Exploiting Suspected Redundancy without Proving It

Hari Mony[1]   Jason Baumgartner[1]   Viresh Paruthi[1]   Robert Kanzelman[2]

[1] IBM Systems Group, Austin, TX   [2] IBM Engineering & Technology Services, Rochester, MN

Abstract

We present several improvements to general-purpose sequential redundancy removal. First, we propose using a robust variety of synergistic transformation and verification algorithms to process the individual proof obligations. Experiments confirm that this more general approach enables greater speed and scalability, and identifies a significantly greater degree of redundancy, than previous approaches. Second, we demonstrate how we may generalize upon traditional redundancy removal applications and utilize the speculatively-reduced model to enhance bounded search, without needing to complete any proofs.

1 Introduction

Redundancy is inherent in most design representations. A variety of computer-aided design tasks, such as logic synthesis, property checking, and equivalence checking, tend to substantially benefit in performance if this redundancy is eliminated. Redundancy removal often relies upon the process of demonstrating that two gates in a design representation always evaluate to the same value. Once a pair of redundant gates are identified, the design may be simplified by replacing each reference to one gate by a reference to the other; i.e., by *merging* one of the gates onto the other.

Techniques for identifying redundancy in combinational designs have become quite robust, using an efficient hybrid of random simulation, BDD-based analysis, and SAT-based analysis [1]. These techniques may be readily extended to sequential designs to merge gates which may be demonstrated to be redundant in *any possible* state (reachable or unreachable), effectively treating state elements as primary inputs during their analysis.[1] While such an approach is useful for efficiently reducing the size of sequential designs, it is too weak to identify gates which are redundant in all reachable states, but do not appear redundant in certain unreachable states.

There have been several approaches to compensate for the weakness of combinational redundancy removal upon sequential designs. For example, Huang et al. proposed a heuristic approximate reachability approach to capture certain unreachability constraints in an equivalence checking framework [3]. The technique of van Eijk uses BDD-based induction to demonstrate that suspected redundancy candidate gates truly are redundant [4]. Bjesse et al. increased the scalability of induction-based sequential redundancy removal by applying SAT instead of BDDs, and by using a *complete* variant of induction [5]. These approaches help to prevent spurious mismatches in unreachable states without suffering the computational expense of exact reachability analysis. However, some inconclusive results – which must be conservatively treated as mismatches – are still a practical inevitability with such techniques. Reachability analysis is computationally expensive, and

the approximation necessary for its scalability weakens its conclusiveness; resource bounds weaken the conclusiveness of even complete techniques such as variable-depth unique-state induction [5] on larger designs.

In this paper we propose a novel approach to general-purpose sequential redundancy removal. Similar to prior approaches [4, 3, 6], we compute a set of suspected redundancy candidates, and then create a speculatively-reduced model by replacing each fanout reference to a candidate gate by a reference to its *representative gate*, and by adding an exclusive-or (XOR) gate, hereafter referred to as a *miter*, over each candidate and its representative. Our novel contributions arise through the ways in which we utilize this model.

First, we propose the use of a larger and more robust variety of synergistic transformation and verification algorithms to solve the miters of the speculatively-reduced model. This is in contrast to prior work which relied upon a smaller set of algorithms, such as inductive [4, 5, 3] or approximate-reachability-based [3] fixed-point computation, refining the candidates based upon failed or inconclusive proof results. The speculative merging synergistically unlocks the reduction potential of subsequent transformations [7], which often result in dramatic further reductions in the size of that model. Certain transformations are particularly well-suited for trivializing the equivalence checking of subcircuits which differ only by commonly-used optimization techniques such as retiming and resynthesis. Furthermore, by applying a *distinct* algorithm flow on a *per-miter* basis, we may attain exponential speedups to the overall redundancy removal process, in addition to greater reduction potential. Such a framework allows maximal flexibility in the selection of which synergistic algorithms are best-suited for each individual miter. We have found this application of transformation-based verification (TBV) [8] to be extremely powerful, enabling efficient and scalable sequential equivalence checking across a variety of design modifications, and proofs in property checking, on large designs with 10,000s of state elements for which previous approaches fail.

Second, we discuss a more general paradigm wherein we may utilize this model for enhanced falsification via incomplete search (e.g., random and symbolic simulation, emulation) without a need to prove any of the miters unreachable. This approach allows us to exploit *even redundancy that holds only for an initial bounded time-frame, but not across all time-frames*, to enable exhaustive bounded search many times faster, and also deeper, than possible on the original design. We also discuss novel optimizations to the candidate guessing and refinement processes.

