Enhanced Diameter Bounding via Structural Transformation

Jason Baumgartner[1] Andreas Kuehlmann[2]

[1] IBM Server Group, Austin, TX
[2] Cadence Berkeley Labs, Berkeley, CA

Abstract

Bounded model checking (BMC) has gained widespread industrial use due to its relative scalability. Its exhaustiveness over all valid input vectors allows it to expose arbitrarily complex design flaws. However, BMC is limited to analyzing only a specific time window, hence will only expose those flaws which manifest within that window and thus cannot readily prove correctness. The diameter of a design has thus become an important concept — a bounded check of depth equal to the diameter constitutes a complete proof. While the diameter of a design may be exponential in the number of its state elements, in practice it often ranges from tens to a few hundred regardless of design size. Therefore, a powerful diameter overapproximation technique may enable automatic proofs that otherwise would be infeasible. Unfortunately, exact diameter calculation requires exponential resources, and overapproximation techniques may yield exponentially loose bounds. In this paper, we provide a general approach for enabling the use of structural transformations, such as redundancy removal, retiming, and target enlargement, to tighten the bounds obtained by arbitrary diameter approximation techniques. Numerous experiments demonstrate that this approach may significantly increase the set of designs for which practically useful diameter bounds may be obtained.

1 Introduction

Due to the complexity of modern hardware designs, formal verification methods are gaining widespread use to augment the coverage shortcomings of simulation-based validation approaches. Unfortunately, formal methods generally require exponential resources with respect to design size. General unbounded approaches, such as symbolic reachability analysis, are PSPACE-complete whereas bounded approaches are NP-complete for a given bound [1]. Due to the lower resource requirements, and the wider variety of available algorithms for discharging a bounded verification problem, **BMC** has gained significant industrial utilization in recent years. **BMC** [2] attempts to find a property violation within k time-steps from the initial state(s) of a design. Though exhaustive, the bounded nature of this approach implies incompleteness; performing a check of depth 0 through k does not necessarily imply that no violation will occur at depths greater than k.

The incompleteness of **BMC** has resulted in a set of research activities to discern a minimal k for which a bounded check is complete. The *diameter* [2] of a design is typically defined as the maximum distance between any two of its states s_i and s_j such that s_j is reachable from s_i. The distance from s_i to s_j is the minimum number of time-steps to transition the design from s_i to s_j. Clearly a bounded check of depth greater or equal to the diameter of the design is complete. However, a general diameter calculation relies upon quantified Boolean formulae (QBF) thus is PSPACE-complete [2]. Sufficient heuristics for solving such problems are not yet available, though some are beginning to emerge [3, 4].

Due to the computational complexity of diameter calculation, various overapproximate techniques have been proposed. For example, the recurrence diameter [2] of a design is its maximum-length irredundant state sequence, and may be calculated by a series of propositional satisfiability problems. This approach is NP-complete as long as the depth is polynomial with respect to design size. However, the recurrence diameter may be exponentially larger than the diameter, hence it is of limited practical utility and may require exorbitant resources to compute.* The technique of [7] performs a fast diameter overapproximation based upon purely structural analysis, and enables compositional diameter bounding. This approach may yield tight bounds for certain designs (primarily acyclic and memory-based) for which the recurrence diameter is loose, though may also result in exponentially-loose bounds for other designs. Other approaches, such as [8], have proposed the use of incomplete algorithms to estimate diameter, though are not guaranteed to yield an upper-bound.

In some cases the use of diameter to bound **BMC** depth is more conservative than necessary; i.e., a smaller bound is sufficient for completeness. For example, we may ignore any vertices outside of the cone of influence of the property, which may decrease the diameter [7]. Additionally, a **BMC** application for the maximum distance from any *initial* state, rather than from any *reachable* state, suffices for property checking [6]. Furthermore, to assess if a given design signal t may ever be asserted, we need to perform a search only deep enough to assess whether signal t may toggle from 0 to 1 relative to an initial state; the amount of time necessary to toggle t from a 1 to a 0 from any state may be exponentially greater. These concepts are revisited in Definition 3, which generalizes the traditional state-based diameter definition, and are used in Theorem 4.

In this paper we demonstrate how structural transformations may be used to enhance arbitrary diameter approximation techniques. This research enables the use of a diameter bound obtained upon a transformed design to yield a tight bound for the original, untransformed design via a constant-time calculation. Due to the reduction potential of these transformations, this theory may enable overapproximate techniques to yield exponentially tighter diameter bounds. Furthermore, the diameter bounding process itself may attain significant speedups by operating on the smaller, transformed designs. Numerous experimental results are provided for benchmark and industrial designs to illustrate our approach.

There are several motivations for this research.

*A hybrid between a QBF and a recurrence diameter approach is proposed in [5] to partially alleviate the respective shortcomings of these techniques. Bounded cone-of-influence analysis is proposed in [6] to tighten recurrence-diameter-based bounds.

1. To our knowledge, this paper is the first to discuss the theoretical impact of these transformations upon design diameter.

2. These transformations often reduce design size, hence often reduce verification resource requirements as demonstrated by the cited papers. Therefore, one may question the utility of back-translating a diameter to the original design vs. attempting a proof solely upon the transformed design. However, note that in some cases, such a transformation may actually hurt verification. E.g., retiming may increase input count while reducing register count [9]; retiming and phase-abstraction [10] may increase the size of next-state functions. These transformations may vary both resource requirements and tightness of the obtained approximation using diameter approximation techniques; this variance may not correlate to their impact upon verification. Therefore, this research constitutes yet another practical mechanism which may be attempted to discharge difficult verification problems.

2 Design Semantics: The Netlist

Definition 1. A *netlist* is a tuple $N = \langle\langle V, E\rangle, G, Z\rangle$ comprising a directed graph with vertices V and edges $E \subseteq V \times V$. Function $G : V \mapsto types$ represents a semantic mapping from vertices to gate *types*, including constants, primary inputs (i.e., nondeterministic bits), registers (denoted as R), and combinational gates with various functions. A *state* is a mapping from registers to $0,1$ values; Z represents the set of initial states.

