# Performance Dataset for Hardware Model Checking on BTOR2 Benchmarks (Technical Report, May 2025)

Po-Chun Chien<sup>1</sup>, Nian-Ze Lee<sup>2,1</sup>, and Zhengyang Lu<sup>3</sup>

<sup>1</sup>LMU Munich, Munich, Germany <sup>2</sup>National Taiwan University, Taipei, Taiwan <sup>3</sup>University of Waterloo, Waterloo, Canada

#### 1 Overview

This technical report presents a performance evaluation of several hardware modelchecking tools on a collection of benchmark tasks in the BTOR2 format [25]. The resulting dataset is intended to support machine-learning research for hardware model checking, particularly in areas such as algorithm selection [27], performance prediction [31], and automated tool configuration [17]. It has been used, for example, in the development and evaluation of BTOR2-SELECT,<sup>1</sup> a machine-learningbased algorithm-selection framework for hardware model checking [18, 19].

To construct the dataset, we benchmarked a diverse set of model-checking tools and algorithmic configurations. Each verification engine was evaluated on a common set of BTOR2 tasks and the performance measurements, including CPU time, wall time, and memory usage, were collected.

The report is organized as follows: Sect. 2 outlines the sources of the BTOR2 benchmark tasks; Sect. 3 introduces the evaluated model-checking engines and their configurations; Sect. 4 describes the benchmarking environment; and Sect. 5 summarizes the collected performance data. All data, including scripts and files required to reproduce the experiments, are publicly available at: https://gitlab.com/sosy-lab/research/data/perf-eval-hwmc/-/tree/1.0.

## 2 BTOR2 Benchmark Set

We collected verification tasks in BTOR2 format from a wide range of domains to capture diverse problem characteristics. The sources include the benchmark sets used in Hardware Model Checking Competitions (HWMCC) in 2019, 2020, and 2024 [1,6], as well as other problem sets such as parametrized model-checking tasks obtained from the BEEM project [26], problems derived from hardware refinement checking [14], random circuits generated by FUZZBTOR2 [30], and the translated reachability-safety tasks from SV-COMP 2024 [2,10]. Following the same categorization used in HWMCC, the collected benchmarks were divided

<sup>&</sup>lt;sup>1</sup> https://gitlab.com/sosy-lab/software/btor2-select

			/			
Input	Tool	Version	Algorithms			
AIGER	$ABC^{bv}$	af1de4fa	BMC, IMC, PDR, sc-IMC, sc-PD			
AIGEN	$RIC3^{bv}$	HWMCC24	BMC, KI, IC3			
Btor2	AVR	bdbfc83b	BMC, KI, IC3sa			
	BtorMC	6603 ed7	BMC, KI, KI-sp			
	Pono	508b380	BMC, KI-sp, IMC, IC3 <sup>bv</sup> , IC3ia			
С	CBMC	5.95.1	BMC, KI			
	Esbmc	7.4.0	BMC, $KI^{bv}$			

Table 1: Benchmarked verification tools and algorithms (tools or algorithms that are only benchmarked on the **bitvec** set are marked with bv)

into two sets: (1) bitvec set: tasks with only bit-vector sorts, and (2) array set: tasks involving both bit-vector and array sorts. In total, the benchmark set contains 3 498 BTOR2 tasks, comprising 1977 tasks in the bitvec set and 1521 tasks in the array set.

### 3 Model-Checking Engines

We evaluated a selection of state-of-the-art open-source symbolic model checkers to obtain the performance data. Many of these tools have been the winners or top contenders in recent editions of HWMCC and SV-COMP. Table 1 provides an overview of the tools, their supported input formats, versions, and the evaluated verification algorithms.

The following tools were benchmarked on the collected BTOR2 tasks: (1) bitlevel hardware model checkers for AIGER: ABC [9] and RIC3 [29], (2) word-level hardware model checkers for BTOR2: AVR [16], BTORMC [25], and PONO [20], and (3) software verifiers for C programs: CBMC [12] and ESBMC [22]. Task translators BTOR2AIGER [24] (at commit a1bb6932) and BTOR2C [4] (at commit 7cf65f00) are bundled with the verifiers for AIGER and C, respectively, to translate the BTOR2 tasks to the input formats of these tools.

For each verifier, multiple verification algorithms were evaluated. These include BMC [7], k-induction (KI) [28], interpolation-based model checking (IMC) [21], and IC3/PDR [8,13]. Some algorithms may incorporate additional techniques. For example, k-induction can be enhanced with simple-path constraints (denoted with a suffix "-sp"). IC3 can be combined with syntax-guided abstraction [15] or implicit predicate abstraction [11] (denoted with a suffix "sa" and "ia", respectively). ABC provides logic-optimization techniques such as speculative reduction [23] (denoted with a prefix "sc-"), which can be utilized to reduce the circuit size before verification. Certain verifier-algorithm combinations were executed only on the bitvec set (as indicated in Table 1), if they do not support arrays in BTOR2 or produced incorrect verification results on tasks involving arrays.

