Towards Compositional Hardware Model Checking Certification

Emily Yu* emily.yu2019@gmail.com
Nils Froleyks* nils.froleyks@jku.at
Armin Biere† biere@cs.uni-freiburg.de
Keijo Heljanko‡§ keijo.heljanko@helsinki.fi

*Johannes Kepler University, Linz, Austria
†University of Freiburg, Freiburg, Germany
‡Helsinki Institute for Information Technology and
§University of Helsinki, Helsinki, Finland

***Abstract*—In this paper, we revisit and formalize temporal decomposition, as one of the most basic, widely-used and effective preprocessing techniques in hardware model checking. The main contribution is a certification framework for hardware model checking using temporal decomposition. Our approach enables generation of a single inductive invariant in a compositional way using inductive invariant certificates provided by existing certifying model checkers on the result of preprocessing a model through temporal decomposition. We implement and evaluate the method on hardware model checking competition benchmarks. The experiments confirm the effectiveness of temporal decomposition. The proposed certification approach makes it feasible to generate a generic proof for model checking and preprocessing.**

I. Introduction

The study of compositional reasoning for safety-critical systems can be traced back to a few decades ago [1]. Compositional model checking breaks the model checking procedure down into several smaller problems, thus enabling faster and more efficient verification. For example, preprocessing techniques are widely used in current industry in combination with standard model checking algorithms.

Among these, temporal decomposition [2] is considered important in industrial hardware model checking [3], [4]. It computes an over-approximation of reachable states to efficiently find a set of transient signals that stabilize to their constant values during any possible execution. A sequence of transformations is applied to the design under verification, including time-shifting it from the reset states and elimination of transient logic. The verification problem thus consists of verifying that the property holds within the time frame the design has been shifted, and that the transformed circuit is safe. For the latter an existing model checker that can provide certificates is employed, such as k-induction [5], symbolic model checking using BDDs [6], and IC3/PDR [7].

While progress in verification using compositional reasoning continues, the number of certifiable approaches are limited [8], [9]. One central objective of verification is to develop a *standardised* method to generate machine-checkable proofs for certifying model checking [9]. This is especially crucial in safety-critical industrial environments, as a faulty processor design can be extremely costly for a hardware manufacturer. Even though a number of single-engine model checkers are able to generate proofs, a key difficulty in this area is to produce a single generic certificate for complex verification pipelines. Furthermore, as in SAT solving [10]–[12], proof generation becomes rather involved when using preprocessing techniques, as the proofs from the preprocessors need to be lifted to proofs for the original model checking problem. This problem is exacerbated by the fact that the certificate given by a model checker can be more complex than a simple inductive invariant [8], [9]. For these reasons there is currently no viable industrial-strength model checker that provides certificates.

In this paper, we make a contribution toward this direction by revisiting temporal decomposition and developing a novel, practical, compositional framework for certifying the model checking result of the base engine and the employed preprocessing technique in a single proof. The distinguishing feature of our approach is to generate a *single witness circuit* as a certificate for the entire verification procedure, while related work [13], [14] relies on conceptually more complex deductive frameworks or has not been applied to industrial relevant hardware model checking problems that we are targeting. The approach in [15] also uses a deductive proof system for temporal decomposition that requires multiple independent parts to be checked, whereas our goal is to design a format such that only one witness circuit is checked, instead of a multi-stage proof. In theory, a single semantically simple proof format can be much easier to check by both untrusted and formally verified certificate checkers. Proof formats and checkers which follow the actual reasoning more closely will require to be adapted for every new technique used in the model checker. In contrast to [15], [16] that the underlying certificate provided by base model checkers is an inductive invariant, a witness circuit is more general. Moreover, different from [13]–[15], [17] as well as [18] which aims at handling more expressiveness, our work focuses on certifying *safety properties* which is the most prominent part in hardware model checking competitions and arguably in an industrial setting too. Additionally others have focused on either verifying model checkers themselves or lifting it directly to theorem proving [19]–[21]. Fully verifying model checkers can be a heavy task, as an update in the

Funded by FWF project W1255-N23, the LIT AI Lab funded by the State of Upper Austria, and Academy of Finland project 336092.

https://doi.org/10.34727/2023/isbn.978-3-85448-060-0_12

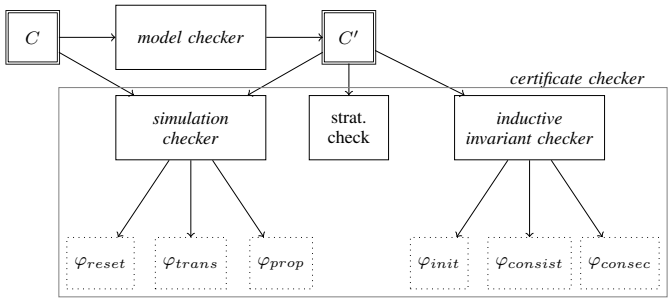

Fig. 1: An outline of the certification flow where C' is the certificate generated from the model checker (formally defined in Def. 12) and C is the original circuit. The certificate checker consists of three components and internally generates six SAT formulas and calls an underlying SAT solver.

optimization techniques often requires the entire procedure to be verified again. Lastly, similar problems have also been addressed in the context of software verification [22], [23].

The resulting certificate in our framework can be checked via six simple SAT checks and a polynomial check following the certification flow established in [8] as illustrated in Fig. 1, thus reducing trusting a PSPACE hard model checking flow into verifying a simple certificate circuit in co-NP. Our compositional approach breaks the formal proofs into manageable parts while enabling more certifiable techniques. For demonstration we focus on the k-induction algorithm as the base engine for the transformed circuit that provides a more complex certificate, as well as a BDD-based model checker. The reason is that our proposed flow requires model checkers to provide certificates for models with reset functions, which is technically rather involved to add to existing complex multi-engine model checkers, but this is a feature we want to encourage to be implemented.

Based on this, we have implemented a prototype certifying hardware model checker CHMC that performs temporal decomposition and provides a certificate that can be verified using a SAT solver. We also present two optimisations for detecting transient logic in a circuit. The experiments show our tool is able to solve 29 (out of 818) additional instances enabling temporal decomposition with the k-induction base engine, and an additional of 39 instances using the BDD backend. Our method can produce certificates for all instances solved by the model checker, and effectively verify them.

II. BACKGROUND

This paper extends the set of certifiable model checking techniques thus adopts the certificate format from [8], [9].

We assume a set of Boolean variables $\mathcal{V}$. A literal l is either a variable $l \in \mathcal{V}$ or its negation. We denote $\mathbb{B}(\mathcal{V})$ as the set of all Boolean functions over $\mathcal{V}$ conveniently represented and with formulas (Boolean expressions). A *cube* is a non-contradictory set of literals. For $f \in \mathbb{B}(\mathcal{V})$ we write $f|_l$ to denote the formula after replacing all occurrences of l in f with $\top$ and all occurrences of $\neg l$ with $\bot$. This naturally extends to cubes (interpreted as conjunction of literals). We also write $vars(f)$ to denote the variables that occur in f.

In the following we use a symbolic representation of hardware circuits matching the notation of [8] summarised below. This definition has the advantage of being highly compatible with the AIGER format [24] used for hardware model checking in practice. Note that we use "$\Rightarrow$" for semantic implication, "$\rightarrow$" for syntactic implication and "$\equiv$" for semantic equivalence cf. [9].

