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. **(1)** We propose using a robust variety of synergistic transformation and verification algorithms to process the individual proof obligations. This enables greater speed and scalability, and identifies a significantly greater degree of redundancy, than prior approaches. **(2)** We generalize upon traditional redundancy removal and utilize the speculatively-reduced model to enhance bounded search, without needing to complete any proofs.

Categories and Subject Descriptors: B.6.3 [Design - Aids]: Verification

General Terms: Verification

Keywords: sequential redundancy removal, sequential equivalence checking, correctness-preserving transformations.

1. INTRODUCTION

Combinational redundancy removal techniques [1] are too weak to identify gates which are redundant in all reachable states, but do not appear redundant in certain unreachable states. Several approaches have been proposed to compensate for this weakness, e.g., via approximate reachability [2] or induction [3]. However, some inconclusive results – which must be conservatively treated as mismatches – are still a practical inevitability. 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 [4].

In this paper we propose a novel approach to general-purpose sequential redundancy removal. Similar to prior approaches [3, 2, 5], 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 contributions arise through the ways in which we utilize this model.

```
1. Guess the redundancy candidates – sets of equivalence classes
   of gates, where each gate $g$ in equivalence class $Q(g)$ is suspected
   to be equivalent to every other gate in the same class.
2. Select a representative gate $R(Q(g))$ from each equivalence class.
3. Construct the speculatively-reduced model by replacing the
   source gate $g$ of every edge $(g,h) \in E$ by $R(Q(g))$. Additionally,
   for each gate $g$, add a miter $T_g$ over $g$ and $R(Q(g))$.
4. Attempt to prove that each of the miters is unreachable.
5. If any miters cannot be proven, refine the equivalence classes to
   separate the corresponding gates; go to Step 2.
6. All miters have been proven unreachable; the equivalence classes
   reflect true redundancy, hence may be merged.
```

Figure 1: Generic redundancy removal algorithm

First, we propose the use of a larger and more robust variety of synergistic transformation and verification algorithms to solve the miters. The speculative merging unlocks the reduction potential of subsequent transformations [6], which often result in dramatic further reductions in the size of that model. Certain transformations may trivialize 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 dramatic improvements to the overall process. We have found this application of transformation-based verification (TBV) [7] to be extremely powerful, enabling scalable sequential equivalence checking across a variety of design modifications, and proofs in property checking, on large designs for which previous approaches fail. Second, we discuss a more general paradigm wherein we may utilize this model for enhanced falsification via incomplete search without a need to prove any of the miters unreachable. This allows us to exploit *even redundancy that holds 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.

2. PRELIMINARIES

We represent our design as a *netlist*, comprising a directed graph with vertices V and edges E, and a mapping G from vertices to gate *types* (including primary inputs, registers, and combinational gates with various functions). Registers have designated *initial values*, which may be arbitrary combinational functions. A *trace* is an assignment of $0, 1$ valuations to gates over time consistent with G.

In property checking, 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 equivalence checking, one will build a composite netlist as the union of the two individual netlists, replacing the primary inputs of one by the corresponding primary inputs of the other. In addition, targets (miters) are added over each pair of gates to be proven as equivalent between the two designs – typically gates that correlate to primary outputs.

Redundancy removal frameworks [3, 2] operate as per the algorithm of Figure 1.[1] There are two potential causes of failure to prove a miter as unreachable in Step 4. First, some of the candidates may not be 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. 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 reducing the amount of speculative merging, each refinement effectively weakens the redundancy *induction hypothesis* – requiring greater proof resources across refinements.

Theorem 1. *Assume that we apply the same sequence of test vectors to the original and the speculatively-reduced model. Let i be the earliest time at which a mismatch is exposed within the equivalence classes, between gates A and their respective representatives. Consider the non-empty subset $B \subseteq A$ such that for every $b \in B$, neither b nor $R\big(Q(b)\big)$ contain any other elements of A in their combinational fanin. This simulation process will necessarily also assert miters T_B in the speculatively-reduced model at time i. Conversely, if miters T_A are asserted at time i, there must exist a non-empty subset $B \subseteq A$ for which the simulation exposes a mismatch in the original design at time i.*

Guessing Redundancy Candidates. Various techniques have been proposed to guess redundancy candidates; e.g., using traces produced by random simulation and approximate symbolic search. All prior approaches are applicable in our framework. In Section 3, we discuss a novel candidate guessing technique, exploiting the enhanced transformation and falsification ability on the speculatively-reduced model.

