Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Trust Layer for AI-Native EDA: Certificate-Bearing Verification of Generated Designs from Logic to Yield

Christophe Bianchi

christophe.bianchi@octinnov.com

École Centrale de Lyon https://orcid.org/0000-0003-2714-4031

Research Article

Keywords: AI-native EDA, design verification, equivalence checking, GF(2), uncertainty quantification, design certificates, functional safety

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11807582/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

A Trust Layer for AI-Native EDA: Certificate-Bearing Verification of Generated Designs from Logic to Yield

Christophe Bianchi, PhD
TensorEDA, Nice, France | christophe.bianchi@tensoreda.com

***Abstract*—AI-native EDA tools can generate RTL, floorplans and routes in days rather than months, yet the evidence that a generated design is correct, robust and manufacturable still comes largely from spot checks: random simulation, nominal timing and sampled design-rule checks. We present a trust layer, an independent verification stage that checks a design package against five claims (function, timing and power yield, physical rules, routability and assembly thermal validity) and returns, for each, a verdict with a typed evidence class. Functional claims are decided by a portfolio of bit-parallel simulation, cut-point sweeping with local GF(2) algebraic-normal-form proofs, and a global GF(2) difference whose minimum-cardinality monomial yields a replayed counterexample. Yield claims use sampling with Wilson intervals; routing and thermal claims use exhaustive enumeration of declared finite models; spacing rules use a hierarchical check proved equal to the flat check. Verdicts are recorded in a hash-chained certificate with an independent checker. On 600 mutated 8-bit adders with exhaustive ground truth, the functional gate refuted all 581 non-equivalent mutants with replayed witnesses and proved all 19 equivalent mutants, with no wrong verdicts, whereas 4,096-vector random simulation missed 146 of 300 rare-trigger bugs (analytic expectation: 141). On seven synthetic design packages the layer rejected all six defective ones, five of which a nominal spot-check flow accepted. All fixtures and variation models are synthetic, and we state the limits explicitly.**

***Index Terms*—AI-native EDA, design verification, equivalence checking, GF(2), uncertainty quantification, design certificates, functional safety.**

I. INTRODUCTION

Generative models are entering the chip-design flow. Large language models now produce Verilog that is scored by simulation against testbenches [1]–[4], and AI-native design systems extend the idea to microarchitecture exploration, verification and physical-design decisions. Faster generation moves the bottleneck: the question is no longer how quickly a design can be produced, but how much evidence supports it.

Simulation-based scoring is necessary but not sufficient. A defect that fires on one input pattern in 2^{25} is invisible to any feasible random test, yet it is a defect. The same asymmetry appears beyond logic. A placement can meet timing at nominal conditions while failing in a large fraction of process, voltage and temperature scenarios; a route can be overflow-free at nominal track capacity while one shared corridor fails whenever it loses a track; a layout assembled from repeated tiles can pass a check on sampled tiles while violating spacing at a seam.

Functional-safety practice frames the same issue as tool confidence. ISO 26262-8 asks whether a tool error could introduce or fail to detect a fault, and whether another measure would detect it; comparing input and output netlists, static timing analysis and gate-level simulation are cited as ways to raise confidence in synthesis-class tools [5]. An independent check downstream of a generator therefore has a standing role, provided its verdicts are honest about what they establish.

We contribute: (i) a *claim contract* in which every gate returns a verdict from a four-valued set together with an evidence class that states how the verdict was obtained; (ii) a functional gate that combines simulation, cut-point sweeping with local GF(2) proofs and a global GF(2) difference, and that never converts an exhausted budget into an equivalence verdict; (iii) timing, physical, routing and thermal gates that each return exact or statistically bounded verdicts relative to declared models; (iv) a hash-chained certificate and an independent checker, with tamper tests that show what the checker can and cannot detect; and (v) a mutation campaign with exhaustive ground truth and a package benchmark. The techniques are individually established; the contribution is their composition under a single verdict contract across logic, timing, yield and physics. The results are preliminary and synthetic.