#### 4 Benchmarking Setup and Environment

For each pair of a BTOR2 task and a verification engine (i.e., a tool running a specific algorithm), we collected the produced verification results, along with the consumed CPU time, walltime, and memory. All executions were conducted on machines running Ubuntu 24.04 (64-bit, Linux 6.8.0), each equipped with a 3.4 GHz CPU (Intel Xeon E3-1230 v5) with 8 processing units and 33 GB of RAM. Each run was limited to a single CPU core, 900 s of CPU time, and 15 GB of memory. To facilitate large-scale and reliable performance benchmarking, the platform BENCHCLOUD [3] was employed, which leverages BENCHEXEC [5] to accurately limit and measure resource usage.

#### 5 Summary of Performance Data

In addition to the collected performance data, we computed the PAR-2 score of each benchmark run based on CPU time. The scoring scheme assigns a penalty of twice the CPU-time limit (1800s) to each run that either failed or ran out of resources, whereas successful runs were assigned their actual CPU time. The PAR-2 score emphasizes both correctness and efficiency by penalizing unsuccessful or slow runs and rewarding faster ones.

On the **bitvec** benchmarks, ABC·sc-PDR was the single best solver (SBS) in term of PAR-2, correctly solving 1306 out of 1977 tasks, while the virtual best solver (VBS), which represents the hypothetical oracle choosing the best tool per instance, solved 1541 tasks. For the **array** benchmarks, AVR·KI was the SBS, correctly solving 797 out of 1521 tasks, whereas the VBS solved 943. These SBS-VBS gaps underscore the potential of machine-learning approaches, such as algorithm selection, to close the performance gap by leveraging complementary strengths among tools.

We provide an overview of the performance data obtained from benchmarking model-checking engines (see Sect. 3) on the collected BTOR2 verification tasks (see Sect. 2). The benchmarks suite was partitioned into training (80%) and testing sets (20%) for the evaluation of downstream machine-learning models. Tables 2 and 3 summarize the performance of each verification engine on the **bitvec** and **array** tasks, respectively. The statistics include the number of solved tasks, the average PAR-2 score, the number of tasks each engine contributes to the VBS, and the number of uniquely-solved tasks. Additionally, Figs. 1 and 2 show the cactus plots comparing the VBS and each verifier's best-performing algorithm on the corresponding benchmark set. A data point (x, y) in the plot indicates that there are x instances, each of which can be solved by the respective engine within a time bound of y seconds.

4

	Alg.	Training set (1582)				Tosting set (305)			
Tool		#solved	PAR-2 <sub>avg</sub>	#in-VB	#uniq	#solved	PAR-2 <sub>avg</sub>	#in-VB	#uniq
VBS	-	1249	<b>412.4</b>	-	-	292	498.9	-	-
ABC	BMC	374	1384.7	0	0	79	1445.4	0	0
	IMC	837	869.6	35	3	192	946.7	4	0
	PDR	1 0 2 2	674.2	112	2	240	744.3	22	1
	sc-IMC	869	839.7	31	1	206	889.2	13	1
	sc-PDR	1055	638.1	127	2	251	695.9	37	1
rIC3	BMC	422	1332.3	111	3	93	1392.8	21	1
	IC3	1 0 4 0	652.2	114	8	241	733.2	30	2
	KI	694	1029.7	64	2	155	1108.2	16	0
AVR	BMC	404	1354.7	31	0	86	1418.7	2	0
	IC3sa	813	900.6	167	19	190	954.7	34	<b>5</b>
	KI	656	1069.3	4	0	144	1156.2	0	0
	BMC	383	1373.6	113	0	81	1436.5	25	0
BTORMC	KI	538	1201.9	55	0	123	1249.4	13	0
	KI-sp	695	1024.7	67	0	153	1114.9	14	0
RIC3 AVR BTORMC PONO	BMC	398	1363.9	65	0	84	1428.3	15	0
	IC3	627	1107.0	77	0	142	1168.6	19	0
	IC3ia	618	1115.4	16	0	144	1161.6	3	0
	IMC	535	1204.3	7	0	121	1263.5	3	0
	KI-sp	759	955.2	38	0	170	1047.5	11	0
Свмс	BMC	373	1384.5	13	0	85	1423.5	10	0
	KI	586	1150.9	2	1	131	1222.7	0	0
Fanua	BMC	343	1419.4	0	0	73	1476.9	0	0
ESBMC	KI	441	1308.3	0	0	94	1381.4	0	0

