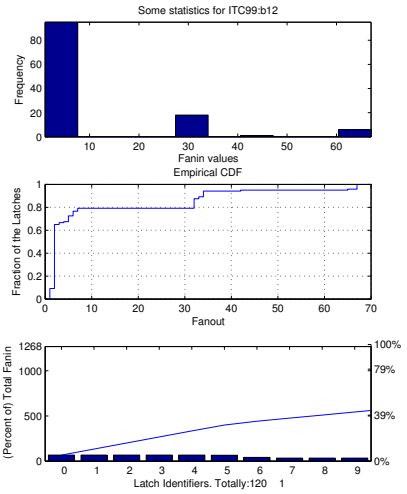


Fig. 4. Fanout and fan-in statistics for circuit b12 in the ITC'99 benchmarks

We observe that a relatively small portion of a circuit generates the majority of the clock enables. Figure 3 shows a Pareto plot of the contribution of some latches to the total fan-in of a microprocessor core. We can see that just 5 latches out of 3358 contribute to 85% of the fan-in of the circuit. Consider the `b12` circuit from the `ITC'99` benchmarks. The first plot in Figure 4 is a histogram showing that for about 90 of 120 latches in the circuit, the fan-out is less than 10. Notice that that 10 latches contribute to nearly 40% of the fan-in of the circuit.

We call such registers *significant*. If the cone-of-influence of such registers is small, computing the reachable states for this sub-circuit is feasible. This provides approximate reachability information about the circuit, because the reachable states of other registers is not known. This information can then improve the precision of MCP identification.

V. Evaluation

Our ideas have been implemented in a tool for finding multi-cycle register pairs. Circuits can be described in Verilog or the

Circuit	#MCPs	Time(s)
b01	7	.03
b02	3	0
b03	122	.75
b04	121	.86
b05	108	.94
b06	4	.02
b07	418	3.18
b08	76	.28
b09	251	1.10
b10	30	.22
b11	162	2.52
b12	1260	38.79
b13	257	1.11

TABLE II
MCP DETECTION ON ITC BENCHMARKS

ISCAS'89 net-list format. Our results of MCP detection on the ITC'99 and ISCAS'89 benchmarks are given in Tables II and III. The combination of accuracy and efficiency in our tool exceeds that of currently reported results on these benchmarks. In some cases, the number of MCPs found is an order of magnitude higher than those in existing literature.

Approximate reachability information is obtained by unwinding small sub-circuits up to their reachability diameter. This method is convenient with a SAT solver and allows combining reachability analysis with MCP detection. The co-sensitisation criterion from this paper is directly encoded.

Our technique scales to large circuits and has also been tested on industrial net-lists a few megabytes large and faced no capacity problems. In fact, the figures reported in Tables II and III can further be reduced because we do not currently use simulation-based filtering to eliminate easily sensitisable register pairs. Despite this difference, our figures are comparable to those obtained by methods using simulation and a combination of non-formal, and hence far more efficient, heuristics.

VI. CONCLUSION

In this paper, we have revisited the foundations of Multi-Cycle Path detection. We provide two main theoretical results and a practical evaluation. The set of MCPs is characterised by a fixed point equation. Conservative approximations of this set can then be computed with fixed point approximation techniques. We mathematically relate different MCP detection methods and identify an unsound criterion in the literature, which is quite easily rectified. Our conclusion is that a feasible yet precise approach to MCP detection must use approximate reachability information. This conclusion is supported by convincing experimental results.

Acknowledgements: We thank Ranan Fraer, Horatiu Jula, Gila Kamhi, Sasha Novakovsky, and Hadas Ronen for discussions.

Circuit	#MCPs	Time(s)
s1423	1281	51.51
s1488	18	.23
s1494	18	.23
s2081	34	.03
s208a	34	.03
s208	34	.03
s298	62	.12
s386a	16	.06
s386	16	.06
s400	140	.34
s420a	134	.28
s420	86	.16
s444	140	.36
s510	25	.10
s526a	136	.42
s526	136	.42
s5378	714	36.85
s641	109	.58
s713	109	.69
s820	0	.11
s832	0	.11
s838a	526	2.78
s838	190	.68
s9234	2273	208.87
s953	82	1.12

TABLE III
MCP DETECTION ON ISCAS'89 BENCHMARKS

REFERENCES

[1] P. Ashar, S. Dey, and S. Malik. Exploiting multi-cycle false paths in the performance optimization of sequential circuits. In *International Conference on Computer-aided Design (ICCAD)*, pages 510–517. IEEE, 1992.

[2] H.-C. Chen and D. H.-C. Du. Path sensitization in critical path problem. In *ICCAD*, pages 208–211, 1991.

[3] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Principles of Programming Languages (POPL)*, pages 238–252. ACM, 1977.

[4] A. P. Gupta and D. P. Siewiorek. Automated multi-cycle symbolic timing verification of microprocessor-based designs. In *Design Automation Conference (DAC)*, pages 113–119. ACM, 1994.

[5] H. Higuchi. An implication-based method to detect multi-cycle paths in large sequential circuits. In *Conference on Design automation*, pages 164–169, New York, NY, USA, 2002. ACM Press.

[6] H. Higuchi and Y. Matsunaga. Enhancing the performance of multi-cycle path analysis in an industrial setting. In *Asia South Pacific Design Automation Conference (ASP-DAC)*, pages 192–197. IEEE, 2004.

[7] R. Marculescu, D. Marculescu, and M. Pedram. Switching activity analysis considering spatiotemporal correlations. In *International Conference on Computer-Aided Design (ICCAD)*, pages 294–299. IEEE, 1994.

[8] K. Nakamura, S. Maruoka, S. Kimura, and K. Watanabe. Multi-clock path analysis using propositional satisfiability. In *Asia South Pacific Design Automation Conference (ASP-DAC)*, pages 81–86. ACM Press, 2000.

[9] K. Nakamura, K. Takagi, S. Kimura, and K. Watanabe. Waiting false path analysis of sequential logic circuits for performance optimization. In *International Conference on Computer-aided Design (ICCAD)*, pages 392–395. ACM, 1998.

[10] C. Papachristou and M. Nourani. False path exclusion in delay analysis of RTL-based datapath-controller designs. In *Conference on European design automation (Euro-DAC)*, pages 336–341. IEEE, 1996.

[11] A. Saldanha, H. Harkness, P. C. McGeer, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Performance optimization using exact sensitization. In *Design Automation Conference (DAC)*, pages 425–429. ACM Press, 1994.

[12] J. P. M. Silva and K. A. Sakallah. An analysis of path sensitization criteria. In *ICCD*, pages 68–72, 1993.

[13] K. Yang and K.-T. Cheng. Efficient identification of multi-cycle false path. In *Asia South Pacific Design Automation Conference (ASP-DAC)*, pages 360–365. IEEE, 2006.