

# Stratified Certification for $k$ -Induction (Extended Abstract)

originally presented at FMCAD'22

Emily Yu<sup>1</sup>, Nils Froyleys<sup>1</sup>, Armin Biere<sup>2</sup>, Keijo Heljanko<sup>3</sup>

<sup>1</sup>Johannes Kepler University Linz, Linz, Austria

<sup>2</sup>University of Freiburg, Freiburg, Germany

<sup>3</sup>University of Helsinki, Helsinki, Finland

MBMV'23

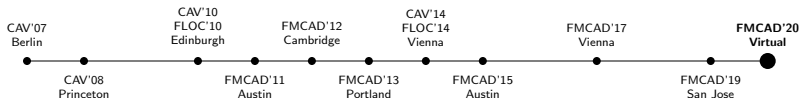
March 23 - 24, 2023

Freiburg, Germany





# Hardware Model Checking Competition 2020



- AIGER format <http://fmv.jku.at/aiger>
- Tracks
  - **SINGLE** safety (bad state) property track
  - how **DEEP** model checkers go on unsolved SINGLE instances  
Oski Technology award \$500
  - **LIVENESS** track (single “justice” property)
- BTOR2 format <https://github.com/boolector/btor2tools>
  - **HWMCC'19** first year with **word-level** tracks



### Certification of model checking

- Generic machine checkable certification is still in its infancy
  - proofs are mandatory in SAT competition
  - currently not possible in hardware model checking competitions

### Temporal induction/ $k$ -induction [SheeranSS00] as a powerful technique

- Reduces model checking to a series of SAT problems
  - highly integratable with modern SAT/SMT solvers
  - also used in other contexts: infinite-state systems, SEC etc.
- State-of-the-art: certification of  $k$ -induction
  - only  $k$ -induction specific certificates (no inductive invariant) [Gurfinkell17]
  - exponential certificates [BjørnerGMR15]
  - one-alternation QBF [YuBH20]



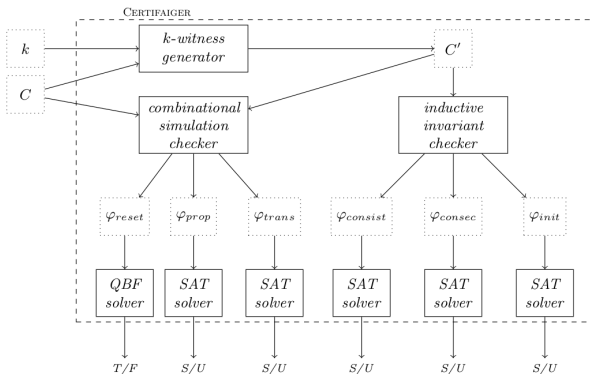
## Inductive Invariant $\phi$

Condition	Formula	The inductive invariant ...
“ <i>initiation</i> ”	$R(L) \Rightarrow \phi(I, L)$	... must hold at all initial states.
“ <i>consistency</i> ”	$\phi(I, L) \Rightarrow P(I, L)$	... must hold at all good states.
“ <i>consecution</i> ”	$U_1 \wedge \phi(I_0, L_0) \Rightarrow \phi(I_1, L_1)$	... is preserved during the transition.



# Old Tool Flow : CERTIFAIGER

## CAV'21



**Definition 7 (Combinational simulation).** Given circuits  $C = (I, L, R, F, P)$  and  $C' = (I', L', R', F', P')$  where  $C'$  combinational extends  $C$ , we say that  $C'$  combinational simulates  $C$ , if the following holds:

1.  $f_l(I, L) \equiv f'_l(I', L')$  for  $l \in L$ ,
2.  $P'(I, L') \Rightarrow P(I, L)$ , and
3.  $R(L) \Rightarrow \exists(L' \setminus L)R'(L')$ .

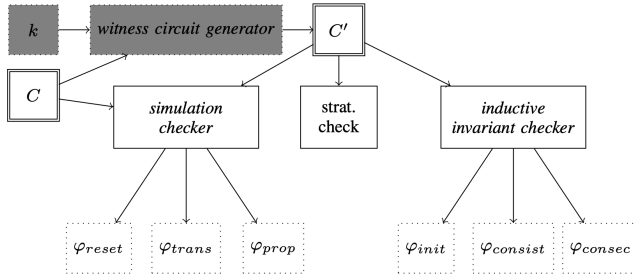
“transition”

“property”

“reset”



