Auditing published circuit tables against exact computation

Luiz Antonio Busnello (Independent Researcher, ORCID 0009-0007-5911-8141)

Version 1.1 · 1 November 2026 · doi:10.5281/zenodo.23861706

Abstract

Logic synthesis and circuit tools ship tables of small circuits that are called optimal: rewriting databases indexed by NPN class, lookup tables for three-input subcircuits, tables of minimum formula sizes, and per-function gate counts for cryptographic S-boxes. A tool that consults such a table inherits the label, and the label is rarely accompanied by anything a third party can check. We describe an audit protocol that tests a published table against ground truth produced by exact synthesis: a fixed snapshot, an exact encoding of the table's cost model, a descending SAT search in which a timeout is recorded as undecided, DRAT proofs checked by drat-trim for the decisive refutations, an independent re-simulation of every circuit, a reading of the consuming tool that separates the entries it can reach from those it cannot, a partition into classes on which cost is invariant so that one decided entry settles its class, and cross-checks between databases used only to refute. We report what the protocol found on seven published sources. On the 614,187-entry AIG database of the Simplifier tool it decided all 7,433 classes the tool can reach, lowered two of them, verified 204 proofs of up to 4.0 GB, and showed that 10,390 of the 10,449 improvements lie in entries the tool never looks up. A second database by several of the same authors stores the same two classes at the same non-minimal sizes. In a widely used synthesis library, a database labelled size-optimum since 2018 is above the minimum in 13 of the 135 classes it covers, and the complete database in the same file already shows it without a solver; we reported it as lsils/mockturtle#714 on 15 October 2026. A table of minimum formula sizes prints a total that its own columns contradict; we reported it to the authors on 15 October 2026. In a recent preprint on gate elimination, one lemma does not state a tree condition that its induction needs, the base case of one algorithm fails under one reading of its input condition, and one step of another algorithm undoes its own restriction as we read it; each is shown on an instance small enough to check by hand. An audit of a paper on simple circuit extensions found no error of content. In a paper on S-box implementations, the depth-1 AND counts of 26 of the 27 five- and six-bit S-boxes match what our exact depth-1 routine returns as the minimum, with the certification status of each lower bound in Section 6, one count can be lowered from 9 to 8, and one linearity value disagrees with the S-box it describes. As far as we can tell, neither reproducing the published pipeline nor counting how many sources agree would have exposed the findings on the Simplifier and Cirbo databases or on the algorithm steps of the preprint: reproduction re-derives the upper half of an optimality claim, and agreement multiplies upper halves. The table of formula sizes, the linearity value, the supplementary circuit for Q2260 and the omitted hypothesis of the preprint's Lemma 21 mark the boundary of this claim, because recomputing, simulating or re-reading the source exposes all four, and the library label is not an instance of it. We relate the protocol to the refutation gap recently named by Pandey and state which of our own verdicts are still bare solver output.

1. Introduction

1.1 Published tables of "optimal" circuits

Exact synthesis of small circuits is old, expensive and widely reused. Its results are rarely recomputed by the people who use them: they are frozen into tables and shipped with tools. The ABC rewriting pass [Mishchenko et al. 2006] and its descendants replace subgraphs by precomputed structures that cover the NPN classes of four-input functions. The mockturtle library ships databases of XAGs and AIGs for the same classes, and one of them is described in its header as "size-optimum". The Simplifier tool looks up three-input, up-to-three-output

subcircuits in a database of 614,187 AIGs whose authors state that part of it could not be proven optimal. Tables of minimum formula sizes for all NPN classes of up to five inputs have been published with a workshop paper. In lightweight cryptography, papers report the number of AND gates needed for each S-box, and later papers compare against those numbers.

Each table carries one of two claims. The weak claim is that every stored circuit computes the function it is stored under. The strong claim is that no smaller circuit exists. The weak claim can be checked by simulation in milliseconds. The strong claim is universal over circuits, so checking it needs a proof of non-existence: a counting or algebraic bound where one applies, and otherwise a checked refutation of the encoding, which is the route we take where the proof was retained (Section 6 lists where it was not). A bare solver answer is not that proof; Section 6 labels ours as solver output wherever the proof is missing. The strong claim is the one that tools and later papers lean on: a pass that draws its replacements from the table can do no better than the stored circuit on each lookup, and a paper that compares its S-box circuit with a published count treats that count as the bar.

1.2 The refutation gap, applied to other people's data

Pandey has recently named the asymmetry in these claims the *refutation gap*: the upper half of an optimality claim comes with a circuit that anyone can simulate, while the lower half usually rests on a bare UNSAT answer [Pandey 2026, arXiv:2609.20873]. We adopt his term, in his sense, and credit it to him. He closes the gap for 121 of his own results on minimum linear straight-line programs over GF(2), with n from 6 to 9: 111 by a DRAT refutation checked by drat-trim and 10 by a counting bound. He also reports that the published gate counts he examined for the AES MixColumns matrix carry no certificate for either half.

This paper applies the same requirement to optimality data that other authors published and that tools consume. Our thesis is that exact computation run against a published table finds defects that neither agreement between sources nor reproduction of the published number can reach, because both operate on the upper half of the claim; where a proof was retained, it is what makes a corrected value, or a row with nothing wrong, more than solver output, and Section 6 lists where it was not. We support it with seven audits, one for each published source that has its own subsection in Section 3: the Simplifier and Cirbo databases, the mockturtle databases, the IWLS'19 tables, and the three papers of Sections 3.5 to 3.7 (a gate-elimination preprint, a paper on simple circuit extensions, and an S-box paper). The count is by source, so the two Carmosino–Dang–Jackman papers are two. Six found something the source does not state, and one found no defect in the intended results, though one of its lemmas is false as literally stated and is repaired by a hypothesis its own proof already uses. The findings are of five kinds: a stored circuit or a reported count above the minimum; stored entries that the consuming tool can never reach; entries that the database should hold and does not; a mathematical statement that fails as printed on a small instance; and a printed value that the source's own data contradict. Five findings do not fit the thesis. The IWLS'19 error is found by re-summing a table (Section 3.4), one linearity value of the S-box paper by recomputing it from the S-box (Section 3.7), the supplementary circuit for Q2260 by simulating it against the table it is given for (Section 3.7), the omitted hypothesis of Lemma 21 in the gate-elimination preprint by reading its statement against its own proof (Section 3.5), and the mockturtle label is refuted by a smaller circuit that the same file already ships (Section 3.3); we report all five, and Section 4 says what each does and does not show.

Some of the other findings fall inside gaps the authors declared themselves (Section 1.4). The audit without errors of content is reported because the protocol is supposed to say so when that happens.

The thesis has precedents outside circuit tables. When SAT solvers with DRAT certificates were used to redo the search behind the nonexistence of a projective plane of order ten, the counts of one intermediate stage disagreed with both the original search of 1989 and its independent verification of 2011, which also disagreed with each other [Bright et al. 2021]. A checker extracted from a Coq formalization confirmed, without finding an error, the published optimal sizes of sorting networks on up to eight inputs [Cruz-Filipe and Schneider-Kamp 2015]. For optimal Golomb rulers, whose values up to order 28 were settled by distributed search, Niu proposes deciding order 29 with a SAT encoding that emits checkable proofs, and reports LRAT proofs up to order 12, which the note validates structurally rather than with a checker [Niu 2026].

