Hardware Model Checking Competition 2024

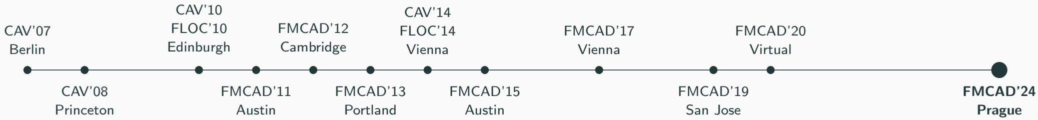
(12th Edition)

Mathias Preiner Nils Froleyks Armin Biere

https://hwmcc.github.io/2024

FMCAD, October 14-18, 2024, Prague

HWMCC Editions



Goals of HWMCC

- Collect large set of publicly available bit-level and word-level benchmarks
- Encourage researchers to work on novel model checking engines
- Provide a platform for comparison

Previous Years

- **Bit-level** tracks
  - AIGER format (`https://github.com/arminbiere/aiger`)
  - **SINGLE** safety (bad state) property track
  - How **DEEP** model checkers go on unsolved SINGLE instances
  - **LIVENESS** track (single "justice" property)

- **Word-level** tracks
  - BTOR2 format (`https://github.com/boolector/btor2tools`)
  - Introduced in **HWMCC'19** for the first time
  - **SINGLE** safety property tracks
    - **Bit-vectors**
    - **Bit-vectors+arrays**

This Year

Word-level Tracks

- BTOR2 format
- **SINGLE** safety property tracks
  - Bit-vectors
  - Bit-vectors+arrays

Bit-level Track

- AIGER 1.9 format
- Benchmarks translated from word-level BV track
- **Certificates mandatory** for sat/unsat
  - **Sat:** AIGER witness
  - **Unsat:** Certifaiger certificates
  - Answer only counted if certificate is valid

Competition Setup

- 3600s wall-clock limit
- 120GB memory limit
- 16 core/32 threads
- One machine per job
- Stanford CENTAUR cluster

Certificate Validation

- 36000s wall-clock limit
- 16GB memory limit
- 2 core/4 threads

Benchmarks

2024 Submissions

- 3 bit-vector benchmarks
  - **877** safety properties
  - submitted by Jannis Harder (YosysHQ)
- 384 bit-vector, 310 array benchmarks
  - **972** safety properties
  - submitted by Zhiyuan Yan, Guangyu Hu, Ziyue Zheng, You Li, Guannan Zhao, Yangdi Lyu, Hongce Zhang, Xiaofeng Zhou (Hong Kong University, Northwestern University)
- 3376 bit-vector, 753 array benchmarks
  - **4129** safety properties
  - submitted by Po-Chun Chien, Nian-Ze Lee (LMU Munich)

From Previous Years

- 2020: **35** total: 30 bit-vector, 5 array benchmarks
- 2019: **4802** total: 2289 bit-vector, 2513 array benchmarks

Benchmark Selection

- **10815** BTOR2 benchmarks in total
  - 7026 bit-vector benchmarks, 3581 bit-vector+array benchmarks
- **Removed** "easy" benchmarks
  - Solved within 10s wall-clock by all 2024 submissions
  - 1008 bit-vector, 1576 array benchmarks
- **Removed** ineligible benchmarks
  - 12 benchmarks without properties
  - 31 benchmarks with reset functions
- Divided all benchmarks into **90 families**
  - Pick at least **one from each family**
- **Random selection** of ∼300 benchmarks from remaining benchmarks
  - **25% old** benchmarks (2019, 2020)
  - **75% new** benchmarks (2024)
- **Selected**: **319 bit-vector** and **321 bit-vector+array** benchmarks
  - 319 bit-vector benchmarks translated to AIGER with btor2aiger

Model Checker Submissions

Submissions from 13 Teams (+7 from 2020)

- **8 bit-level** competitive (+5 from 2020), 1 non-competitive
- **7 bit-vector** competitive (+5 from 2020), 1 non-competitive
- **4 bit-vector+array** competitive (+1 from 2020), 1 non-competitive