# New Tool Flow : CERTIFAIGER++ FMCAD'22



- Each component is **independent** of others
- **Certification success**: all seven checks pass
- Faulty witness generation?
- Certification on failure? (incorrect value of  $k$ )



## Our contribution

- **Generic** certification framework for  $k$ -induction-based model checking
  - generates seven SAT problems
  - a simple **inductive invariant** as proof certificate
  - **linear** to the circuit size and  $k$
- A tool-suite CERTIFAIGER++
  - extension of CERTIFAIGER
  - independent of any (hardware) model checker
  - compatible with modern  $k$ -induction-based model checkers
  - evaluated empirically
- A word-level certifier CERTIFAIGER-WL
  - word-level certification





**Definition** A circuit is a tuple  $C = (I, L, R, F, P)$  such that:

- $I$  is a finite set of **input** variables.
- $L$  is a finite set of **latch** variables.
- $R = \{r_l(L) \mid l \in L\}$  is a set of **reset functions**.
- $F = \{f_l(I, L) \mid l \in L\}$  is a set of **transition functions**.
- $P(I, L)$  is a function that evaluates to a Boolean output, encoding the **(good states) property**.

**Definition** reset function  $r_l$  for latch  $l$  **depends** on  $l'$  if  $l'$  occurs in  $r_l$

**Definition** set of reset functions  $R = \{r_l(L) \mid l \in L\}$  is **stratified** iff dependencies graph acyclic

**Definition**

A circuit  $C = (I, L, R, F, P)$  is said to be **stratified** iff  $R$  is stratified.

- symbolic representation to reduce syntactic clutters
- compatible with AIGER format



**Definition** Given  $C = (I, L, R, F, P)$  and  $C' = (I', L', R', F', P')$ ,  $C'$  **combinationally extends**  $C$  if  $I = I'$  and  $L \subseteq L'$ .

- Allows us to interpret the inputs and latches of a circuit as being part of another circuit.

**Definition** Given two stratified circuits  $C$  and  $C'$ , where  $C'$  combinational extends  $C$ . There is a **stratified simulation** between  $C'$  and  $C$  iff,

- ①  $r_l(L) \equiv r'_l(L')$  for  $l \in L$ , "reset"
- ②  $f_l(I, L) \equiv f'_l(I, L')$  for  $l \in L$ , and "transition"
- ③  $P'(I, L') \Rightarrow P(I, L)$ . "property"

- Can be verified by SAT checks.

**Theorem** Given two circuits  $C$  and  $C'$ , where  $C'$  **simulates**  $C$ . If  $C'$  is **safe**, then  $C$  is also **safe**.



## Certification: witness circuit

**Definition** Given a circuit  $C = (I, L, R, F, P)$ , and  $k \in \mathbb{N}^+$ , the **witness circuit**  $C' = (I', L', R', F', P')$  of  $C$  is defined as follows:

- ①  $I' = I$ . For simplicity we also refer to  $I'$  as  $X^{k-1}$ .
- ②  $L' = X^0 \cup \dots \cup X^{k-2} \cup L^0 \cup \dots \cup L^{k-1} \cup B$ , such that:
  - ①  $X^i$  is a copy of the original inputs, for all  $i \in [0, k-2]$ .
  - ②  $L^i$  is a copy of the original latches, for all  $i \in [0, k-1]$ .
  - ③  $B = \{b^0, \dots, b^{k-1}\}$  is the set of initialisation bits.
- ③  $R'$ :
  - for  $l \in L^{k-1}$ ,  $r'_l = r_l(L^{k-1})$ .
  - for  $l \in L^0 \cup \dots \cup L^{k-2} \cup X^0 \cup \dots \cup X^{k-2}$ ,  $r'_l = l$ .
  - $r'_{b^{k-1}} = \top$ .
  - for  $i \in [0, k-1]$ ,  $r'_{b^i} = \perp$ .
- ④  $F' = \{f'_i(I', L') \mid l \in L'\}$  is defined as follows:
  - ① For  $i \in [0, k-1]$ ,  $f'_{x^i}(I', L') = x^{i+1}$ .
  - ② For  $l \in L^{k-1}$ ,  $f'_l(I', L') = f_l(X^{k-1}, L^{k-1})$ .
  - ③ For  $i \in [0, k-1]$ ,  $f'_i(I', L') = i^{i+1}$ .
  - ④ For  $i \in [0, k-1]$ ,  $f'_{b^i}(I', L') = b^{i+1}$ , and  $f'_{b^{k-1}}(I', L') = b^{k-1}$ .
- ⑤ The property  $P'$  is defined as  $P'(I', L') = \bigwedge_{i \in [0,4]} p_i(I', L')$  such that:
  - ① For  $i \in [0, k-1]$ ,  $h^i = (L^{i+1} \simeq F(X^i, L^i))$ .
  - ②  $p_0(I', L') = \bigwedge_{i \in [0, k-1]} (b^i \rightarrow b^{i+1})$ .
  - ③  $p_1(I', L') = \bigwedge_{i \in [0, k-1]} (b^i \rightarrow h^i)$ .
  - ④  $p_2(I', L') = \bigwedge_{i \in [0, k]} (b^i \rightarrow P(X^i, L^i))$ .
  - ⑤  $p_3(I', L') = \bigwedge_{i \in [1, k]} ((\neg b^{i-1} \wedge b^i) \rightarrow R(L^i))$ .
  - ⑥  $p_4(I', L') = b^{k-1}$ .