Definition 2. The *semantics of a netlist N* are defined in terms of semantic traces: $0,1$ valuations to gates over time. We denote the set of all legal traces associated with a netlist by $P \subseteq [V \times \mathbb{N} \mapsto \{0,1\}]$, defining P as the subset of all possible functions from $V \times \mathbb{N}$ to $\{0,1\}$ which are consistent with G. The value of gate v at time i in trace p is denoted by $p(v,i)$.

Our verification problem consists of a set of *targets* $T \subseteq V$ correlating to a set of properties $AG(\neg t), \forall t \in T$. We say that target t is *hit* in trace p at time i iff $p(t,i) = 1$. Due to the ability to synthesize safety properties into automata [11], this invariant-checking model is rarely a practical limitation.

Definition 3. The *diameter* $d(U)$ of vertex set U is the minimum number such that for any trace p and any increasing succession[†] $k_1, \ldots, k_c$, there exists another trace p' and another increasing succession $l_1, \ldots, l_c$ such that $\bigwedge_{j=1}^{c}(l_j \leq k_j)$ and $\left(l_c \leq l_{c-1} + d(U)\right)$, taking $l_0 = -1$, which satisfies $\forall u \in U.\ \bigwedge_{j=1}^{c}\left(p(u,k_j) = p'(u,l_j)\right)$.

Note that our definition of diameter is generally one greater than the standard definition for graphs; this matches the number of time-steps necessary to ensure completeness of **BMC**, and is always greater than zero. Definition 3 may be viewed as a generalization of a traditional state-based diameter definition because the diameter of vertices U actually need not correlate to that of $coi(U) \cap R$. This definition provides an opportunity to bound diameter without a need to analyze the underlying state space representation. For example, if a vertex u encodes an XOR function of a primary input and a sequentially-driven signal A, then $d(u) = 1$ regardless of $d(A)$ since any valuation to u will be producible at any time-step. From this example, one may be tempted to view this definition as using a set of vertices U as a *filter* through which the state space of the netlist is observed; while this is an intuitive interpretation, the following example provides a distinction. Consider a netlist comprising a primary input i_0, which fans out to register r_1 (whose initial value is primary input i_1), which in turn fans out to register r_2 (whose initial value is primary input i_2). The diameter of (r_1, r_2) is two, since it will take two time-steps to view $(1,1)$ after $(0,0)$. However, the diameter of (r_2) is one since r_2 may produce any valuation at any time-step independently of other time-steps, being trace-equivalent (refer to Definition 4) to any primary input.

[†] An increasing succession is an ordered set of natural numbers $k_1, \ldots, k_c$ for $c \geq 1$ which satisfies the relation $k_i < k_{i+1}, \forall i \in [1, c-1]$.

3 Diameter Bounding Transformed Netlists

In this section, we discuss the impact of various types of transformations upon the diameter of a design. This theory will enable a diameter bound $\hat{d}(U')$ obtained upon vertex set U' of a transformed netlist N' to immediately imply a diameter bound $\hat{d}(U)$ on the corresponding vertex set of the original, untransformed netlist N via a constant-time calculation. In many cases, this theory enables that a tight $\hat{d}(U')$ can yield a tight $\hat{d}(U)$. In the worst case, for all of the semantics-preserving transformations to be discussed, the resultant $\hat{d}(U)$ will be at most a linear factor of $\hat{d}(U')$ with respect to $|R|$. Recall that diameter overapproximation techniques may yield exponentially loose bounds with respect to register count, and may require exponential resources with respect to design size. Structural transformations are capable of yielding arbitrarily large reductions in design size, and often require polynomial resources. This collectively implies that the theory of this section may enhance both the resource requirements and the tightness of the diameter overapproximation obtained via any technique. We also discuss how diameter bounds obtained through approximate transformations (which are not both sound and complete) cannot be used in general to bound the diameter of the original netlist.

3.1 Trace-Equivalence-Preserving Transformations

The following definition and theorem illustrates that trace-equivalence-preserving transformations do not alter diameter.

Definition 4. Vertex sets A and A' of netlists N and N', respectively, are said to be *trace equivalent* iff there exists a bijective mapping $\psi : A \mapsto A'$ which satisfies the following conditions.

- $\forall p \in P. \exists p' \in P'. \forall i \in \mathbb{N}. \forall a \in A.\ p(a,i) = p'\left(\psi(a), i\right)$
- $\forall p' \in P'. \exists p \in P. \forall i \in \mathbb{N}. \forall a \in A.\ p(a,i) = p'\left(\psi(a), i\right)$

Theorem 1. Let N and N' be netlists which are trace-equivalent with respect to vertex sets A and A' and bijective mapping $\psi : A \mapsto A'$. The diameter of A is equal to that of A'.

Proof. This theorem follows immediately from Definitions 3 and 4. □

It is well-known that *bisimilarity*, relating state transition graphs of netlists, is more restrictive than trace equivalence – bisimilarity implies trace equivalence, though the latter does not imply the former. Numerous approaches have been proposed for reducing the size of a design while preserving bisimilarity (thus also preserving trace equivalence); for example, those of [12, 13]. Such techniques often require the analysis of the state space of the design, which is computationally too expensive to consistently offer benefits for invariant checking as noted in [13]. However, the technique of *redundancy removal* [14, 15] is widely applied for enhancing verification. The idea of this approach is to attempt to identify two semantically-equivalent vertices u and v; when two such vertices are found, all fanout edges from v are moved to u and v is made a simple buffer

(one-input AND gate) sourced by u. Clearly, redundancy removal does not alter the semantics of any vertex. Such merging is beneficial since it reduces the number of vertices in the cone of influence of a target, and may enable further reductions of the fanout cone of the merged vertex v. Identification of semantically-equivalent vertices may be performed efficiently by structural analysis or by BDD and SAT sweeping [14, 15] with no need to analyze the state space of the netlist. Note also that a cone-of-influence reduction preserves trace-equivalence of all vertices in the cone.

The technique of parametric re-encoding of a netlist [16, 17] replaces the fanin cone C of a cut $\langle C, \overline{C} = V \setminus C \rangle$ of a netlist with a trace-equivalent cone C'. Such re-encoding preserves trace-equivalence of any vertex set in $\overline{C}$.