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. However, 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 such a 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 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, Theorem 1 nonetheless indicates a minimal subset of gates $B \subseteq A$ which must be refined given only the set A of gates whose miters have been asserted.

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

For miters which cannot be proven due to resource limitations, Theorem 1 implies 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.

3. EXPLOITING THE REDUCED MODEL

Redundancy Removal Applications. In redundancy removal applications, we wish to prove that each of the miters is unreachable. If we succeed, we may perform the corresponding merging in a sound and complete manner. Unlike prior approaches 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) [7, 6] 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 reduce resources, regardless of the 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 [6] alone are often able to trivialize the resulting miters without a need for more costly inductive or reachability analysis. Several techniques have been proposed to explicitly simplify the verification of such design optimizations; e.g., [8, 9]. 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 [6]. 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 pronounced benefit we have noted lies in the use of localization in the per-miter transformation flows. Consider an equivalence checking problem where only a portion of the netlist has been redesigned. Each gate in the non-redesigned portion of the first netlist has a unique correspondent in the second. Only those miters over gates which have a source edge from the redesigned portion will be non-trivial. The redesigned logic may in cases not rely upon any constraints in its fanin cone to ensure the equivalence of these miters, in which case localization will extract only the redesigned portion. Otherwise, in addition to the redesigned portion, localization will extract only that logic which is needed to establish equivalence of the miters.[2] Localization often prunes such subproblems 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 [6].

Note 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 methodological; e.g., equivalence checking frameworks may use name- and structure-based comparisons to guess candidates. 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 most effective when applied *after* the speculative merging. Furthermore, 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. We have found that applying TBV only to the suboptimally-merged netlist (e.g., derived using induction alone) cannot compensate for the strength of applying TBV to solve all miters of the speculatively-reduced model.

Falsification Applications. By Theorem 1, we may perform incomplete search on 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 generalizes the miters from being *assumption checkers* [2] 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 candidates are correct – even if they are not correct in all reachable states.

As an example, any number of symbolic evaluation steps performed without asserting any of the miters preserves the results obtained for 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.*[3] We leverage the speculatively-reduced model for the following types of falsification applications.

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.

Third, we may 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 other refinements. Otherwise, if we refine this equivalence class resulting in miters (a XOR c) and (b XOR d), we may infer that these two new miters cannot be asserted for times $0, \ldots, i$ because refinements only split a class into several new classes, but never group gates from previously-incompatible classes. This application enables us to reuse the discharging of the induction hypothesis across refinements. Also, one may intermix semi-formal analysis to assert miters with proof analysis to assess their unreachability; this optimization allows reuse of falsification effort across refinements. In [5], 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 valid after the refinement. We may generalize upon [5] by noting that the assertion of a miter over a and $R\big(Q(a)\big)$ at time i only risks invalidating results *beyond* time i for miters which contain a in their fanin cones.

[2]TBV-based localization is a generalization of using design hierarchy boundaries [11] or speculative merge points [2] to insert cutpoints. TBV offers greater reduction potential 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.

[3]This also allows us to use the speculatively-reduced model in proof-incapable frameworks such as simulators and hardware emulators.