1.3 Contributions

1. An audit protocol for published optimality data, with the certificate each verdict needs and the label it gets when the certificate is missing (Section 2).
2. One step that we have not found in earlier audits of published optimality data (the search behind this statement is not closed, Sections 5 and 6): deciding which entries the consuming tool can reach, by reading and then tracing its lookup code (Section 2.6). Transferring a verdict across a class on which cost is invariant (Section 2.7) is not ours: such tables are usually built that way, one representative per class, as in Knuth's tables [TAOCP 4A, §7.1.2], in Haaswijk [2019] and in Cirbo [Averkov et al. 2025b]. We use it in the audit and state each transfer as such.
3. Case studies with counts and sources: the Simplifier and Cirbo AIG databases, the mockturtle NPN databases, a table of minimum formula sizes, a preprint on gate elimination, a paper on simple circuit extensions, and a paper on low-AND-depth S-box implementations (Section 3).
4. An account of why agreement and reproduction miss most of these findings, with the cases in which reproduction does catch one and the case in which the file itself already refutes the label (Section 4).
5. A list of our own verdicts that are still bare solver output (Section 6).

1.4 What this paper does not claim

We do not claim that the audited authors made claims they did not make. Simplifier's authors state their gap; the S-box paper of Section 3.7 does not use the word optimal for its tables. We do not claim to be the first to certify a minimum circuit size with DRAT (Section 5). We do not claim lower bounds beyond the small sizes that exact synthesis reaches: every audited object has at most six inputs. And we do not claim that certified ground truth will find something in every published table. The thesis is about what the three instruments can reach, and the evidence for it is the seven sources audited here; one of the seven yielded no defect of content, which is the outcome the protocol is supposed to report when it happens.

---

2. The audit protocol

The protocol has nine steps. Each step names what it produces and what it cannot establish.

2.1 Snapshot. Fix the audited object by commit and SHA-256 (for a file) or by arXiv version and PDF hash (for a paper), and re-check before writing that the upstream still serves the same bytes. All line indices and counts refer to the snapshot, and a verdict holds for the snapshot only. A file that changes after the audit needs a new audit before any verdict is cited for the new bytes, and the old line references no longer apply.

2.2 Cost model and encoder. Read the cost model off the source (which gates, whether inverters are free, whether outputs may share gates, whether an output may be an input), and build a CNF whose models are circuits of k gates in that model. Normalisations that shrink the search (no duplicate gates, every gate used, an order on gate inputs) must keep at least one circuit of every size that exists, and when an UNSAT answer at one size is read as excluding all smaller sizes, that step needs its own argument (for example, padding a smaller circuit to the size asked for). Both arguments must be written down, because a DRAT proof certifies the CNF, not the encoding.

2.3 Descending SAT search, with timeouts kept as timeouts. For an entry stored with k gates, ask for $k - 1$. UNSAT means the stored size is minimum; SAT yields a smaller circuit, and the search descends until UNSAT. A call that hits its time cap is recorded as undecided, never as either answer. The cap is a parameter of the run, not of the problem: in the Simplifier audit a 30-second cap left 15,937 entries undecided, and rerunning one entry per class with a cap of seven days decided every class, with a median of 121 s per call and a maximum of about an hour; the other entries took the verdict of their class (Section 2.7).

2.4 Proofs for the decisive refutations. An UNSAT answer presented as certified requires a checked proof: re-solve with proof logging, have drat-trim check the proof, and record the SHA-256 of the CNF. An UNSAT answer kept without one is retained as solver output and labelled as such wherever it appears. Where proofs are too large

to keep, keep the CNF hash and the command that regenerates the proof. Verdicts without a checked proof are labelled solver output in every table.

2.5 Independent re-simulation. Every circuit we report, ours or theirs, is re-simulated by a checker that shares no code with the encoder and derives each disputed convention (bit order, input order, output polarity) from the data, aborting when the evidence is not unanimous. This step confirmed a defect in our own pipeline that a model reviewer had flagged: the first emission of improved circuits had the input variables reversed (Section 3.1).

2.6 Reach by the consumer. A database entry matters only if some tool can return it. Read the consuming code to find which keys it builds and which outputs it strips before the lookup, and classify every entry as reachable or not. Then run the tool with tracing, on constructed inputs that exercise each stripping rule and on its own benchmarks, and check that no lookup ever uses a key the reading says is impossible. The reading covers every entry; the run covers only the lookups it made.

2.7 Partition into classes, and transfer by class. When cost is invariant under a group (for AIG size with free inverters: permutation and negation of inputs, negation and permutation of outputs), partition the entries into orbits. One decided entry settles the minimum of every entry in its class, and one improved entry shows that every entry of its class stored at the same size can be improved. We report which entries were solved and which inherit a verdict, and for inherited improvements we transport the circuit through the class transformation and re-simulate it.

2.8 Cross-checks between databases, used only to refute. Comparing two published tables of the same functions is cheap and useful in one direction only. A circuit in another table that is smaller than our minimum would refute our UNSAT; we look for that. A circuit in another table that equals the first table's size says nothing about minimality: it is a second upper bound. Section 3.2 shows two databases agreeing on two non-minimal sizes.

2.9 Reporting. Each verdict is labelled with the certificate behind it: checked proof, counting or algebraic bound, solver output, re-simulated witness, or transfer by class. Totals are reported per label, and each total is produced by a tool over a registry rather than by a search over the working repository: every artefact that carries a certificate is classified as counting or not counting towards that total, with a written reason, and an artefact that no one has classified stops the tool instead of silently moving the number. We adopted this after answering one such question three times by hand in one evening and getting three different answers, two of which were wrong for reasons the registry now names: a column that mixes functions of four and five inputs, and a row whose certificate belongs to a function built from the one the row names. A statement about a paper is classified as an error of content, an editorial omission, or a difference of convention, and the instance behind it is printed small enough to check by hand.

---

3. Case studies

3.1 The Simplifier AIG database

Object. `databases/database_aig.txt` of `github.com/SPbSAT/simplifier` at commit `5b088f2f`, 614,187 entries: AIGs for Boolean functions with three inputs and up to three outputs, with size the number of ANDs, free inverters and shared gates. The Simplifier paper states that for several classes its authors have efficient circuits but no proof of minimality, and its Table 1 marks three AIG rows as upper bounds ("≤ 9", "≤ 10", "≤ 11"). The caption of its Figure 4 calls a size-11 circuit "presumably optimal".

Verdicts. The original pass, with a cap of 30 s per call, confirmed 587,801 entries as minimum (solver output, no proof kept; see Section 6), improved 10,449 (saving 11,447 ANDs in total) and left 15,937 undecided, all of them three-output entries stored with 10 or 11 ANDs.

Classes and reach. Under permutation and negation of inputs and outputs the 2,763,520 sets of three distinct three-input functions fall into 8,886 classes, which is the sum of the AIG column of the paper's Table 1. The tool's only consumer of the database strips constant, literal, duplicate and complementary outputs before it builds a lookup key. 7,433 classes are reachable and 1,453 are not. Every reachable class was decided: 7,431 are confirmed

at their stored size and 2 are lowered, the class of the truth tables (22, 24, 52) from 10 ANDs to 9 and the class of (24, 45, 105) from 11 to 10. Both lie in rows the paper marks as upper bounds. Under our transcription of the paper's Figure 4, its function lies in one of the five classes confirmed at 11, so the circuit the paper calls presumably optimal would be minimum. The transcription is ours and has not been checked by the authors. Three short edges have unclear polarity at print resolution; of the eight readings, only one computes a function in the six classes stored with 11 ANDs, where the paper places its function, and we use that one. The certified minimum is the one of line 595150, carried to the function of the figure by the invariance of cost on the class.

Unreachable improvements. 10,390 of the 10,449 improvements, 99.4%, are on entries that the Simplifier lookup path can never return — unreachability is a property of that consumer's code, not of the file, and other software reading the same file may reach them: each has an output equal to an input literal (7,777) or two complementary outputs (2,613). The other 59 lie in the two lowered classes. Running the tool with tracing on 11 constructed subcircuits and 11 of its 12 benchmark circuits gave 978,616 lookups, none of them with a key the reading of the code excludes.