3.2 Retiming

Leiserson and Saxe [18] proposed the technique of retiming as a synthesis optimization for reducing the number of registers of a design (and thereby its size), or for reducing the clock period of a design by decreasing its longest purely-combinational path. More recently, several researchers have proposed the use of retiming to enhance verification [19, 20, 9].

Definition 5. A *retiming* of netlist N is a gate labeling $r : V \mapsto \mathbb{Z}$, where $r(v)$ is the *lag* of vertex v, denoting the number of registers that are moved backward through v. A *normalized retiming* r' may be obtained from an arbitrary retiming r, and is defined as $r' = r - \max_{v \in V} r(v)$.

In [9], the use of normalized retiming is proposed for enhanced invariant checking. The retimed netlist $\tilde{N}$ has two components: 1) a sequential *recurrence structure* $\tilde{N}'$ which has a unique representative in the original netlist N for each combinational vertex, and whose registers are placed according to Definition 5, and 2) a combinational *retiming stump* $\tilde{N}''$ obtained through unfolding, representing retimed initial values as well as the functions of all gates for prefix time-steps that were effectively discarded from the recurrence structure. It is demonstrated that each gate $\tilde{u}'$ within $\tilde{N}'$ is trace-equivalent to the corresponding u within N, modulo a temporal skew of $-r(\tilde{u}')$ time-steps. Furthermore, there will be $-r(\tilde{u}')$ correspondents to this u within $\tilde{N}''$, each being trace-equivalent to $\tilde{u}'$ for one time-step during this temporal skew.

Theorem 2. If the diameter of a set of vertices $\tilde{U}'$ of the recurrence structure of a normalized-retimed netlist is $d(\tilde{U}')$, and $\exists i. \forall \tilde{u}' \in \tilde{U}'. -r(\tilde{u}') = i$, then the diameter of the original set of vertices U satisfies $d(U) \leq d(\tilde{U}') + i$.

Proof. In [7], it is shown that for a given set of vertices $u_1, \ldots, u_n$ with diameter d, the diameter of a set of acyclic registers $v_1, \ldots, v_n$ whose input edges are sourced by $u_1, \ldots, u_n$, respectively, will be at most $d+1$. Note that we may compose a series of i registers to each $\tilde{u}'$, whose initial values are determined by corresponding values from the retiming stump. This yields a set of vertices U' which are trace-equivalent to U. Each stage of this pipeline is an acyclic register, hence increments diameter by at most 1. This proof therefore follows from Theorem 1 and the results of [9, 7]. □

Each lag will be between $-|R|$ and 0 for any optimal normalized retiming, thus guaranteeing that $d(U)$ is a linear factor of $d(\tilde{U}')$. Retiming is potentially capable of eliminating all registers from a netlist; e.g., for a pipeline with a primary input driving a set of registers connected in series. Because the diameter of a combinational netlist is 1, the diameter of the i-th register of this pipeline will be bounded at $i+1$ by Theorem 2, which is likely to be exact. As demonstrated in [9], redundancy removal plus retiming yields average reductions in register count of 27% for ISCAS89 benchmarks, and of 62% for IBM Gigahertz processor netlists.

3.3 State-Folding Abstractions

In this section we discuss two structural transformations which entail state folding: *phase abstraction* and *c-slow abstraction*. Phase abstraction [10, 17] is a technique to yield a register-based netlist from one composed of level-sensitive latches: gates with two inputs (*data* and *clock*), which act as buffers when their *clock* is active and hold their last-sampled *data* values otherwise. C-slow abstraction [21, 17] is directly applicable to register-based netlists. Both abstractions are applicable to netlists in which the state elements may be c-colored such that state elements of color i may only combinationally fan out to state elements of color $(i+1) \bmod c$.[‡] By eliminating all but one color of state elements (transforming others into combinational logic), both abstractions reduce the number of state elements of a netlist by a factor of $1/c$ or greater. The semantic effect of these abstractions is to temporally fold the resulting netlist modulo-c.

Theorem 3. If the diameter of a set of identically-colored vertices $\tilde{U}$ of phase abstracted or c-slow abstracted netlist $\tilde{N}$ is $d(\tilde{U})$, then the diameter of the corresponding vertex set U of the original netlist N is at most $c \cdot d(\tilde{U})$.

Proof. By Definition 3, if the diameter of the phase- or c-slow-abstracted vertex set $\tilde{U}$ is $d(\tilde{U})$, then the longest required duration to witness a particular valuation to $\tilde{U}$ is $d(\tilde{U})$ time-steps. From [10, 21, 17], we know that phase abstraction and c-slow abstraction fold time modulo-c. Therefore, any transition of states in $\tilde{N}$ correlates to c transitions in N, and the corresponding valuation to U must occur within $c \cdot d(\tilde{U})$ time-steps. □

3.4 Target Enlargement

Target enlargement is a technique to render a target which may be hit at a shallower depth from the initial states of a netlist, and with a higher probability, than the original target [22, 23]. Target enlargement is based upon preimage computation to calculate the set of states which may hit the target within k time-steps. Inductive simplification may be performed upon the i-th preimage to eliminate states which hit the target in fewer than i time-steps. The result of this calculation is the characteristic function of the set of states S which is a subset of all states that can hit the target in $0, \ldots, k$ steps, and a superset of those that can hit the target in exactly k steps minus those that can hit the target in $0, \ldots, k-1$ steps. Prior research has noted the benefit of representing the enlarged target structurally to enable better synergy with simulation and SAT-based analysis [24], and to enable a reduction in the size of the cone-of-influence of the enlarged target [7].

Theorem 4. If the diameter of a k-step enlarged target t' is $d(t')$, then the original target t is hittable within $d(t') + k$ time-steps, if at all.

