Introducing Certificates to the Hardware Model Checking Competition

Nils Froleyks[1(✉)], Emily Yu[2], Mathias Preiner[3], Armin Biere[4], and Keijo Heljanko[5,6]

[1] Johannes Kepler University, Linz, Austria
Nils.froleyks@gmail.com
[2] Institute of Science and Technology Austria, Klosterneuburg, Austria
[3] Stanford University, Stanford, CA, USA
[4] University of Freiburg, Freiburg, Germany
[5] University of Helsinki, Helsinki, Finland
[6] Helsinki Institute for Information Technology, Helsinki, Finland

Abstract. Certification was made mandatory for the first time in the latest hardware model checking competition. In this case study, we investigate the trade-offs of requiring certificates for both passing and failing properties in the competition. Our evaluation shows that participating model checkers were able to produce compact, correct certificates that could be verified with minimal overhead. Furthermore, the certifying winner of the competition outperforms the previous non-certifying state-of-the-art model checker, demonstrating that certification can be adopted without compromising model checking efficiency.

1 Introduction

Competitions have played a key role in advancing the state of the art in automated reasoning tools by enabling direct performance comparisons across a wide range of solvers, offering challenging benchmarks, and fostering new research. However, many of these tools operate as black boxes by providing only *true* or *false* as an output. Certification addresses this limitation by requiring a counterexample when verification fails and a proof when it succeeds. Since certificates can be independently validated, they significantly enhance confidence in the correctness of verification results, thereby improving the reliability of solvers.

One goal of using certificates in hardware model checking is to repeat the success story of proof certificates in SAT for this automated reasoning domain with a large industrial user base. Besides increasing trust in verification results, certificates enable more complex design optimizations, allow to continue using legacy code and can streamline and improve efficiency of tool development in both verification and synthesis. The simple proof certificate format used in SAT still allows to capture a wide range of solving optimizations at industrial scale. In this case study, we investigate whether the simple model checking certificate

R. Piskac and Z. Rakamarić (Eds.): CAV 2025, LNCS 15931, pp. 281–295, 2025.
https://doi.org/10.1007/978-3-031-98668-0_14

format employed in the recent hardware model checking competition has the potential to achieve the same in the field of hardware model checking.

The Hardware Model Checking Competition (HWMCC) has its roots in a rather lively discussion at the 2nd Alpine Verification Meeting (AVM) in 2006 among Daniel Kröning, Dirk Beyer and Armin Biere. The debated question was if and how model checking research as well as industrial applications can benefit from competitions in the same way the SAT competitions were instrumental in advancing SAT. Daniel Kröning and Armin Biere argued to focus on hardware gate-level models with simple and clear semantics.[1]

This argument prompted the development of the AIGER format [3] used in the first (hardware) model checking competition, affiliated to CAV'07. This first version of AIGER (20071012) came with a library for parsing and other essential tools, including a translator from SMV and BLIF to AIGER. The challenge of the first competitions in 2007, 2008 and 2010 was to collect benchmarks.

For the 2011 competition the first major revision of the AIGER 1.9 format [6] included *liveness* properties and *constraints*. The following competitions from 2012–2015 [8] and in 2017 [4] included a *deep bound track* to emphasize the common industrial practice of relying on incomplete but deep bounded model checking. In 2019 a word-level track was established based on the BTOR 2.0 format [22] proposed at CAV'18. After focusing on word-level in 2020 the organizers decided in 2024 [5] to reintroduce a bit-level track but take the chance to force all participating model checkers to produce certificates.

The introduction of mandatory certification in HWMCC'24 significantly impacted participation and competition dynamics. The 2024 competition saw a record nine participants, up from three in the previous edition, reflecting growing interest and accessibility. Discussions with participants revealed that new rules, *particularly the requirement for certification*, leveled the playing field by encouraging the development of verifiable solvers. Feedback indicated that participants successfully implemented certificate generation based on our published results [14,29–31]. It was also noted that implementing correct model checking algorithms demanded substantially more effort than generating certificates.

The certificate format itself has undergone several iterations with the ultimate aim of its use in the competition. In HWMCC'24, all participating model checkers were required to produce proofs alongside the model checking results for both safe and unsafe instances. For unsafe instances, the certificate is a trace serving as a counterexample, which can be validated via simulation; as for safe instances, it is a proof witness circuit. For the competition we use an extended version of the witness format defined in [14], that supports constraints, an essential feature of AIGER 1.9.

In this case study, we first describe the certificate format used in the competition, then present experimental findings. We investigate the overhead introduced by certificate checking in model checking and results show that it accounts for only *a fraction of the total verification time*. Moreover, we compare the

[1] Dirk Beyer proposed to use C as input language, which is much harder to master, due to its complex semantics. Accordingly the first SV-COMP took place in 2012.

certifying winner of HWMCC'24, RIC3, against the state-of-the-art model checker ABC which does not support certificate generation. Results show that RIC3 outperforms ABC even when including the time required for witness validation.

