Formal Algebraic and $\mathbb{F}_2$-Circuit Certification for the NP-Completeness of Optimal CNOT Synthesis on Linear Oracles

Lluis Eriksson
Independent Researcher · CNOT-HARD-CERT-1.0
https://github.com/lluiseriksson/optimal-cnot-hardness-certificates

November 26, 2026

Abstract

We present a machine-checked constructive Lean 4.35.0 kernel and zero-float $\mathbb{F}_2$ circuit verification suite (`CNOT-HARD-CERT-1.0`) auditing Andreas Pavlogiannis's proof that optimal CNOT synthesis is NP-complete even for involutive linear oracles $B_G \in \mathrm{GL}(n+1+m, 2)$ and even with free SWAP gates or ancillas (`arXiv:2611.34138v1`, Nov. 2026). Our artifact combines 18 constructive Lean 4.35.0 theorems (`0 sorry`, `0` custom axioms, `0 Classical.choice`, `0` warnings), an AST-audited zero-float Python $\mathbb{F}_2$ bitmask verifier, 16 canonical JSON certificates (`C01`–`C16`), and 36 unit and forgery-rejection tests. In Lean we formalize linear-oracle involution, 3-CNOT SWAP equivalence, free-ancilla reduction rules, Lemma 2 clique-cover dominance and evenness, Lemma 3 gate-elimination Case (a)/(b) projections and telescoping step, Lemma 4 Hamiltonian-path CNOT wire invariants, z-wire restoration, and exact gate count $2m + \tau(G) + 1$, and Theorem 5 parity-rounding reduction soundness. In Python we execute the Lemma 2 reduction $H \mapsto H_1 \cup H_2 \cup K_{2n}$ with exact branch-and-bound vertex covers, step-by-step Lemma 3 gate elimination on optimal and Case-(b) XOR circuits, full $\mathbb{F}_2$ matrix simulation of synthesized Lemma 4 CNOT circuits up to $\mathrm{GL}(33, 2)$ with $B_G^2 = I$, and syntactic free-SWAP/clean-ancilla to XOR translation.

1 Introduction and Audited Scope

Given an invertible binary matrix $A \in \mathrm{GL}(d, 2)$ and a gate budget $K \in \mathbb{N}$, the *optimal CNOT synthesis* problem asks whether A can be implemented using at most K in-place CNOT gates $x_j \leftarrow x_j \oplus x_k$. Building on the XOR-circuit edge-form lower bound of Boyar, Matthews, and Peralta [2] ($D_{\mathrm{XOR}}(F_G) = m + \tau(G)$ for $F_G = \{z \oplus x_u \oplus x_v\}_{\{u,v\} \in E}$, where $\tau(G)$ is the minimum vertex cover size of $G = (V, E)$ with $|V| = n, |E| = m$), Pavlogiannis [1] proves NP-completeness directly on *linear oracles*

$$B_G = \begin{pmatrix} I_{n+1} & 0 \\ F_G & I_m \end{pmatrix} \in \mathrm{GL}(n+1+m, 2), \qquad B_G^2 = I, \tag{1}$$

via four modular pillars: (i) Lemma 2 reduces general Vertex Cover on H to $G = H_1 \cup H_2 \cup K_{2n}$ with $\tau(G) = 2\tau(H) + 2n$ even and a minimum vertex cover inducing a Hamiltonian path; (ii) Lemma 3 proves $D_{\mathrm{XOR}}(F_G) = m + \tau(G)$ by localized gate elimination; (iii) Lemma 4 proves $D_{\mathrm{XOR}}(B_G) = 2m + \tau(G)$ and $D_{\mathrm{CNOT}}(B_G) \in \{2m + \tau(G), 2m + \tau(G) + 1\}$ via an explicit Hamiltonian-path CNOT synthesis; and (iv) Theorem 5 reduces $\tau(G) \le k$ to $D_{\mathrm{CNOT}}(B_G) \le 2m + k' + 1$ (k' rounded down to even), persisting under free SWAPs and free ancillas.

This note provides an independent machine-checked audit (`CNOT-HARD-CERT-1.0`) with an explicit scope split:

1. **Constructive Lean 4.35.0 kernel** (`lean/CnotHardnessCert.lean`): 18 universally quantified Boolean/$\mathbb{F}_2$ and $\mathbb{N}$-arithmetic theorems (`0 sorry`, `0 Classical.choice`, audited via `#print axioms`) formalizing all algebraic, wire-update, gate-count, telescoping, and parity-reduction steps.

2. **AST-audited zero-float $\mathbb{F}_2$ verifier** (`src/cnot_hardness_verifier.py`): executes the finite-graph constructions, gate-elimination sequences, full $\mathbb{F}_2$ CNOT matrix simulations, and free-resource syntactic translations across 16 certificates.