Proof. If $d(t') = i$, then t' must be hittable at time $0, \ldots, i-1$ if at all, as per Definition 3. Because the enlarged target is a subset of the states that hit the target within k time-steps, and a superset

[‡]Note that c may readily be bounded by $|R|$. In [17], a netlist preprocessing technique is formalized to allow c-slow abstraction to be applied to any netlist where each directed cycle comprises a factor of $c > 1$ registers.

of those that hit the target in exactly k time-steps minus those that hit the target in fewer than k time-steps, the original target must be hittable within $0, \ldots, k+i-1$ time-steps, if at all. □

Due to the temporal union and input quantification inherent in target enlargement, it may be the case that a transition of the enlarged target t' from 1 to 0 may be skewed and possibly eliminated with respect to such a transition of the original target t. For example, target t may be an OR of signals A and B, where B encodes the function *counter* $\not\equiv 0$ for a mod-c *counter*. The first hit of t via A may cause the *counter* to unconditionally begin counting, such that t may thereafter only be deasserted one time-step of every c time-steps. Target enlargement may obscure this deasserted time-step such that once hit, t' will never be deasserted. The number of time-steps necessary to drive a binary 1 to t from any reachable state of N may be exponentially smaller than the number necessary to subsequently drive a binary 0 onto t. Therefore, target enlargement does not entail as clean of an impact on diameter as we may hope; we cannot use a target enlargement approach to bound the diameter of an intermediate component of a partitioned netlist [7], for example. However, the result of Theorem 4 is sufficient to allow a bound derived from the target-enlarged netlist to imply an upper-bound on the number of time-steps sufficient to perform **BMC** in a complete manner for the original target. In [7], it is demonstrated that target enlargement may yield average register count reductions of 14% for ISCAS89 benchmarks and of 71% for IBM Gigahertz processor netlists.

3.5 Overapproximate Abstraction

Various techniques have been developed for attempting to reduce netlist size while providing an overapproximation of the behavior of that netlist. In other words, any target assessed to be unreachable after overapproximation is guaranteed to be unreachable before the overapproximation. However, a *target hit* obtained for the overapproximated netlist may not imply that the target is hittable in the original netlist. For example, cut-point insertion [25] is commonly used in equivalence checking, and consists of replacing an internal netlist vertex by a primary input. Localization [26] is another common overapproximation technique which replaces all vertices sourcing crossing edges of a cut of the netlist by primary inputs. There are two byproducts of overapproximate abstractions. First, states which are not truly reachable may become reachable, which may clearly increase the diameter. Second, state transitions which are not reachable may become reachable, which may decrease the diameter. Therefore, diameter bounds obtained upon an overapproximated netlist cannot be used in general to obtain a bound for the original netlist.

3.6 Underapproximate Abstraction

An underapproximate abstraction attempts to reduce the netlist size while providing an underapproximation of its behavior. In other words, any trace hitting a target of the abstracted netlist is valid to illustrate a hit of the unabstracted target. However, *target unreachable* results obtained upon the underapproximate netlist may not imply that the target is unreachable in the original netlist. For example, case splitting may replace certain primary inputs by constant values. While a special case of case splitting is semantics-preserving (if the image of a netlist cut is preserved as described in Section 3.1), underapproximate abstractions may generally cause two byproducts. First, reachable states may become unreachable, which may decrease the diameter. Second, reachable state transitions may become unreachable, which may increase diameter. Therefore, diameter bounds obtained upon an underapproximated netlist cannot generally be used to bound the original netlist.

4 Experimental Results

In this section we provide the experimental results of our transformation-based diameter overapproximation theory. All experiments were run on an IBM ThinkPad model T21 running RedHat Linux 6.2, with an 800 MHz Pentium III and 256 MB main memory. As transformations, we utilized a redundancy removal engine COM [27], a retiming engine RET [9], and for the IBM Gigahertz Processor examples, a phase abstraction engine [10].

The diameter overapproximation algorithm utilized for our results is the fast structural technique presented in [7]. To briefly summarize this technique, the approach first partitions the netlist into an acyclic sequence of components, and then compositionally derives an overapproximate diameter bound $\hat{d}$ from this partition. While the diameter of each component is generally exponential in its register count, it is demonstrated that certain commonly-occurring types of components have a much smaller impact on diameter. Four classes of components are considered: *constant components* (*CC*s) — these do not increase the diameter; *acyclic components* (*AC*s), which represent a one-stage pipeline of arbitrary width — these increment the diameter by one regardless of how many bits wide the pipeline stage is; *memory/queue components* (*MC*s/*QC*s) — these multiply the diameter by the number of atomically-updated *rows* plus one regardless of the bit-width of each row; and *general components* (*GC*s) — these are the catch-all category for all other strongly-connected components, and may have a diameter exponential in their register count. Rather than using more expensive diameter bounding techniques such as QBF or recurrence diameter for *GC*s, we assume an exponential diameter increase in these experiments for speed.

Our first set of experiments summarized in Table 1 are on the ISCAS89 benchmarks, using each primary output as a target for lack of any more meaningful available targets. We categorized the registers in the netlist into the various component types. We additionally ran the diameter overapproximation algorithm of [7] on all targets; any with a diameter of less than 50 were enumerated in set $T' \subseteq T$ and the average of these corresponding diameters is reported. The bound of 50 was arbitrarily chosen as being a reasonable cut-off size for discharging with **BMC**. In the bottom row we report the cumulative sum of registers of the corresponding types, and the cumulative sum of $|T'|$ and $|T|$. We performed these experiments on the original netlists; on redundancy-removed netlists (COM); and on netlists after redundancy removal and retiming (COM,RET,COM), using Theorems 1 and 2. We perform the identical set of experiments on randomly-selected phase-abstracted GP netlists in Table 2. We do not report per-line computational resources for these experiments; the structural diameter overapproximation algorithms consume less than 1 second and 1 MB per target. In contrast to the results reported in [7] which do not reflect the results of retiming nor redundancy removal, these experiments are intended to illustrate precisely the effects of these transformations.

Many registers are initially non-complex for the ISCAS netlists: 21% are acyclic registers, and 5% are table cells (elements of *MC*s/*QC*s). A total of 477 original targets (30%) have a diameter of less than 50. After redundancy removal, 24% of the registers are acyclic, and 10% are table cells; 556 of the targets (34%) have a diameter of less than 50. After redundancy removal and retiming, 10% of the registers are acyclic and 11% are table cells. This drop

in acyclic registers is due primarily to their elimination by retiming. A total of 639 targets (40%) have a diameter of less than 50.