Definition 1 (Circuit [9])**.** *A circuit is a tuple* $C = (I, L, R, F, P)$ *where* I *is a set of Boolean* input *variables,* L *is a set of Boolean* latch *variables,* $R = \{r_l(L) \in \mathbb{B}(L) \mid l \in L\}$ *is a set of* reset functions*,* $F = \{f_l(I, L) \in \mathbb{B}(I, L) \mid l \in L\}$ *is a set of* transition functions*, and* $P(I, L) \in \mathbb{B}(I, L)$ *is the* property *formula.*

We write $R(L') = \bigwedge_{l \in L'} (l \simeq r_l(L))$ for some $L' \subseteq L$, where "$\simeq$" is used for syntactic equivalence [25]. Furthermore L_i denotes a copy of L in the temporal direction, thus $U_m = \bigwedge_{i \in [0,m)} (L_{i+1} \simeq F(I_i, L_i))$ denotes an unrolling of length m.

Definition 2 (Stratified circuit [8])**.** *Given a circuit* $C = (I, L, R, F, P)$ *with* $R = \{r_l \mid l \in L\}$*. The dependency graph* G_R *has latch variables* L *as nodes and contains a directed edge* (a, b) *from* a *to* b *iff* $a \in vars(r_b)$ *and* $r_b \neq b$*. The circuit is stratified iff* G_R *is acyclic.*

Definition 3 (Stratified simulation [8])**.** *Given two stratified circuits* $C = (I, L, R, F, P)$ *and* $C' = (I, L', R', F', P')$ *where* $L \subseteq L'$*. The circuit* C *is simulated by* C' *iff: (i)* $r_l(L) \equiv r'_l(L')$ *for* $l \in L$*; (ii)* $f_l(I, L) \equiv f'_l(I, L')$ *for* $l \in L$*; and (iii)* $P'(I, L') \Rightarrow P(I, L)$*.*

Intuitively, a circuit is stratified if its reset functions are acyclic. The stratification assumption allows the designed certificate in [8] to be checked by simple SAT checks instead of having a one-alternation QBF check as in [9]. The stratified simulation relation can be therefore verified via three SAT checks as stated in the definition above and one polynomial time check for stratification of reset functions of C'.

Definition 4 (k-induction [9])**.** *Given a circuit* C*,* P *is* k*-*inductive *in* C *iff, (i)* $U_{k-1} \wedge R(L_0) \Rightarrow \bigwedge_{i \in [0,k)} P(I_i, L_i)$*; and (ii)* $U_k \wedge \bigwedge_{i \in [0,k)} P(I_i, L_i) \Rightarrow P(I_k, L_k)$*.*

We make use of k-induction [5] formulated as a combination of BMC check and a consecution check. It generalizes the concept of checking an *inductive invariant* which is equivalent to 1-induction where the invariant is simply the property.

III. OVERVIEW

Temporal decomposition helps to simplify model checking by removing parts of the circuit that are only needed for initialisation. Using it as a preprocessing technique, the model checking problem is decomposed into smaller sub-problems.

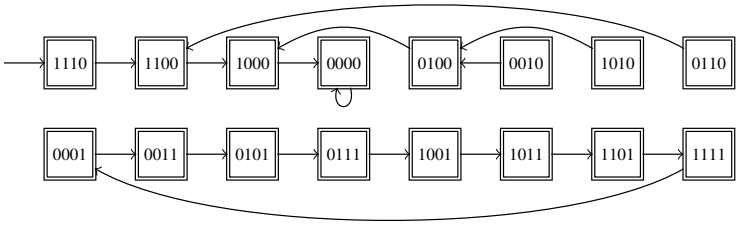

Fig. 2: State space of a 4-bit shift counter.

A series of transformation is applied to the circuit by time-shifting it and removing transient signals found via an over-approximation of the reachable states. We begin with an example to show that it can be quite useful and complementary to model checking techniques such as k-induction.

Example 1 (Shift Counter)**.** *Figure 2 shows the state transition diagram of the considered circuit, i.e., a shift counter over 4 bits, with an initial state 1110, where the least significant bit controls the operational mode and never changes. When this bit is set to 0, a left shift is performed; if it is set to 1, the other bits operate as a three-bit binary counter.*

Consider the property that at least one bit is zero. As the diagram shows, it is 8-inductive (as the single bad state with all bits one is not reachable, but the longest path only having it as last state has 8 states). In the more general case, with an arbitrary large number of bits n, the inductive depth k is exponentially large (2^{n-1}) in n, for a k-induction-based model checker. The size of the certificate for k-induction using only the approach presented in [8] without temporal decomposition would then be in $\mathcal{O}(k \cdot n)$ and thus exponential too.

However, if we consider only the reachable part of the state space, all bits are transient signals as they all eventually stabilise to zero. Therefore we can use temporal decomposition to simplify the design by time-shifting it and removing transient logic. The time-shifting depth (later we call it the *duration*) is linear in n as it grows linearly in the number of bits. In this particular scenario, the model checking problem for the terminal part becomes trivial. We only need a BMC check for the initial $n-1$ time frames. This leaves us with the problem: how to certify model checking with temporal decomposition?

The overall certification approach we propose is outlined in Fig. 3. For this example suppose we have a set of transient signals found that stabilise to constant values with duration 3. In practice the time-shifting namely *circuit-forwarding* of a design is implemented implicitly by computing the successor states originating from reset, however, with the objective of producing elegant and compositional proofs, we construct intermediate circuits forwarded by one transition only.

At each forwarding, the circuit gets unrolled by one time step from reset. For the sequence of circuits $C_0, ..., C_3$, a bounded model checker is used to verify that all initial states are good states. The last forwarded circuit along the pipeline (C_3) is simplified with transient elimination to obtain the factor circuit C_3', which is given to a base model checker (e.g., k-induction, BDD or IC3/PDR) that also produces a certificate (i.e., a witness circuit [8]). We now construct a composite witness circuit certifying both the preprocessing algorithm for the transients and the safety property. We then build a sequence of backward witness circuits while adding the BMC check for the initial states each time. At each step W_i is a witness circuit of C_i. In the end, we get W_0 as the final witness circuit with an inductive invariant for the entire procedure.

The final witness can be checked by an external proof checker [8]. If the check passes, the original circuit is guaranteed to be safe. It is thus not necessary to trust the correctness of the presented framework nor its implementation. The formal proofs provided in the following sections serve to show that if the original circuit is safe, a valid certificate can be produced.

IV. Temporal Decomposition

As one of our contributions we revisit temporal decomposition by defining a precise formalism and proving its correctness, which is an improvement over the theory in [2]. We show that our method is complete and will provide a valid certificate whenever temporal decomposition is employed.

In practice, temporal decomposition uses ternary simulation [26] to find transient signals [2]. As generalization we define *cube semantics* and the notion of *cube simulation*, that subsumes ternary simulation as well as other optimisations. We make use of this in our implementation (see Section VI).

Definition 5 (Cube simulation)**.** *Given a circuit $C = (I, L, R, F, P)$, then a cube simulation $c_0, \cdots, c_\delta, \cdots, c_{\delta+\omega}$ for $\delta, \omega \in \mathbb{N}$ is a sequence of cubes over latch variables L such that,*

- $R(L) \Rightarrow c_0$.
- *For $i \in [0, \delta + \omega)$ we require validity of $c_i \wedge (L' \simeq F(I, L)) \Rightarrow c'_{i+1}$ (L' and c'_{i+1} denote primed copies).*
- *In addition, it is called a* cube lasso *iff $c_{\delta+\omega} \wedge (L' \simeq F(I, L)) \Rightarrow c'_\delta$.*

Note that for $\delta = 0$ and $\omega = 0$, it would simply be a self-loop. This symbolic definition of cube simulation could be implemented directly by a symbolic engine. However, ternary simulation is more appropriate for industrial benchmarks [2].