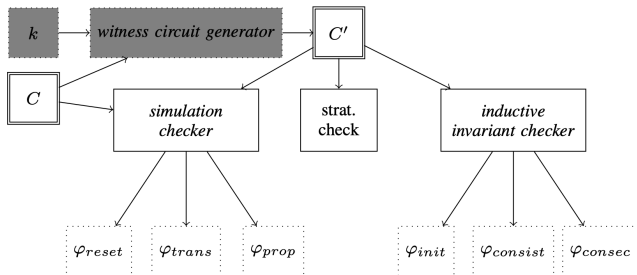
- Saves the previous  $k - 1$  observations; with  $k$  initialisation bits
- The oldest state is discarded after each transition; the youngest state unrolls in the same way as the original one
- The property is composed of five subproperties

**Theorem** The witness circuit  $C'$  simulates the original circuit  $C$ .

**Theorem** Given a circuit  $C$ , a fixed  $k \in \mathbb{N}^+$ , and its witness circuit  $C'$ ,  $P$  is  $k$ -inductive in  $C$  if and only if  $P'$  is 1-inductive in  $C'$ .



## Implementation: CERTIFAIGER++



- Each component is **independent** of others
- **Certification success**: all seven checks pass
- Faulty witness generation?
- Certification on failure? (incorrect value of  $k$ )





## Experimental Results: bit level

TABLE I: Summary of certification results for the bit-level TIP suite.

Benchmarks	$\varphi_{init}$		$\varphi_{consist}$		$\varphi_{consec}$		$\varphi_{trans}$		$\varphi_{prop}$		$\varphi_{preset}$	
	$t_1$	$t_2$	$t_1$	$t_2$	$t_1$	$t_2$	$t_1$	$t_2$	$t_1$	$t_2$	$t_1$	$t_2$
c.periodic	7.78	0.06	0.06	0.06	56.82	56.29	0.15	0.14	0.05	0.05	84.04	0.00
n.guidance <sub>1</sub>	0.19	0.01	0.01	0.01	3.73	3.79	0.12	0.12	0.01	0.01	1.21	0.00
n.guidance <sub>7</sub>	4.09	0.02	0.02	0.02	18.40	18.17	0.12	0.12	0.02	0.02	25.22	0.00
n.tcasp <sub>2</sub>	0.17	0.01	0.01	0.01	2.64	2.68	0.23	0.23	0.01	0.02	1.79	0.00
n.tcasp <sub>3</sub>	0.11	0.01	0.01	0.01	1.82	1.70	0.23	0.26	0.02	0.02	1.01	0.00
v.prodcell <sub>12</sub>	2.35	0.03	0.03	0.03	59.05	59.22	0.12	0.12	0.03	0.03	8.48	0.00
v.prodcell <sub>13</sub>	0.22	0.01	0.01	0.01	2.99	2.99	0.12	0.12	0.01	0.01	0.20	0.00
v.prodcell <sub>14</sub>	0.64	0.02	0.02	0.02	13.69	13.69	0.12	0.12	0.02	0.02	1.45	0.00
v.prodcell <sub>15</sub>	2.22	0.02	0.03	0.03	32.66	32.28	0.12	0.12	0.02	0.02	2.26	0.00
v.prodcell <sub>16</sub>	0.01	0.01	0.01	0.01	1.19	1.20	0.12	0.12	0.01	0.01	0.06	0.00
v.prodcell <sub>17</sub>	2.34	0.03	0.03	0.03	48.51	48.17	0.12	0.12	0.03	0.03	6.86	0.00
v.prodcell <sub>18</sub>	0.67	0.01	0.01	0.01	8.67	8.78	0.12	0.12	0.02	0.02	0.79	0.00
v.prodcell <sub>19</sub>	1.66	0.02	0.02	0.03	31.98	31.78	0.12	0.12	0.03	0.03	3.73	0.00
v.prodcell <sub>24</sub>	3.32	0.04	0.04	0.04	112.12	115.18	0.12	0.12	0.04	0.04	17.64	0.00