2 Related Work

Certification in other Competitions. Certification has been an essential part in many other competitions. In SAT competitions [13], certification has been mandatory for almost a decade, as a fundamental requirement. Solvers must produce certificates for both SAT and UNSAT instances: a satisfying truth assignment for SAT and a proof in the DRAT [28] format for UNSAT. A solver is disqualified from the main track if a singe certificate is found invalid. The software verification community is following suit. At SV-COMP'24, it is the second year of having a dedicated track for witness validation, with a range of participating witness validators [2]. The MaxSAT Evaluation [1] has also taken a step forward in 2024 by requesting proofs for the first time. In QBF Evaluations [25], there used to be a dedicated Evaluate & Certify track, where solvers are required to produce proofs that are easier to check than the solving task; however, as the organizers pointed out, only a few QBF solvers support certificate generation. SMT competitions (SMT-COMP) [27] and ATP System Competitions (CASC) [26] feature a wide variety of theories and have yet to adopt a universal certification standard. Classical Planning is similar to verification, but usually more focused on finding solutions (plans). Nevertheless, a deductive certificate format [11] has been introduced, and extended to support UNSAT certificates produced by an underlying SAT solver [10].

Related Work in Model Checking Certification. Deductive proof systems have been used for generating proofs of model checking. For example, the author of [21] addresses μ-calculus, while the authors of [15] focus on liveness and several pre-processing techniques. These approaches require model checkers to provide deductive proofs. The works in [7,16] explore the use of inductive invariants as certificates for k-induction. Notably, the certificate format employed in HWMCC'24 is also compatible with these inductive invariants. The authors of [18] use liveness-to-safety reduction techniques to certify liveness properties. The problem of certifying model checking has also been addressed in infinite-state systems [9,19] where SMT solvers are leveraged for unbounded state spaces. An alternative approach to providing certificates, is to formally verify the model checker itself, as demonstrate in [12].

3 Certificate Format

We assume the standard notions and terminology of Boolean logic. In the following, we consider hardware designs modeled as Boolean circuits encoded as sequential and-inverter graphs (AIGs) [3,6,17,20]. Such a Boolean circuit is given

as a tuple $M = (I, L, R, F, P, C)$ where I is an ordered set of inputs, L is an ordered set of latches, R defines the set of reset states, represented as a reset predicate that holds when every latch $l \in L$ equals its reset function r_l; F is the transition predicate that refers to two consecutive states, and encodes that each latch in one state is equal to its corresponding transition function f_l applied to the previous state; P and C are predicates that define the set of good states and the set of states valid under the *constraint*, respectively. These predicates, along with the reset and transition functions, are encoded in the circuits as binary AND-gates with possible negation at the incoming gates.

We use the notion of a reset predicate R being stratified; for space reasons we refer to [30] for formal definitions. In essence, it means that the dependencies that the reset functions introduce among the latches are acyclic. For $K \subseteq L$, $R\{K\}$ and $F\{K\}$ restrict these predicates so only the latches in K are required to be equal to their reset or transition. When referencing a sequence of states, we use indices on the predicates to represent the corresponding copy of the predicate at a certain state in the sequence.

A trace of length n is a sequence of $n + 1$ states, where the first state needs to satisfy R, every pair of consecutive states satisfies F (written $F_{i,i+1}$ for the i-th and its successor state) and all states satisfy the constraint C. If the last state violates P, the trace is bad. Thus a satisfying assignment to the following formula *certifies* that a circuit is *unsafe*:

$$R_0 \wedge \bigwedge_{i \in [0,n)} F_{i,i+1} \wedge \bigwedge_{i \in [0,n]} C_i \wedge \neg P_n.$$

For *safe* instances, the certificate format employed in HWMCC'24 takes the form of witness circuits, defined as follows.

Definition 1 (Witness Circuit). *The circuit $W = (I', L', R', F', P', C')$ is a witness circuit of $M = (I, L, R, F, P, C)$, if R' is stratified and for $K = L \cap L'$:*

1. *Reset:* $R\{K\} \wedge C \Rightarrow R'\{K\} \wedge C'$;
2. *Transition:* $F_{0,1}\{K\} \wedge C_0 \wedge C_1 \wedge C'_0 \Rightarrow F'_{0,1}\{K\} \wedge C'_1$;
3. *Property:* $(C \wedge C') \Rightarrow (P' \Rightarrow P)$;
4. *Base:* $R'\{L'\} \wedge C' \Rightarrow P'$;
5. *Step:* $P'_0 \wedge F'_{0,1}\{L'\} \wedge C'_0 \wedge C'_1 \Rightarrow P'_1$.

The five conditions described above are simple SAT checks. An additional polynomial-time check is required to verify that R' is stratified. If all checks pass, M' is a valid certificate for M, certifying its safety property. The first three conditions in Definition 1 establish a simulation relation between two circuits, such that if M' is safe, M is also safe. The intuition is that an initial state in the original circuit M corresponds to an initial state in the witness circuit, and each valid transition in M corresponds to a transition in M'.

Property P' is a strengthening of P. Consequently, safety of M' implies safety of M. In summary, a bad trace in M corresponds to a bad trace in M'. A sketch of the traces for both M and M' is provided in Fig. 1. The latter two checks

(Definition 1.4 and Definition 1.5) prove P' to be an inductive invariant, entailing the safety of M'. We provide a high-level intuitive illustration of Definition 1 in Fig. 1.