The rest of this paper is organized as follows. In Section 2 we review sequential redundancy removal basics. In Section 3, we discuss the speculatively-reduced model. In Section 4.1, we discuss the use of this model for traditional redundancy removal applications. In Section 4.2, we discuss the more general use of this model to falsify properties without first discharging any miters. In Section 5, we provide experimental results to illustrate the power of these techniques. In Section 6, we conclude this paper.

---

[1]The use of combinational redundancy removal techniques for sequential equivalence checking often requires the two designs to share the same state encoding, and correlates the state elements [2] in addition to primary inputs.

Figure 1: Generic redundancy removal algorithm

2 Preliminaries

We represent our design as a *netlist*, which is a tuple $\langle \langle V, E \rangle, G \rangle$ comprising a directed graph with vertices V and edges $E \subseteq V \times V$. Function $G : V \mapsto types$ represents a mapping from vertices to gate *types*, including constants, primary inputs, registers, and combinational gates with various functions. Registers have designated *initial values*, which may be arbitrary combinational functions. The *semantics of a netlist* are defined in terms of semantic *traces*: $0, 1$ valuations to gates over time which are consistent with G.

In a property checking environment, certain gates are labeled as *targets*, where the verification goal is to obtain a trace illustrating an assertion of a target, or to prove that no such trace exists. In an equivalence checking environment, one will build a composite netlist as the union of the two individual netlists, though replacing the primary inputs of one by the corresponding primary inputs of the other, and adding XOR targets (miters) over each pair of gates to be proven as equivalent between the two designs – typically gates that correlate to primary outputs.

Given a netlist, redundancy removal frameworks such as those of [4, 3] operate as per the algorithm of Figure 1. There are two potential causes of failure to prove a miter as unreachable in step 4. First, some of the candidates may be incorrect, i.e., not truly equivalent in all reachable states. This is reflected by the generation of a trace asserting the corresponding miters. Second, resource limitations (or an incomplete proof technique) may preclude the solution of a subset of the miters. For example, induction may become computationally expensive after a particular depth, hence miters that are not provable within that threshold will remain unsolved. Discarding miters due to resource limitations not only immediately prunes the equivalence classes, but also tends to result in a significant amount of future resource-gated refinements. This is because, by diminishing the amount of speculative merging, each refinement effectively weakens the redundancy *induction hypothesis* – requiring greater resources to complete proofs across refinements.

To illustrate this problem, assume that we wish to perform sequential equivalence checking over primary outputs for two versions of the netlist depicted in Figure 2. Only the darkest-shaded portion of the circuit was redesigned, hence each gate outside of this region will have a unique correspondent in the modified design. Only those miters over gates which have a source edge from the redesigned region will initially be nontrivial; the others will be trivially of the form $g \not\equiv g$ due to the speculative merging. If any of these cannot be solved due to resource limitations, the subsequent refinement will cause a set of gates in the immediate fanout of the refined gates to become nontrivial. Since their fanin gates were too difficult to prove equivalent, these gates also are likely to be too difficult to prove. This refinement ripples forward, increasing the resources needed for the subsequent proofs, pushing more
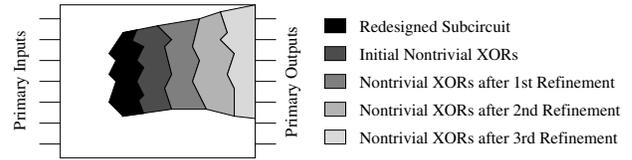

Figure 2: Failed proofs increase post-refinement complexity

miters from provable to unsolvable. Ultimately, many gates in the shaded region may remain unsolved, including some of the primary outputs which are the goal of the equivalence check.

3 The Speculatively-Reduced Model

In this section we discuss the speculatively-reduced model, constructed as per step 3 of the algorithm of Figure 1.[2] Our framework requires *all gates to be equivalent to their representatives in the initial states*, and *the selected representatives to be chosen arbitrarily among the combinationally-shallowest gates in the equivalence classes* to prevent combinational loops in the speculatively-reduced model. Theorem 1 provides the theoretical justification of how we may utilize the speculatively-reduced model both for proofs and falsification, as will be discussed in Section 4.

Theorem 1. Assume that we apply the same sequence of test vectors to the corresponding inputs of the original and the speculatively-reduced model. This simulation process will expose its first mismatch within the equivalence classes in the original design at time i if and only if it first asserts one or more miters in the speculatively-reduced model at time i.

Proof. We prove this theorem by induction on the length of the trace produced by the simulation run.