Non-Competitive Model Checkers (submitted by organizers)

- **voiraig**: Reference model checker for AIGER + certificates
  - Nils Froleyks (JKU)
- **BtorMC**: Reference model checker for BTOR2
  - Aina Niemetz, Mathias Preiner, Armin Biere (Stanford, Freiburg)

Teams

- **avr**: Aman Goel (AWS), Karem Sakallah (University of Michigan)
- **avr_dp**: Hongyu Fan, Baiting Jiang, Fei He (Tsinghua University)
- **btor2-cert**: Po-Chun Chien, Nian-Ze Lee, Salih Ates, Dirk Beyer (LMU Munich), Zsófia Ádám (BME Budapest)
- **btor2-selectmc**: John-Lu (UWaterloo), Po-Chun Chien, Nian-Ze Lee (LMU Munich), Vijay Ganesh (Georgia Tech)
- **fric3**: Tobias Seufert (University of Freiburg)
- **mc-zhulf**: Lingfeng Zhu (Chinese Academy of Sciences)
- **ncip**: Tobias Faller (University of Freiburg)
- **nuxmv**: Alberto Griggio (FBK), Martin Jonáš (Masaryk University)
- **pavy**: Yakir Vizel, Basel Khouri, Andrew Luka (Technion), Arie Gurfinkel (UWaterloo)
- **pono**: Áron Ricardo Perez-Lopez, Makai Mann, Ahmed Irfan, Florian Lonsing, Yahan Yang, Samantha Archer, Clark Barrett (Stanford University)
- **ric3**: Yuheng Su, Qiusong Yang, Yiwei Ci (Chinese Academy of Sciences)
- **satvik**: Arun Chandrasekharan (Hobbyist)
- **supercar**: Yibo Dong, Yechuan Xia, Hongtai Zhu, Jianwen Li, Geguang Pu (East China Normal University)

AVR: Astractly Verifying Reachability

https://github.com/aman-goel/avr

Aman Goel (AWS), Karem Sakallah (University of Michigan)

- AVR proof race: 16 parallel configurations racing proof or counterexample
  - 10 variants of IC3+EA
    word-level IC3 with syntax-guided abstraction, plus add-ons:
    - data abstraction
    - incremental refinement
    - hybrid abstractions
    - property-directed word splitting
    - extract/concat handler
    - datapath abstractions
  - 3 variants of BMC, plus data abstraction
  - 3 variants of K-induction
- Abstract only heavy operators (new)
- Limited datapath propagation (new)
- Several bug fixes

- Dissertation Dive deep into AVR (Chapters 3 & 4)
- TACAS'20 AVR at a high level
- NFM'19 Syntax-guided equality abstraction technique of AVR
- SOSP'19 Using AVR for verifying distributed protocols

AVR_dp

Hongyu Fan, Baiting Jiang, Fei He (Tsinghua Univ.)

- Built upon IC3 + Data-path abstraction (implemented in AVR)
- Introduces a new technique: Data-path propagation (DPP)
- Moderately repay the semantics of data-path operations
- Executes DPP prior to each invocation of the SMT solver
- Lemmas generated during DPP are reused to optimize SMT solving
- Support Verilog, Btro2, VMT format (enabled by AVR)
- Paper published in TCAD 2023

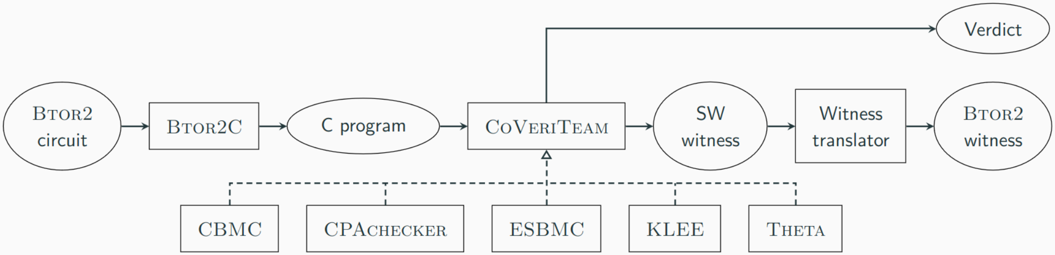
AVY

Basel Khouri, Andrew Luka, Yakir Vizel (Technion), Arie Gurfinkel (UWaterloo)