Table 2: Summary of each verifier's performance on 1977 bitvec tasks



Fig. 1: Cactus plot of VBS and each verifier's best-performing algorithm on  $1\,977$  <code>bitvec</code> tasks

Tool	Alg.	Training set (1.217)				Testing set $(304)$			
		#solved	PAR-2av	$_{\rm g}$ #in-VE	$3 \ \#uniq$	#solved	PAR-2avg	$_{g} \# \text{in-VB}$	#uniq
VBS	-	756	700.1	-	-	187	712.9	-	-
AVR	BMC	352	1305.6	40	0	73	1392.9	7	0
	IC3sa	445	1165.6	85	<b>46</b>	110	1168.0	28	11
	KI	<b>644</b>	872.8	17	1	153	918.8	1	0
BTORMC	BMC	370	1261.6	110	0	74	1369.9	20	0
	KI	606	927.9	133	0	143	979.1	36	0
	KI-sp	611	921.3	73	0	143	976.4	22	0
Ρονο	BMC	377	1250.6	209	2	79	1343.9	42	2
	IC3ia	365	1290.8	23	1	92	1289.9	8	0
	IMC	404	1234.8	3	1	100	1247.5	0	0
	KI-sp	612	916.8	57	9	149	936.7	19	3
Свмс	BMC	111	1640.8	6	0	17	1701.9	4	0
	KI	245	1458.0	0	0	57	1484.1	0	0
ESBMC	BMC	90	1668.9	0	0	15	1711.7	0	0

Table 3: Summary of each verifier's performance on 1521 array tasks



Fig. 2: Cactus plot of VBS and each verifier's best-performing algorithm on  $1\,521$  array tasks

# References

- Hardware model-checking competition (HWMCC). https://hwmcc.github.io/, accessed: 2025-01-14
- Beyer, D.: State of the art in software verification and witness validation: SV-COMP 2024. In: Proc. TACAS (3). pp. 299–329. LNCS 14572, Springer (2024). https://doi.org/10.1007/978-3-031-57256-2\_15
- Beyer, D., Chien, P.C., Jankola, M.: BENCHCLOUD: A platform for scalable performance benchmarking. In: Proc. ASE. pp. 2386–2389. ACM (2024). https: //doi.org/10.1145/3691620.3695358
- Beyer, D., Chien, P.C., Lee, N.Z.: Bridging hardware and software analysis with BTOR2C: A word-level-circuit-to-C translator. In: Proc. TACAS (2). pp. 152–172. LNCS 13994, Springer (2023). https://doi.org/10.1007/978-3-031-30820-8\_12
- Beyer, D., Löwe, S., Wendler, P.: Reliable benchmarking: Requirements and solutions. Int. J. Softw. Tools Technol. Transfer 21(1), 1–29 (2019). https://doi.org/10. 1007/s10009-017-0469-y
- Biere, A., Froleyks, N., Preiner, M.: Hardware model checking competition 2024. In: Proc. FMCAD. pp. 7–7. TU Wien Academic Press (2024). https://doi.org/ 10.34727/2024/isbn.978-3-85448-065-5\_6
- Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Adv. Comput. 58, 117–148 (2003). https://doi.org/10.1016/S0065-2458(03) 58003-2
- Bradley, A.R.: SAT-based model checking without unrolling. In: Proc. VM-CAI. pp. 70-87. LNCS 6538, Springer (2011). https://doi.org/10.1007/ 978-3-642-18275-4\_7
- Brayton, R., Mishchenko, A.: ABC: An academic industrial-strength verification tool. In: Proc. CAV. pp. 24–40. LNCS 6174, Springer (2010). https://doi.org/10. 1007/978-3-642-14295-6\_5
- Chien, P.C., Lee, N.Z.: CPV: A circuit-based program verifier (competition contribution). In: Proc. TACAS (3). pp. 365-370. LNCS 14572, Springer (2024). https://doi.org/10.1007/978-3-031-57256-2\_22
- Cimatti, A., Griggio, A., Mover, S., Tonetta, S.: IC3 modulo theories via implicit predicate abstraction. In: Proc. TACAS. pp. 46–61. LNCS 8413, Springer (2014). https://doi.org/10.1007/978-3-642-54862-8\_4
- Clarke, E.M., Kröning, D., Lerda, F.: A tool for checking ANSI-C programs. In: Proc. TACAS. pp. 168–176. LNCS 2988, Springer (2004). https://doi.org/10. 1007/978-3-540-24730-2\_15
- Eén, N., Mishchenko, A., Brayton, R.K.: Efficient implementation of property directed reachability. In: Proc. FMCAD. pp. 125–134. FMCAD Inc. (2011). https: //dl.acm.org/doi/10.5555/2157654.2157675
- Fang, W., Hu, G., Zhang, H.: r-map: Relating implementation and specification in hardware refinement checking. IEEE Trans. on CAD of Integrated Circuits and Systems 42(12), 5113-5126 (2023). https://doi.org/10.1109/TCAD.2023.3294454
- Goel, A., Sakallah, K.: Model checking of Verilog RTL using IC3 with syntax-guided abstraction. In: Proc. NFM. pp. 166–185. Springer (2019). https://doi.org/10. 1007/978-3-030-20652-9\_11
- Goel, A., Sakallah, K.: AVR: Abstractly verifying reachability. In: Proc. TACAS. pp. 413–422. LNCS 12078, Springer (2020). https://doi.org/10.1007/ 978-3-030-45190-5\_23

- Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proc. LION. pp. 507–523. LNCS 6683, Springer (2011). https://doi.org/10.1007/978-3-642-25566-3\_40
- Lu, Z., Chien, P.C., Lee, N.Z., Ganesh, V.: Algorithm selection for word-level hardware model checking (student abstract). In: Proc. AAAI. vol. 39, pp. 29426– 29427 (2025). https://doi.org/10.1609/aaai.v39i28.35275
- Lu, Z., Chien, P.C., Lee, N.Z., Gurfinkel, A., Ganesh, V.: BTOR2-SELECT: Machine learning based algorithm selection for hardware model checking. In: Proc. CAV. Springer (2025)
- Mann, M., Irfan, A., Lonsing, F., Yang, Y., Zhang, H., Brown, K., Gupta, A., Barrett, C.W.: PONO: A flexible and extensible SMT-based model checker. In: Proc. CAV. pp. 461–474. LNCS 12760, Springer (2021). https://doi.org/10.1007/ 978-3-030-81688-9\_22
- McMillan, K.L.: Interpolation and SAT-based model checking. In: Proc. CAV. pp. 1– 13. LNCS 2725, Springer (2003). https://doi.org/10.1007/978-3-540-45069-6\_1
- Menezes, R., Aldughaim, M., Farias, B., Li, X., Manino, E., Shmarov, F., Song, K., Brauße, F., Gadelha, M.R., Tihanyi, N., Korovin, K., Cordeiro, L.: ESBMC v7.4: Harnessing the power of intervals (competition contribution). In: Proc. TACAS (3). pp. 376–380. LNCS 14572, Springer (2024). https://doi.org/10.1007/ 978-3-031-57256-2\_24
- Mony, H., Baumgartner, J., Mishchenko, A., Brayton, R.: Speculative reductionbased scalable redundancy identification. In: Proc. DATE. pp. 1674–1679. IEEE (2009). https://doi.org/10.1109/DATE.2009.5090932
- 24. Niemetz, A., Preiner, M., Wolf, C., Biere, A.: Source-code repository of BTOR2 tools. https://github.com/hwmcc/btor2tools, accessed: 2025-05-01
- Niemetz, A., Preiner, M., Wolf, C., Biere, A.: BTOR2, BTORMC, and BOOLECTOR 3.0. In: Proc. CAV. pp. 587–595. LNCS 10981, Springer (2018). https://doi.org/ 10.1007/978-3-319-96145-3\_32
- Pelánek, R.: BEEM: Benchmarks for explicit model checkers. In: Proc. SPIN. pp. 263–267. LNCS 4595, Springer (2007). https://doi.org/10.1007/978-3-540-73370-6\_17
- Rice, J.R.: The algorithm selection problem. Adv. Comput. 15, 65–118 (1976). https://doi.org/10.1016/S0065-2458(08)60520-3
- Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a SAT-solver. In: Proc. FMCAD, pp. 127–144. LNCS 1954, Springer (2000). https://doi.org/10.1007/3-540-40922-X\_8
- Su, Y., Yang, Q., Ci, Y.: Predicting lemmas in generalization of IC3. In: Proc. DAC. ACM (2024). https://doi.org/10.1145/3649329.3655970
- Xiao, S., Zhang, C., Li, J., Pu, G.: FUZZBTOR2: A random generator of word-level model checking problems in BTOR2 format. In: Proc. TACAS. pp. 36–43. Springer (2023). https://doi.org/10.1007/978-3-031-30820-8\_5
- Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. 32, 565-606 (2008). https: //doi.org/10.1613/JAIR.2490