Base Case: In our framework, all gates are equivalent to their representatives in the initial states. This implies that at time 0, all redundancy candidates are correct – hence the speculative merging does not alter any gate's valuation in the trace at time 0, and no miter assertions may occur at time 0.

Inductive Step: By the induction hypothesis, assume that no mismatches nor miter assertions occured at times $0, \ldots, j$. We prove that the theorem holds at time $j + 1$. If no mismatches are exposed, and no miters are asserted, this trend trivially holds. Otherwise, first consider the case that at time $j + 1$, there is a nonempty set of gates A which each differ from their representatives in the trace over the original design. Consider the subset $B \subseteq A$ such that for every $b \in B$, neither b nor $R(Q(b))$ contain any other elements of A in their combinational fanin cones – the set of gates which may be reached fanin-wise without traversing through a register. Clearly B is non-empty assuming the netlist is free of combinational cycles, since we select representatives as the combinationally shallowest gates in their classes. Note that any speculative merging in the fanin cone of $b \in B$ and $R(Q(b))$ cannot alter their valuation at time $j + 1$, since that merging was correct for all such gates (other than those in B themselves) at times $0, \ldots, j + 1$. Therefore, if the simulation exposes a mismatch on set B in the original design at time $j + 1$, the miters T_B must be asserted in the speculatively-reduced model at time $j + 1$. By the same argument, if miters T_A are asserted in the speculatively-reduced model at time $j + 1$, we may similarly compute a nonempty subset $B \subseteq A$ for which the simulation must expose a mismatch in the original design at time $j + 1$. □

---

[2] If both g and $R(Q(g))$ are registers, we instead create the miter over their next-state functions to reduce the size of the speculatively-reduced model, similarly to [3]. Our technique also includes *antivalent* gates – which evaluate to opposite values in all reachable states – in the equivalence classes, similarly to [4].

3.1 Guessing Redundancy Candidates

A variety of techniques have been proposed to guess redundancy candidates. *Non-functional* approaches approximate the set of truly redundant gates using name- and structure-based comparisons [2], which are useful in equivalence checking applications since often only a fraction of the circuit is redesigned. However, they are often not applicable for property checking, and tend to break down within substantially redesigned subcircuits. *Functional* approaches [2] overapproximate the set of redundant gates by analyzing traces produced by random simulation and approximate symbolic search, and are useful in all redundancy removal applications.

Techniques such as [4, 3, 5] attempt to yield the exact set of truly redundant gates, beginning the process of Figure 1 with an overapproximate set of candidates, and relying upon refinements to yield a maximal set of candidates which may all be proven equivalent. Given adequate resources and an underlying complete proof technique (such as unique-state variable-depth induction [5]), these approaches will yield the set of all redundant gates as a *byproduct* of their proof process. However, the set of algorithms used in these approaches is incomplete in practice; the final set of redundancy candidates they yield on more complex problems may thus be a proper subset of those which are truly redundant, which is precisely the weakness we address in Section 4.1.

All prior work in guessing candidates is applicable in our framework. One novel candidate guessing technique we have developed is discussed in Section 4.2, exploiting the enhanced transformation and falsification ability on the speculatively-reduced model.

3.2 Refining the Redundancy Candidates

If a miter cannot be proven unreachable, its equivalence class must be refined to separate the corresponding candidate from its representative. As discussed in Section 2, there are two causes of such failed proofs: a miter is asserted (i.e., a trace is generated which differentiates the corresponding candidates), or a miter cannot be solved within the available resource limits.

As per Theorem 1, a trace asserting any of the miters in the speculatively-reduced model must differentiate at least one pair of redundancy candidates in the original design. Note, however, that the speculative merging may cause certain gates in the fanout of incorrectly-merged gates to either mismatch when they should not, or to not mismatch when they should. An effective way to determine precisely which candidates have been differentiated by the corresponding miter-hitting trace is to simulate the original design with the input sequence illustrated in that trace. By additionally injecting random stimulus to any don't-cares therein and extending the sequential length of that trace, we may obtain a useful set of patterns from which we may refine all candidates. However, if the processing of a trace is not possible, the proof of Theorem 1 nonetheless indicates a minimal subset of gates $B \subseteq A$ which must be refined given only the knowledge of the set A of gates whose miters have been asserted.

For miters which cannot be proven due to resource limitations, we may again utilize the proof of Theorem 1 to choose a minimal subset of candidates to refine by taking A to be the set of miters which could not be proven unreachable. However, this subsetting tends to be of lesser practical utility since the gates in the fanout of this refined subset will tend to become even more difficult to prove after the refinement as illustrated in Figure 2, unless the subset B includes truly incorrect candidates whose incorrectness was the cause of the difficulty of proving $A \setminus B$.