For brevity we omit (the natural) formal definitions here, but remark that a "bounded version" of cube simulation is subsumed by model checking and in particular ternary simulation does not yield more transients than those produced by Boolean constraint propagation in the SAT solver. As a result, for SAT based bounded model checkers (and therefore

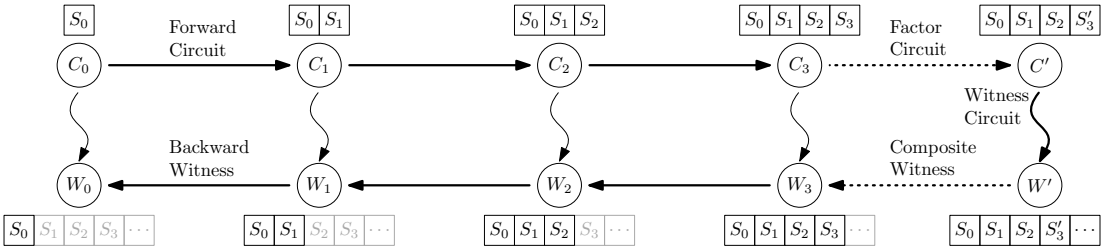

Fig. 3: An outline of our certification framework (for duration of 3).

k-induction-based model checkers on satisfiable instances), there is hardly any gain using temporal decomposition.

Proposition 1. *Bounded model checking subsumes bounded cube simulation.*

Under cube simulation, transient signals can be found by identifying those that stay constant from a certain point onwards along a cube lasso.

Definition 6 (Transients via cube lasso)**.** *Given a circuit C and a cube lasso $c_0, \cdots, c_\delta, \cdots, c_{\delta+\omega}$. A set of transient signals T is a cube over latches that satisfies $c_i \Rightarrow \bigwedge_{l \in T} l$ for $i \in [\delta, \delta+\omega]$.*

For the purpose of temporal decomposition, we could always time-shift the circuit by δ, however, it can make model checking of the final circuit easier to reduce this value as much as possible. For a set of transients found via cube lasso, we formally define the *unstable duration* of a circuit. It is the smallest value for which the last transient becomes constant.

Definition 7 (Unstable duration)**.** *Given a cube lasso $c_0, \cdots, c_\delta, \cdots, c_{\delta+\omega}$ and a set of transients T, the unstable duration $d \leq \delta$ is the lowest index that satisfies $c_i \Rightarrow \bigwedge_{l \in T} l$ for $i \in [d, \delta+\omega]$.*

For the rest of the paper, we simply refer to it as duration.

By taking the disjunction of all cubes from the duration onwards, we get an over-approximation of all the reachable states in the time-shifted circuit.

Definition 8 (Cube loop invariant)**.** *Given a cube lasso $c_0, \cdots, c_\delta, \cdots, c_{\delta+\omega}$, and a set of transients T with a duration d. The cube loop invariant ϕ_T is defined as:*

$$\phi_T = \bigvee_{i \in [d, \delta+\omega]} c_i.$$

An immediate observation is that the cube loop invariant is simply the inductive invariant that implies the stabilised transients in the time-shifted circuit by the duration. We formalise this in the following lemma.

Lemma 1. *Given a circuit C, a cube lasso $c_0, \cdots, c_d, \cdots, c_\delta, \cdots, c_{\delta+\omega}$, and a set of transients T with a duration d. Let C' be the resulting circuit of applying circuit forwarding to C iteratively d times. The cube loop invariant is an inductive invariant of C' for $\bigwedge_{l \in T} l$.*

To formalize "time shifting" we introduce the notion of *stable variables* which have the same values throughout the entire execution and do not appear in any other transition function nor in the property.

Definition 9 (Stable variable)**.** *Given a circuit $C = (I, L, R, F, P)$, the set $\Lambda \subseteq L$ is a set of stable variables if for all $l \in \Lambda$, the following holds: (i) $f_l = l$; (ii) for all $l' \in L \backslash \{l\}, l \notin vars(f_{l'})$; and (iii) $l \notin vars(P)$.*

This simple purely syntactic definition of stable variables is the weakest condition that allows us to avoid a spurious exponential blowup during circuit forwarding (Def. 10). Note that here identifying stable variables is a simple polynomial time check. Replacing it with stronger conditions, such as cone-of-influence reduction [27], after every forward circuit construction would potentially yield a smaller certificate, which we leave to future work.

A forward circuit is constructed based on a given circuit by copying the set of active variables (i.e., non-stable), and updating the resets of the original active variables to one transition ahead using oracles (uninitialised latches) instead of inputs. The formal definition is given below.

Definition 10 (Forward circuit)**.** *Given a circuit $C = (I, L, R, F, P)$ with a set of stable variables $\Lambda \subseteq L$ (we refer to $A = L \backslash \Lambda$ as the set of active variables). The forward circuit $C' = (I, L', R', F', P)$ is defined as follows:*

- *$L' = L \cup I' \cup A'$ where I' and A' are copies of I and A.*
- *$R' = \{r'_l \mid l \in L'\}$:*
  - *For $l \in \Lambda \cup A', r'_l = r_l(I, \Lambda \cup A')$.*
  - *For $l \in I', r'_l = l$.*
  - *For $l \in A, r'_l = f_l(I', A')$.*
- *$F' = \{f'_l \mid l \in L'\}$:*
  - *For $l \in L, f'_l = f_l(I, L)$.*
  - *For $l \in I' \cup A', f'_l = l$.*

An alternative to Def. 10 is to simply copy all latches when forwarding a circuit, however, the resulting circuit would be exponential in the duration when doing so iteratively. In the following we show that the forward circuit is stratified

Lemma 2. *Given a stratified circuit C. Its forward circuit C' is also stratified.*

Proof. Based on the reset function definition in Def. 10, the reset functions of $\Lambda \cup A'$ are the same as in $R(L)$, which

are acyclic. The variables in I' are uninitialised, thus do not depend on the rest of the variables. As $R'(A)$ only uses variables distinct from A, we conclude C' is stratified. □

For the forward circuit, we proceed to simplify it by removing the set of transients that have been found. The resulting simplified circuit (namely *factor circuit*) is formally defined in the following definition.

Definition 11 (Factor circuit)**.** *Given a circuit $C = (I, L, R, F, P)$ and a set of transients T from cube simulation. Its factor circuit is a tuple $C|_T = (I, L', R', F', P')$ such that,*

- $L' = L \backslash vars(T)$.
- $R' = \{r_l|_T \mid l \in L'\}$.
- $F' = \{f_l|_T \mid l \in L'\}$.
- $P' = P|_T$.

In the following proposition, we state that the conjunction of the factor property $P|_T$ and $\bigwedge_{l \in T} l$ (*i.e.,* the transients are constant) implies the original property. The same follows for reset and transition functions.

Proposition 2. $(P|_T \wedge \bigwedge_{l \in T} l) \Rightarrow P$, $(R|_T \wedge \bigwedge_{l \in T} l) \Rightarrow R$, *and* $(F|_T \wedge \bigwedge_{l \in T} l) \Rightarrow F$.

An important observation we make is that the inductive depth of the property does not increase after temporal decomposition, which is later confirmed in our experimental evaluation. However, we have already seen in Example II an exponential reduction in the inductive depth. We formally prove it and summarise it in the following theorem, which follows directly from Lemmas 3, and 4.