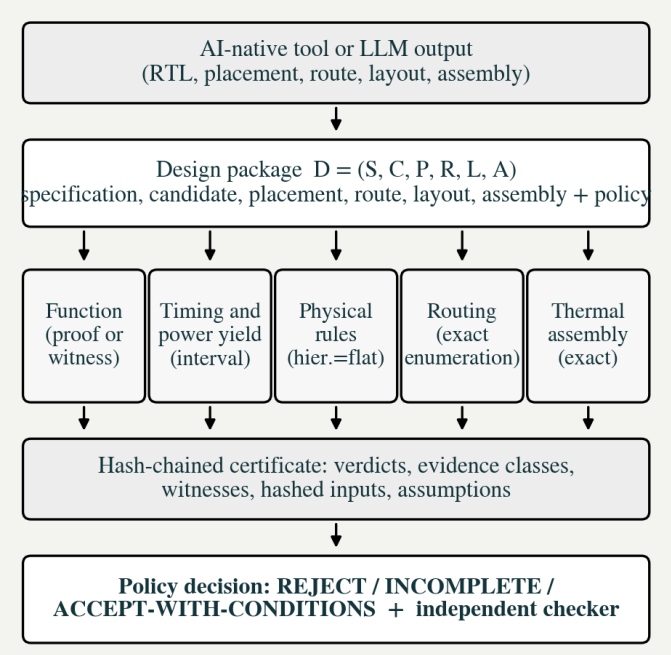

Fig. 1. Trust-layer architecture: one design package, five gates, one hash-chained certificate and a policy decision.

II. RELATED WORK

LLM-generated hardware. VerilogEval [1], RTLLM [2], RTL-Repo [3] and earlier fine-tuning studies [4] evaluate generated RTL by simulation against reference testbenches. Their metrics measure agreement on the stimuli supplied; they do not bound the probability of residual rare-trigger faults.

Equivalence checking. Canonical representations [6], structural cuts [7] and SAT sweeping [8] underpin industrial combinational checkers. Word-level and Gröbner-basis methods address arithmetic datapaths and finite-field multipliers [9], [10]. Our functional gate uses cut-point sweeping in the classical sense, with GF(2) polynomial identity as the local prover, and follows the portfolio principle that an engine may discharge only the obligations its own semantics supports. Tensor-network formulations of Boolean problems [11] and tensor-train decompositions [12] motivate the rank diagnostics of the companion TensorEQC workflow but are not required by the gates reported here.

Variation and tensor methods. Tensor-train and ANOVA decompositions for high-dimensional uncertainty quantification in circuits [13], [14] are the basis for the statistical gates of the wider platform. The present timing gate deliberately uses plain Monte Carlo with an exact-coverage-motivated Wilson interval [15], so that its verdict does not depend on surrogate accuracy.

Certificates. Proof-producing solvers and checkers, such as clausal proofs for SAT [16], separate proof search from proof checking. Hash chaining of records dates to timestamping schemes [17]. Our certificates are tamper-evident but not signed, and we do not yet emit machine-checkable equivalence proofs.

III. TRUST MODEL

Claims and verdicts. A design package is $D = (S, C, P, R, L, A, \Pi)$: a specification netlist, a candidate netlist, a placement, a route, a layout, an assembly, and a policy that fixes thresholds and declared models. The generator implicitly asserts five claims. A gate returns a verdict in {VERIFIED, REFUTED, INCONCLUSIVE, UNRESOLVED}, an evidence class, a witness for refutations, and the assumptions on which the verdict depends.

TABLE I
EVIDENCE CLASSES

Class	Meaning	Gates
PROOF_EXACT	Equality proved under gate semantics	Function
WITNESS_REPLAYED	Counterexample re-executed on both models	Function
EXACT_FINITE_MODEL	Full enumeration of a declared model	Routing, thermal
EXACT_HIERARCHICAL	Hierarchical result proved equal to flat	Physical
STATISTICAL_BOUND	Sampling interval, declared model	Timing, yield

Decision policy. The package is rejected if any claim is REFUTED; it is INCOMPLETE if any claim is UNRESOLVED or INCONCLUSIVE; otherwise it is accepted with conditions, and the conditions list the declared models. The policy is asymmetric by design: a replayed witness refutes conclusively, whereas a verification is only as strong as its evidence class. While models are uncalibrated the layer never issues an unconditional acceptance.