Coverage. The reachable three-output keys form 432,678 orbits under input permutation, the only symmetry the lookup tries. The database has a key in every orbit but one: six keys, (44, 107, 142) and five others, are absent, so a subcircuit computing any of them finds nothing.

Proofs. The 204 deciding refutations of the rerun (one per open class, plus one for the class whose minimum the original pass had only bounded) were solved again with proof logging. drat-trim verified 204 of 204 proofs; the median proof is 189 MB and the largest 4.0 GB, and checking took a median of 2.25 times the solve time (1.49 times on the largest proof). Proof logging and checking took 22.1 h of kissat and 83.9 h of drat-trim, summed over the calls. The other 587,801 confirmations of the original pass are solver output: the CNFs were not kept.

Our own defect. The first emission of the improved circuits had the input variables in reverse order. Gate counts and verdicts were unaffected, since the minimum is invariant under input permutation, but the circuits computed the mirrored functions. A model reviewer found the defect and our independent verifier confirmed it; the deposited circuits are the remapped ones, and the verifier now aborts when the derived input permutation is not the identity.

A cost model that does not fit. No single one of two natural cost models makes all the unreachable entries minimum. Under the audit's model, 10,390 of the 31,910 unreachable entries can be improved. With outputs taken as computed instead (an output may be an input, a negated output costs a gate), the minimum equals the stored size on 9,573 of the 10,390 improved entries, is smaller on 729 and larger on 88; on the 21,520 unreachable entries that the audit confirmed minimum, it exceeds the stored size on every one, so those stored circuits rely on free output inversion. Neither model accounts for the whole file, and the generator is not public.

3.2 The Cirbo AIG database

Cirbo, a tool by several of the same authors, ships its own AIG database for the same functions (commit `ca19a9f2`), and its paper gives the same upper-bound counts at "≤ 10" (224 classes) and "≤ 11" (6 classes). All 349,724 of its circuits compute the truth tables they are stored under. Of its 8,344 three-output classes, 8,063 also occur in the Simplifier file. In no shared class does Cirbo store a circuit below our minimum. The sizes differ in 632 classes: in 630 an output is an input literal and Cirbo spends one to five ANDs more, and the other two are exactly the two classes lowered in Section 3.1, which Cirbo stores at 10 and 11 ANDs, the same sizes as the Simplifier.

Two databases, released separately by overlapping teams, agree on the size of these two classes, and both are one gate above the minimum. We do not know whether the two files were generated independently; either way, their agreement is two upper bounds that coincide. It is the clearest instance in this paper of why agreement between sources does not certify a minimum (Section 4).

3.3 The mockturtle NPN databases

Cross-check (no defect). The mockturtle library ships, in `xag_npn.hpp`, complete AIG and XAG databases for the 222 NPN classes of four-input functions (added on 2021-04-29), and in `mig_npn.hpp` a MIG database described

as size-optimum. Against our exact values, every one of the 221 non-constant classes has an AIG and an XAG at our minimum, none smaller and none larger, and the MIG database matches our minimum in 222 of 222 classes. Used as Section 2.8 prescribes, this comparison is a test of our side: a smaller circuit in their files would have refuted one of our UNSAT answers, and there is none. It does not show that their circuits are minimum. Our lower bounds would, but for these four-input bases they are, with three exceptions, solver output without proofs (Section 6).

The label. The header of `xag_npn.hpp` has read "Replace with size-optimum XAGs and AIGs from NPN (from ABC rewrite)" since the file's first commit (604950f, 2018-12-06), almost two and a half years before the complete databases were added, and the class template defaults to the older, incomplete database. Under the registration rule of the class's own database builder, the incomplete database covers 135 classes; in 122 its size equals our XAG minimum and in 13 it is one or two gates larger (for example 7 against 5 for the class `0x036e`). The complete database in the same file has the smaller circuit in all 13, so the defect can be shown from the file itself, with no solver. A smaller circuit that is already shipped refutes the label, and no certificate is needed for that; our exact values show only that the complete database is itself at the minimum, and for XAG they are solver output (Section 6). The incomplete database has had the same cost per class since 2018. A search of the repository's issues and pull requests on 14 October 2026, over twelve terms covering the file name, the class name, the words of the label and their negations, returned only the pull requests that added the databases and unrelated subjects, and none that compared the two; this is a bounded search, and it does not establish that no one had noticed. We opened one on 15 October 2026 (issue 714). A maintainer replied on 22 October, confirmed the two provenance facts we had asserted and proposed a corrected header; at the snapshot audited here the file still carries the old one.

3.4 A table of minimum formula sizes

Lee, Jiang, Mishchenko and Brayton (IWLS 2019) enumerate minimum fan-out-free circuits for all NPN classes of up to five inputs and publish the formula length L(f) of each class. Their Table 2 ($n = 4$) prints 32 classes at $L(f) = 4$, while the columns of that row sum to 34, the "# classes" column then sums to 220 against a printed total of 222, their own Table 1 gives 34, and both their published data and our own computation of fan-out-free size give 34. The error is typographical and affects no result. We include it because it marks the boundary of our thesis: an arithmetic check of the published table, which is a form of reproduction, finds it, and no certificate is needed.

3.5 A preprint on gate elimination

Object. Carmosino, Dang and Jackman, *Constructive Separations from Gate Elimination*, arXiv:2604.23958v1 (27 April 2026). It is a theory paper; its statements about small circuits have finite instances that exact synthesis and exhaustive simulation can test.

Lemma 21. The lemma states that a circuit of size $3(n - 1)$ that can be partitioned into XOR_2 widgets computes XOR_n or its negation. As printed, it does not require the widgets to form a tree over disjoint variable sets, and as printed it is false: with $W(u, v) = (u \vee v) \wedge \neg(u \wedge v)$, the usual three-gate XOR block, the circuit $W(W(x_1, x_2), x_1)$ on the declared inputs x_1, x_2, x_3 (x_3 unused) has $6 = 3(3 - 1)$ gates, partitions into two widgets and computes x_2. The proof combines two circuits computing XOR "on distinct subsets of variables", and here the two subsets, $\{x_1, x_2\}$ and $\{x_1\}$, are distinct but not disjoint. The paper's own introduction speaks of binary trees of widgets, and a sufficient additional condition is that contracting each widget to a node yields a binary tree whose leaves are the n declared inputs, each occurring once; with it the induction goes through. We therefore classify the point as a hypothesis missing from the statement, which the proof as printed also needs, and not as an error in the intended result. We asked the authors whether the recognition procedure is meant to require that the leaves are exactly the live variables, each used once, as the induction in Lemma 21 needs. The question matters inside the paper's checker (Algorithm 4), which relies on a recognition test for such partitions. In our implementation, on a 9-gate circuit with four inputs and one choice of gate that the algorithm permits, a test without the tree condition accepts a restricted circuit that is not parity, and 4 of the 28 choice paths end without returning an error. The circuit is $C = W(W(x_1, x_2), m)$, with W as above, where $m = (x_4 \wedge \neg x_3) \vee (\neg x_4 \wedge x_1)$ takes three more gates, nine in all. Write each truth table as a number with x_j as bit $j - 1$ of the row index; C is 0x69cc. For $x_4 = 1$ it equals the negation of $x_1 \oplus x_2 \oplus x_3$ (the high byte, 0x69), so the restriction is correct; for $x_4 = 0$ it equals x_2 (the low byte, 0xcc), which is not parity. The choice is $\alpha = x_1 \wedge \neg x_4$ with the variable x_4: fixing it sets $x_4 = 0$, the restricted

circuit passes the local test, and the checker recurses on the side where C is correct and finds no error. With the tree condition enforced, and with step 22 changed as described below under Algorithm 3, no path failed in our tests up to four inputs.