4 Exploiting the Speculatively-Reduced Model

In this section we address the various ways in which the speculatively-reduced model may be utilized for enhanced verification. First, this model may be utilized to simplify the proof obligation inherent in the use of this technique as a "redundancy removal engine." Second, this reduced model may be utilized to attempt to falsify targets, without any need for a prior proof obligation to enable the reductions already reflected in the model.

4.1 Redundancy Removal Applications

As follows from Theorem 1, in redundancy removal applications, we wish to prove that each of the miters is unreachable. If we succeed, each of the redundancy candidates used in the construction of the speculatively-reduced model is correct - hence we may perform the corresponding merging in a sound and complete manner. Unlike prior approaches to sequential redundancy removal which rely upon induction or approximate reachability to discharge the miters, we propose a more general scheme which leverages arbitrary synergistic transformation-based verification (TBV) [8] flows to solve the resulting miters. This approach has several benefits.

First, TBV often substantially reduces the size of the speculatively-reduced model, and the number of distinct miters therein. This tends to significantly reduce resource requirements, regardless of the proof techniques used to discharge the miters.

Second, the transformations themselves are sufficient to solve many of the miters. For example, given two subcircuits which differ only by retiming and resynthesis, polynomial-resource retiming and redundancy removal engines [7] alone are often able to trivialize the resulting equivalence checking problem without a need for more costly inductive analysis. We note that several past researchers have proposed techniques explicitly intended to simplify the verification of retiming- and resynthesis-optimized designs. For example, in [9] a technique is proposed for extracting a *retiming invariant* to be inductively discharged more simply than could direct equivalence candidates. Such solutions are somewhat complementary to ours, in that our technique could be used to prove the resulting simplified proof obligation. However, we have found that our approach tends to be more robust, and a practical superset, of the prior research in several ways. For example, such approaches tend to require a preprocessing step on a rigidly-transformed intermediate version of the design, e.g., one derived only through retiming with limited resynthesis. In contrast, many design evolutions intertwine such transformations [10] along with a variety of other (often manually-performed) transformations such as state-machine re-encoding. Overall, our TBV approach is able to efficiently discharge these simpler retiming and resynthesis subproblems without dedicated preprocessing, and also more generally scales to efficiently compensate for more aggressive design modifications.

The third, and possibly greatest, benefit of the use of TBV in sequential redundancy removal comes through the ability to leverage independent algorithm flows on each individual miter. Different transformation and verification algorithms are better-suited for different problems – often exponentially so [7]. Even the most complex miter may often be rendered sufficiently small to be reliably solved using a variety of proof techniques, instead of relying solely upon possibly inconclusive induction or approximate analysis.

One particularly pronounced benefit we have noted lies in the use of localization in the per-miter transformation flows. Figure 3 illustrates an example problem, where only a portion of the two cones has actually been redesigned. Each gate in the non-shaded region

(a) Original designs.
Redesigned subcircuit darkly shaded.

(b) Speculatively-reduced model.
Localized circuit needed to prove
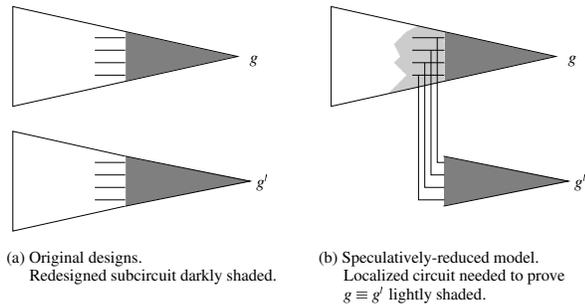$g \equiv g^l$ lightly shaded.

Figure 3: Use of localization to prune speculatively-reduced model

of the top design of Figure 3a has a unique correspondent in the bottom design, resulting in trivial miters. Gate g has g^l as a correspondent, resulting in a nontrivial miter over the two darkly shaded regions plus the original cone driving one of those regions as depicted in Figure 3b. In cases, the redesign may not rely upon any constraints in its fanin cone to ensure this equivalence, hence localization may require no gates outside of the darkly shaded region. Otherwise, localization may further reduce the amount of logic from that cone needed to discharge the miter as depicted by the lightly shaded region in Figure 3b.[3] This localization often prunes the resulting subproblem to tens or (few) hundreds of registers, regardless of the size of the corresponding cone. Localization also tends to greatly unlock the reduction potential of other subsequent transformations such as retiming and parametric re-encoding [7].