2 Constructive Lean 4.35.0 Kernel

All 18 theorems in `CnotHardnessCert.lean` build in 0.93 s under Lean 4.35.0 with zero warnings and depend exclusively on at most `propext` and `Quot.sound` (zero `Classical.choice`, with 9 theorems depending on zero axioms). Table 1 summarizes all 18 quantified Lean statements in abbreviated mathematical syntax. Note honestly that Lean certifies the Boolean wire/gate identities, telescoping step, exact gate counts, and Theorem 5 parity-rounding soundness, while finite-graph vertex-cover branch-and-bound, explicit circuit execution, and gate-elimination replays are certified by our zero-float $\mathbb{F}_2$ Python verifier in Section 3.

Table 1: Summary of all 18 constructive theorems in `CnotHardnessCert.lean` (Lean 4.35.0, `0 sorry`, `0 Classical.choice`, 0 warnings).

#	Lean Theorem Name	Quantified Lean Statement (abbreviated mathematical syntax)
1	`linear_oracle_involution`	$\forall y, f \in \mathbb{B},\ (y \oplus f) \oplus f = y$ (0 axioms)
2	`reversible_edge_form_involution`	$\forall y, z, x_u, x_v \in \mathbb{B},\ (y \oplus (z \oplus (x_u \oplus x_v))) \oplus (z \oplus (x_u \oplus x_v)) = y$ (0 axioms)
3	`three_cnot_swap_exact`	$\forall a, b \in \mathbb{B},\ a_1 = a \oplus b,\ b_1 = b \oplus a_1,\ a_2 = a_1 \oplus b_1 \implies (a_2, b_1) = (b, a)$ (0 axioms)
4	`ancilla_zero_control_and_target`	$\forall c, t \in \mathbb{B},\ (t \oplus \bot = t) \wedge (\bot \oplus c = c)$ (0 axioms)
5	`lemma2_clique_cover_dominance`	$\forall n \geq 1,\ \tau_H \leq n-1,\ 2\tau_H + 2n < 4n - 1$
6	`lemma2_even_and_threshold_iff`	$\forall n, \tau_H, k,\ (2\tau_H + 2n) \bmod 2 = 0 \wedge (\tau_H \leq k \iff 2\tau_H + 2n \leq 2k + 2n)$
7	`lemma2_hamiltonian_interleaving`	$\forall n \geq 1,\ \tau_H \leq n-1,\ 2\tau_H < 2n \wedge 2\tau_H + (2n - 2\tau_H) + 2\tau_H = 2\tau_H + 2n$
8	`gate_elim_case_a_projection`	$\forall z, x_v \in \mathbb{B},\ (z \oplus z = \bot) \wedge (z \oplus (z \oplus x_v) = x_v)$ (0 axioms)
9	`gate_elim_case_b_projection`	$\forall z, x_u, x_w, x_v,\ x_u{=}x_w \to x_u \oplus x_w{=}\bot \wedge z \oplus (x_u \oplus x_w){=}z \wedge z \oplus (x_u \oplus x_v){=}z \oplus (x_w \oplus x_v)$ (0 ax.)
10	`gate_elim_telescoping_step`	$\forall c_1 \leq c_0, e_1 \leq e_0, t_1 \leq t_0,\ (e_0 - e_1) + (t_0 - t_1) \leq c_0 - c_1 \wedge e_1 + t_1 \leq c_1 \to e_0 + t_0 \leq c_0$
11	`lemma4_xor_lower_bound_lift`	$\forall c, c', m, \tau,\ c' + m \leq c \wedge m + \tau \leq c' \to 2m + \tau \leq c$
12	`lemma4_cnot_wire_invariant`	$\forall z, y, x_{s_i}, x_{s_{i+1}}, x_u,\ y_{2a}{=}y \oplus z \oplus x_{s_i} \oplus x_u \wedge y_{2b}{=}y \oplus z \oplus x_{s_i} \oplus x_{s_{i+1}} \wedge z_2{=}z \oplus x_{s_{i+1}}$ (0 ax.)
13	`lemma4_z_wire_restored`	$\forall z, x_{s_k} \in \mathbb{B},\ (z \oplus x_{s_k}) \oplus x_{s_k} = z$ (0 axioms)
14	`lemma4_cnot_gate_count_exact`	$\forall k \geq 1,\ k - 1 \leq m,\ 2 + 3(k-1) + 2(m - (k-1)) = 2m + k + 1$
15	`theorem5_even_budget_rounding`	$\forall \tau, k,\ \tau \bmod 2 = 0 \to k' \bmod 2 = 0 \wedge (\tau \leq k \iff \tau \leq k')$
16	`theorem5_reduction_iff`	$\forall m, \tau, k, d_{\text{cnot}},\ \tau \bmod 2 = 0 \wedge 2m + \tau \leq d_{\text{cnot}} \leq 2m + \tau + 1 \to (\tau \leq k \iff d_{\text{cnot}} \leq 2m + k' + 1)$
17	`free_resource_sandwich`	$\forall m, \tau, d_{\text{free}}, d_{\text{cnot}},\ 2m + \tau \leq d_{\text{free}} \leq d_{\text{cnot}} \leq 2m + \tau + 1 \to d_{\text{free}} \in \{2m + \tau, 2m + \tau + 1\}$
18	`xor_transitive_cancellation`	$\forall a, b, c \in \mathbb{B},\ (a \oplus b) \oplus (b \oplus c) = a \oplus c$ (0 axioms)