Theorem 1. *Given a circuit C where the property is k-inductive. Let $C|_T$ be the circuit resulting from temporal decomposition of C. Its property is k-inductive.*

Lemma 3. *Given a circuit $C = (I, L, R, F, P)$ where the property is k-inductive. Let $C' = (I, L', R', F', P')$ be the factor circuit. P' is k-inductive in C'.*

Proof. We do a proof by contradiction by assuming P' is not k-inductive in C'. First we consider the case where the BMC check fails in C' (i.e., $U'_{k-1} \wedge R'(L'_0) \wedge \neg \bigwedge_{i \in [i,k)} P'(I_i, L'_i)$ has a satisfying assignment). By Lemma 1, the transients stay constant in C, and by Propositions 2, we can construct a satisfying assignment for $U_{k-1} \wedge R(L_0) \wedge \neg \bigwedge_{i \in [i,k)} P(I_i, L_i)$. This contradicts our assumption. The second scenario where the consecution check fails in C' follows the same logic. □

Lemma 4. *Given a circuit $C = (I, L, R, F, P)$ where the property is k-inductive. Let $C' = (I, L', R', F', P)$ be the forward circuit. P is k-inductive in C'.*

Proof. We provide a proof by contradiction assuming P is not k-inductive in C'. Similarly we assume $R'(L'_0) \wedge U'_{k-1} \wedge \neg \bigwedge_{i \in [0,k)} P(I_i, L_i)$ has a satisfying assignment. By Def. 10 the same satisfies $R(L_0) \wedge U_k \wedge \neg \bigwedge_{i \in [1,k]} P(I_i, L_i)$. This implies that a bad state is reachable from the initial states thus contradiction. We then consider the consecution check fails in C' such that $U'_k \wedge \bigwedge_{i \in [0,k)} P(I_i, L_i) \wedge \neg P(I_k, L_k)$ has a satisfying assignment. By Def. 10 the transition function stays the same for the common latches thus the same assignment satisfies the formula $U_k \wedge \bigwedge_{i \in [0,k)} P(I_i, L_i) \wedge \neg P(I_k, L_k)$. Therefore we have reached a contradiction. □

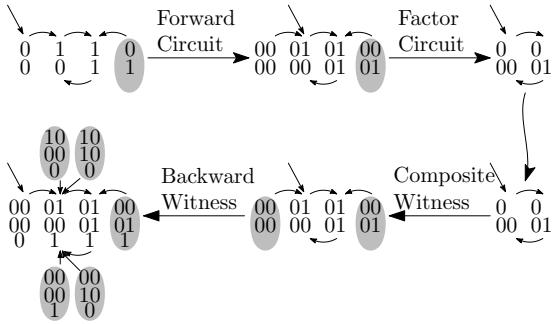

Fig. 4: An illustration of the overall certification flow for the delayed clock example. Initial states are marked with an additional arrow; bad states are marked gray. In the original circuit (top left), the lower bit is oscillating, and the upper bit is the enabler bit.

V. Certification

In this section, we present a compositional certification framework that is complete. Along the model checking procedure with temporal decomposition, a certificate can be automatically generated by the model checker following the format defined in this section.

Example 2. *Consider the scenario of a delayed clock (Fig. 4). The clock has one bit (the bottom bit in the diagram) that oscillates between zero and one after the enabler bit (first bit) is set to one. There is only one initial state where the clock is set to zero and enabler bit not set. A state is bad if the clock is high without being enabled.*

After preprocessing, a base model checker is called for verifying the factor circuit (In Fig. 4 the property of the factor circuit is already an inductive invariant thus the factor circuit is simply the certificate). It is required to provide a certificate that is then used to build the final certificate. We give a formal definition of the general certificate format below.

Definition 12 (Witness circuit)**.** *Given a circuit C, a witness circuit $W = (I, M, S, G, Q)$ of C satisfies the following:*

- *W simulates C under stratified simulation relation.*
- *Q is an inductive invariant in W.*

Since the method is compositional, with the witness circuit of the factor circuit produced from the model checker, we combine it with the loop invariant for cube lasso to construct a composite witness circuit that certifies both.

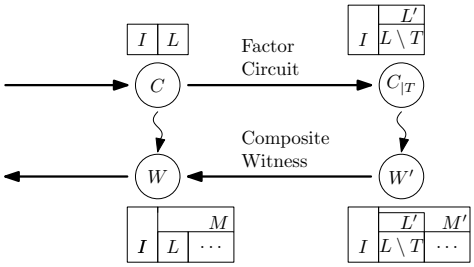

Fig. 5: Constructing composite witness circuit.

Definition 13 (Composite witness circuit)**.** *Given a stratified circuit $C = (I, L, R, F, P)$ with the cube loop invariant ϕ_T for the set of transients T, and its factor circuit $C|_T = (I, L', R', F', P')$ with its witness circuit $W' = (I, M', S', G', Q')$. The composite witness circuit $W = (I, M, S, G, Q)$ is constructed as follows:*

- $M = L \cup (M' \backslash L')$.
- $S = \{s_l \mid l \in M\}$:
  - *For* $l \in L, s_l = r_l$;
  - *For* $l \in M' \backslash L', s_l = s'_l$.
- $G = \{g_l \mid l \in M\}$:
  - *For* $l \in L, g_l = f_l$;
  - *For* $l \in M' \backslash L', g_l = g'_l$.
- $Q = \phi_T(L) \wedge Q'(I, M')$.

Intuitively, in the composite witness circuit we add back the previously eliminated transients. The new property simply combines the cube loop invariant from cube simulation and the property of the witness circuit of the factor circuit, which is an inductive invariant in the composite witness circuit.

Theorem 2. *Given a stratified circuit C, its factor circuit $C|_T$ with an inductive invariant ϕ_T, a witness circuit W' of $C|_T$, and the composite witness circuit W. Then W is a witness circuit of C.*

Proof. We show that W simulates C. The factor circuit $C|_T$ is obviously stratified as C is stratified by Def. 11. The witness circuit W' is also stratified. The reset functions of L remain the same in W, and $R(L)$ do not contain latches in $M' \setminus L'$. Thus W is stratified. Based on Def. 13, the common latches $L \subseteq M$ and inputs are the same in both circuits. As the reset functions and transition functions of L remain the same, this satisfies the reset check and transition check. Since W' is a witness of $C|_T$, we have $Q' \Rightarrow P'$, where $P' \equiv P|_T$ by Def. 11. Since ϕ_T is an inductive invariant for transients and by Lemma 1, we have $\phi_T \Rightarrow \bigwedge_{l \in T} l$. By Proposition 2, we have $Q \Rightarrow P$. Therefore W simulates C.

We then show the BMC check passes ($S(M) \Rightarrow Q(I, M)$) by assuming that $S(M)$ and thereby $R(L) \wedge S'(M' \setminus L')$ holds, and shall proceed to the conclusion $\phi_T(L) \wedge Q'(I, M')$, by Def. 13. We begin with $R(L)$ and may deduce $\phi_T(L)$ immediately since Lemma 1 applies to C and the circuit is in its reset state.

From this point on, our attention will be on the subset of L that is free of transients, L' by Def. 11. We have $R(L') = R'(L')$ again by Def. 11, and then $R'(L') = S'(L')$ since W' simulates $C|_T$, which by Def. 3 implies the resets to be the same. We combine this with the second half of our initial assumption to arrive at $S'(M')$. The witness circuit W' is therefore at reset, and its inductive invariant $Q'(I, M')$ holds. We conclude with $Q(I, M)$.

We make the observation that $G(I, L) \equiv F(I, L)$, $F(I, L \setminus T) \equiv F'(I, L \setminus T) \equiv G'(I, L \setminus T)$ by Def. 12 and Def. 11. The latter together with Def. 13 gives us $G(I, M') \equiv G'(I, M')$. Assume a fixed satisfying assignment for $V_1 \wedge Q(I_0, M_0)$, where V_1 is the unrolling of length 1 in W. By the observation above, this also satisfies U_1 and V'_1 which are the unrollings of C and W' respectively. By Def. 13 the assignment satisfies $\phi_T(L_0) \wedge Q'(I_0, M'_0)$ which are inductive invariants in C and W' respectively. We thus have $\phi_T(L_1) \wedge Q'(I'_1, M'_1)$. As we have shown the inductiveness of Q with the BMC check and consecution check, together with the simulation relation, we conclude W is a witness circuit of C. □

We now introduce *backward witness circuit* built based on a given witness circuit for one backward step, as illustrated in Fig. 6. Intuitively, at each backward step, the backward witness circuit certifies the BMC check for the initial states of the corresponding circuit C while maintaining and delaying the behaviours of the given witness circuit W' by one step.