Name / Metric	Original Design	Reduced Model	After Merge via Induction	After Merge via TBV
FPU			1485s / 453MB	748s / 451MB[1]
Registers	22988	11504	21702	11504
ANDs	314500	160477	305000	159879
Targets	263	263	258	0
IFU			1060s / 460MB	1710s / 497MB[2]
Registers	66421	33231	58643	33231
ANDs	588090	304990	47145	296950
Targets	756	1068	718	0
IOC			3170s / 918MB	2698s / 890MB[3]
Registers	29607	241	2354	241
ANDs	213929	3777	17807	3000
Targets	35	301	23	0
SMM			842s / 237MB	4060s / 229MB[4]
Registers	2460	1284	2166	1284
ANDs	70938	62108	69314	60318
Targets	1	567	1	0
S3384 [3]			2190s / 55MB	3088s / 311MB[5]
Registers	689	406	412	383
ANDs	2297	2258	2223	2159
Targets	26	494	25	0
S6669 [3]			35s / 46MB	24s / 94MB[6]
Registers	506	325	321	272
ANDs	4460	3992	4056	3981
Targets	55	256	17	0

Table 1: Results. TBV flows used on speculatively-reduced model within EQV: [1]COM,LOC,CUT,COM,RCH; [2]COM,LOC,CUT,COM,LOC,CUT,COM,RCH; [3]COM,RET,COM,IND; [4]COM,RET,COM,IND,LOC,COM,CUT,COM,IND,RCH; [5]COM,RET,COM,CUT,COM,EQV; [6]COM,CUT,RET,CUT,COM,SAT

4. EXPERIMENTAL RESULTS

All experiments were run on an IBM pSeries 7040-681, with a 1.1 GHz POWER4 Processor using the IBM internal verification tool *SixthSense*. The TBV engines used include:

- **COM**: a combinational redundancy removal engine[1].
- **RET**: a min-area retiming engine [7].
- **CUT**: a reparameterization engine [13].
- **LOC**: a SAT-based localization refinement engine [12].
- **RCH**: a symbolic reachability engine.
- **IND**: a SAT- and BDD-based induction engine.
- **EQV**: our sequential redundancy removal engine.
- **SAT**: a structural SAT solver [1], combining redundancy removal with BDD- and SAT-based analysis, correlating to the multi-algorithm solution in [5].

We present three sets of experiments in Table 1. The first consists of two industrial equivalence checking examples over manually redesigned circuits. The second consists of two difficult industrial invariant checking problems. The third consists of two equivalence checking problems from

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 on speculatively-reduced model

SIS-optimized ISCAS89 benchmarks [3]. 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; phase abstraction was used to preprocess the industrial examples [6]. The second column reflects the size of the target cones of the original, unreduced verification problem. The 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.[4] The fifth column is the size after merging the gates proven 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. 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 [3]. 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 [6]. One flow terminated in a recursive application of sequential redundancy removal, enabling the merging of gates that were equivalent modulo a time skew [9], which became truly equivalent through min-area retiming.

In Table 2, we detail the power of TBV in processing the speculatively-reduced model, 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 [6]. For IFU, localization renders three orders of magnitude reduction on the size of one of the miters. For S3384, retiming 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 an induction-based redundancy removal engine which yielded a combinational *or* inductive problem.

[4]The induction depths were chosen on a per-design basis to minimize runtimes. The initial induction was limited to one hour to find the maximum depth at which any of the miters was proven; the reported results bounded each induction run to that depth. We also disabled unique-state constraints [4] if they did not yield greater merging.

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
IOC	75051	3226	679575	283 s
SMM	210	199	315	1310 s
S3384	70	39	345	117 s
S6669	84	25	243	1056 s

Table 3: Search depths reached within one hour

We illustrate the advantage of utilizing the speculatively-reduced model for SAT-based bounded search in Table 3. We ran 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 analysis to guess the redundancy candidates (column 3); and on the 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; 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 deeper validation of the original targets.

5. REFERENCES

[1] A. Kuehlmann, V. Paruthi, F. Krohm, and M. Ganai, "Robust Boolean reasoning for equivalence checking and functional property verification," *TCAD*, Dec. 2002.

[2] 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 Trans. Computers*, May 2000.

[3] C. A. J. van Eijk, "Sequential equivalence checking without state space traversal," in *DATE*, Mar. 1998.

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

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

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

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

[8] M. Mneimneh and K. Sakallah, "REVERSE: Efficient sequential verification for retiming," in *IWLS*, May 2003.

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

[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*, June 2004.

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

[13] 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.