This is a slight extension to the format in [14], as it supports constraints and now covers all AIGER 1.9 [6] features except liveness. In HWMCC'24, witness circuits are also produced as AIGER files. The witness circuit validation is implemented in the certificate checker CERTIFAIGER[2] used for the competition, but has not been described in detail before. For efficient certification, CERTIFAIGER leverages the SAT solver Kissat 4.0.0, winner of the SAT competition 2024.

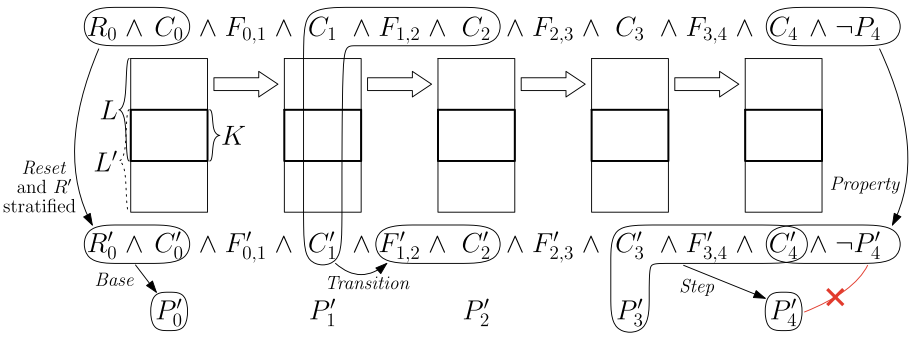

Fig. 1. An illustration for the correctness of Definition 1. Assuming that a circuit M with a valid witness M' has a bad trace leads to a contradiction. Depicted are the overlapping sets of variables and how conditions of the witness check are used to construct a bad trace in M', and arrive at a contradiction. For *Transition* and *Step* only one application is illustrated.

3.1 Soundness of the Certificate Format

We present a proof that the existence of a witness circuit as defined in Definition 1 indeed certifies the safety of a model. The proof extends what is presented in [14] by considering constraints.

Theorem 1. *Given two circuits M and M', with $M = (I, L, R, F, P, C)$, and $M' = (I', L', R', F', P', C')$. If M' is a valid witness circuit for M, then M is safe.*

Before proving the main theorem, we first introduce some additional notation: An assignment maps a subset of the gates to true or false, and is always consistent with the valuation of the AND-gates. Extending an assignment means assigning more gates while leaving previously assigned gates unchanged. We refer to the reset gate associated with latch l as r_l and the primed version r'_l, when referencing the reset gates used by R'.

[2] https://github.com/Froleyks/certifaiger.

Every gate g refers to the Boolean function defined by its fan-in cone, and we write $g(s)$ to denote that we consider the function under an assignment s, i.e., the variables in g which are assigned by s are replaced with the corresponding constants. A function g *semantically depends* on a variable v if an assignment exists under which $g(s_v)$ and $g(s_{\neg v})$ evaluate to different truth values.

We first show that a reset state in M corresponds to a reset state in M'.

Lemma 1. *For circuits $M = (I, L, R, F, P, C)$ and $M' = (I', L', R', F', P', C')$ satisfying the reset check (Definition 1.1) and R' stratified, any assignment to $I \cup L$ satisfying $R\{K\} \wedge C$, where $K = L \cap L'$, can be extended to satisfy $R'\{L'\} \wedge C'$.*

Proof. Assuming the reset check passes and R' is stratified, let s be an arbitrary but fixed assignment to $I \cup L$ satisfying $R\{K\} \wedge C$. The assumptions of the Lemma further imply that s satisfies $R'\{K\} \wedge C'$. To show that s can be extended to satisfy $R'\{L'\}$, we first prove for each latch $l \in K$, $r'_l(s)$ has no semantic dependency outside $(I \cup L) \cap (I' \cup L')$. Assume, for contradiction, there is a latch $l \in K$ with $r'_l(s) \not\Leftrightarrow r'_l(s_u)$ where s_u is the same as s except for the value of some gate $u \in (I' \cup L') \setminus (I \cup L)$. We have $l \not\Leftrightarrow r'_l(s_u)$ and therefore $R'\{K\}$ does not hold under s_u. However, u is not in $I \cup L$ and $R\{K\} \wedge C$ still evaluates to true under s_u, thus implying $R'\{K\}$, and leading to the desired contradiction.

Since R' is stratified, the semantic dependencies of the reset gates r'_l can be seen as a topologically sorted graph. Given the above result, when considering $r'_l(s)$, the remaining dependency graph can be sorted topologically such that the variables in $(I \cup L) \cap (I' \cup L')$ are at the bottom. Thus, s can be extended to satisfy $R'\{L'\}$ by assigning the remaining latches in the reverse of that order. The extended assignment still satisfies $R\{K\} \wedge C$ and thereby C'.

We can now move on to prove the correctness of the certificate format, i.e., the proof of the main Theorem 1. Refer to Fig. 1 for a visualization of the proof.