This is achieved by using one bit b which becomes constant true exactly one step after initialisation. At reset, W has the same values as C, with the additional latches holding the reset values from W'; once b is set, it operates as W'. Recall that when constructing the forward circuit, an additional copy of inputs and active latches is added for copying the initial values of those in the given circuit. Since input values are non-deterministic, this needs to be recovered when constructing W such that the inputs are copied to ensure matching values.

Definition 14 (Backward witness circuit)**.** *Given a stratified circuit $C = (I, L, R, F, P)$, and its forward circuit $C' = (I, L', R', F', P)$. Let $W' = (I, M', S', G', Q')$ be the witness circuit C'. The backward witness circuit $W = (I, M, S, G, Q)$ is defined as follows:*

1) $M = M' \cup \{b\}$.
2) $S = \{s_l \mid l \in M\}$ *such that:*
   - *For* $l \in L, s_l = r_l$.
   - *For* $l \in M' \backslash L, s_l = s'_l$.
   - $s_b = \bot$.
3) $G = \{g_l \mid l \in M\}$ *such that:*
   - *For* $l \in L, g_l = f_l$.
   - *For* $l \in M' \backslash L', g_l = ite(b, g'_l, s'_l)$.
   - *For* $l' \in L' \backslash L, g_{l'} = ite(b, l', l)$ *where* l *is the literal with the same index in* $I \cup A$ *as* l' *in* $I' \cup A'$ $(= L' \backslash L)$.
   - $g_b = \top$.
4) $Q = \bigwedge_{i \in [0,3]} q_i$, *where*

- $q_0 = P(I, L)$.
- $q_1 = b \rightarrow Q'(I, M')$.
- $q_2 = \neg b \rightarrow R(L)$.
- $q_3 = \neg b \rightarrow S'(M' \backslash L)$.

We can thus obtain a witness circuit that certifies (i) the property holds at reset for forward circuits and (ii) the transformed property holds in the factor circuit. In an iterative manner eventually we can construct a witness circuit for the entire pipeline illustrated in Fig. 3. Here the property consists of four subproperties: q_0 is the original property; q_1 states that the property from W' needs to hold when b is set; q_2 states that b is false only at initialisation; and q_3 states that all latches except those in L need to hold the reset values as in W'.

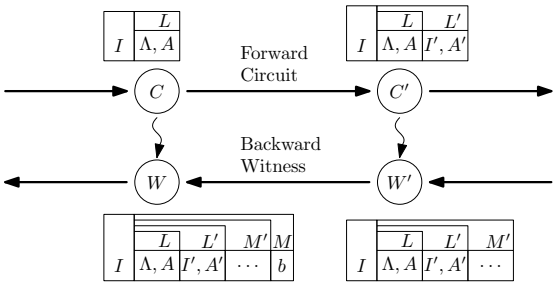

Fig. 6: An illustration of the latch contents of circuits. C is a time-shifted circuit with latches of Λ (stable variables) and A (active variables). We obtain C' by adding a copy of I' and A' which are used for unrolling the initial states. W' is the witness circuit simulating C'. In addition to the common latches L', M' can also include a set of unknown latches (potentially generated from the model checker). W is the backward witness circuit, containing all latches from M' and an additional bit b used for an initialisation step.

By Def. 12 the backward witness needs to pass the six SAT checks for the stratified simulation relation and the witness inductiveness, as well as the stratification check of its reset functions. We summarise this in the following theorem.

Theorem 3. *Given a stratified circuit C where the property holds in all reset states, the forward circuit C' with its witness circuit W', and the backward witness circuit W as defined in Def. 14. Then W is a witness circuit of C.*

Proof. First we show W is stratified. By Def. 14, $S(L) \equiv R(L)$, therefore stratified and does not depend on latches outside L, and $S(\{b\})$ is independent of other variables. Furthermore, for the rest of latches $M' \backslash L$, the reset functions are the the same as in W' therefore stratified. We conclude that W is stratified. By Def. 14, $L \subseteq M$ and the inputs I stay the same in W. Based on Def. 14, for the common latches L, their reset function and transition function are the same as in C, and $q_0 \equiv P$. Therefore W simulates C.

We proceed to prove that the BMC check passes in W, i.e., $S(M) \Rightarrow Q(I, M)$. Since b is false at reset, q_1 is trivially satisfied. The reset also directly implies $R(L)$ and $S'(M' \backslash L)$, which gives us q_2 and q_3. We have q_0 since the property holds at reset in C. We then move on to prove the consecution check passes ($V_1 \wedge Q(I_0, M_0) \Rightarrow Q(I_1, M_1)$). We consider two cases based on the value of b_0 and begin with the case $b \equiv \top$. By Def. 14, since b always transitions to true, $q_2(I_1, L_1)$ and $q_3(I_1, L_1)$ are trivial. Additionally, $q_1(I_1, M_1)$ implies $Q'(I_1, M_1')$ which by Def. 12 implies $P(I_1, L_1)$, as C and C' share the same property. Thus we only need to show $Q'(I_1, M_1')$ holds after one transition to satisfy $q_1(I_1, M_1)$. Consider a fixed satisfying assignment for $V_1 \wedge Q(I_0, M_0)$ and b is true. We show that the same assignment satisfies V_1' (the unrolling of W'). For latches in $M' \backslash L'$ this follows directly from the definition. By Def. 12 and Def. 10 L has the same transition function in all 4 circuits (Fig. 6). The latches in $L' \backslash L$ stay constant which matches Def. 10. The rest follows from Def. 12. With this, $q_1(I_0, M_0')$ gives us $Q'(I_0, M_0')$ which is an inductive invariant in W'. $Q'(I_1, M_1')$ follows.

Now consider a satisfying assignment where b is false thereby $R(L_0)$ and $S'(M_0' \backslash L_0)$. By Def. 10, for the latches in L the transition function matches the reset function in C', thus we get $R'(L_1)$. The latches in $L' \backslash L$ copy the values of $I \cup A$. By Def. 10 $R'(I_1')$ holds trivially and $R'(A_1')$ follows from $R(L_0)$. We get $R'(L_1' \backslash L_1)$. Together with the previous result we have $R'(L_1')$, which by Def. 3 yields $S'(L_1')$. The reset for the rest of the latches $M' \backslash L'$ follows directly from Def. 14 and we get $S'(M_1')$. Since the inductive invariant of W' holds in its reset we conclude with $Q'(I_1, M_1')$. □

The above concludes the description and formal proof of our certification framework. To perform temporal decomposition on a given circuit under verification, we use cube simulation to find a set of transient signals and determine the duration d. The original circuit is then time-shifted by constructing forward circuits iteratively d times. After eliminating transient signals we obtain a simplified circuit that is then given to a base model checker for verifying the simplified safety property while producing a witness circuit. We construct a final witness circuit by building backward witness circuit d times again in an iterative manner. This final witness circuit serves as a single certificate for both preprocessing and backend model checking. It can be easily verified by an external certifier.

VI. Implementation and Experimental Evaluation

We implemented the method proposed in Section IV and V in the certifying model checker CHMC [28]. Our model checker CHMC supports two configurations for the backend solver performing certification on simplified models (*i.e.,* factor circuits) after preprocessing: k-induction and BDD. As factor circuits use reset functions, we extended the k-induction-based open-source model checker MCAIGER [29] to support reset functions and also extended the tool presented by the authors of [8] to generate witness circuits. For the BDD-based configuration, we used the simple BDD-based model checker AIGTRAV developed by one of the authors of this paper at JKU Linz which already supports reset functions. We reencoded the convergent BDD to an AIG encoding the membership in the set of reachable states, which is simply the inductive invariant. The witness circuit is the factor

circuit having the inductive invariant as the new property. Our model checker CHMC then performs witness composition and backwarding to generate the final witness verified by CERTIFAIGER++ [30].