Threat model. The layer addresses unintentional generator error. The certificate protects against accidental or casual modification; an adversary who can recompute hashes is out of scope without signing, and Section VI quantifies the consequence.

IV. GATES

A. Functional gate

For matched outputs F (specification) and G (candidate) define

$$D_i(x) = F_i(x) \oplus G_i(x),\ M(x) = \vee_i D_i(x),\ \text{equivalent} \Leftrightarrow M \equiv 0.$$

Stage 1 evaluates m random patterns plus the all-zero and all-one vectors bit-parallel. A mismatch is a witness, replayed on both netlists. A clean run is never an equivalence proof.

Stage 2 is cut-point sweeping. Every net of both netlists receives a simulation signature. For each specification net x in topological order, a candidate net y with the same signature is tested by computing the GF(2) algebraic normal forms p_x and p_y over the primary inputs and the cut variables introduced so far, where a cut variable replaces any net already proved equal. If $p_x = p_y$, both nets are replaced by one fresh variable. If all outputs end with identical polynomials the verdict is VERIFIED.

Proposition 1 (soundness of sweeping). If polynomial identity holds over inputs and cut variables, the nets are functionally equal over the primary inputs.

Proof. Each cut variable stands for a net already proved equal in both netlists, so its value under any input assignment is that common function value. A polynomial identity holds for all assignments of its variables, in particular for the consistent assignments that arise from inputs. By induction over the cut order, identity at an output implies equality as a function of the inputs. Signature matches are heuristic; only the polynomial test is trusted, so a bad match can cost completeness but not soundness. ∎

Stage 3 computes global ANFs under term and operation budgets. For a nonzero difference polynomial P, let S be the variable set of a minimum-cardinality monomial and set exactly the variables in S to one. Only monomials contained in S can survive, and minimality leaves exactly one; hence P evaluates to one, giving a witness that is replayed on both netlists. If a budget is exhausted the verdict is UNRESOLVED. Summing mismatch indicators over GF(2) is never used as an existence test, because two mismatches cancel.

B. Timing and yield gate

Let ξ follow the declared variation model and let Y be the pass indicator for the policy metrics. From k passes in n samples with $\hat{p} = k/n$, the Wilson interval is

$$[l, u] = \left(\hat{p} + z^2/2n \pm z \sqrt{\left(\hat{p}(1-\hat{p})/n + z^2/4n^2 \right)} \right) / \left(1 + z^2/n \right).$$

with $z = 1.96$ [15]. The verdict is VERIFIED if $l \geq \tau$, REFUTED if $u < \tau$, and INCONCLUSIVE otherwise, for target yield τ. Asymptotically, if the true yield is below τ the probability of a VERIFIED verdict is about 2.5% or less. Variance drivers are reported as total-effect indices estimated by the Jansen pair estimator over the latent variation sources [18].

C. Physical gate

Let a layout consist of instances of tile types, each tile a set of rectangles, and let d be the minimum spacing; two rectangles violate the rule if their Euclidean gap is below d. The hierarchical

check evaluates each unique tile internally once and then evaluates rectangle pairs only between instances whose bounding boxes are closer than d.

Proposition 2 (hierarchical equals flat). The hierarchical check reports exactly the violating pairs of the flat check.

Proof. A violating pair either lies in one instance, and is then a violating pair of its tile type, found by the internal check and replicated by offset; or it spans two instances, whose bounding-box gap is at most the pair gap and hence below d, so the pair is examined at the seam stage. No other pair can violate. ∎

D. Routing and thermal gates

For routing, lane l loses 0, 1 or 2 tracks with declared probabilities $\pi_l(s_l)$. With edge usage u_e and state-dependent capacity $c_e(s)$, the gate computes

$$P_{fail} = \Sigma_s \, \pi(s) \cdot 1[\, \exists e : u_e > c_e(s) \,], \; \pi(s) = \Pi_l \, \pi_l(s_l),$$

by enumeration of all 81 states and returns the most probable failing state as a witness. The thermal gate evaluates a weighted scenario table with integer arithmetic: slot temperatures follow from a coupling matrix and slot powers, and the claim holds if the weights of passing scenarios sum to at least the policy threshold. Both verdicts are exact relative to the declared model and say nothing about the model itself.