Algorithm 1, step 4. The base case of the refuter for the multiplexer evaluates the candidate circuit on four points and relies on the claim that any circuit with fewer than three gates errs on one of them. For $d = (0, 0)$, the one-gate circuit $x_1 \vee x_2$ agrees with MUX_1 on all four points, and for each of the four values of d some function of AIG cost at most 2 does. At $n = 2$, the 6-gate circuit $(x_1 \vee x_2) \wedge \neg a_1 \wedge \neg(x_3 \wedge a_2) \wedge \neg(x_4 \wedge a_2)$ is reduced by the recursion to $x_1 \vee x_2$ and passes the four points, although it does not compute MUX_2. This holds when the input condition of the algorithm is read as a bound on size, as Theorem 13 states it. The printed input condition bounds a different measure, defined later in the paper; under that reading the base case is safe, but Theorem 13 no longer guarantees the condition for this circuit. Evaluating all eight assignments repairs the step under both readings and keeps the running time polynomial; the theorem the algorithm serves still holds.

Algorithm 3, step 22. As we read it, the step flips the bit of the variable whose restriction made the circuit constant, which undoes the restriction. Take the three-gate circuit $g = x_1 \wedge x_2$, $t = \neg x_3 \wedge \neg g$, $C = x_1 \wedge t$, which computes $x_1 \wedge \neg x_2 \wedge \neg x_3$ ($n = 3$), with $p = 1$: the two points the step produces, (0, 0, 0) and (1, 0, 0), both agree with XOR_3, so no error is found, while flipping the other live variable gives (0, 1, 0), where C errs. A companion paper of the same authors flips another variable [Carmosino et al. 2026b].

A bound measured. The paper's Lemma 1 gives a lower bound of $2 \cdot 2^n + n - 2$ gates for the multiplexer MUX_n in the De Morgan basis, 8 at $n = 2$. We computed the exact AIG size of MUX_2 (six inputs): 9. Every size $k = 1, \ldots, 8$ is refuted with a DRAT proof checked by drat-trim, under a normalisation lemma proved on paper, and a 9-gate circuit is re-simulated on all 64 inputs. This is a measurement, not a defect: the paper says the bound is not tight asymptotically.

We reported the following privately to the authors on 13 October 2026, among other remarks: the reading of step 22 of Algorithm 3; the question on the tree condition in the recognition test of Algorithm 4, with the nine-gate instance above; the reading of the input condition of Algorithm 1, with the instance at $n = 2$; and the value $\mathrm{opt}(MUX_2) = 9$, reporting that the 7- and 8-gate refutations had been checked by drat-trim. The same message carried, among other remarks, the two textual points of Section 3.6, bibliographic notes and a citation point. It offered to share scripts and data. The counterexample to the literal statement of Lemma 21 was not part of it; we sent it to the authors on 1 November 2026, with the sufficient condition above, before printing it here. The refutations for the sizes below 7 were in neither message.

3.6 A paper on simple circuit extensions

Object. Carmosino, Dang and Jackman, *Simple Circuit Extensions for XOR in PTIME*, STACS 2026 (LIPIcs 364, 23:1–23:20), full version arXiv:2511.16903v1. We read its De Morgan circuits in normal form as AIGs with complemented edges and the same number of binary gates.

What was tested. We enumerated every optimal normalized circuit where that is feasible and tested each statement with finite content on all of them; the completeness of each enumeration rests on the solver's unsatisfiability answer at the end of it, without a proof (Section 6). For simple extensions with $n + m \leq 4$ (123 classes, 1,319 optimal circuits), Corollary 13 and Theorem 19 hold on every circuit, and so does Lemma 17 under the reading of "layered" discussed below. Claim 16 is tested on the 1,639 all-stops restrictions found while testing Lemma 17 on those circuits, and in all 1,639 the step is a single passing rule. For Lemma 14, none of the 55 orbits of pairs in which the extension is degenerate admits a circuit of the form the lemma describes (55 of 55 unsatisfiable), as the lemma requires. For XOR, the whole of Theorem 37, Lemma 35 and Corollary 34 hold at $n = 2$ and 3 (28 optimal circuits). Lemma 4 holds at $n = 1$ and 2, and the size inequality of Lemma 10 held on 20,000 random single-sink circuits (its layered property was not tested there).

What diverged. Two textual points, both in parts of the full version that are not in the conference text. Lemma 38 is stated for any circuit computing XOR_n, while its proof starts from an optimal normalized one; the 11-gate DNF circuit for XOR_3 computes XOR_3 and contains no block. We classify it as editorial. Definition 8 asks the depth of successive gates to be non-decreasing, while the proof of Lemma 10 picks the depth-maximal gate at

each step; under the literal reading, 54 of the 1,319 circuits have no layered restriction, and under the reading of the proof all have one. We classify it as a difference of convention. We found no result that depends on either.

3.7 Low-AND-depth S-box implementations

Object. Bolzer, Duval and Minier, *A New Tool to Find Lightweight (AND, XOR) Implementations of Quadratic Vectorial Boolean Functions up to Dimension 9*, arXiv:2601.08368 (IEEE TCAS-I 73(1)). For quadratic S-boxes of 5 to 9 bits it reports the number of ANDs of implementations with AND-depth exactly 1. It does not call those numbers optimal; it says that from 8 bits upwards it can no longer guarantee them. The supplementary material, with lookup tables and AND/XOR circuits for the S-boxes of these tables, was obtained from the authors on request and published in their repository on 15 October 2026 (github.com/seduval/implem-quad-sbox, commit 44d249cf).

What was tested. The 5- and 6-bit tables of the paper (Tables VII and IX) have 27 rows. We rebuilt the lookup tables of 17 of them from primary specifications and took the other 10 from the supplementary material, reading each one from the page image and from the text layer of the PDF; the two readings agree on every entry. Where both exist, the supplementary tables agree with ours up to the order of the bits (ASCON) and up to an isomorphism of fields (the two 6-bit power maps are given over an irreducible polynomial different from the one printed in the paper). Differential uniformity, linearity and branch number match the paper's tables in 26 of the 27 rows. For BL_ECCZ_11 the paper gives linearity 16, while the lookup table, which is also the function that the authors' circuit computes, has linearity 32: one component function has a quadratic part of rank 2, and two independent Walsh computations give 32.

The supplementary material gives a circuit for 26 of the 27 rows (all but SYCON). We simulated each on all 2^n inputs. Every one has AND-depth 1 and the numbers of ANDs and XORs printed in the tables, and 25 compute their lookup table. The circuit given for Q2260 (S_2260 in the supplementary material) differs from its table on 32 of the 64 inputs. In the line $q5 = (l13 \times l8) \oplus x5$, replacing x5 by $l0 = x4 \oplus x5$ is the only change of one operand that makes the circuit compute the table, and it keeps the same 8 ANDs and 29 XORs, so we read it as a typographical error.

At AND-depth 1 the minimum number of ANDs is a problem of linear algebra over GF(2), which we solved exactly. In 26 of the 27 rows the published count is what our exact depth-1 routine returns as the minimum; the witnesses are simulated, and the certification status of each lower bound is given in Section 6. For BL_ECCZ_1 the paper reports 9 ANDs; the exact routine finds a circuit with 8 ANDs at depth 1, which two independent simulators check against the supplementary lookup table on all 64 inputs, and refutes 7. The paper does not call its counts optimal, so this lowers one cell of its table and contradicts no statement in it. Each minimum circuit was re-simulated. On inputs of three and four bits, the exact routine agrees with an independent breadth-first search on every subspace of quadratic parts (16 and 2,825 subspaces). The 7- to 9-bit tables were not audited.