It is noteworthy that the benefits of our technique to enhance redundancy removal cannot be fully realized without the ability to process the speculatively-reduced model using TBV flows. This is partially because certain methodologies may limit the type of transformations which may be applied prior to the redundancy removal engine. For example, equivalence checking frameworks may use name- and structure-based comparisons to guess candidates [2]. Prior transformations such as retiming with intertwined resynthesis [10] may render such comparisons ineffective. More generally, this is because the speculative merging often enables a pronounced synergistic increase in the reduction potential of other transformations. Thus, TBV flows are more effective *after* the speculative merging, hence applying such flows only *prior* to the speculative merging will not yield as great of reductions. Furthermore, as illustrated in Figure 2, on the most complex problems it is imperative to leverage as robust a variety of transformation and verification algorithms as possible to ensure that the truly correct candidates may be proven equivalent; otherwise, an avalanche of resource-gated refinements may occur, resulting in suboptimal merging. On such complex problems, we have found that applying TBV only to the suboptimally-merged netlist (e.g., derived using induction alone to discharge the miters) cannot compensate for the strength of applying TBV to solve all miters of the speculatively-reduced model.

4.2 Falsification Applications

Theorem 1 implies that we may perform incomplete search upon the speculatively-reduced model, and provided that none of the miters are asserted during that search, the verification results obtained during that effort are directly valid for the corresponding

targets on the original design. This basically casts the miters from being *assumption checkers* [3] for a redundancy removal proof to being *filters* through which to confirm that any analysis performed upon the speculatively-reduced model remains within the state-space for which the redundancy candidates are correct – even if they are not correct in all reachable states. Note that such an incomplete application is a generalization of the traditional use of these miters for redundancy-removal proofs; any proven miter may be safely discarded, as it cannot *invalidate* the analysis performed on the speculatively-reduced model.

For example, if we apply a sequence of test vectors to both models, and none of the miters in the speculatively-reduced model are asserted during that simulation, a target in the reduced model will be asserted if and only if that target is asserted in the original design under that simulation. This observation in turn implies that any number of symbolic evaluation steps – whether exact or underapproximate – performed without asserting any of the miters preserves the results obtained upon the targets during that analysis. This result allows us to exploit *even redundancy that holds only for an initial bounded time-frame, but not across all time-frames*, to leverage the speculatively-reduced model for applications such as increasing the depth to which bounded falsification may be performed on the targets.[4] This result also allows us to construct and use the speculatively-reduced model in alternative proof-incapable frameworks such as simulators and hardware emulators. We have found several applications for this theory.

First, we have found many cases where we could perform bounded falsification on targets many times faster, and also deeper, on the speculatively-reduced model than on the original design. Second, we have found this approach useful in the candidate guessing process; after preliminary guessing via low-cost analysis of the original design, we build a speculatively-reduced model and apply a sequence of transformations to further reduce that model, and then apply more extensive semi-formal analysis on that reduced model to further attempt to differentiate the candidates.

Our third application allows us to reuse the knowledge that a and b cannot be differentiated for times $0, \ldots, i$ across refinements. In particular, if the equivalence class containing a and b is unaltered, we may immediately infer that the XOR of a and b cannot be asserted for times $0, \ldots, i$ regardless of the refinement of any other classes. Furthermore, if we refine the equivalence class containing a and b resulting in miters (a XOR c) and (b XOR d), we may immediately infer that these two new miters cannot be asserted for times $0, \ldots, i$. This optimization holds because refinements only split a class into several new classes, but never group gates from previously-incompatible classes. Practically, this application serves two goals. First, it enables us to reuse the discharging of the induction hypothesis from prior proof attempts to speed up later ones. Second, one may wish to intermix semi-formal analysis to assert miters with proof analysis to demonstrate their unreachability; this optimization helps reuse falsification effort across refinements. In [6], it is noted that one need not re-prove any miters that had no refined gates in their fanin cones because the prior proof is guaranteed valid after the refinement. We may extend our third application to generalize that of [6] by noting that the assertion of a miter over a and $R(Q(a))$ at time i only risks invalidating results beyond time i for miters which contain a in their fanin cones.

---

[3]TBV-based localization may be viewed as a generalization of using design hierarchy boundaries [11] or speculative merge points [3] to insert cutpoints. TBV offers greater reduction potential through localization since it is not limited to specific boundaries, and offers more robust refinement algorithms (e.g., [12]) in case the chosen cutpoints result in spurious mismatches. TBV additionally enables a robust variety of synergistic transformations before and after localization, unlike prior approaches.