Proof. Suppose, for contradiction, M is unsafe. Then there is a bad trace of some finite length n in the form of an assignment to $n + 1$ copies of $I \cup L$ satisfying:

$$R_0\{L\} \wedge C_0 \wedge F_{0,1}\{L\} \wedge C_1 \wedge \cdots \wedge C_{n-1} \wedge F_{n-1,n}\{L\} \wedge C_n \wedge \neg P_n.$$

We extend this assignment to each copy of the gates in $I' \setminus I \cup L' \setminus L$ that satisfies:

$$R'_0\{L'\} \wedge C'_0 \wedge F'_{0,1}\{L'\} \wedge C'_1 \wedge \cdots \wedge C'_{n-1} \wedge F'_{n-1,n}\{L'\} \wedge C'_n \wedge \neg P'_n.$$

Let $X' = (I' \cup L') \setminus (I \cup L)$. The assignment satisfying $R_0\{K\} \wedge C_0$ can by Lemma 1 can be extended to X'_0 satisfying $R'_0\{L'\} \wedge C'$. With that and the transition check $F'_{0,1}\{K\} \wedge C'_1$ is satisfied and the assignment can be extended to X'_1 satisfying $F'_{0,1}\{L'\} \wedge C'_1$ by the definition of transition functions.

Applying the same argument n times yields an assignment to $(I \cup L \cup I' \cup L')^n$ satisfying $F'_{i,i+1}\{L\}$ for $i \in [0, n)$ and C_i for $i \in [0, n]$. Lastly, the property check guarantees $\neg P'_n$, giving us the desired assignment. However, the base and step check together ensure that the property P' holds on all reachable states of M', thus contradicting the initial assumption that a bad trace exists in M.

4 Evaluation

In this section, we present a comprehensive analysis of the competition results[3], focusing on the overhead of certificate generation and checking. Specifically, we address the following three questions:

1. What is the runtime overhead associated with validating certificates?
2. What is the space overhead associated with storing certificates?
3. How do certifying model checkers compare to the state of the art?

Experimental Setup. The 2024 competition ran on a cluster of 48 compute nodes equipped with an AMD Ryzen 9 7950X 16-core processor at 4.5 GHz and 128 GB or RAM, running Ubuntu 20.04 LTS. For fairness, the experiment described in Sect. 4.3 ran on the cluster used for the last competition in 2020. Each node has access to two Xeon E5-2620 v4 CPUs, for a total of 16 cores running at 2.1 GHz, and 128 GB of RAM.

We focus on (all) the 319 bit-level benchmarks of HWMCC'24, which were translated from the word-level (BTOR/bit-vector) track of HWMCC'24. The majority of the benchmarks (250) are new benchmarks submitted in 2024 by three different groups, including benchmarks for checking safety properties of open source RISC-V cores, sequential equivalence checking, branch coverage problems, as well as software verification problems, which were translated from SV-COMP'24 [2]. The remaining 69 benchmarks were selected randomly from previous competition years (2019 and 2020). Each model checker had exclusive access to a node, with a 120 GB memory limit and a one-hour wall-clock limit. A separate limit of 10 h was imposed for certificate checking.

Note that for precision and reliability of measurements, the competition cluster uses `runexec` to measure resource consumption of the model checkers. We further rely on it to properly isolate the processes and to enforce both the time and memory resource limits.

4.1 Certificate Checking Overhead

We now evaluate the overhead introduced by certificate checking. For each solver, we consider the model checking time, t_{MC}, the time required to validate the produced certificate, t_{CERT}, and the total time $t_{\text{TOTAL}} = t_{\text{MC}} + t_{\text{CERT}}$. The certificate checking overhead for a model checker refers to the additional time required to run all benchmarks when certification is enabled. Note that benchmarks unsolved by the model checker are excluded from this metric. The results are displayed in Fig. 2 where both safe and unsafe instances are considered.

The clear winner of the competition is RIC3, demonstrating superior performance on both safe and unsafe benchmarks. When considering only safe benchmarks, the ranking remains virtually unchanged, with FRIC3 narrowly outperforming SUPERCAR. As for unsafe instances, which constitute approximately 30%

[3] https://hwmcc.github.io/2024.