Depth above 1 (pending). For eight of the 27 (the 5- and 6-bit chi maps, ASCON, SYCON, S_51, S_52, S_53 and S_61) the published count equals the dimension of the span of the quadratic parts of the coordinates, a lower bound that holds at any depth, so these eight are minimum without a depth limit. For the other 19, whether a deeper circuit needs fewer ANDs than the depth-1 minimum is open, and whether minimum AND count and minimum AND count at depth 1 coincide for quadratic vectorial functions in general is listed as an open question [Bilgin et al. 2020]. For the 5-bit S-box of PRIMATEs (FIDES-5), an exact search left the answer at 6 or 7 after several months on 120 cores [Stoffelen 2016]; a later paper reports 7 [Zhang and Huang 2023], and its tables carry no unsatisfiability answer at 6 for this S-box: the one row they report for it is unsatisfiability at 5, while the row at 6 belongs to the inverse S-box. Our own refutation of 6 ANDs, below, supports the value by an independent route, under the reduction stated with it. SAT instances without a depth limit are running for the 13 of these 19 whose lookup tables we had before the supplementary material arrived, and for none of the other six; four of those rows have since answered, each UNSAT one AND below the published count, which closes them at the depth-1 minimum: X^9 over GF(2^6) at 6, and FIDES-5, X^3 and X^5 over GF(2^5) at 7. The first needs no assumption beyond the encoder. The other three fix the first AND product, a reduction that is sound for these orbits because the stabiliser of the span of the quadratic parts acts transitively on planes, and we have not yet reproduced them without it. None of the four carries a proof log, so each rests on the solver's answer and on the soundness of the encoding. Of the nine instances that had not answered, eight were stopped on 23 October after 214.6 hours of

CPU each, with no line of output beyond the start marker, and one is still running; six rows have no instance at all. For those fifteen nothing here goes beyond depth 1. The eight were not dropped for impatience. Splitting one of them with lookahead produces exactly 2^d cubes at depths 8, 10, 12, 16 and 20, which is to say the search tree is full and no branch is refuted; and with 2^20 cubes, eleven of twelve sampled still do not close in 300 seconds. Extrapolating from that sample, and assuming the sampled cubes are representative, the cube-and-conquer route for a single one of the eight would cost on the order of 80,000 core-hours, some 373 times what the sequential run had already spent without answering. This is an estimate from twelve cubes, not a proved lower bound.

Of the 27 rows, 26 match what that routine returns as the depth-1 minimum, and one reports a count that a depth-1 circuit lowers by one AND. One linearity value disagrees with the S-box it describes, and one supplementary circuit has a typographical error that a single operand repairs. We reported these points to the authors on 15 October 2026, among other remarks.

---

4. What agreement and reproduction do not find

We use *agreement* for any count of sources that report the same value: two databases, a database and the papers that cite it, several reviewers, or several model families. We use *reproduction* for re-deriving the published value with the producer's method: rerunning the generator, re-summing a table, re-reading a proof.

The structural reason. An optimality claim has two halves. Reproduction re-derives the upper half: rerunning the producer's search yields the same circuit, and a solver that timed out for the producer times out again under the same cap. Agreement multiplies upper halves: two tables that store the same circuit size are two witnesses for the same upper bound. Neither touches the lower half, which needs a refutation. In every database audited here, the published data contains no refutation to reproduce: what is published is the circuit and the count. A defect in the lower half (the stored circuit is not minimum) is invisible to both until someone produces either a smaller circuit or a proof. The smaller circuit finds the defect; only a proof says where the improvement stops, or that a row with nothing wrong is at the minimum.

Table F: how each finding was made.

case	finding	would reproduction expose it?	would agreement?	what exposed it
Simplifier, two lowered classes	stored size one above the minimum	no, as far as we know: the producers report that their solver could not settle rows like these (we do not know their cap)	no: Cirbo stores the same sizes	SAT descent with a cap long enough, then DRAT
Simplifier, 10,390 unreachable improvements	entries the tool can never return	no: unreachability is a property of the consumer's code, not of the file	no	reading and tracing the consumer
Simplifier, missing orbit	six keys absent	not assessed: we did not find the generator in the public repository at the audited commit	no	enumerating the key universe
mockturtle label	"size-optimum" above the minimum in 13 classes	no: building the database reproduces the same circuits	in part: the complete database in the same file disagrees in all 13, and a bounded search of the repository's issues and pull requests on 14 October 2026 found none that compared them (lsils/mockturtle#714)	comparing the incomplete database with the complete one in the same file, with no solver; our exact values only add that the complete one is at the minimum, and for XAG they are solver output
IWLS'19 Table 2	32 printed for 34	yes: the row and column sums disagree	n/a	arithmetic

case	finding	would reproduction expose it?	would agreement?	what exposed it
arXiv:2604.23958, Lemma 21	tree condition not stated	yes, in the wider sense defined above: comparing the statement with its own induction exposes it. In the narrow sense no, because the proof uses the condition, so re-reading the proof passes	n/a	the omission: reading the statement against the induction in its proof; its effect: running the checker of Algorithm 4, which rests on the statement, on a nine-gate circuit
arXiv:2604.23958, Algorithm 1 step 4	base case misses a 1-gate circuit	not by re-deriving the paper's results: the proof asserts the property the instance violates	n/a	enumerating all circuits of cost ≤ 2
arXiv:2604.23958, Algorithm 3 step 22	the step undoes the restriction that made the circuit constant	not by re-deriving the paper's results, which do not exercise this branch: the step as printed is what fails	n/a	following the step on a three-gate instance
S-box paper, BL_ECCZ_1	9 ANDs reported at depth 1, where 8 suffice	not assessed: we did not rerun the producers' tool	n/a: not assessed	exact minimization at depth 1, with the 8-AND circuit simulated
S-box paper, BL_ECCZ_11	linearity 16 printed, 32 for the S-box	yes: recomputing the Walsh spectrum from the lookup table	n/a	recomputation
S-box paper, supplementary circuit for Q2260	the circuit does not compute its table	yes: simulating the circuit	n/a	simulation

The mockturtle row is not an instance of this claim. The label "size-optimum" is refuted by a smaller circuit that the same file already contains. A smaller circuit is a witness for the upper half, so the finding needs no refutation and no certificate. We keep the row because the finding is real. What the protocol adds is the exact value of each class: by our solver the complete database is itself at the minimum, though those minima are, with the exceptions of Section 6, solver output without proofs.

Where reproduction does work. Reproduction is not useless. The IWLS'19 case is caught by re-summing a table, the linearity value of Section 3.7 by recomputing it from the lookup table, and the supplementary circuit for Q2260 by simulating it against the table it is given for. Pandey reports a case in which reproducing a published count exposed a matrix built with the wrong structure, which his internal checks had not caught. The claim of this section is narrower: reproduction and agreement establish the upper half of an optimality claim, and the findings that live in the lower half, or outside the table in the code that consumes it, need a different instrument.

Agreement in the literature. Exact-synthesis tools have been validated in the same way. One thesis checks the distribution of minimum sizes that its tool computes against the one Knuth published, and its tool against other exact-synthesis implementations such as those in ABC; it compares the difficulty with that of verifying a SAT solver, for which, barring formal verification, "the best we can do is to test it against known results and other trusted solver implementations" [Haaswijk 2019, Appendix A.2].

Agreement as a control. Agreement between two exact computations is a different object from the agreement of Section 2.8: each side claims the minimum, not only a circuit, so a match bears on both halves of the claim, within the limits below. For the Flix compiler, Madsen, van de Pol and Henriksen generate a minimum formula for each of the 65,536 Boolean functions of four inputs by dynamic programming over increasing size, counting the occurrences of $\wedge$ and $\vee$ in negation normal form with negations free, which by De Morgan is formula size over AND with free inversion, and print how many functions have each size [Madsen et al. 2023]. Our formula sizes for the 222 NPN classes, from our own dynamic program over formulas (the atlas in Section 7), expanded by their orbits, give all 16 of those counts. The two programs share no code, but they share the algorithm, so the agreement is evidence against defects the two do not have in common and not proof that neither has one; and it compares distributions, not classes, so two errors that cancel in the histogram would pass.