[4]A noteworthy dual of this reduction is that of [13], allowing unfoldings to exploit redundancies that are proven to hold only *after* a certain number of time-steps. TBV will yield such a reduction only if a technique such as retiming sufficiently skews those gates to enable their merging; refer to the discussion of Table 1.

5 Experimental Results

All experiments were run on an IBM RS/6000 Model 681, with a 1.1 GHz POWER4 Processor using the IBM internal verification tool *SixthSense*. As our solution is more general than previous approaches, we have chosen our experiments from difficult cases for which induction alone – even with variable-depth unique-state constraints [5] – was inadequate to solve the targets. The engines used in the TBV flows include the following.

- **COM**: a redundancy removal engine which uses combinational techniques such as structural hashing and resource-bounded BDD- and SAT-based analysis to identify gates which are functionally redundant across all states [1].
- **RET**: a min-area retiming engine [8], which reduces the number of registers by shifting them across combinational gates.
- **CUT**: a range-preserving parametric re-encoding engine [14], which replaces the fanin-side of a *cut* of the netlist graph with a trace-equivalent, yet simpler, piece of logic.
- **LOC**: a localization engine, which isolates a cut of the netlist local to the targets by replacing gates by primary inputs. **LOC** is an overapproximate transformation, and uses a SAT-based refinement scheme [12] to prevent spurious counterexamples.
- **SAT**: a structural SAT solver [1], which interleaves redundancy removal with BDD- and SAT-based analysis, hence correlates to the multi-algorithm solution used in [6].
- **RCH**: a symbolic reachability engine.
- **IND**: a SAT- and BDD-based induction engine.
- **EQV**: a sequential redundancy removal engine using the techniques proposed in this paper.

We present three sets of experiments in Table 1. The first set consists of industrial sequential equivalence checking examples, where manually redesigned circuits are compared against the original circuit. The redesigns include a variety of techniques, from retiming and resynthesis to replication of subcircuitry for reduced propagation delay to outright re-encoding of portions of the design. FPU is a floating-point unit; IFU is an instruction-fetch unit; and SDQ is an issue queue. The second set consists of difficult industrial invariant checking problems. IOC is an I/O controller; SLB is an address translation unit; SMM is a memory management unit. The third set consists of sequential equivalence checking problems from IS-CAS89 benchmarks, where the original circuits are compared to the result of applying SIS kerneling, retiming, and *script.rugged* [4].

The first column indicates the name of the corresponding design and the size metric being tracked in the corresponding row. All designs are mapped onto a netlist representation containing only constants, primary inputs, two-input AND gates, inverters, and registers, using straight-forward logic synthesis techniques. The second column reflects the size of the target cones of the original, unreduced verification problem; phase abstraction [15] was used to preprocess the industrial examples, and simple structural hashing was used to simplify them all. The sequential equivalence checking targets represent miters over corresponding primary outputs. The third column indicates the size of the speculatively-reduced model; this target count reflects the number of distinct, nontrivial miters added to validate the redundancy. Note that the miters are a superset of the initial targets, each in an equivalence class with the *constant zero* gate. The fourth column is the size of the original design after merging the gates proven equivalent using only induction.[5] The fifth column is the size after merging the gates proven

---

Name / Metric	Original Design	Speculatively-Reduced Model	After Merging via Induction	After Merging via TBV
FPU			1485 s / 453 MB	748 s / 451 MB[1]
Registers	22988	11504	21702	11504
ANDs	314500	160477	305000	159879
Targets	263	263	258	0
IFU			1060 s / 460 MB	1710 s / 497 MB[2]
Registers	66421	33231	58643	33231
ANDs	588090	304990	47145	296950
Targets	756	1068	718	0
SDQ			355 s / 62 MB	283 s / 178 MB[3]
Registers	3665	1833	3473	1833
ANDs	21855	10924	21211	10803
Targets	309	199	261	0
IOC			3170 s / 918 MB	2698 s / 890 MB[4]
Registers	29607	241	2354	241
ANDs	213929	3777	17807	3000
Targets	35	301	23	0
SLB			41 s / 61 MB	41 s / 88 MB[5]
Registers	6886	139	143	139
ANDs	40374	873	865	845
Targets	3	71	3	0
SMM			842 s / 237 MB	4060 s / 229 MB[6]
Registers	2460	1284	2166	1284
ANDs	70938	62108	69314	60318
Targets	1	567	1	0
S3271 [4]			20s / 43 MB	34 s / 68 MB[7]
Registers	305	249	239	213
ANDs	2207	2446	2118	2079
Targets	14	110	13	0
S3384 [4]			2190 s / 55 MB	3088 s / 311 MB[8]
Registers	689	406	412	383
ANDs	2297	2258	2223	2159
Targets	26	494	25	0
S6669 [4]			35 s / 46 MB	24 s / 94 MB[9]
Registers	506	325	321	272
ANDs	4460	3992	4056	3981
Targets	55	256	17	0