E. Certificates and verification

Each gate appends an entry e_j holding the claim, verdict, evidence class, witness and assumptions. Entries are chained by

$$h_j = \text{SHA-256}(\, \text{canon}(e_j) \,\|\, h_{j-1} \,), \; h_0 = 0,$$

and the certificate stores a hash of its canonicalized inputs and the policy decision. An independent checker, which shares no code with the engines, verifies the chain, replays functional, spacing, routing and thermal witnesses with its own evaluator, recomputes the exact routing and thermal gates, and recomputes intervals from counts. For a VERIFIED functional claim the default checker can only confirm the absence of mismatches on random vectors, a necessary condition. An optional re-execution mode reruns the equivalence engine.

V. EXPERIMENTAL SETUP

Packages. The specification is a 12-bit ripple-carry adder with carry-in (25 inputs, 13 outputs). The clean candidate is a NAND-only rewrite. A rare-trigger defect XORs sum bit 11 with an AND of 25 literals (probability 2^{-25}); a frequent defect replaces an XOR by an OR in the sum at bit 5. Timing uses a six-macro placement proxy with two critical paths, correlated process, droop and temperature sources and an activity term; the policy is 800 ps, 10.2 W and a target yield of 85%, estimated from 40,000 samples. The physical fixture has 1,600 tiles of 16 rectangles (25,600 rectangles), minimum spacing 10, six irregular tiles of pitch 25 and a 35-unit shift from tile row 17 onward that creates a seam violation. Routing has seven nets over four lanes of widths 4, 4, 3, 3, lane-loss probability 0.15, two-track probability 0.25 and a 5% failure limit. The thermal fixture has four typed slots, six dies, six weighted scenarios and a 90/100 threshold. Seven packages result: one clean and six with a single injected defect (function twice, timing, physical, routing, thermal).

Baseline. A spot-check flow of our own design stands in for nominal-only practice: 65,536 random vectors, nominal timing and power, ten sampled tiles, nominal track capacity and the typical thermal scenario. It does not represent any vendor or AI-native product.

Mutation campaign. For ripple-carry, NAND and carry-lookahead 8-bit adders (17 inputs) we generated 100 local mutants (gate-type change or input rewiring) and 100 rare-trigger mutants (a random AND of 8 to 17 literals XORed into a random output) each, 600 in total. Ground truth comes from exhaustive simulation of all 2^{17} inputs. In the campaign the first stage uses 4,096 patterns, so simulation alone and the full gate see the same stimulus. Metrics are the detections by 4,096-vector simulation, the verdicts of the functional gate, the number of wrong verdicts and the number of UNRESOLVED outcomes. All code is Python with NumPy on a single machine; times are for unoptimized code.

VI. RESULTS

A. Design packages

TABLE II
PACKAGE VERDICTS

Package	Refuted gate	Layer	Spot-check
P0 clean	none	Accept*	Accepts
P1 rare logic bug	function	Reject	Accepts
P2 frequent logic bug	function	Reject	Rejects
P3 fragile timing	timing/yield	Reject	Accepts
P4 spacing, seam	physical	Reject	Accepts
P5 route bottleneck	routing	Reject	Accepts
P6 thermal assignment	thermal	Reject	Accepts

Table II lists the outcomes; * denotes acceptance with conditions. The layer rejected 6 of 6 defective packages and accepted the clean one; the spot-check flow accepted 5 of the 6 defective packages and caught only the frequent logic bug. The clean package passed every gate: timing yield 90.20% (95% interval 89.91 to 90.49%), routing failure probability 3.75%, thermal weight 94/100.

Function. For the 12-bit NAND rewrite, global ANF exhausted its budget (400,000 terms or 60 million operations) after 10 s, whereas cut-point sweeping proved equality in 0.075 s with 60 proved internal pairs. The rare-trigger candidate survived 131,072 random patterns; the global difference had 4,096 terms and its minimum-cardinality monomial contained 13 variables (all twelve a bits and b_0 set, remaining inputs zero), which the checker replayed independently.

Timing and yield. The fragile layout had a nominal slack of +60.6 ps and passed the nominal check, but its yield was 81.06% (80.67 to 81.44%) against the 85% target. The total-effect indices of its slack were 0.48 (process), 0.35 (voltage droop), 0.17 (temperature) and 0.00 (activity).