Agreement among reviewers. The same limit applies to review by models. In the program in which these audits were run, one model family proposed a gate-ordering constraint for the encoder and a second confirmed it as sound; a third produced a concrete circuit that the constraint wrongly excluded. Model reviews did find real defects here, including the reversed input order of Section 3.1 and, in drafting the note on the Simplifier, a citation to a deposit that did not yet exist. In each case what settled the matter was a circuit simulated or a file checked, not the number of reviewers who agreed.

Agreement between solvers. Two solvers that return UNSAT on the same CNF are also a vote. In the audit of Section 3.6, the 55 unsatisfiable instances behind Lemma 14 were answered by Glucose 4 and CaDiCaL 1.9.5 in agreement, without a proof. We report them as solver output (Section 6).

---

5. Relation to the refutation gap and to earlier work

The refutation gap. Pandey's paper [Pandey 2026, arXiv:2609.20873] is the closest work. It names the gap, states explicitly for this literature that optimality claims in circuit minimization should carry a certified refutation, measures the cost of doing so for linear straight-line programs over GF(2) (111 DRAT-checked refutations among 121 results, proofs up to 301 MB, checking at a median of 1.9 times the solve time), and reports that a time cap manufactured a tail of apparent failures. We adopt his term and credit it to him. The name, the explicit statement of the problem in this literature, his pipeline and his measurements are his; the principle of certifying both halves is older, as he writes himself: it is "a discipline which is standard in combinatorial optimization" that "has not reached this literature". Our work differs in its object and its scale. The object is optimality data published by others and consumed by tools, which adds the questions of reach (Section 2.6) and of transfer by class (Section 2.7). The bases are non-linear (AIG, MIG, XAG). His circuits also have several outputs, one per row of the matrix, but they are XOR only. Our proofs reach 4.0 GB in the Simplifier audit, and elsewhere in the same program a single proof of 59.19 GB was checked in 16.4 hours (it certifies the unsatisfiability of the CNF; reading it as a bound on circuits depends on the encoder and on a symmetry-breaking lemma); Pandey states that he has no evidence at gigabyte scale. His observation about time caps matches ours: in Section 3.1, a 30-second cap produced 15,937 undecided entries, and rerunning one entry per class with a longer cap decided the classes, with a median of 121 s per call over the 204 reruns; the other entries took the verdict of their class (Section 2.3).

Pandey writes that the discipline of certifying both halves "has not reached circuit minimization". The narrower statement in his introduction, about the literature on minimum XOR counts, is consistent with what we know. The general one is not: DRAT refutations for minimum AIG sizes, checked by drat-trim (the checker his own pipeline uses), with the scripts that generate them, were part of a public deposit of ours dated 27 July 2026 (doi:10.5281/zenodo.21630763), seven weeks before his first version. We say this to date the record, not to dispute his contribution; a data deposit is easy to miss, and the name and the thesis are his.

Proof logging and certified optimality. Certifying algorithms [McConnell et al. 2011], DRAT and drat-trim [Wetzler et al. 2014], and pseudo-Boolean proof logging with parity reasoning [Gocht and Nordström 2021] supply the instruments. Certified symmetry breaking is the subject of Bogaerts et al. [2023]. Our level of use is the simplest one: plain DRAT on encodings whose symmetry-breaking constraints are argued on paper (Section 6).

Exact synthesis. The encoders follow the SAT-based exact synthesis line of Kojevnikov, Kulikov and Yaroslavtsev [2009], Soeken et al. [2018a] and Haaswijk [2019], on the model of Boolean chains of Knuth [TAOCP 4A, §7.1.2]; Fuhs and Schneider-Kamp [2010] and Stoffelen [2016] used the same kind of encoding for linear and multiplicative complexity. In that line the minimality of a network rests on the unsatisfiability answers below it, and those answers are taken from the solver. Soeken et al. raise the gate count until the instance becomes satisfiable, and state that the procedure "clearly finds a Boolean logic network of minimal size"; Haaswijk writes that "if the problem is UNSAT, we know that no solution with r gates exists", and checks the satisfiable side by simulating the chain the solver returns. The easy half is checked and the hard half is not, and neither work reports a proof log for the second. Pandey makes the same observation for the linear case.

Audits of published artefacts. The databases audited here are those of Simplifier [Averkov et al. 2025a], Cirbo [Averkov et al. 2025b] and mockturtle [Soeken et al. 2018b], the formula tables of Lee et al. [IWLS 2019] and the S-box tables of Bolzer et al. [2026]. Earlier audits of published optimality data exist. Two outside circuits

appear in Section 1.2: a certified redo of a classification step whose counts disagreed with both earlier searches [Bright et al. 2021], and a verified checker that confirmed the sizes of optimal sorting networks on up to eight inputs [Cruz-Filipe and Schneider-Kamp 2015]. In cryptography, Zhang and Huang showed that a published SAT encoding for multiplicative complexity [Stoffelen 2016] can return UNSAT for an S-box whose algebraic normal form has a constant term, and re-checked with a corrected encoding the values that had been published with it; those values held [Zhang and Huang 2023]. For quantum circuits, simulation and translation validation have exposed optimized benchmark circuits that did not compute the intended unitary [Nam et al. 2018; Kissinger and van de Wetering 2020]. One circuit found wrong in the later paper is an output of the earlier one, which had itself found an error in a circuit produced by another optimizer. Closest to ours, Toledo's cert-machine re-derived two published records for 3×3 matrix multiplication in August 2026: a 55-addition circuit, checked separately for computing the right map and for its count [Karunaratne and Idamekorala 2026], and the certificate for a rank lower bound of 20 over GF(2) [Wang 2026], where it computed the true minimum on 18 of the 496 certificate nodes, refuted none, and left the other 478 resting on the author's rules [Toledo 2026]. Except for that lower-bound check, those audits test the weak claim. We found no audit of a database shipped with a logic-synthesis tool against certified lower bounds, in searches of 14 and 15 October 2026 over peer-reviewed venues, arXiv and the web. A negative search does not establish absence.

Our earlier audits. The same protocol, before it had this name, refuted a theorem of a preprint on the "unit gap" of AND-inverter circuits and closed the two entries that a published catalogue of four-input AIG minima had left open, at 10 gates. The refutations up to 8 gates are deposited with DRAT proofs; those at 9 gates (proofs of 4.79 and 3.87 GB) were checked with drat-trim on two machines, and only their CNFs and the command that regenerates the proofs were kept. The author of the preprint later withdrew it. These are reported elsewhere [Busnello 2026a; Busnello 2026b].

---

6. Limitations

Verdicts that are solver output, without a checked proof.