Table 1: Redundancy removal results. TBV configurations used within EQV on the speculatively-reduced model:
[1]COM,LOC,CUT,COM,RCH; [2]COM,LOC,CUT,COM,LOC,CUT,COM,RCH;
[3]COM,LOC,CUT,COM,LOC,CUT,COM,LOC,CUT,COM,RCH;
[4]COM,RET,COM,IND; [5]COM,LOC,COM,RET,COM,CUT,COM,IND,RCH;
[6]COM,RET,COM,IND,LOC,COM,CUT,COM,IND,RCH; [7]COM,RET,COM,EQV;
[8]COM,RET,COM,CUT,COM,EQV; [9]COM,CUT,RET,CUT,COM,SAT

equivalent using TBV flows, after a low-cost induction preprocessing to eliminate the easier miters. The reported resources include all aspects of the verification process, including candidate guessing.

These experiments clearly demonstrate that the TBV approach results in significantly greater merging than possible with induction alone, and ultimately completes all unreachability proofs even though induction alone fails on most. To our knowledge, we are the first to report a method for solving all primary outputs as equivalent in these particular S3384 and S6669 benchmarks [4]. Additionally, TBV is often faster than induction alone. Most of the TBV flows ultimately relied upon localization and reachability analysis to solve the more complex miters. Several exploited the ability of transformations such as retiming to enhance the inductiveness of targets [7]. Two flows terminated in a recursive application of sequential redundancy removal, capturing redundancy which heuristically arose through min-area retiming. In particular, certain gates that were equivalent modulo a time skew [16] were retimed such that they became truly equivalent, hence could be merged.

To illustrate the power of TBV in processing the speculatively-reduced model, we detail several TBV results in Table 2, where the columns indicate the size of the problem *after* the corresponding transformation engine (indicated in the top row) was run. Note how the transformation engines synergistically reduce the size of the netlist [7]. For IFU, localization renders three orders of magnitude reduction on the size of one of the miters. For S3384, re-

IFU	Initial	COM	LOC	CUT	COM	
Registers	33231	30362	19	19	19	
ANDs	304990	276795	86	76	71	
Inputs	1371	1329	23	10	10	
Targets	1	1	1	1	1	
S3384	**Initial**	**COM**	**RET**	**COM**	**CUT**	**COM**
Registers	406	362	66	66	66	66
ANDs	2236	1735	3019	1966	1830	1723
Inputs	43	17	33	33	31	31
Targets	38	37	37	32	32	31
S6669	**Initial**	**COM**	**CUT**	**RET**	**CUT**	**COM**
Registers	325	186	138	0	0	0
ANDs	3992	3067	1747	2186	1833	1788
Inputs	83	61	40	40	24	24
Targets	17	16	16	16	16	15

Table 2: TBV results on speculatively-reduced model

Name	Bounded Steps Completed			Col. 4 Time to Surpass Col. 2 Depth
	Original Design	Original with Miters	Spec. Reduced with Miters	
FPU	16	12	23	584 s
IFU	8	5	10	595 s
SDQ	18	12	94	22 s
IOC	75051	3226	679575	283 s
SLB	3272	2163	3731	2646 s
SMM	210	199	315	1310 s
S3271	119	44	135	2508 s
S3384	70	39	345	117 s
S6669	84	25	243	1056 s

Table 3: Search depths reached within one hour

timing and resynthesis alone solve nearly 20% of the miters, and reduce register count by more than 80%. For S6669, retiming transforms the sequential netlist into a combinational one, enabling us to use exclusively combinational techniques such as SAT to solve the miters, which otherwise were not inductive. Without the prior CUT, there are retiming traps which preclude retiming from rendering a combinational netlist. Without applying TBV to the speculatively-reduced model, we could not find a sequence of transformations exclusively before or after the (inductive) redundancy removal engine which yielded a combinational or inductive problem.