A larger fraction of the registers is originally non-complex for the GP netlists, as is intuitive for highly-pipelined gigahertz designs: 1% are constants, 57% are acyclic, and 13% are table cells. A total of 95 targets (33%) have a diameter of less than 50. After redundancy removal, 0.5% of the registers are constants, 58% are acyclic, and 15% are table cells. A total of 111 targets (39%) have a diameter of less than 50. After retiming and redundancy removal, 1% of the registers are constants, 19% are acyclic, and 34% are table cells. A total of 126 (44%) of these targets have a diameter of less than 50.

Overall, the results clearly indicate that the use of structural transformations, coupled with the theories presented in this paper, are capable of significantly increasing the set of targets for which a practically useful overapproximate diameter bound may be obtained. We increase the percentage by 10% or more of such targets in both ISCAS89 and GP netlists. This result is particularly profound noting that we did not employ any (possibly costly) techniques to attempt to tighten *GC* diameter bounds; our experiments thus reflect a very fine line between being able to attain a small diameter bound and a huge bound. As techniques emerge for efficiently improving diameter bounding for *GC*s, the transformation-based theory we have developed should prove even more useful to obtain superior results with lesser resources.

In some cases, the diameter bound computed for a retimed netlist is slightly larger than that of the original netlist – for example, with S1196 and S15850_1. This is partially due to the inequality in Theorem 2; we must add the negated target lag to its diameter bound, even though retiming may not have reduced register count for that target. Use of a normalized retiming helps minimize this potential increase, as would retiming and normalizing a single target cone at a time (instead of the entire netlist). However, the potential for increase tends to be very small (since most lags tend to be very small and are bounded by $-|R|$), whereas the potential for decrease is exponentially greater. Transformations also impact table identification and clustering heuristics for *MC*s/*QC*s. Due to the speed of these heuristic algorithms, it may be beneficial to run them on every possible netlist representation to enable the best possible result.

5 Conclusion

We have presented a practical approach with supporting theory to enable the use of structural transformations (such as retiming, redundancy removal, and target enlargement) to enhance diameter overapproximation techniques, and thus to help enable completeness of bounded verification approaches. The theory enables a diameter bound obtained upon a transformed design to backward-imply a tight bound for the original, untransformed design. Due to the reduction potential inherent in these transformations, this theory is capable of significantly increasing the set of designs for which practically useful diameter bounds may be calculated efficiently. This capability is validated by numerous experimental results. A promising future research direction is to apply this theory for speeding up quantified-Boolean-formulae-based diameter calculation, and to exponentially tighten alternate overapproximate techniques such as ones based upon recurrence diameter.

References

[1] A. Aziz, V. Singhal, and R. K. Brayton, "Verifying interacting finite state machines: Complexity issues," Tech. Rep. UCB/ERL M93/52, University of California at Berkeley, July 1993.

[2] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Tools and Algorithms for Construction and Analysis of Systems*, March 1999.

[3] L. Zhang and S. Malik, "Conflict driven learning in a quantified Boolean satisfiability solver," in *Int'l Conference on Computer Design*, Nov. 2002.

[4] M. N. Mneimneh and K. A. Sakallah, "SAT-based sequential depth computation," in *ASP Design Automation Conference*, January 2003.

[5] M. Sheeran, S. Singh, and G. Stalmarck, "Checking safety properties using induction and a SAT-solver," in *Formal Methods in Computer-Aided Design*, Nov. 2000.

[6] D. Kroening and O. Strichman, "Efficient computation of recurrence diameters," in *Int'l Conference on Verification, Model Checking, and Abstract Interpretation*, Jan. 2003.

[7] J. Baumgartner, A. Kuehlmann, and J. Abraham, "Property checking via structural analysis," in *Computer-Aided Verification*, July 2002.

[8] C.-C. Yen, K.-C. Chen, and J.-Y. Jou, "A practical approach to cycle bound estimation," in *Int'l Workshop on Logic & Synthesis*, 2002.

[9] A. Kuehlmann and J. Baumgartner, "Transformation-based verification using generalized retiming," in *Computer-Aided Verification*, July 2001.

[10] J. Baumgartner, T. Heyman, V. Singhal, and A. Aziz, "Model checking the IBM Gigahertz Processor: An abstraction algorithm for high-performance netlists," in *Computer-Aided Verification*, July 1999.

[11] E. A. Emerson, "Temporal and modal logic," *Handbook of Theoretical Computer Science*, vol. B, 1990.

[12] A. Aziz, V. Singhal, G. M. Swamy, and R. K. Brayton, "Minimizing interacting finite state machines: A compositional approach to language containment," in *Int'l Conference on Computer Design*, 1994.

[13] K. Fisler and M. Vardi, "Bisimulation and model checking," in *Correct Hardware Design and Verification Methods*, Sept. 1999.

[14] A. Kuehlmann, V. Paruthi, F. Krohm, and M. Ganai, "Robust Boolean reasoning for equivalence checking and functional property verification," *IEEE Transactions on Computer-Aided Design*, vol. 21, Dec. 2002.

[15] J. Baumgartner and A. Kuehlmann, "Min-area retiming on flexible circuit structures," in *Int'l Conference on Computer-Aided Design*, 2001.

[16] I.-H. Moon, H. H. Kwak, J. Kukula, T. Shiple, and C. Pixley, "Simplifying circuits for formal verification using parametric representation," in *Formal Methods in Computer-Aided Design*, Nov. 2002.

[17] J. Baumgartner, *Automatic Structural Abstraction Techniques for Enhanced Verification*. PhD thesis, University of Texas, Dec. 2002.

[18] C. Leiserson and J. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, 1991.

[19] A. Gupta, P. Ashar, and S. Malik, "Exploiting retiming in a guided simulation based validation methodology," in *Correct Hardware Design and Verification Methods*, Sept. 1999.

[20] G. Cabodi, S. Quer, and F. Somenzi, "Optimizing sequential verification by retiming transformations," in *Design Automation Conference*, June 2000.

[21] J. Baumgartner, A. Tripp, A. Aziz, V. Singhal, and F. Andersen, "An abstraction algorithm for the verification of generalized C-slow designs," in *Computer-Aided Verification*, July 2000.

[22] J. Yuan, J. Shen, J. Abraham, and A. Aziz, "On combining formal and informal verification," in *Computer-Aided Verification*, June 1997.

[23] C. H. Yang and D. L. Dill, "Validation with guided search of the state space," in *Design Automation Conference*, June 1998.