- Bit-level model checker
  - Combines sequence interpolants with Property Directed Reachability
  - CAV 2014, 2015, 2019, FMCAD 2014
- Interpolants are computed using DRUP proofs
  - Configurations include the SAT solvers: Glucose and CaDiCaL 2.0
  - A new implementation of DRUP-based interpolants in CaDiCaL 2.0 (paper under submission)
- Other configurations include BMC and PDR
- Two configurations use a novel PDR engine implemented in **RUST**
  - One of the configurations include a novel technique as part of PDR

Btor2-Cert: Certifying Btor2 Verification Using Software Analyzers

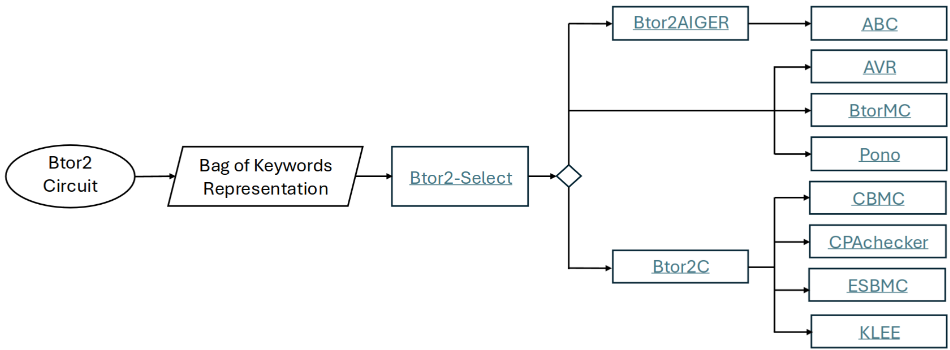
Zsófia Ádám (BME Budapest), Salih Ates, Dirk Beyer, Po-Chun Chien, Nian-Ze Lee (LMU Munich)



- Translate model-checking tasks using Btor2C
- Backend: Parallel portfolio of software analyzers (CoVeriTeam)
- Verification techniques: BMC, k-induction, interpolation, symbolic execution
- Translate witnesses back to Btor2 domain

Btor2-SelectMC: Algorithm Selection for Btor2 Model Checking

John Lu (UWaterloo), Po-Chun Chien, Nian-Ze Lee (LMU Munich), Vijay Ganesh (Georgia Tech)



- Btor2-Select: decision-tree-based algorithm selector
- Backend tools: state-of-the-art hardware model checkers and software analyzers
- Algorithms used by backend tools: BMC, k-induction, interpolation, IC3/PDR, predicate abstraction, symbolic execution

FrIC3

Tobias Seufert, University of Freiburg

FrIC3 is a portfolio combining different certifying model checkers (mostly IC3-derived).

- FrIC3 does *not preprocess* the AIGER spec (only the transition relation CNF with frozen inputs / states).
- In total, it runs 9 tools portfolio-style (no communication).
  - 1: fbPDR: alternating execution of standard and reverse PDR with additional collaboration between both variants.
  - 2: Reverse PDR
  - 3-4: Basic IC3 with MiniSAT and another version with CaDiCaL
  - 5-7: PROGRESS-PDR (with input and statebit restrictions) in three variants
  - 8: k-induction without unrolling; similar to kInd from (Gurfinkel and Ivrii, 2017)
  - 9: one basic BMC engine with MiniSAT

NCIP: Next Craig Interpolant Prover

Tobias Faller, Florian Pollitt, Bernd Becker (University of Freiburg)