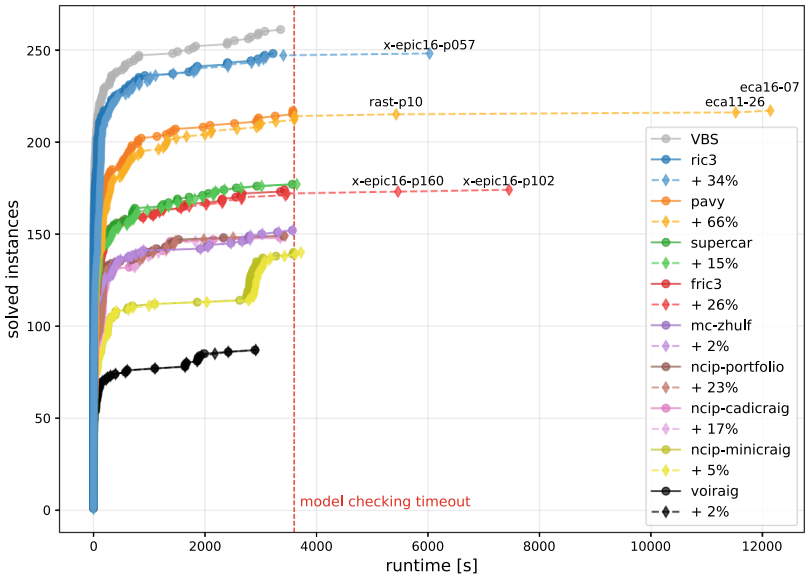

Fig. 2. HWMCC'24 results (319 benchmarks). The plots show the number of solved instances as a function of time. For each model checker, we present (1) the model checking time and (2) the total time for model checking and certificate validation. Diamonds represents the time taken to model check a circuit and validate the produced witness, while dots indicate model checking time only. Benchmarks whose certificates were especially time-consuming to verify are labeled. The Virtual Best Solver (VBS) indicates the top solver performance on each instance. The legend includes the overall certification overhead. The results clearly indicate, that certificate validation only adds minimal overhead.

of solved benchmarks, SUPERCAR slightly outperforms PAVY. In both scenarios, RIC3 maintains its lead and performs impressively close to the virtual best solver.

In Fig. 2, we also identify six outliers where the combined model checking and certification time exceeded the one-hour model checking timeout by more than 5%. The difficulty in their certification seems to be related to the witness circuit generation process *within* the model checker, as for each of these instances, another model checker found a witness circuit, which could be validated under 100 s. An exception is the x-epic16-p057 benchmark, which was solved exclusively by RIC3. Certificate checking never exceeded the 10-hour limit.

As Fig. 2 shows, the overall certification overhead only gives rise to a small fraction of the model checking time, which is highly promising and highlights the effectiveness of the certificate format. For instance, when using RIC3 to model check all 248 instances it solved, the total time is increased by only 34% when all produced certificates are validated. In general, validating certificates for safe instances is a more challenging task than validating simulation traces for unsafe ones, a trend similar as in SAT solving. In fact, simulation time accounts for only 2% of the overhead in RIC3, and even less for all other solvers.

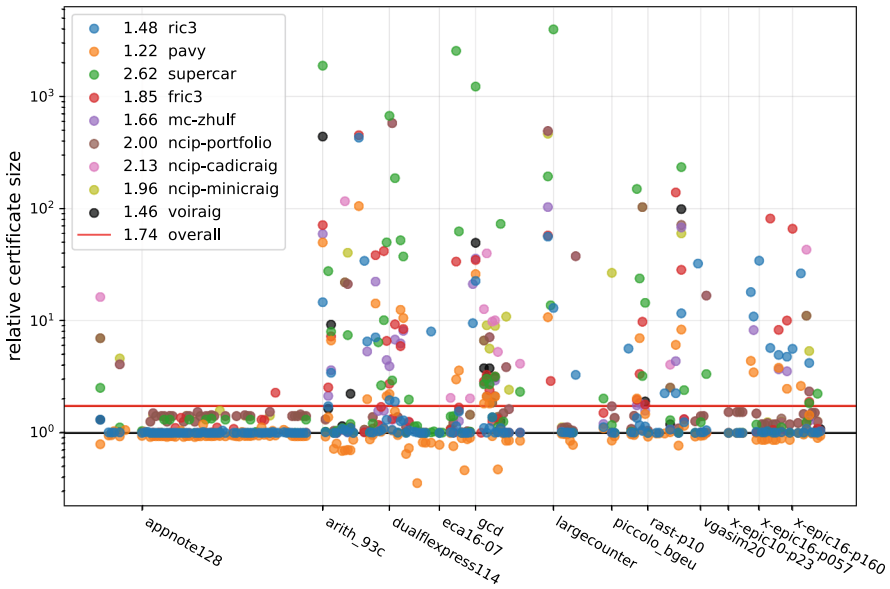

Fig. 3. Size of produced witness circuits relative to their original model circuit. The x-axis represents the set of benchmarks, sorted alphabetically, whereas the y-axis indicates the certificate size (gates) relative to the model. The legend also shows the geometric mean of the relative certificate size for each model checker and all produced certificates combined. Dots stacked vertically correspond to the same benchmark. Since the x-axis is sorted by benchmark name, neighboring instances are likely to belong to the same family. For clarity and space reasons, only a select few benchmarks are explicitly labeled. We observe an overall relative certificate size of 1.74, which indicates the compactness of the certificates.

4.2 Certificate Size

Next, we evaluate the size of witness circuits for safe instances, where circuit size is measured in terms of gates, which includes the number of inputs, latches, and AND-gates. The relative certificate size is defined as $\frac{\text{certificate size}}{\text{model size}}$. Figure 3 presents the relative certificate sizes for all solved instances. Note that appnote and x-epic families, comprising 52 and 13 benchmarks respectively, depicted in the plot, include several multi-property benchmarks. In these cases, the benchmarks represent the same model, differing only in the property to be checked.

We observe that PAVY produces smallest witnesses, with a geometric mean ratio of 1.22, whereas SUPERCAR exhibits the highest ratio of 2.62. Overall, more than 80% of the produced witnesses are less than twice as large as the certified model, with a geometric mean ratio of 1.74 across all produced witness circuits.