[24] M. Ganai, *Algorithms for Efficient State Space Search*. PhD thesis, University of Texas, May 2001.

[25] A. Kuehlmann and F. Krohm, "Equivalence checking using cuts and heaps," in *Design Automation Conference*, June 1997.

[26] R. P. Kurshan, *Computer-Aided Verification of Coordinating Processes*. Princeton University Press, 1994.

[27] A. Kuehlmann, M. Ganai, and V. Paruthi, "Circuit-based Boolean reasoning," in *Design Automation Conference*, June 2001.

Design	Original Netlist		COM		COM,RET,COM	
	$\|R\| \in \overline{CC}; \overline{AC};$ $MC+QC; GC$	$\|T''\| / \|T\|;$ Avg. $\hat{d}(t')$	$\|R\| \in \overline{CC}; \overline{AC};$ $MC+QC; GC$	$\|T''\| / \|T\|;$ Avg. $\hat{d}(t')$	$\|R\| \in \overline{CC}; \overline{AC};$ $MC+QC; GC$	$\|T''\| / \|T\|;$ Avg. $\hat{d}(t')$
PROLOG	0 ; 107 ; 1 ; 28	14 / 73 ; 8.9	0 ; 103 ; 1 ; 28	**16 / 73 ; 11.9**	0 ; 16 ; 1 ; 28	**24 / 73 ; 21.0**
S1196	0 ; 18 ; 0 ; 0	14 / 14 ; 3.3	0 ; 18 ; 0 ; 0	14 / 14 ; 3.3	0 ; 16 ; 0 ; 0	14 / 14 ; 4.3
S1238	0 ; 18 ; 0 ; 0	14 / 14 ; 3.3	0 ; 18 ; 0 ; 0	14 / 14 ; 3.3	0 ; 16 ; 0 ; 0	14 / 14 ; 4.3
S1269	0 ; 9 ; 17 ; 11	2 / 10 ; 10.0	0 ; 9 ; 17 ; 11	2 / 10 ; 10.0	0 ; 8 ; 17 ; 11	2 / 10 ; 10.0
S13207_1	0 ; 314 ; 128 ; 196	49 / 152 ; 2.0	0 ; 315 ; 128 ; 195	49 / 152 ; 2.1	0 ; 77 ; 89 ; 183	**79 / 152 ; 6.4**
S1423	0 ; 3 ; 16 ; 55	1 / 5 ; 1.0	0 ; 3 ; 16 ; 55	1 / 5 ; 1.0	0 ; 1 ; 12 ; 59	1 / 5 ; 2.0
S1488	0 ; 0 ; 0 ; 6	19 / 19 ; 33.0	0 ; 0 ; 0 ; 6	19 / 19 ; 33.0	0 ; 0 ; 0 ; 6	19 / 19 ; 33.0
S1494	0 ; 0 ; 0 ; 6	19 / 19 ; 33.0	0 ; 0 ; 0 ; 6	19 / 19 ; 33.0	0 ; 0 ; 0 ; 6	19 / 19 ; 33.0
S1512	0 ; 0 ; 1 ; 56	0 / 21 ; 0.0	0 ; 0 ; 0 ; 57	0 / 21 ; 0.0	0 ; 0 ; 0 ; 57	0 / 21 ; 0.0
S15850_1	0 ; 99 ; 124 ; 311	115 / 150 ; 2.7	0 ; 96 ; 107 ; 328	115 / 150 ; 2.7	0 ; 73 ; 81 ; 292	115 / 150 ; 4.7
S208_1	0 ; 0 ; 0 ; 8	0 / 1 ; 0.0	0 ; 0 ; 0 ; 8	0 / 1 ; 0.0	0 ; 0 ; 0 ; 8	0 / 1 ; 0.0
S27	0 ; 1 ; 2 ; 0	1 / 1 ; 4.0	0 ; 1 ; 2 ; 0	1 / 1 ; 4.0	0 ; 1 ; 2 ; 0	1 / 1 ; 4.0
S298	0 ; 0 ; 1 ; 13	0 / 6 ; 0.0	0 ; 0 ; 1 ; 13	0 / 6 ; 0.0	0 ; 0 ; 1 ; 13	0 / 6 ; 0.0
S3271	0 ; 6 ; 0 ; 110	1 / 14 ; 7.0	0 ; 6 ; 0 ; 110	1 / 14 ; 7.0	0 ; 0 ; 0 ; 110	1 / 14 ; 7.0
S3330	0 ; 103 ; 1 ; 28	16 / 73 ; 11.9	0 ; 103 ; 1 ; 28	16 / 73 ; 11.9	0 ; 16 ; 1 ; 28	**33 / 73 ; 25.3**
S3384	0 ; 111 ; 0 ; 72	6 / 26 ; 16.5	0 ; 111 ; 0 ; 72	6 / 26 ; 16.5	0 ; 0 ; 0 ; 72	6 / 26 ; 16.5
S344	0 ; 0 ; 4 ; 11	3 / 11 ; 5.0	0 ; 0 ; 4 ; 11	3 / 11 ; 5.0	0 ; 0 ; 4 ; 11	3 / 11 ; 5.0