## Experimental Results: word level

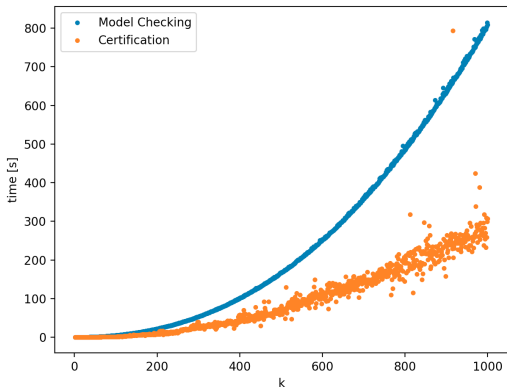
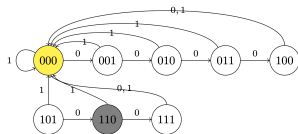
TABLE II: Summary of certification results word-level benchmarks from the HWMCC20

Benchmarks	k	#model	#witness	ModelCh.	Certifi.	Consec.	Ratio
paper_v3	256	35	12801	10.25	1.14	0.90	0.11
VexRiscv-regch0-15-p0	17	2149	43077	10.31	4.04	3.29	0.39
zipcpu-pfcache-p02	37	1818	105874	13.95	4.40	2.73	0.32
zipcpu-pfcache-p24	37	1818	105874	14.35	4.49	2.83	0.31
zipcpu-busdelay-p43	101	950	145466	15.29	6.14	3.86	0.40
dspfilters_fastfir_second-p42	15	6732	115388	16.11	14.80	12.96	0.92
zipcpu-pfcache-p01	41	1818	117434	18.33	6.34	4.47	0.35
dspfilters_fastfir_second-p10	11	6732	84348	24.56	9.76	8.44	0.40
zipcpu-busdelay-p15	101	950	145466	58.17	8.18	5.89	0.14
qspiflash_dualflexpress_divfive-p120	97	3100	394412	63.58	22.07	14.58	0.35
zipcpu-pfcache-p22	93	1818	267714	166.07	23.66	19.06	0.14
VexRiscv-regch0-20-p0	22	2149	55862	240.50	16.76	15.76	0.07
dspfilters_fastfir_second-p14	15	6732	115388	354.01	21.27	19.44	0.06
dspfilters_fastfir_second-p11	21	6732	161948	627.69	46.88	44.30	0.07
dspfilters_fastfir_second-p45	17	6732	130908	1094.11	30.14	28.06	0.03
VexRiscv-regch0-30-p1	32	2150	81464	1444.47	83.38	81.95	0.06
dspfilters_fastfir_second-p43	19	6732	146428	2813.61	58.02	55.69	0.02



## Experimental Results: word level

- Number of bits = 500,  
module bound = 32
- Value of  $k$  ( $= b - m + 1$ ) scaled up to 1000
- model checking vs. certification time with increasing values of  $k$










## Ongoing Work

- optimizing the witness circuits
- formally verified checkers (Isabelle)
- combine with SAT certificates
- witnesses for preprocessing:
  - temporal decomposition
  - two-phase abstraction
  - retiming



## References

-  Sheeran, M., Singh, S., Stalmarck, G.: Checking safety properties using induction and a SAT-solver. In: FMCAD. Lecture Notes in Computer Science, vol. 1954, pp. 108–125. Springer (2000)
-  Gurfinkel, A., Ivrii, A.: K-induction without unrolling. In: FMCAD. pp. 148–155. IEEE (2017)
-  N. Bjørner, A. Gurfinkel, K. L. McMillan, and A. Rybalchenko, “Horn Clause Solvers for Program Verification,” in Fields of Logic and Computation II - Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday, 2015, pp. 24–51. [Online]. Available: [http://dx.doi.org/10.1007/978-3-319-23534-9\\_2](http://dx.doi.org/10.1007/978-3-319-23534-9_2)
-  Biere, A., Brummayer, R.: Consistency checking of all different constraints over bit-vectors within a SAT solver. In: FMCAD. pp. 1–4. IEEE (2008)
-  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.