- Easy-to-use (Un-)Bounded Model Checker with Craig interpolation
- Inspired by CIP (Craig Interpolant Prover) Stefan Kupferschmid[1]
- Modern design based on C++17 and std containers
  - CaDiCraig (via CaDiCaL tracer, upstream, special thanks Mathias Fleury)
  - KittenCraig (via tracer, fast for small problems)
  - MiniCraig v2 (based on MiniCraig)
  - Symmetric, Asymmetric, Dual Asymmetric Interpolants (Union, Intersection, Smallest, Largest)
- Easy-to-use AIG API (with Constraints)
  - AIG certificates (Certifaiger, special thanks to Nils Froleyks)
- Easy-to-use CNF API (Init, Trans, Target)
  - Supports non-total transition relations
  - CNF certificates are future work
- Intuitive CLI supporting AIG (Aiger 1.9) and CIP formats
- Includes thread-parallel portfolio version
- Fuzzing of Craig interpolants, and AIG / CNF formats

[1] Stefan Kupferschmid, "Über Craigsche Interpolation und deren Anwendung in der formalen Modellprüfung". University of Freiburg, Der Andere Verlag 2013, ISBN 978-3-86247-411-0, pp. 1-247

nuXmv

Alberto Griggio (FBK), Martin Jonáš (Masaryk University)

Portfolio approach: 7 engines in parallel, no communication

SAT-based engines

(all employing counterexample-guided array abstraction using prophecy variables)

- BMC
- k-induction
- IC3
- IC3 with lazy abstraction

SMT-based engines

- BMC
- k-induction
- IC3 with implicit predicate abstraction

Pono

Áron Ricardo Perez-Lopez, Makai Mann, Ahmed Irfan, Florian Lonsing, Yahan Yang, Samantha Archer, Clark Barrett (Stanford University)

- Lightweight, adaptable SMT-based model checker
  - Built on solver-agnostic SMT API, smt-switch
- Competition portfolio configuration
  - BMC
  - K-Induction
  - Interpolation-based
  - IC3SA: IC3 with Syntax-guided Abstraction
  - IC3IA: IC3 with Implicit predicate Abstraction
  - Model-based IC3 (BV only)
  - Bit-level IC3 (BV only)
  - IC3 with SyGuS-based lemma generation (BV only)
- SMT Solvers
  - Bitwuzla (most BV and array solving), MathSAT5 (interpolation, IC3IA)
  - Many thanks to the SMT solver developers!

rIC3

Yuheng Su, Qiusong Yang, Yiwei Ci (Institute of Software, Chinese Academy of Sciences)

- Bit level model checker
- Competition portfolio configuration
  - IC3 with CTG, Internal Signals, Local abstraction
  - BMC
  - K-induction
- The SAT solver in IC3 is deeply optimized
- All algorithms are implemented in Rust
- Using CaDiCaL and Kissat for BMC and K-induction

Satvik

Arun Chandrasekharan (Hobbyist)

- BTOR2 frontend
- Multi-process, multi-threaded framework
- Main engines (bit-level):
  - PDR
  - BMC
- SAT solvers:
  - rewritten-from-scratch MiniSat
  - Kissat
  - Gimsatul

SuperCAR: Complementary Approximate Reachability

Yibo Dong, Yechuan Xia, Hongtai Zhu, Jianwen Li, Geguang Pu (East China Normal University)

- 15 Parallel Running Variants
  - 9 Backward-CAR
    - 5 with different i-good lemma
    - 2 with different Localization
    - 1 with mUC
    - 1 basic
  - 5 Forward-CAR
  - 5 with different i-good lemma
  - 1 simple BMC
- Related Publications
  - Safety model checking with complementary approximations. Li et al. ICCAD'17
  - SimpleCAR: An Efficient Bug-Finding Tool Based on Approximate Reachability. Li et al. CAV'18
  - Searching for i-Good Lemmas to Accelerate Safety Model Checking. Xia et al. CAV'23