- The 587,801 confirmations of the original Simplifier pass, and the classes of Section 3.1 confirmed in that pass. The CNFs were not kept; they can be regenerated and re-solved, not checked from the deposit. Only the 204 deciding refutations of the rerun have checked proofs, and those proofs (up to 4.0 GB) were not kept either; the deposit holds the drat-trim verdicts and the SHA-256 of each CNF.
- The minima of Section 3.1 under the alternative cost model (31,910 entries), and the infeasibility answers of the size–depth fronts in the Simplifier note (for each of 2,000 sampled entries, the minimum size at each depth bound; the fronts are in the note and its deposit, not in this paper), apart from a sample of 200 checked with DRAT.
- The XAG, AIG and MIG minima for four-input functions used in Section 3.3. Of the 222 classes, the lower side can still be checked from retained artefacts for four in AIG — in full for two of them, and only as far as 9 of the minimum 10 for the other two — one in MIG and none in XAG; every other lower bound in the three bases is solver output without a proof. Three of those five minima are certified on both sides. In AIG, `0x0006` is unsatisfiable at up to 4 gates and satisfiable at 5, and `0x16e9` is unsatisfiable at up to 8 and satisfiable at 9, with every refutation checked by drat-trim and both witnesses re-simulated, so their minima of 5 and 9 rest on nothing else. In MIG, `0x6996`, four-input parity, is the same case: unsatisfiable at up to 5, satisfiable at 6, the witness simulated on all 16 rows, minimum 6. The other two are in AIG. One is `0x1669`, whose minimum is 10 and whose ladder is checked only to 8: the retained artefacts support 9, and the refutation at 9 gates that separates 9 from 10 was verified with drat-trim on two machines in July 2026 and then deleted, together with the one for `0x166b`, whose minimum is also 10 and whose retained artefacts likewise support 9: the deposit named in Section 7 carries DRAT proofs up to 8 gates for both classes. For those two what remains is the CNF hashes and the command that regenerates the proofs (Section 5). We report these counts separately by basis and do not add them. A lower bound can sometimes be carried from one of these bases to another through an inclusion or a gate simulation, but such a transfer needs an argument of its own and does not preserve tightness; and a count of certified classes in one basis is a count of a different property from a count in another, which is why the registry of Section 2.9 keeps the basis as a

column and will not aggregate over it. The defect of Section 3.3 does not depend on any of these minima: the file itself shows it.
- The 55 unsatisfiable instances behind Lemma 14 in Section 3.6 (two solvers in agreement, no proof), and the completeness of the all-solutions enumeration of optimal circuits in Section 3.6, which rests on the solver's UNSAT at the end of each enumeration.
- In Section 3.7, the depth-1 minima come from an exact algorithm of ours, cross-checked against an independent search at $n = 3$ and 4, with every witness simulated. The lower side has a DRAT certificate only for X^3 and X^5 over GF(2^5), and for FIDES-5, which is in the orbit of X^3: the SAT refutation of 6 ANDs at depth 1 was checked by drat-trim, on a CNF whose symmetry-breaking constraints are argued in prose. Elsewhere at $n = 5$ and 6 there is none; in particular, that BL_ECCZ_1 needs 8 ANDs at depth 1 and not 7 rests on the algorithm alone, while the 8-AND circuit is simulated.

What the proofs do not cover. A checked DRAT proof shows that a CNF is unsatisfiable. That the CNF encodes "no circuit of k gates" rests on the encoder and its normalisations, argued in prose and tested on known tables; transfer by class rests on the invariance of cost, also argued in prose. A DRAT proof certifies the CNF it was given: a published encoding for multiplicative complexity could return UNSAT for a function with a constant term in its algebraic normal form [Zhang and Huang 2023, Remark 1], and a proof for that CNF would have checked just as well. The same gap appears at scale in machine-produced proofs: in October 2026 OpenAI released a Lean proof claiming to refute the Exponential Time Hypothesis, the kernel's acceptance could be reproduced from the published sources, and what remained for the examiners was whether the formal statement says what the informal claim says [OpenAI 2026; Busnello, in preparation]. drat-trim itself is not formally verified; checking LRAT proofs with a verified checker would close that link, and we have not done it.

Other limits. Reach is read from the source and observed on traced runs, not exercised entry by entry. The statement about the Simplifier paper's Figure 4 rests on our transcription of a drawing. Every audited object is small (at most six inputs), so the protocol says nothing about how published heuristic results for large circuits compare with their optima. All audits were run by one operator in one research program. Each case depends on a snapshot; a changed upstream file needs a new audit (Section 2.1). The assignment of the classes to the rows of the Simplifier paper's Table 1 is inferred from the class counts, and the statement that both lowered classes lie in rows marked as upper bounds rests on it. We read the Cirbo database with a decoder of our own, without running Cirbo's code. In Section 3.1 the transcription of the Figure 4 circuit rests on one of eight readings of three unclear edges, chosen because it is the only one that lands in the classes of 11 ANDs, and the certified minimum is that of line 595150, carried to the figure by invariance of cost. The statement that we have not found the step of Section 1.3, item 2, in earlier audits, and the negative search reported in Section 5, depend on a prior-art search that is not closed.

---

7. Data availability

- Simplifier audit: per-entry verdicts, the 10,449 improved circuits, encoder, driver and independent verifier, doi:10.5281/zenodo.22726276 (v1.0.0); the rerun of the undecided tail — the 203 open classes and the one entry whose minimum the original pass had only bounded — doi:10.5281/zenodo.23328069 (v1.0.1); the drat-trim verdicts and CNF hashes of the 204 deciding calls (the proofs themselves are not in the deposit), the comparison with Cirbo and the size–depth fronts, doi:10.5281/zenodo.23375519 (v1.0.2).
- Note on the Simplifier audit: doi:10.5281/zenodo.23329260 (v1.0) and doi:10.5281/zenodo.23376253 (v1.1).
- CNFs, DRAT proofs for $k \leq 8$ and a normalisation lemma for the two four-input classes of AIG size 10: doi:10.5281/zenodo.23329385.
- Atlas of the 222 NPN classes of four-input functions in three bases: doi:10.5281/zenodo.23329555 (v1.0) and doi:10.5281/zenodo.23353481 (v1.1).
- Artefacts of the unit-gap refutation, including the July 2026 DRAT proofs: doi:10.5281/zenodo.21630763 (v1.0.0, published 2026-07-27). The concept doi is 10.5281/zenodo.21630762.
- The scripts of Sections 3.3 to 3.7 (mockturtle and IWLS'19 checks, gate-elimination refuters, the CDJ audit, the S-box audit, MUX_2), with the verbatim run captures beside them: doi:10.5281/zenodo.23629701 (concept doi:10.5281/zenodo.23629700, which always resolves to the latest version).

Use of generative AI

This work used AI models extensively, under the author's direction: Anthropic's Claude for the analysis, code, encodings and drafting, and models of other families for read-only review. The author is responsible for all claims. Correctness rests on the proofs checked by drat-trim, on the re-simulation of every reported circuit by an independent checker, and on exhaustive enumeration where stated; the remaining verdicts are solver output without proof certificates (Section 6). It does not rest on the models.