To illustrate the advantage of utilizing the speculatively-reduced model for incomplete search, we performed experiments to assess the effectiveness of SAT-based bounded analysis which are detailed in Table 3. We ran independently on the targets of the original design (column 2); on the miters validating the equivalence classes on the original design *without* speculative merging, e.g., if using such bounded analysis to guess the redundancy candidates (column 3); and on the corresponding miters of the speculatively-reduced model after a sequence of inexpensive transformations (column 4). Recall that the miters are a superset of the original targets, and that our SAT-solver integrates redundancy removal algorithms; we additionally ran COM on the designs of columns 2 and 3 prior to unfolding. These columns indicate the bounded depth reached within a one-hour time limit, inclusive of all transformations. The fifth column indicates how long it took to reach the depth of the second column using the speculatively-reduced model. As expected, the third column attains a lesser depth than the second, illustrating how SAT-based candidate guessing tends to degrade due to the increased number of targets. Note that the fourth column enables orders of magnitude deeper miter validation in cases, in addition to significantly deeper validation of the original targets. The fifth column further illustrates the speed benefit of the fourth.

6 Conclusions

We have presented several improvements to sequential redundancy removal. First, we have generalized upon traditional inductive and approximate-reachability-based approaches by using a flexible and synergistic set of transformation and verification algorithms to discharge the subproblems. This enables greater speed and also greater reduction potential, since the transformations sufficiently simplify even the most complex subproblems to enable low-cost proofs. Second, we have more generally exploited *suspected* redundancy by constructing a speculatively-reduced model which may be used for enhanced falsification with no need for completing any proofs. We have also presented improvements to the redundancy candidate guessing and refinement procedures. Experiments demonstrate the efficiency and scalability of our techniques, en-

abling sequential equivalence checking across a variety of design modifications, and enhanced proofs as well as falsification in property checking, on large designs for which previous approaches fail.

References

[1] A. Kuehlmann, V. Paruthi, F. Krohm, and M. Ganai, "Robust Boolean reasoning for equivalence checking and functional property verification," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 12, 2002.

[2] H. Cho and C. Pixley, "Apparatus and method for deriving correspondences between storage elements of a first circuit model and storage elements of a second circuit model," in *U.S. Patent 5,638,381*.

[3] S.-Y. Huang, K.-T. Cheng, K.-C. Chen, C.-Y. Huang, and F. Brewer, "AQUILA: An equivalence checking system for large sequential designs," *IEEE Transactions on Computers*, vol. 49, May 2000.

[4] C. A. J. van Eijk, "Sequential equivalence checking without state space traversal," in *Design,Automation,and Test in Europe*, Mar. 1998.

[5] P. Bjesse and K. Claessen, "SAT-based verification without state space traversal," in *FMCAD*, Nov. 2000.

[6] K. Ng, M. R. Prasad, R. Mukherjee, and J. Jain, "Solving the latch mapping problem in an industrial setting," in *DAC*, June 2003.

[7] H. Mony, J. Baumgartner, V. Paruthi, R. Kanzelman, and A. Kuehlmann, "Scalable automated verification via expert-system guided transformations," in *FMCAD*, Nov. 2004.

[8] A. Kuehlmann and J. Baumgartner, "Transformation-based verification using generalized retiming," in *CAV*, July 2001.

[9] M. Mneimneh and K. Sakallah, "REVERSE: Efficient sequential verification for retiming," in *Int'l Workshop on Logic & Synthesis*, 2003.

[10] J. Baumgartner and A. Kuehlmann, "Min-area retiming on flexible circuit structures," in *ICCAD*, Nov. 2001.

[11] D. Anastasakis, L. McIlwain, and S. Pilarski, "Efficient equivalence checking with partitions and hierarchical cut-points," in *DAC*, 2004.

[12] D. Wang, *SAT based Abstraction Refinement for Hardware Verification*. PhD thesis, Carnegie Mellon University, May 2003.

[13] A. Kuehlmann, "Dynamic transition relation simplification for bounded property checking," in *ICCAD*, Nov. 2004.

[14] I.-H. Moon, H. H. Kwak, J. Kukula, T. Shiple, and C. Pixley, "Simplifying circuits for formal verification using parametric representation," in *FMCAD*, Nov. 2002.

[15] J. Baumgartner, T. Heyman, V. Singhal, and A. Aziz, "An abstraction algorithm for the verification of level-sensitive latch-based netlists," *Formal Methods in System Design*, no. 23, 2003.

[16] S.-Y. Huang, K.-T. Cheng, and K.-C. Chen, "On verifying the correctness of retimed circuits," in *Great Lakes Symp. on VLSI*, Mar. 1996.