HWMCC'24 Ranking

- **3 Tracks**
  - Bit-level
  - Word-level: bit-vectors
  - Word-level: bit-vectors+arrays
- **Ranked** by number of solved benchmarks (sat+unsat)
  - Gold: 1st place
  - Silver: 2nd place
  - Bronze: 3rd place
- 3 medals for each track: **9 medals** in total

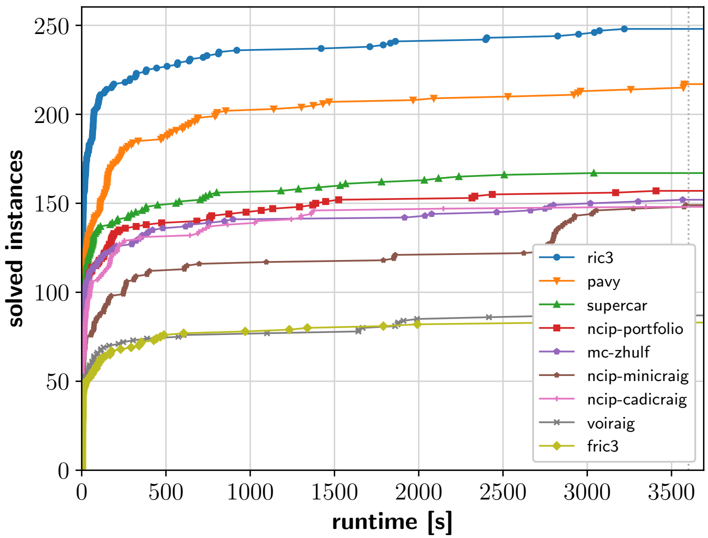
Results

Bit-Level Track: Solved

	solved	sat	unsat	real [s]	cpu [s]	mem [mb]	best	uniq
ric3	248	72	176	291344	4153052	4493952	104	23
pavy	217	51	166	390161	3786142	7301625	24	3
supercar	167	46	121	438161	5242815	4928159	8	4
ncip-portfolio	157	47	110	565994	1695151	14808792	1	0
mc-zhulf	152	51	101	450169	2190356	5056390	23	1
ncip-minicraig	149	43	106	702887	702828	3201387	42	1
ncip-cadicraig	148	47	101	626479	626813	12445563	26	0
voiraig	87	19	68	856670	856462	498891	16	0
fric3	83	14	69	300617	349652	388817	17	0

319 benchmarks, 1h wall-clock time limit, 120GB memory limit

Bit-Level Track: Solved



Bit-Level Track: Certified

		certified	sat		unsat		uniq
1	ric3	248	72		176		25
2	pavy	217	51		166		3
3	supercar	162	41	(-5)	121		4
	mc-zhulf	152	51		101		1
	ncip-portfolio	149	47		102	(-8)	0
	ncip-cadicraig	148	47		101		0
	ncip-minicraig	140	43		97	(-9)	0
	voiraig	87	19		68		0
	fric3	83	14		69		0

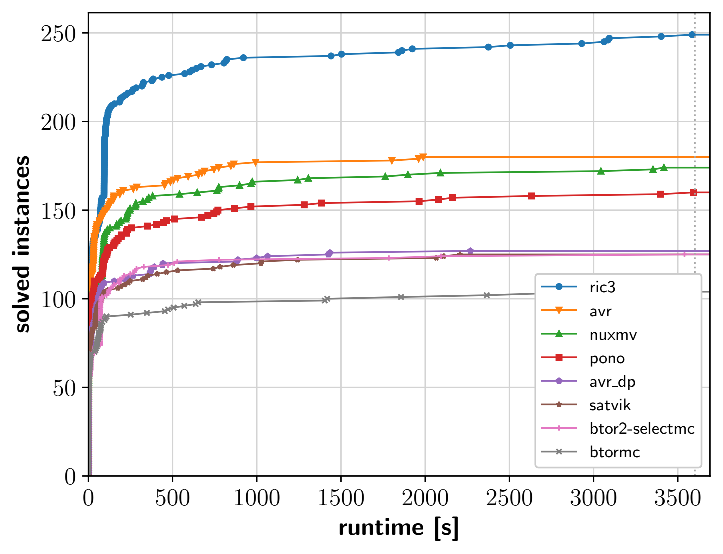
Negative numbers in orange are invalid certificates.