S349	0 ; 0 ; 4 ; 11	3 / 11 ; 5.0	0 ; 0 ; 4 ; 11	3 / 11 ; 5.0	0 ; 0 ; 4 ; 11	3 / 11 ; 5.0
S35932	0 ; 0 ; 0 ; 1728	0 / 320 ; 0.0	0 ; 0 ; 0 ; 1728	0 / 320 ; 0.0	0 ; 0 ; 0 ; 1728	0 / 320 ; 0.0
S382	0 ; 6 ; 0 ; 15	0 / 6 ; 0.0	0 ; 6 ; 0 ; 15	0 / 6 ; 0.0	0 ; 0 ; 0 ; 15	0 / 6 ; 0.0
S38584_1	0 ; 47 ; 4 ; 1375	56 / 304 ; 1.0	1 ; 203 ; 366 ; 854	**133 / 304 ; 14.9**	0 ; 170 ; 345 ; 832	**110 / 304 ; 16.7**
S386	0 ; 0 ; 0 ; 6	7 / 7 ; 33.0	0 ; 0 ; 0 ; 6	7 / 7 ; 33.0	0 ; 0 ; 0 ; 6	7 / 7 ; 33.0
S400	0 ; 6 ; 0 ; 15	0 / 6 ; 0.0	0 ; 6 ; 0 ; 15	0 / 6 ; 0.0	0 ; 0 ; 0 ; 15	0 / 6 ; 0.0
S420_1	0 ; 0 ; 0 ; 16	0 / 1 ; 0.0	0 ; 0 ; 0 ; 16	0 / 1 ; 0.0	0 ; 0 ; 0 ; 16	0 / 1 ; 0.0
S444	0 ; 6 ; 0 ; 15	0 / 6 ; 0.0	0 ; 6 ; 0 ; 15	0 / 6 ; 0.0	0 ; 0 ; 0 ; 15	0 / 6 ; 0.0
S4863	0 ; 62 ; 0 ; 42	0 / 16 ; 0.0	0 ; 83 ; 0 ; 21	0 / 16 ; 0.0	0 ; 16 ; 0 ; 21	0 / 16 ; 0.0
S499	0 ; 0 ; 0 ; 22	0 / 22 ; 0.0	0 ; 0 ; 0 ; 22	0 / 22 ; 0.0	0 ; 0 ; 0 ; 22	0 / 22 ; 0.0
S510	0 ; 0 ; 0 ; 6	7 / 7 ; 33.0	0 ; 0 ; 0 ; 6	7 / 7 ; 33.0	0 ; 0 ; 0 ; 6	7 / 7 ; 33.0
S526N	0 ; 0 ; 1 ; 20	0 / 6 ; 0.0	0 ; 0 ; 1 ; 20	0 / 6 ; 0.0	0 ; 0 ; 1 ; 20	0 / 6 ; 0.0
S5378	0 ; 115 ; 0 ; 64	4 / 49 ; 1.5	0 ; 126 ; 0 ; 53	4 / 49 ; 1.5	0 ; 56 ; 0 ; 56	**7 / 49 ; 3.9**
S635	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0
S641	0 ; 7 ; 0 ; 12	3 / 24 ; 1.0	0 ; 7 ; 0 ; 12	3 / 24 ; 1.0	0 ; 4 ; 0 ; 10	**7 / 24 ; 2.0**
S6669	0 ; 181 ; 0 ; 58	37 / 55 ; 3.4	0 ; 181 ; 0 ; 58	37 / 55 ; 3.4	0 ; 18 ; 0 ; 58	37 / 55 ; 4.0
S713	0 ; 7 ; 0 ; 12	3 / 23 ; 1.0	0 ; 7 ; 0 ; 12	3 / 23 ; 1.0	0 ; 7 ; 0 ; 7	**7 / 23 ; 2.3**
S820	0 ; 0 ; 0 ; 5	19 / 19 ; 17.0	0 ; 0 ; 0 ; 5	19 / 19 ; 17.0	0 ; 0 ; 0 ; 5	19 / 19 ; 17.0
S832	0 ; 0 ; 0 ; 5	19 / 19 ; 17.0	0 ; 0 ; 0 ; 5	19 / 19 ; 17.0	0 ; 0 ; 0 ; 5	19 / 19 ; 17.0
S838_1	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0
S9234_1	0 ; 45 ; 9 ; 157	22 / 39 ; 1.2	0 ; 49 ; 5 ; 157	22 / 39 ; 1.2	0 ; 14 ; 25 ; 133	22 / 39 ; 2.0
S938	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0	0 ; 0 ; 0 ; 32	0 / 1 ; 0.0
S953	0 ; 23 ; 0 ; 6	3 / 23 ; 2.0	0 ; 23 ; 0 ; 6	3 / 23 ; 2.0	0 ; 0 ; 0 ; 6	**23 / 23 ; 29.8**
S967	0 ; 23 ; 0 ; 6	3 / 23 ; 2.0	0 ; 23 ; 0 ; 6	3 / 23 ; 2.0	0 ; 0 ; 0 ; 6	**23 / 23 ; 29.8**
S991	0 ; 0 ; 0 ; 19	17 / 17 ; 8.8	0 ; 0 ; 0 ; 19	17 / 17 ; 8.8	0 ; 0 ; 0 ; 19	17 / 17 ; 8.8
$\sum$	0; 1317; 313; 4622	477 / 1615	1; 1503; 653; 4086	**556 / 1615**	0; 509; 583; 3992	**639 / 1615**