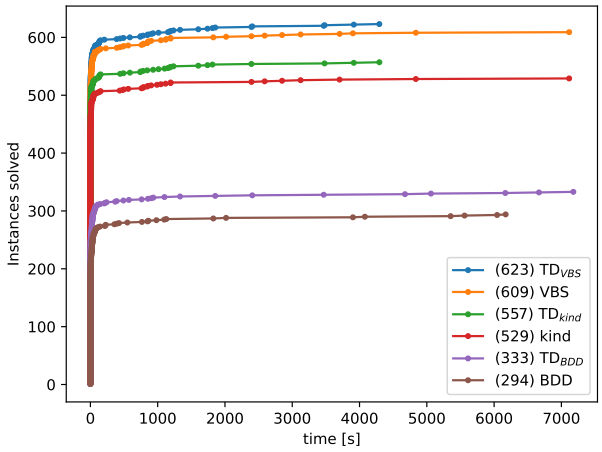

Fig. 7: Comparison for temporal decomposition with k-induction (TD_{kind}) and BDDs (TD_{BDD}) and the base engines on HWMCC10 instances. We show the results of the best performing solver for each instance with temporal decomposition (TD_{VBS}) as well as the best performing solver without it (VBS).

In principle our approach naturally works for any model checker that produces a certificate (e.g., IC3/PDR, interpolation), however, currently the available implementations for these do not support reset functions nor do they follow the pre-defined certificate format. It seems non-trivial to do so for certain model checkers. Note that a simple constraint-based construction to eliminate reset functions does not solve the problem, as the produced certificate needs to be lifted for the overall certification. It is however natural to modify symbolic model checkers to support reset functions, which we strongly encourage for extending the certification capability to a larger set of model checking tools.

State-of-the-art model checkers such as ABC [3] use additional preprocessing techniques and are multi-engines for efficient verification, which are hard to separate from each other thus making it difficult to certify a complete model checker with a multitude of preprocessing algorithms. However, we have shown how the certification of an important preprocessing technique, namely temporal decomposition, can be certified in a compositional manner. We believe that similar compositional certification techniques can be identified for other preprocessing techniques. In particular the approach used here can be used for any other preprocessing technique subsumed by cube simulation.

To increase trust in our tool we generated 42 million random circuits [24] and checked all produced certificates. We used the HWMCC'10 benchmarks [31] and additionally scaled the shift counter example for further evaluation. All experiments were conducted in parallel on 32 nodes of a cluster. Each node has

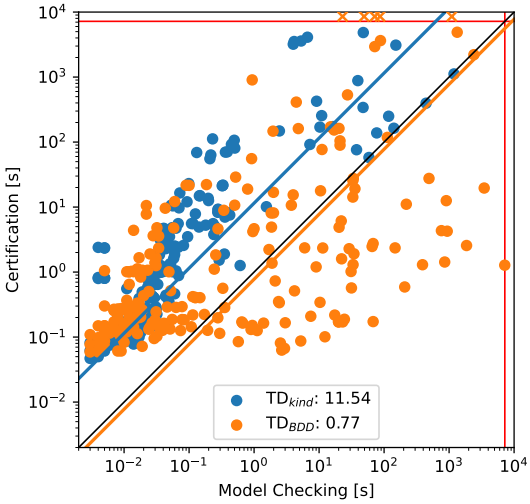

Fig. 8: Certification vs. model checking time.

access to two 8-core Intel Xeon E5-2620 v4 CPUs running at 2.10 GHz (turbo-mode disabled) and 128 GB main memory. We allocate 8 instances to every node with a timeout of 2 hours and memory limit of 16 GB per instance.

We studied the impact of the preprocessing technique on k-induction and BDD-based model checking. The preprocessing (ternary simulation and circuit transformation) terminated within 200ms on all instances. Fig. 7 displays the number of instances solved over time by the four configurations. We observe that CHMC with k-induction (TD_{kind}) outperforms the rest. It verified a total of 557 instances, among which 235 are UNSAT. In particular, compared with base MCAIGER (kind), we gain 29 UNSAT instances. We further inspected the inductive depths of the original circuits and their simplified ones; the results match the claim of Theorem 1. In our benchmark set k was reduced for 19 instances. As for the BDD configuration (TD_{BDD}), CHMC performed well on SAT cases, solving an additional of 39 instances. These results clearly demonstrate that simplifying transient logic in circuits can be beneficial for model checking.

We now proceed to evaluate the certification framework on both k-induction and BDD configurations. Table I summarises the results obtained on certifying the HWMCC10 benchmarks, where we display a subset of interesting instances (sorted by witness size). CHMC was able to produce certificates for all 235 unsatisfiable instances that it was able to model-check. For the k-induction variant, all certificates were successfully verified, with a mean ratio of certification time and model checking time of 11.54, which can be further inspected in Fig. 8. This shows the practicality of our method.

On average, duration d tends to be small, which can be partially explained as the starting sequence in designs playing a role. On the BDD variant, the average ratio of certification and model checking time is 0.77. However, 5 instances experienced time-outs when verifying the certificates produced by

TABLE I: Columns report cycle length (ω), stem length of cube lasso (δ), duration (d), no. transients found (τ), inductive depth of factor circuits (k), original model size ($\#_M^K$), model checking time ($t_M(s)$), certification time ($t_C(s)$), and certificate size ($\#_M^K$). All circuit sizes ($\#_M^K$) are measured in no. thousands of gates including latches and inputs. The certification time is the total of generation time and SAT solving time. We report three mean values on different subsets of benchmarks. Mean TD_{BDD} and TD_{kind} are computed over 223 and 235 instances respectively, where certification checks terminated. Mean $\text{TD}_{\cap}$ concerns the intersection of both sets, from which we display 20 instances that produced the biggest certificate sizes for TD_{kind}. Additionally, we list 5 instances at the bottom for which the BDD-based algorithm succeeded in model checking the simplified circuit, but produced certificates that could not be checked within the time limit.