It further turns out that PAVY consistently generates witnesses substantially smaller than their corresponding models. Notably, this was not possible in earlier versions of the certificate format [29–31], which required the entire model to be embedded within the witness circuit. The original format was revised in [14]

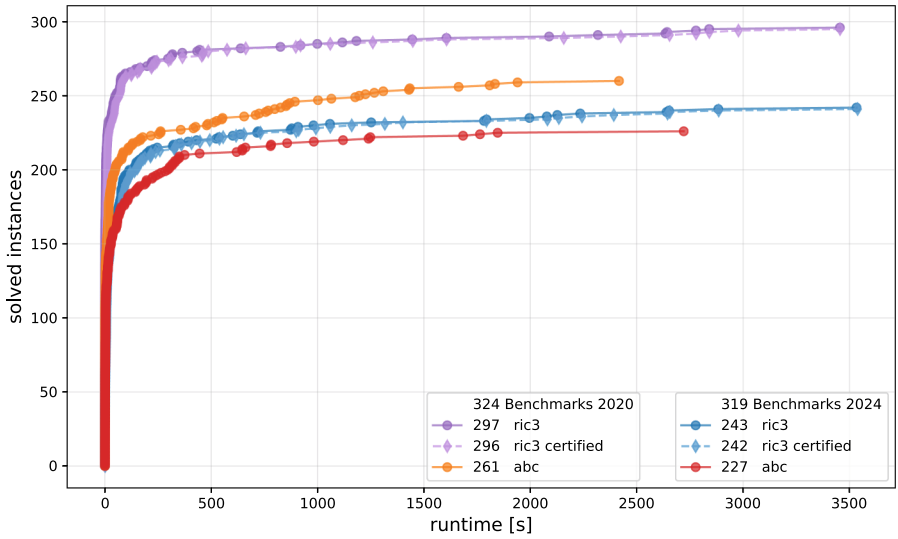

Fig. 4. Comparing RIC3 (2024 winner) with ABC (2020 winner). The same HWMCC'20 hardware setup is used, for both benchmarks sets (2020 and 2024). RIC3 is also run in a fully certified mode, where each result is confirmed by checking the certificate. Note that for these certified runs the shown run-time not only includes model checking time but also certificate production and certificate checking time. We observe that on both sets RIC3 consistently outperforms ABC, even when accounting for certificate validation time.

and went through another update for the competition, which is described in Sect. 3. This version allows, beside constraints, optimized witnesses that focus on a subset of the certified model, enabling significant reduction in witness size.

Witness size only correlates weakly with validation time. The biggest witness, produced by SUPERCAR for the largecounter benchmark, contains over 7 million gates for a model with fewer than 2 thousand gates, yet is verified within 900 s, which is 30% faster than model checking. Conversely, the two difficult-to-check eca witnesses produced by PAVY are 20% smaller than the model.

4.3 Comparison to State of the Art

Since participating model checkers in HWMCC'24 generate certificates, i.e., are *certifying model checkers*, it remains to show data on how certificate generation affects solver speed. We thus compare RIC3, the HWMCC'24 winner, with the state-of-the-art model checker ABC, the winner of the previous HWMCC edition in 2020, where witness circuits were not yet introduced. It is worth noting that the industrial-strength model checker ABC has dominated the bit-level track of the HWMCC since its debut in 2008. However, it could not participate in the 2024 competition, as certificates are now mandatory.

For our experiment, we use the version of ABC, which was submitted to the HWMCC'20, and was tailored specifically for the competition thus distinct from its public releases. To ensure that ABC is used with the same hardware specifications as expected by the participants in 2020 we run our experiment on the HWMCC'20 hardware. Note that this hardware is significantly older than the cluster used for HWMCC'24.

Note that the benchmarks from HWMCC'20 and HWMCC'24 were both included (there was no competition in between). The two sets are mostly distinct with only 8 benchmarks in common. This is following the SAT competition practice: HWMCC uses mostly new benchmarks every year, adhering to the SAT Practitioner's Manifesto.

Figure 4 shows that RIC3 convincingly outperforms ABC on both benchmark sets. Notably, in 2020, RIC3 solves 36 more benchmarks and is faster on 247 out of the 256 benchmarks solved by both model checkers. For RIC3, we also include a certified version, which represents its performance if it did internal certificate validation, and every benchmark is only reported as solved after the certificate has been successfully validated. Even in its certified mode, RIC3 still holds a clear lead, losing only one instance per year due to certificate validation exceeding the remaining time before the one-hour model checking timeout.

One minor exception is the performance on the 2024 benchmarks within the first 30 s, where the certificate checking adds a significant enough overhead for ABC to catch up to the certified version. Nevertheless, certificate production introduces *no measurable overhead* to overall model checking performance. These results demonstrate that RIC3, is a robust and efficient model checker, that presents superior performance while providing added benefits of certifying.