References

1. R. Pandey. *The Refutation Gap: Certifying Both Halves of an Optimality Claim.* arXiv:2609.20873v1 [cs.LO], 16 September 2026.
2. D. Averkov, G. Emdin, V. Krivogornitsyna, A. S. Kulikov, F. Kurmazov, A. Smal, V. Vaskin. *Simplifier: A New Tool for Boolean Circuit Simplification.* arXiv:2503.19103v1, 2025. [Averkov et al. 2025a]
3. D. Averkov, T. Belova, G. Emdin, M. Goncharov, V. Krivogornitsyna, A. S. Kulikov, F. Kurmazov, D. Levtsov, G. Levtsov, V. Vaskin, A. Vorobiev. *Cirbo: A New Tool for Boolean Circuit Analysis and Synthesis.* AAAI 2025. arXiv:2412.14933. [Averkov et al. 2025b]
4. lsils/mockturtle, `include/mockturtle/algorithms/node_resynthesis/xag_npn.hpp` at commit `1c9c3af`, https://github.com/lsils/mockturtle.
5. S.-Y. Lee, J.-H. R. Jiang, A. Mishchenko, R. K. Brayton. *Enumeration of Minimum Fanout-Free Circuit Structures.* IWLS 2019.
6. M. L. Carmosino, N. Dang, T. Jackman. *Constructive Separations from Gate Elimination.* arXiv:2604.23958v1, 2026. [Carmosino et al. 2026c]
7. M. L. Carmosino, N. Dang, T. Jackman. *Simple Circuit Extensions for XOR in PTIME.* STACS 2026, LIPIcs 364, 23:1–23:20. Full version arXiv:2511.16903v1. [Carmosino et al. 2026a]
8. M. Bolzer, S. Duval, M. Minier. *A New Tool to Find Lightweight (AND, XOR) Implementations of Quadratic Vectorial Boolean Functions up to Dimension 9.* IEEE Transactions on Circuits and Systems I: Regular Papers 73(1):478–491, 2026. doi:10.1109/TCSI.2025.3602151. arXiv:2601.08368.
9. N. Wetzler, M. J. H. Heule, W. A. Hunt Jr. *DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs.* SAT 2014, LNCS, pp. 422–429. doi:10.1007/978-3-319-09284-3_31.
10. A. Kojevnikov, A. S. Kulikov, G. Yaroslavtsev. *Finding efficient circuits using SAT-solvers.* SAT 2009, LNCS, pp. 32–44. doi:10.1007/978-3-642-02777-2_5.
11. D. E. Knuth. *The Art of Computer Programming, Volume 4A.* Addison-Wesley, 2011. Section 7.1.2.
12. M. Soeken, W. Haaswijk, E. Testa, A. Mishchenko, L. Amarù, R. Brayton, G. De Micheli. *Practical exact synthesis.* DATE 2018, pp. 309–314. doi:10.23919/DATE.2018.8342027. [Soeken et al. 2018a]
13. W. Haaswijk. *SAT-Based Exact Synthesis for Multi-Level Logic Networks.* PhD thesis, EPFL, 2019.
14. K. Stoffelen. *Optimizing S-box Implementations for Several Criteria Using SAT Solvers.* FSE 2016, LNCS 9783, pp. 140–160. doi:10.1007/978-3-662-52993-5_8.
15. C. Fuhs, P. Schneider-Kamp. *Synthesizing Shortest Linear Straight-Line Programs over GF(2) Using SAT.* SAT 2010, LNCS 6175, pp. 71–84. doi:10.1007/978-3-642-14186-7_8.
16. R. M. McConnell, K. Mehlhorn, S. Näher, P. Schweitzer. *Certifying algorithms.* Computer Science Review 5(2):119–161, 2011. doi:10.1016/j.cosrev.2010.09.009.
17. B. Bogaerts, S. Gocht, C. McCreesh, J. Nordström. *Certified dominance and symmetry breaking for combinatorial optimisation.* JAIR 77:1539–1589, 2023. doi:10.1613/jair.1.14296.
18. S. Gocht, J. Nordström. *Certifying Parity Reasoning Efficiently Using Pseudo-Boolean Proofs.* AAAI 2021, 35(5):3768–3777. doi:10.1609/aaai.v35i5.16494.
19. B. Bilgin, L. De Meyer, S. Duval, I. Levi, F.-X. Standaert. *Low AND Depth and Efficient Inverses: a Guide on S-boxes for Low-latency Masking.* IACR ToSC 2020(1):144–184. doi:10.46586/tosc.v2020.i1.144-184.
20. L. A. Busnello. *The Unit Gap Is Base-Dependent: A Mechanically Verified Refutation and a Cross-Base Census.* Zenodo, doi:10.5281/zenodo.23226733 (DOI de conceito, resolve sempre para a última versão; v1.0 doi:10.5281/zenodo.23226734, v2.1 doi:10.5281/zenodo.23328820). [Busnello 2026a]
21. L. A. Busnello. *opt(0x1669) = opt(0x166b) = 10: CNFs, DRAT proofs for $k \leq 8$ and a normalization lemma for the last two open entries of the NPN-4 exact AIG catalog.* Zenodo, doi:10.5281/zenodo.23329385 (v1.0; conceito doi:10.5281/zenodo.23329384). [Busnello 2026b]

22. L. A. Busnello. *An exact-synthesis audit of the Simplifier AIG database: a quantified complement to its stated optimality gap.* Zenodo, doi:10.5281/zenodo.23376253 (v1.1; conceito doi:10.5281/zenodo.23329259).
23. M. L. Carmosino, N. Dang, T. Jackman. *Convergent Gate Elimination and Constructive Circuit Lower Bounds.* arXiv:2602.17942v1, 2026. [Carmosino et al. 2026b]
24. F. Zhang, Z. Huang. *Optimizing S-box Implementations Using SAT Solvers: Revisited.* Cryptology ePrint Archive, Paper 2023/1721, 2023. https://eprint.iacr.org/2023/1721
25. A. Mishchenko, S. Chatterjee, R. Brayton. *DAG-aware AIG rewriting: a fresh look at combinational logic synthesis.* DAC 2006, pp. 532–535. doi:10.1145/1146909.1147048.
26. C. Bright, K. K. H. Cheung, B. Stevens, I. Kotsireas, V. Ganesh. *A SAT-based Resolution of Lam's Problem.* Proc. AAAI 2021, 35(5):3669–3676. doi:10.1609/aaai.v35i5.16483. arXiv:2012.04715.
27. L. Cruz-Filipe, P. Schneider-Kamp. *Formalizing Size-Optimal Sorting Networks: Extracting a Certified Proof Checker.* ITP 2015, LNCS, doi:10.1007/978-3-319-22102-1_10; arXiv:1502.05209v2, 2015. Journal version: L. Cruz-Filipe, K. Larsen, P. Schneider-Kamp. *Formally Proving Size Optimality of Sorting Networks.* J. Autom. Reasoning 59(4):425–454, 2017. doi:10.1007/s10817-017-9405-9.
28. Y. Nam, N. J. Ross, Y. Su, A. M. Childs, D. Maslov. *Automated optimization of large quantum circuits with continuous parameters.* npj Quantum Information 4:23, 2018. doi:10.1038/s41534-018-0072-4. arXiv:1710.07345.
29. A. Kissinger, J. van de Wetering. *Reducing the number of non-Clifford gates in quantum circuits.* Phys. Rev. A 102(2):022406, 2020. doi:10.1103/PhysRevA.102.022406. Preprint: *Reducing T-count with the ZX-calculus*, arXiv:1903.10477.
30. M. Soeken, H. Riener, W. Haaswijk, E. Testa, B. Schmitt, G. Meuli, F. Mozafari, S.-Y. Lee, A. Tempia Calvino, D. S. Marakkalage, G. De Micheli. *The EPFL Logic Synthesis Libraries.* arXiv:1805.05121 (v1 2018; v3 2022). [Soeken et al. 2018b]
31. M. Madsen, J. van de Pol, T. Henriksen. *Fast and Efficient Boolean Unification for Hindley-Milner-Style Type and Effect Systems.* Proc. ACM Program. Lang. 7(OOPSLA2):516–543, Article 240, 2023. doi:10.1145/3622816.
32. T. Niu. *Toward a Formally Verified Optimality Certificate for OGR(29): A SAT-Encoding Methods Note with Small-Case Demos.* arXiv:2609.05421, 2026.
33. OpenAI. *Working with academic communities to build scientific understanding.* 26 October 2026. https://openai.com/index/building-scientific-understanding-with-academic-communities/. The Lean challenge is `lean/ComparatorChallenges/SubexponentialSAT.lean` in `openai/math` at commit `7ef5c0cc`, with its solution in the module `OAI.Algorithms.SubexponentialSAT.Main`. [OpenAI 2026]
34. L. A. Busnello. *Certify, don't vote.* In preparation. [Busnello, in preparation]
35. C. Toledo. *cert-machine: independent exact certification of machine-generated mathematics.* Zenodo, version v2026.10.27, 2026. doi:10.5281/zenodo.22225860. The audits cited are https://carlostoledo.co/reports/tensor-rank-bounds.html and https://carlostoledo.co/reports/matmul-additions.html (August 2026). [Toledo 2026]
36. S. Karunaratne, A. Idamekorala. *55 Additions Suffice for 3x3 Matrix Multiplication at Rank 23.* arXiv:2607.28676, 2026.
37. C. Wang. *Automated Lower Bounds for Bilinear Complexity over Finite Fields.* arXiv:2603.07280, 2026.