Routing and thermal. The shortest route had no nominal overflow but a failure probability of 18.19%; its most probable failing state (probability 6.9%), in which one lane loses a track, overflowed six edges (usage 4, capacity 3). The selected route had 3.75%. The thermal assignment of package P6 scored 74/100, failing the hot and shared-hot scenarios, against 94/100 for the clean assignment.

B. Physical gate

TABLE III
HIERARCHICAL VERSUS FLAT SPACING CHECK

Layout	Violations	Pair tests	Time (ms)
Clean	0 / 0	240 / 357,000	7 / 477
Defective	300 / 300	10,480 / 358,698	9 / 478

Table III gives hierarchical / flat values. The violation sets were identical in both layouts, as Proposition 2 requires. Pair tests fell by a factor of 1,488 on the clean regular layout and 34 on the defective one, where seam checks add work. These ratios depend on the regularity of the synthetic layout and should not be generalized; the ten-tile sampling baseline missed the defects.

C. Mutation campaign

TABLE IV
MUTATION CAMPAIGN, 600 8-BIT ADDER MUTANTS

Measure	Result
Non-equivalent mutants (truth)	581
Refuted with replayed witness	581
Missed by 4,096-vector simulation	146 (all rare-trigger)
Equivalent mutants proved equal	19 of 19
Wrong verdicts / UNRESOLVED	0 / 0

Local mutants were all detected by simulation (281 of 281); rare-trigger mutants separated the methods (Table IV). The simulation miss probability for a trigger of k literals is

$$P_{miss}(k, m) = (1 - 2^{-k})^{m} \approx \exp(-m / 2^{k}).$$

Averaged over the trigger widths used (k uniform on 8 to 17), 300 mutants and $m = 4{,}096$ give an expected 141 misses against 146 observed (Fig. 2); at $m = 65{,}536$ the expectation is 34. The functional gate refuted all 581 non-equivalent mutants and proved all 19 equivalent ones; the whole campaign took 13.6 s.

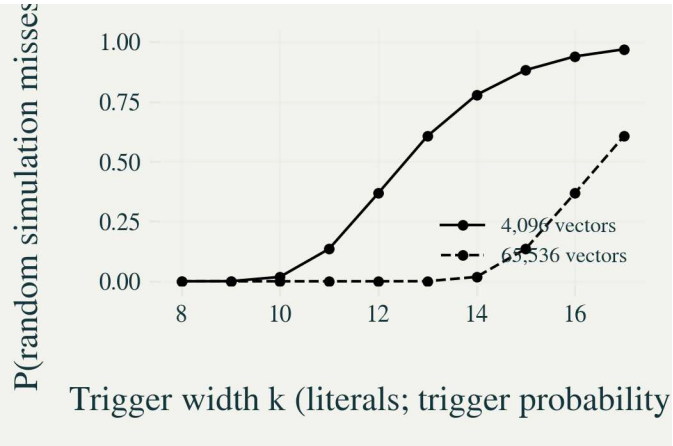

Fig. 2. Probability that random simulation misses a rare-trigger bug as a function of trigger width, for 4,096 and 65,536 vectors.

D. Certificate integrity

Three tamper tests were run on the certificates. Flipping a verdict without recomputing hashes was detected by the chain. Flipping a routing verdict and recomputing the chain was detected by exact recomputation of the routing gate. Flipping the verdict of the rare-trigger functional claim to VERIFIED and recomputing the chain was *not* detected by the default checker, because 2,000 random vectors cannot expose a 2^{-25} defect; it was detected only by the re-execution mode. The hash chain is therefore tamper-evident, and signing plus a proof-checking path would be required against an adversary.

VII. DISCUSSION AND LIMITATIONS

Synthetic evidence. Every fixture, variation model and threshold is synthetic. Yield, routing and thermal verdicts are exact or bounded only relative to declared models that are not calibrated to foundry or silicon data, which is why acceptance is always conditional. The spot-check baseline is illustrative; its weakness is by construction and should not be read as a statement about any deployed flow.