Invalid Certificates. In HWMCC'24 and the experiments presented above, producing an invalid certificate causes the benchmark to count as unsolved. Out of the 1536 certificates generated during the competition, 44 were found to be incorrect. They were produced by four model checkers: SUPERCAR (20), NCIP-MINICRAIG (9), NCIP-PORTFOLIO (8), FRIC3 (7). The incorrect certificates produced by SUPERCAR are all simulation traces, notably 8 of them are for benchmarks which have been proven *safe* by other model checkers. In addition to 3 more incorrect simulation traces from FRIC3, all other invalid certificates were witness circuits failing one of the checks outlined in Definition 1.

Many of the invalid certificates stemmed from bugs uncovered by the organizers before the competition through extensive fuzz testing. The fuzzer and subsequent delta-debugging helped identify minimal failing circuits, shared subsequently with the model checker developers for fixes. Initially, all model checkers produced invalid certificates. After extensive feedback, most solvers passed thousands of fuzzer-generated test cases with correct certificates. This process highlights the benefits of certifying model checkers to improve their robustness.

Summary of Results. Our experimental evaluation entails the following key findings. (i) Minimal overhead: certification adds only a small runtime overhead, representing a fraction of the total model checking time. (ii) Compact certificates: optimized certificate formats reduced storage requirements, with over 80% of

certificates being less than twice the size of the certified model. (iii) Impact on performance: RIC3, the 2024 winner, outperformed the 2020 winner ABC, even when all certificates are verified, demonstrating that certifying approaches can simultaneously provide correctness guarantees and strong performance.

5 Conclusion

HWMCC'24 marks the first time that the Hardware Model Checking Competition has mandated certification for all participating solvers. Our case study confirms that certification can be integrated with minimal overhead while significantly improving confidence in verification results, illustrating the practical benefits of mandatory certification in hardware model checking.

Looking ahead, we call on more participants and model checker developers—both in academia and industry—to adopt and support certification. Building on the success of HWMCC'24, we intend to extend certification to the word-level track, for which a certificate checker CERBOTOR is already publicly available. However, challenges remain, including the need to develop techniques for generating certificates tailored to word-level-specific methods and addressing the use of trustworthy SMT solvers, which require SMT-based certificates.

On the other hand, a certifying liveness track is under planning, although this endeavor requires certificate generation for liveness checking algorithms, which remains another open research challenge. Another direction concerns the degree of trust we can place in the certificate checker. Ultimately, achieving a fully verified certificate checker would ensure an end-to-end correctness in the verification process, further increasing confidence.

Beyond increasing trust in model checkers, certificates have broader applications. An ongoing industry collaboration explores the integration of certifying model checkers as hammers in interactive theorem provers such as Isabelle [23] via Sledgehammer [24]. This entails, the theorem prover encoding an open proof as a model checking problem, invoking a model checker, and lifting the certificate back into the theorem prover.

Acknowledgements. This work is supported in part by the ERC-2020-AdG 101020093, the LIT AI Lab funded by the State of Upper Austria, the Research Council of Finland under the project 336092, and a gift from Intel Corporation.

Furthermore we of course also owe a big thank-you to the submitters of model checkers and benchmarks to the competition over all these years. Without their enthusiasm and support neither the competition nor this study would exist.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this paper.

References

1. Bacchus, F., Berg, J., Järvisalo, M., Martins, R.: MaxSAT evaluation 2020: solver and benchmark descriptions (2020)

2. Beyer, D.: State of the art in software verification and witness validation: SV-COMP 2024. In: Finkbeiner, B., Kovács, L. (eds.) TACAS 2024. LNCS, vol. 14572, pp. 299–329. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-57256-2_15
3. Biere, A.: The AIGER And-Inverter Graph (AIG) format version 20071012. Technical report, 07/1, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria (2007). https://doi.org/10.35011/fmvtr.2007-1
4. Biere, A., van Dijk, T., Heljanko, K.: Hardware model checking competition 2017. In: Stewart, D., Weissenbacher, G. (eds.) Formal Methods in Computer-Aided Design, FMCAD 2017, Vienna, Austria, 02–06 October 2017, p. 9. IEEE (2017)
5. Biere, A., Froleyks, N., Preiner, M.: Hardware model checking competition 2024. In: Narodytska, N., Rümmer, P. (eds.) Proceedings 24th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2024), p. 7. TU Wien Academic Press (2024). https://doi.org/10.34727/2024/isbn.978-3-85448-065-5_6
6. Biere, A., Heljanko, K., Wieringa, S.: AIGER 1.9 and beyond. Technical report, 11/2, Institute for Formal Models and Verification, Johannes Kepler University, Altenbergerstr. 69, 4040 Linz, Austria (2011). https://doi.org/10.35011/fmvtr.2011-2
7. Bjørner, N., Gurfinkel, A., McMillan, K., Rybalchenko, A.: Horn clause solvers for program verification. In: Beklemishev, L.D., Blass, A., Dershowitz, N., Finkbeiner, B., Schulte, W. (eds.) Fields of Logic and Computation II. LNCS, vol. 9300, pp. 24–51. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23534-9_2
8. Cabodi, G., et al.: Hardware model checking competition 2014: an analysis and comparison of solvers and benchmarks. J. Satisf. Boolean Model. Comput. **9**(1), 135–172 (2014). https://doi.org/10.3233/SAT190106
9. Conchon, S., Mebsout, A., Zaïdi, F.: Certificates for parameterized model checking. In: Bjørner, N., de Boer, F. (eds.) FM 2015. LNCS, vol. 9109, pp. 126–142. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19249-9_9
10. Eriksson, S., Helmert, M.: Certified unsolvability for SAT planning with property directed reachability. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 30, pp. 90–100 (2020). https://doi.org/10.1609/icaps.v30i1.6649
11. Eriksson, S., Röger, G., Helmert, M.: Unsolvability certificates for classical planning. In: Barbulescu, L., Frank, J., Mausam, Smith, S.F. (eds.) Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, 18–23 June 2017, pp. 88–97. AAAI Press (2017)
12. Esparza, J., Lammich, P., Neumann, R., Nipkow, T., Schimpf, A., Smaus, J.G.: A fully verified executable LTL model checker. Arch. Formal Proofs **2014** (2014)
13. Froleyks, N., Heule, M., Iser, M., Järvisalo, M., Suda, M.: SAT competition 2020. Artif. Intell. **301**, 103572 (2021)
14. Froleyks, N., Yu, E., Biere, A., Heljanko, K.: Certifying phase abstraction. In: Benzmüller, C., Heule, M.J.H., Schmidt, R.A. (eds.) IJCAR 2024. LNCS, vol. 14739, pp. 284–303. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-63498-7_17
15. Griggio, A., Roveri, M., Tonetta, S.: Certifying proofs for SAT-based model checking. Formal Methods Syst. Des. **57**(2), 178–210 (2021). https://doi.org/10.1007/s10703-021-00369-1
16. Gurfinkel, A., Ivrii, A.: K-induction without unrolling. In: 2017 Formal Methods in Computer Aided Design (FMCAD), pp. 148–155. IEEE (2017)