Word-Level Track: Bit-Vectors

		solved	sat	unsat	real [s]	cpu [s]	mem [mb]	best	uniq
1	ric3	249	72	177	297749	4198237	12412529	88	34
2	avr	180	53	127	485581	4983847	16061230	4	1
3	nuxmv	174	44	130	277837	1654862	5229664	17	2
	pono	160	52	108	389448	4920452	16119475	4	2
	avr_dp	127	26	101	527823	2235222	2993829	17	0
	btor2-selectmc	125	21	104	462172	470502	804063	33	0
	satvik	125	49	76	503167	9657805	11436575	22	0
	btormc	104	48	56	787777	787654	3694299	74	0

319 benchmarks, 1h wall-clock time limit, 120GB memory limit

Word-Level Track: Bit-Vectors

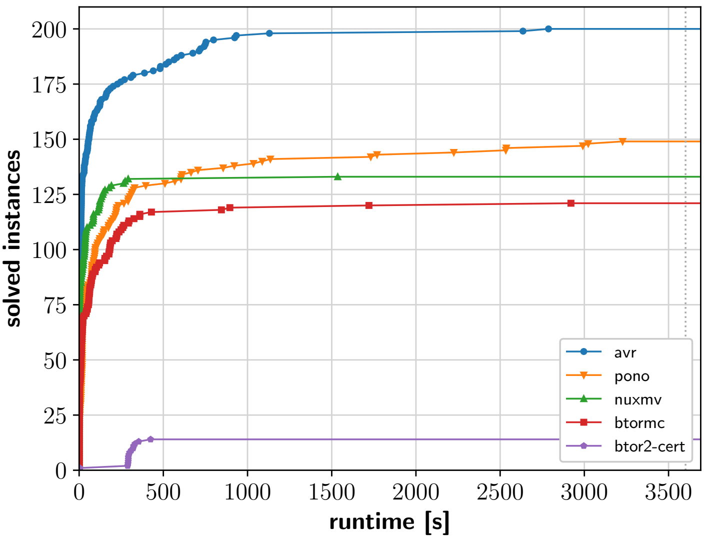


Word-Level Track: Bit-Vectors+Arrays

		solved	sat	unsat	real [s]	cpu [s]	mem [mb]	best	uniq
1	avr	200	102	98	435088	4767808	9604215	23	9
2	pono	149	87	62	609128	5303197	9843296	5	1
3	nuxmv	133	45	88	244252	1401939	1828014	83	3
	btormc	121	96	25	717497	717164	1007327	93	0
	btor2-cert	14	14	0	249048	442224	11768536	0	0

321 benchmarks, 1h wall-clock time limit, 120GB memory limit

Word-Level Track: Bit-Vectors+Arrays



Results Summary

	gold	silver	bronze
ric3	2		
avr	1	1	
pavy		1	
pono		1	
nuxmv			2
supercar			1

Congratulations to the winners!

Bit-Level Solving and Certification Statistics

Solving Statistics

- 1284 hours wall-clock
- 5445 hours CPU time
- 50TB memory usage
- 60 out of 319 unsolved

Certification Statistics

- 21 hours wall-clock
- 380GB memory usage
- no time or memory outs
  - max: 12000s/9.1G memory
  - avg: 26s/135MB
  - med: 1.5s/5.5MB
- 390 sat certificates (5 incorrect), checked with aigsim
- 1018 unsat certificates (17 incorrect), checked with certifaiger

Summary

HWMCC'24

- **13 teams**
- **19 competitive** entries in 3 tracks
- **5978** new single safety benchmarks

HWMCC' ??

- Word-level tracks: mandatory sat witnesses
- Bit-level: Revive **LIVENESS** track

Thank you to all teams and benchmark submitters!