Scope of the functional gate. The gate handles combinational netlists of adders up to 12 bits in the reported tests. Sweeping works because carry chains admit small local cuts; multipliers and designs without matching internal structure will exhaust budgets and return UNRESOLVED, which is the intended failure mode but limits coverage. Sequential equivalence, a general RTL frontend, SAT-based baselines and proof-producing output are not implemented.

Independence. The gates share declared environments, and the checker re-derives exact gates with its own code but trusts the engines for VERIFIED functional and statistical claims unless it re-executes them. A defect in a shared assumption, for example a wrong capacity model, would pass every gate.

Standards. The layer can supply evidence for a tool-error-detection argument [5]. It is not a tool qualification, and decisions about confidence levels remain with the safety process.

Validity of the mutation study. Mutants are generated by simple operators on small adders; the ground truth is exact, but the distribution of real LLM errors is unknown. The agreement between observed and analytic miss counts supports the sampling model, not the realism of the mutants.

VIII. CONCLUSION

We described a trust layer that checks AI-generated design packages against five claims and attaches to each verdict an explicit evidence class, assumptions and, where applicable, a replayable witness. In a synthetic benchmark it refuted every injected defect, including rare-trigger logic bugs that random simulation misses about half the time, and it returned no wrong verdicts in 600 mutants. The next steps are a real RTL frontend with open SAT baselines, calibration of the timing, thermal and routing models on partner data, machine-checkable proofs, and signed certificates. Until then the results show feasibility of the verdict contract rather than coverage of industrial designs.

REFERENCES

[1] M. Liu, N. Pinckney, B. Khailany, and H. Ren, “VerilogEval: Evaluating large language models for Verilog code generation,” in *Proc. ICCAD*, 2023.

[2] Y. Lu, S. Liu, Q. Zhang, and Z. Xie, “RTLLM: An open-source benchmark for design RTL generation with large language model,” in *Proc. ASP-DAC*, 2024.

[3] A. Allam and M. Shalan, “RTL-Repo: A benchmark for evaluating LLMs on large-scale RTL design projects,” in *Proc. IEEE LLM-Aided Design Workshop*, 2024.

[4] S. Thakur *et al.*, “Benchmarking large language models for automated Verilog RTL code generation,” in *Proc. DATE*, 2023.

[5] International Organization for Standardization, *ISO 26262-8:2018, Road vehicles — Functional safety — Part 8: Supporting processes*, clause 11; see also Synopsys, “Understanding how to assess tool confidence levels for ISO 26262,” white paper.

[6] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, 1986.

[7] A. Kuehlmann and F. Krohm, "Equivalence checking using cuts and heaps," in *Proc. DAC*, 1997.

[8] A. Mishchenko, S. Chatterjee, R. Brayton, and N. Een, "Improvements to combinational equivalence checking," in *Proc. ICCAD*, 2006.

[9] N. Shekhar, P. Kalla, and F. Enescu, "Equivalence verification of arithmetic datapaths with multiple word-length operands," in *Proc. DATE*, 2006.

[10] T. Pruss, P. Kalla, and F. Enescu, "Equivalence verification of large Galois field arithmetic circuits using word-level abstraction via Gröbner bases," in *Proc. DAC*, 2014.

[11] J. D. Biamonte, J. Morton, and J. Turner, "Tensor network contractions for #SAT," *J. Stat. Phys.*, vol. 160, pp. 1389–1404, 2015.

[12] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[13] Z. Zhang, X. Yang, I. V. Oseledets, G. E. Karniadakis, and L. Daniel, "Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 1, pp. 63–76, 2015.

[14] Z. Zhang, K. Batselier, H. Liu, L. Daniel, and N. Wong, "Tensor computation: A new framework for high-dimensional problems in EDA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 4, pp. 521–536, 2017.

[15] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *J. Amer. Statist. Assoc.*, vol. 22, no. 158, pp. 209–212, 1927.

[16] N. Wetzler, M. J. H. Heule, and W. A. Hunt Jr., "DRAT-trim: Efficient checking and trimming using expressive clausal proofs," in *Proc. SAT*, 2014.

[17] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *J. Cryptology*, vol. 3, no. 2, pp. 99–111, 1991.

[18] A. Saltelli *et al.*, "Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index," *Comput. Phys. Commun.*, vol. 181, no. 2, pp. 259–270, 2010.