17. Kuehlmann, A., Paruthi, V., Krohm, F., Ganai, M.K.: Robust Boolean reasoning for equivalence checking and functional property verification. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **21**(12), 1377–1394 (2002). https://doi.org/10.1109/TCAD.2002.804386
18. Kuismin, T., Heljanko, K.: Increasing confidence in liveness model checking results with proofs. In: Bertacco, V., Legay, A. (eds.) HVC 2013. LNCS, vol. 8244, pp. 32–43. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03077-7_3
19. Mebsout, A., Tinelli, C.: Proof certificates for SMT-based model checkers for infinite-state systems. In: FMCAD, pp. 117–124. IEEE (2016)
20. Mishchenko, A., Chatterjee, S., Brayton, R.K.: DAG-aware AIG rewriting a fresh look at combinational logic synthesis. In: Sentovich, E. (ed.) Proceedings of the 43rd Design Automation Conference, DAC 2006, San Francisco, CA, USA, 24–28 July 2006, pp. 532–535. ACM (2006). https://doi.org/10.1145/1146909.1147048
21. Namjoshi, K.S.: Certifying model checkers. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 2–13. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44585-4_2
22. Niemetz, A., Preiner, M., Wolf, C., Biere, A.: Btor2 , BtorMC and Boolector 3.0. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10981, pp. 587–595. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_32
23. Nipkow, T., Wenzel, M., Paulson, L.C.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. Springer, Heidelberg (2002)
24. Paulson, L., Nipkow, T.: The Sledgehammer: let automatic theorem provers write your Isabelle scripts (2023)
25. Pulina, L., Seidl, M.: The 2016 and 2017 QBF solvers evaluations (QBFEVAL'16 and QBFEVAL'17). Artif. Intell. **274**, 224–248 (2019)
26. Sutcliffe, G.: Proceedings of the 12th IJCAR ATP System Competition (CASC-J12) (2024)
27. Weber, T., Conchon, S., Déharbe, D., Heizmann, M., Niemetz, A., Reger, G.: The SMT competition 2015–2018. J. Satisfiability Boolean Model. Comput. **11**(1), 221–259 (2019)
28. Wetzler, N., Heule, M., Hunt, W.A.: DRAT-trim: efficient checking and trimming using expressive clausal proofs. In: Sinz, C., Egly, U. (eds.) SAT 2014. LNCS, vol. 8561, pp. 422–429. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09284-3_31
29. Yu, E., Biere, A., Heljanko, K.: Progress in certifying hardware model checking results. In: Silva, A., Leino, K. (eds.) CAV 2021. LNCS, vol. 12760, pp. 363–386. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81688-9_17
30. Yu, E., Froleyks, N., Biere, A., Heljanko, K.: Stratified certification for K-Induction. In: Griggio, A., Rungta, N. (eds.) 22nd Formal Methods in Computer-Aided Design, FMCAD 2022, Trento, Italy, 17–21 October 2022, pp. 59–64. IEEE (2022). https://doi.org/10.34727/2022/ISBN.978-3-85448-053-2_11
31. Yu, E., Froleyks, N., Biere, A., Heljanko, K.: Towards compositional hardware model checking certification. In: Nadel, A., Rozier, K.Y. (eds.) Formal Methods in Computer-Aided Design, FMCAD 2023, Ames, IA, USA, 24–27 October 2023, pp. 1–11. IEEE (2023). https://doi.org/10.34727/2023/ISBN.978-3-85448-060-0_12