	Decomposition				Model		TD_{kind}			TD_{BDD}		
	ω	δ	d	τ	k	$\#_M^K$	t_M	t_C	$\#_M^K$	t_M	t_C	$\#_M^K$
Mean TD_{BDD}(223)	1.9	5.8	0.3	4.9	1.7	4	–	–	–	100.29	77.71	58
Mean TD_{kind}(235)	2.1	8	0.9	46.3	4.1	7	10.40	120.06	93	–	–	–
Mean $\text{TD}_{\cap}$(157)	2.2	5.7	0.4	5.1	1.7	4	0.19	46.40	35	65.33	46.69	36
pj2002	1	4	2	2	9	37	4.13	3522.82	1757	33.09	25.56	129
pj2003	1	4	2	2	9	37	4.00	3203.89	1757	33.57	27.57	129
cmuperiodic	1	1	1	1	96	2	16.41	173.04	480	0.05	0.52	5
mentorbm1p09	1	39	1	122	2	36	0.45	92.00	331	0.18	22.17	123
mentorbm1or	1	39	1	122	1	36	0.32	21.24	123	0.10	21.56	123
nusmvreactorp4	1	1	1	1	13	1	0.20	12.59	85	7177.65	1.28	15
neclaftp5001	1	10	10	21	0	2	0.05	1.34	79	0.03	1.37	79
neclaftp5002	1	10	10	21	0	2	0.05	1.34	79	0.03	1.40	79
bobsynthand	1	38	1	1	0	19	0.12	7.85	73	0.02	7.79	73
139464p0	1	4	1	2	0	21	0.09	12.05	45	0.06	12.16	45
bj08amba4g5	1	6	0	0	3	14	0.13	69.07	42	219.68	11.06	14
139463p0	1	4	1	2	0	15	0.07	9.72	33	0.04	9.60	33
bj08amba3g62	1	4	0	0	3	10	0.10	22.99	31	1.85	4.92	10
139454p0	1	4	1	2	0	13	0.06	4.73	30	0.06	4.78	30
139462p0	1	4	1	2	0	10	0.06	4.82	23	0.03	4.81	23
139453p0	1	4	1	2	0	9	0.05	3.51	21	0.03	3.58	21
bj08amba5g82	1	5	0	3	0	20	0.07	16.49	20	1.98	16.85	20
139444p0	1	4	1	2	0	8	0.05	3.60	19	0.03	3.62	19
pdtvisvsa16a04	5	7	0	0	2	7	0.05	2.57	18	1.95	147.20	258
pdtvisvsa16a00	5	7	0	0	2	7	0.05	2.62	18	0.02	1.12	7
pdtswvtma6x4p3	1	7	0	0	44	2	77.00	138.26	138	71.75	to	13534
pdtswvtma6x4p2	1	7	0	0	37	2	37.69	76.23	116	86.86	to	13327
nusmvreactorp3	1	1	1	1	4	1	0.04	2.30	24	1100.38	to	6588
pdtvisvsar04	5	7	0	0	2	2	0.02	0.61	8	22.97	to	11074
pdtvisminmax2	1	3	0	0	2	1	0.02	0.14	2	49.29	to	2776

TABLE II: Comparison of kind and TD_{kind} for the *Shift Counter*. The n column reports no. of bits. Circuits sizes ($\#_M$) measured in number of gates. Compared with TD_{kind}, as n increases the model checking quickly becomes hard for the k-induction only engine and it experiences time-outs (to), while TD_{kind} was still able to solve the instances within reasonable time.

	Decomposition				Model		kind			TD_{kind}		
n	ω	δ	d	τ	k	$\#_M$	t_M	t_C	$\#_M$	t_M	t_C	$\#_M$
2	1	1	1	2	2	4	0.003	0.051	66	0.011	0.053	23
3	1	2	2	3	4	13	0.003	0.053	250	0.010	0.057	79
4	1	3	3	4	8	22	0.004	0.066	739	0.010	0.057	159
5	1	4	4	5	16	31	0.005	0.093	1974	0.010	0.061	263
6	1	5	5	6	32	40	0.008	0.165	5045	0.013	0.069	391
7	1	6	6	7	64	49	0.031	0.361	12764	0.014	0.069	543
8	1	7	7	8	128	58	0.111	0.987	32883	0.010	0.080	719
9	1	8	8	9	256	67	0.461	3.460	88618	0.011	0.086	919
10	1	9	9	10	512	76	2.409	14.830	255649	0.011	0.084	1143
11	1	10	10	11	1024	85	13.711	59.410	799128	0.012	0.105	1391
12	1	11	11	12	2048	94	86.877	164.968	2698130	0.013	0.108	1663
13	1	12	12	13	4096	103	498.395	501.894	9693060	0.014	0.125	1959
14	1	13	13	14	8192	112	2686.540	2217.410	36368300	0.011	0.108	2279
1000	1	999	999	1000	2^{999}	8986	to	to	to	5.902	1809	11996000

TABLE III: Optimised ternary simulation of three configurations: base version, counter stagnation, cube subsumption with counter stagnation. The number of latches is denoted by $\#_L$ ($\#_M$ is the total number of gates including inputs and latches), and t is time taken (seconds) by ternary simulation. Highlighted numbers indicate the best performing variant for each instance.

			ternary simulation				counter stagnation				cube subsumption			
	$\#_M$	$\#_L$	ω	δ	τ	t	ω	δ	τ	t	ω	δ	τ	t
bob12m04	269935	43950	2	9	199	0.06	2	9	199	0.07	2	6	153	**0.05**
6s376r	113207	4708	131072	262160	145	766.34	1	8198	116	16.30	1	2056	116	**4.20**
bob12m15	2855	448	12288	10379	133	1.62	12288	10379	133	1.60	3	4108	12	**0.40**
bobsmnut1	6301	644	1024	259	107	0.21	1024	259	107	0.19	1024	81	106	**0.18**
shift1add	174	27	1	262145	20	1.20	1	262145	20	1.25	1	2056	1	**0.01**
6s47	4950	815	to	to	to	to	2	262167	8	35.63	2	2085	6	**0.30**
6s100	763775	97598	to	to	to	to	1	2053	36	**37.53**	1	2053	36	41.48
6s107	25447	1568	to	to	to	to	1	32799	746	16.93	1	2079	746	**1.18**
6s149	112623	12781	to	to	to	to	1	20497	4	47.81	1	5137	4	**12.88**
6s342rb$_{122}$	394016	56838	to	to	to	to	to	to	to	to	192	46080	1894	**354.60**
6s202b41	604375	68881	to	to	to	to	1	4136	8574	45.71	1	2088	8574	**28.35**
6s204b16	237048	28986	to	to	to	to	1	4136	4034	19.33	1	2088	4034	**13.51**
6s205b20	603985	68842	to	to	to	to	1	4136	8727	46.56	1	2088	8727	**28.08**
6s355rb$_{8740}$	179445	15091	to	to	to	to	1	8466	221	37.07	1	3346	221	**15.51**
6s400rb$_{7819}$	180067	14665	to	to	to	to	1	8466	221	36.96	1	2322	221	**11.00**
cucnt128	1272	128	to	to	to	to	to	to	to	to	1	61440	0	**1.82**
cucnt32	312	32	to	to	to	to	to	to	to	to	1	1054	0	**0.01**

the BDD backend. Indeed, the BDD algorithm can produce large certificates due to the nature of using the exact set of reachable states as the inductive invariant.

We further investigate the shift counter example by scaling the number of bits (n). The results obtained are reported in Table II. As expected, we found that temporal decomposition significantly simplifies model checking, whereas MCAIGER quickly experiences timeouts. Note that in this particular example, the value of k simply decreases to 0 on the simplified model which becomes trivial after temporal decomposition.

Optimised Ternary Simulation To find transient signals we initially implemented a base version of ternary simulation. We now present two optimisations, also subsumed by cube simulation: (i) *Counter stagnation*: if the current cube size does not decrease in a few thousand steps, we observe a *stagnation* in the search. We thus remove a random latch that has flipped sign between the last two cubes. In case we remove a latch that is part of a large binary counter in the design, all more significant bits will be removed in a linear number of simulation steps. This technique can therefore achieve an exponential reduction in simulation steps. (ii) *Cube subsumption*: the cube lasso can terminate in a cube which implies a previous cube under the transition relation (Def. 5). Ternary simulation on the other hand utilises a hash map to terminate when an exact match of a cube is visited a second time. With cube subsumption it terminates when the current cube is a (not necessarily proper) superset of a previously encountered cube. For this we implement a forward-subsumption algorithm utilising a one-watch-literal data structure [32].

We compare it to two configurations with different levels of optimisation on 20,815 instances including all HWMCC benchmarks (2007-2020) [33]. The optimised versions can miss transients however it does not happen often. Table III displays the instances where the number of transients differ among three configurations. Worth mentioning is that cube subsumption together with counter stagnation did not time out on any instances of the entire benchmark set.

VII. Conclusion

While certification for model checking has been studied for decades, the number of certifiable approaches are still limited. In this paper we revisited the popular preprocessing technique temporal decomposition by formally defining it and proving its correctness. We further present a certification framework in a compositional manner, that builds a single witness circuit. The approach was evaluated on a wide range of benchmarks. Note that our framework requires model checkers to allow reset functions, that is a feature we would like to encourage existing model checkers to implement. Experimental results show that our approach is quite effective in practice.