Table 1: Diameter bounding experiments for ISCAS89 benchmarks.

Design	Original Netlist		COM		COM,RET,COM	
	$\|R\| \in \overline{CC}; \overline{AC};$ $MC+QC; GC$	$\|T''\| / \|T\|;$ Avg. $\hat{d}(t')$	$\|R\| \in \overline{CC}; \overline{AC};$ $MC+QC; GC$	$\|T''\| / \|T\|;$ Avg. $\hat{d}(t')$	$\|R\| \in \overline{CC}; \overline{AC};$ $MC+QC; GC$	$\|T''\| / \|T\|;$ Avg. $\hat{d}(t')$
CP_RAS	0 ; 279 ; 66 ; 315	0 / 2 ; 0.0	0 ; 286 ; 66 ; 307	0 / 2 ; 0.0	0 ; 179 ; 65 ; 238	0 / 2 ; 0.0
CLB_CNTL	0 ; 29 ; 2 ; 19	0 / 2 ; 0.0	0 ; 25 ; 2 ; 19	0 / 2 ; 0.0	0 ; 15 ; 2 ; 20	0 / 2 ; 0.0
CR_RAS	0 ; 96 ; 6 ; 329	0 / 1 ; 0.0	0 ; 100 ; 7 ; 321	0 / 1 ; 0.0	0 ; 52 ; 10 ; 284	0 / 1 ; 0.0
D_DASA	0 ; 16 ; 81 ; 18	1 / 2 ; 35.0	0 ; 10 ; 86 ; 13	**2 / 2 ; 27.0**	0 ; 1 ; 86 ; 13	**2 / 2 ; 28.0**
D_DCLA	0 ; 382 ; 1 ; 754	0 / 2 ; 0.0	0 ; 387 ; 1 ; 748	0 / 2 ; 0.0	0 ; 14 ; 0 ; 736	0 / 2 ; 0.0
D_DUDD	0 ; 30 ; 28 ; 71	4 / 22 ; 9.2	0 ; 21 ; 28 ; 71	4 / 22 ; 10.8	0 ; 1 ; 21 ; 71	**7 / 22 ; 11.0**
I_IBBQn	0 ; 623 ; 1488 ; 0	15 / 15 ; 4.7	0 ; 623 ; 1488 ; 0	15 / 15 ; 4.7	0 ; 0 ; 1488 ; 0	15 / 15 ; 4.7
I_IFAR	0 ; 303 ; 11 ; 99	0 / 2 ; 0.0	0 ; 257 ; 11 ; 93	0 / 2 ; 0.0	0 ; 41 ; 18 ; 79	0 / 2 ; 0.0
I_IFPF	11 ; 893 ; 44 ; 598	0 / 1 ; 0.0	1 ; 923 ; 35 ; 525	0 / 1 ; 0.0	0 ; 191 ; 4 ; 218	0 / 1 ; 0.0
L3_SNP1	25 ; 529 ; 39 ; 82	0 / 5 ; 0.0	6 ; 400 ; 41 ; 62	0 / 5 ; 0.0	0 ; 31 ; 30 ; 41	**1 / 5 ; 1.0**
L_EMQn	5 ; 146 ; 6 ; 66	0 / 1 ; 0.0	5 ; 136 ; 6 ; 66	**1 / 1 ; 1.0**	5 ; 20 ; 14 ; 57	**1 / 1 ; 1.0**
L_EXEC	12 ; 421 ; 0 ; 102	0 / 2 ; 0.0	0 ; 430 ; 0 ; 58	0 / 2 ; 0.0	0 ; 88 ; 0 ; 57	0 / 2 ; 0.0
L_FLUSHn	6 ; 198 ; 0 ; 4	7 / 7 ; 3.7	0 ; 194 ; 0 ; 4	7 / 7 ; 3.7	0 ; 12 ; 0 ; 4	7 / 7 ; 4.0
L_INTRo	14 ; 143 ; 12 ; 5	30 / 30 ; 3.8	0 ; 135 ; 12 ; 5	30 / 30 ; 3.8	0 ; 3 ; 12 ; 4	**30 / 30 ; 3.6**
L_LMQo	28 ; 690 ; 4 ; 133	0 / 16 ; 0.0	24 ; 682 ; 4 ; 141	0 / 16 ; 0.0	24 ; 114 ; 2 ; 132	0 / 16 ; 0.0
L_LRU	0 ; 142 ; 20 ; 75	0 / 12 ; 0.0	0 ; 127 ; 86 ; 9	**12 / 12 ; 15.0**	0 ; 0 ; 86 ; 8	**12 / 12 ; 15.0**
L_PFQo	14 ; 1936 ; 17 ; 84	1 / 67 ; 1.0	8 ; 1929 ; 82 ; 20	1 / 67 ; 1.0	8 ; 192 ; 83 ; 17	1 / 67 ; 1.0
L_PNTRn	3 ; 228 ; 10 ; 11	23 / 31 ; 2.0	0 ; 211 ; 10 ; 11	23 / 31 ; 2.0	0 ; 1 ; 10 ; 11	23 / 31 ; 4.0
L_PRQn	34 ; 366 ; 106 ; 265	10 / 10 ; 15.2	30; 367; 108; 260	10 / 10 ; 15.2	30 ; 12 ; 64 ; 302	**10 / 10 ; 8.0**
L_SLB	3 ; 135 ; 6 ; 27	2 / 3 ; 1.0	0 ; 126 ; 6 ; 26	2 / 3 ; 1.0	0 ; 15 ; 6 ; 23	2 / 3 ; 1.0
L_TBWKn	0 ; 202 ; 117 ; 14	0 / 21 ; 0.0	0 ; 186 ; 119 ; 12	**1 / 21 ; 1.0**	0 ; 1 ; 78 ; 53	**1 / 21 ; 1.0**
M_CIU	0 ; 343 ; 10 ; 424	0 / 6 ; 0.0	0 ; 321 ; 5 ; 417	0 / 6 ; 0.0	0 ; 63 ; 60 ; 286	**6 / 6 ; 1.0**
SIDECAR4	3 ; 109 ; 32 ; 455	0 / 1 ; 0.0	0 ; 60 ; 34 ; 453	0 / 1 ; 0.0	0 ; 24 ; 34 ; 67	0 / 1 ; 0.0
S_SCU1	1 ; 232 ; 4 ; 136	0 / 3 ; 0.0	0 ; 220 ; 6 ; 124	0 / 3 ; 0.0	0 ; 75 ; 4 ; 70	**2 / 3 ; 2.0**
V_CACH	5 ; 94 ; 15 ; 59	0 / 1 ; 0.0	0 ; 93 ; 14 ; 52	0 / 1 ; 0.0	1 ; 22 ; 15 ; 27	**1 / 1 ; 1.0**
V_DIR	6 ; 91 ; 13 ; 68	0 / 2 ; 0.0	0 ; 100 ; 13 ; 55	0 / 2 ; 0.0	0 ; 13 ; 10 ; 20	**2 / 2 ; 8.0**
V_SNPM	65; 846; 134; 376	1 / 2 ; 2.0	3 ; 762 ; 97 ; 401	**2 / 2 ; 1.5**	0 ; 51 ; 26 ; 46	**2 / 2 ; 1.5**
W_GAR	0 ; 159 ; 0 ; 83	1 / 7 ; 1.0	0 ; 158 ; 0 ; 82	1 / 7 ; 1.0	0 ; 10 ; 0 ; 81	1 / 7 ; 1.0
W_SFA	0 ; 22 ; 0 ; 42	0 / 8 ; 0.0	0 ; 22 ; 0 ; 42	0 / 8 ; 0.0	0 ; 0 ; 0 ; 42	0 / 8 ; 0.0
$\sum$	235; 9683; 2272; 4714	95 / 284	77; 9291; 2367; 4397	**111 / 284**	68; 1241; 2228; 3007	**126 / 284**

Table 2: Diameter bounding experiments for phase-abstracted GP netlists.