In the future we plan to investigate other preprocessing techniques such as retiming [34], phase abstraction [35], as well as stabilizing constraints [15], [36]. Furthermore, our simple generic model checking certificate makes it appealing to work on verified proof checkers too.

REFERENCES

[1] E. M. Clarke, D. E. Long, and K. L. McMillan, "Compositional model checking," in *LICS*. IEEE Computer Society, 1989, pp. 353–362.

[2] M. L. Case, H. Mony, J. Baumgartner, and R. Kanzelman, "Enhanced verification by temporal decomposition," in *FMCAD*. IEEE, 2009, pp. 17–24.

[3] R. K. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *CAV*, ser. Lecture Notes in Computer Science, vol. 6174. Springer, 2010, pp. 24–40.

[4] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuXmv symbolic model checker," in *CAV*, ser. Lecture Notes in Computer Science, vol. 8559. Springer, 2014, pp. 334–342.

[5] M. Sheeran, S. Singh, and G. Stålmarck, "Checking safety properties using induction and a SAT-solver," in *FMCAD*, ser. Lecture Notes in Computer Science, vol. 1954. Springer, 2000, pp. 108–125.

[6] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10ˆ20 states and beyond," in *LICS*. IEEE Computer Society, 1990, pp. 428–439.

[7] A. R. Bradley, "SAT-based model checking without unrolling," in *VMCAI*, ser. Lecture Notes in Computer Science, vol. 6538. Springer, 2011, pp. 70–87.

[8] E. Yu, N. Froleyks, A. Biere, and K. Heljanko, "Stratified certification for k-induction," in *FMCAD*. IEEE, 2022, pp. 59–64.

[9] E. Yu, A. Biere, and K. Heljanko, "Progress in certifying hardware model checking results," in *CAV (2)*, ser. Lecture Notes in Computer Science, vol. 12760. Springer, 2021, pp. 363–386.

[10] M. J. H. Heule, "Proofs of unsatisfiability," in *Handbook of Satisfiability - Second Edition*, ser. Frontiers in Artificial Intelligence and Applications, A. Biere, M. Heule, H. van Maaren, and T. Walsh, Eds. IOS Press, 2021, vol. 336, pp. 635–668. [Online]. Available: https://doi.org/10.3233/FAIA200998

[11] M. J. Heule and A. Biere, "Proofs for satisfiability problems," in *All about Proofs, Proofs for all*, B. W. Paleo and D. Delahaye, Eds. College Publications, 2015, ch. 1.

[12] M. Järvisalo, M. Heule, and A. Biere, "Inprocessing rules," in *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, ser. Lecture Notes in Computer Science, B. Gramlich, D. Miller, and U. Sattler, Eds., vol. 7364. Springer, 2012, pp. 355–370. [Online]. Available: https://doi.org/10.1007/978-3-642-31365-3_28

[13] S. Eriksson, G. Röger, and M. Helmert, "Unsolvability certificates for classical planning," in *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017*, L. Barbulescu, J. Frank, Mausam, and S. F. Smith, Eds. AAAI Press, 2017, pp. 88–97.

[14] K. S. Namjoshi, "Certifying model checkers," in *CAV*, ser. Lecture Notes in Computer Science, vol. 2102. Springer, 2001, pp. 2–13.

[15] A. Griggio, M. Roveri, and S. Tonetta, "Certifying proofs for SAT-based model checking," *Formal Methods Syst. Des.*, vol. 57, no. 2, pp. 178–210, 2021.

[16] ——, "Certifying proofs for LTL model checking," in *FMCAD*. IEEE, 2018, pp. 1–9.

[17] A. Abuin, A. Bolotov, U. Díaz-de-Cerio, M. Hermo, and P. Lucio, "Towards certified model checking for PLTL using one-pass tableaux," in *TIME*, ser. LIPIcs, vol. 147. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 12:1–12:18.

[18] T. Kuismin and K. Heljanko, "Increasing confidence in liveness model checking results with proofs," in *Haifa Verification Conference*, ser. Lecture Notes in Computer Science, vol. 8244. Springer, 2013, pp. 32–43.

[19] J. Esparza, P. Lammich, R. Neumann, T. Nipkow, A. Schimpf, and J. Smaus, "A fully verified executable LTL model checker," in *CAV*, ser. Lecture Notes in Computer Science, vol. 8044. Springer, 2013, pp. 463–478.

[20] ——, "A fully verified executable LTL model checker," *Arch. Formal Proofs*, vol. 2014, 2014.

[21] S. Wimmer and J. von Mutius, "Verified certification of reachability checking for timed automata," in *TACAS (1)*, ser. Lecture Notes in Computer Science, vol. 12078. Springer, 2020, pp. 425–443.

[22] D. Beyer, M. Dangl, D. Dietsch, and M. Heizmann, "Correctness witnesses: exchanging verification results between verifiers," in *SIGSOFT FSE*. ACM, 2016, pp. 326–337.

[23] D. Beyer, M. Dangl, D. Dietsch, M. Heizmann, T. Lemberger, and M. Tautschnig, "Verification witnesses," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 4, pp. 57:1–57:69, 2022.

[24] A. Biere, K. Heljanko, and S. Wieringa, "AIGER 1.9 and beyond," Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria, Tech. Rep. 11/2, 2011.

[25] A. Degtyarev and A. Voronkov, "Equality reasoning in sequent-based calculi," in *Handbook of Automated Reasoning (in 2 volumes)*, J. A. Robinson and A. Voronkov, Eds. Elsevier and MIT Press, 2001, pp. 611–706.

[26] C. H. Seger and R. E. Bryant, "Formal verification by symbolic evaluation of partially-ordered trajectories," *Formal Methods Syst. Des.*, vol. 6, no. 2, pp. 147–189, 1995.

[27] E. M. Clarke, O. Grumberg, D. Kroening, D. A. Peled, and H. Veith, *Model checking, 2nd Edition*. MIT Press, 2018. [Online]. Available: https://mitpress.mit.edu/books/model-checking-second-edition

[28] CHMC, "CHMC," 2023, http://fmv.jku.at/chmc.

[29] A. Biere and R. Brummayer, "Consistency checking of all different constraints over bit-vectors within a SAT solver," in *FMCAD*. IEEE, 2008, pp. 1–4.

[30] Certifaiger, "Certifaiger," 2021, http://fmv.jku.at/certifaiger.

[31] A. Biere and K. Claessen, "Hardware model checking competition 2010," 2010, http://fmv.jku.at/hwmcc10/.

[32] L. Zhang, "On subsumption removal and on-the-fly CNF simplification," in *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005, St. Andrews, UK, June 19-23, 2005, Proceedings*, ser. Lecture Notes in Computer Science, F. Bacchus and T. Walsh, Eds., vol. 3569. Springer, 2005, pp. 482–489. [Online]. Available: https://doi.org/10.1007/11499107_42

[33] M. Preiner, A. Biere, and N. Froleyks, "Hardware model checking competition 2020," 2020.

[34] A. Kuehlmann and J. Baumgartner, "Transformation-based verification using generalized retiming," in *CAV*, ser. Lecture Notes in Computer Science, G. Berry, H. Comon, and A. Finkel, Eds., vol. 2102. Springer, 2001, pp. 104–117.

[35] P. Bjesse and J. H. Kukula, "Automatic generalized phase abstraction for formal verification," in *ICCAD*. IEEE Computer Society, 2005, pp. 1076–1082.

[36] K. Claessen and N. Sörensson, "A liveness checking algorithm that counts," in *FMCAD*. IEEE, 2012, pp. 52–59.