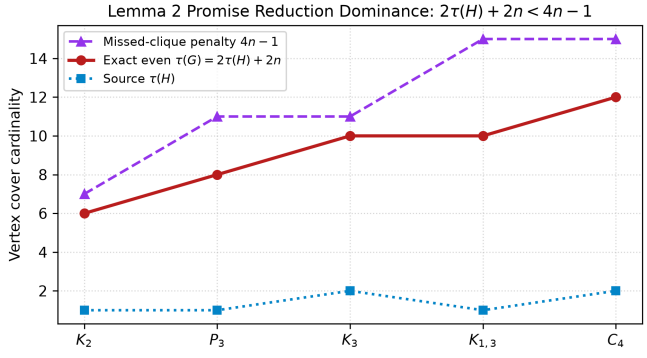

Figure 1: Lemma 2 promise vertex-cover reduction on $H \in \{K_2, P_3, K_3, K_{1,3}, C_4\}$ confirming strict clique-cover dominance $2\tau(H) + 2n < 4n - 1$.

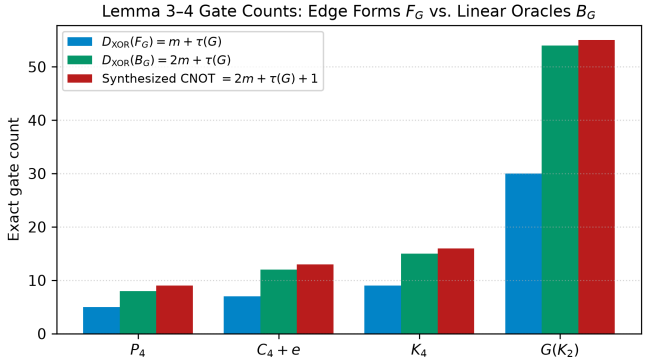

Figure 2: Exact XOR distance $D_{\text{XOR}}(F_G) = m + \tau(G)$, reversible XOR distance $D_{\text{XOR}}(B_G) = 2m + \tau(G)$, and synthesized Lemma 4 CNOT gate count $2m + \tau(G) + 1$ across `C09`–`C12`.

3 Zero-Float $\mathbb{F}_2$ Verification Modes

The verifier `src/cnot_hardness_verifier.py` is statically AST-audited by `pytest` for zero floating-point operations, representing every $\mathbb{F}_2$ linear expression and matrix row as an exact Python `int` bitmask.

Mode 1 (`promise_vc_reduction_lemma2`, `C01`–`C04`). Given a source graph H (n vertices), computes exact $\tau(H)$ via branch-and-bound, builds $G = H_1 \cup H_2 \cup K_{2n}$ (with complete bipartite edges between K_{2n} and $H_1 \cup H_2$), verifies $\tau(G) = 2\tau(H) + 2n$ independently by exact branch-and-bound on G (up to $n_G = 16$, `C03`–`C04`), and checks that the alternating sequence $s_1, u_1, \ldots, s_t, u_t, u_{t+1}, \ldots, u_{2n}$ is a valid vertex cover inducing a Hamiltonian path in $G[S]$ (Fig. 1).

Mode 2 (`gate_elimination_lemma3`, `C05`–`C08`). Given G and a supplied XOR circuit C over $\{z, x_0, \ldots, x_{n-1}\}$, verifies that C computes every edge form $z \oplus x_u \oplus x_v$, then *executes* Pavlogiannis's localized gate-elimination sequence $\langle C_0, G_0 \rangle \to \cdots \to \langle C_q, G_q \rangle$: locating the first two-input gate $g \leftarrow a \oplus b$, projecting $x_u \mapsto z$ (Case a, deleting u) or $x_u \mapsto x_w$ (Case b, merging u into w), removing all redundant gates, and verifying both invariant (1) (C_{i+1} implements $F_{G_{i+1}}$) and invariant (3) ($|C_i| - |C_{i+1}| \geq (|E_i| - |E_{i+1}|) + (\tau(G_i) - \tau(G_{i+1}))$) at every step, plus verifying that the extracted vertices cover G and that the optimal $(m + \tau(G))$-gate upper-bound circuit succeeds. Certificate `C06` explicitly exercises Case (b) vertex merges on a non-optimal 6-gate circuit for P_4.

Mode 3 (`linear_oracle_cnot_lemma4`, `C09`–`C12`). Constructs $B_G \in \mathrm{GL}(n + 1 + m, 2)$, verifies exact $\mathbb{F}_2$ matrix involution $B_G^2 = I$, validates the supplied Hamiltonian minimum vertex cover $s_1, \ldots, s_\tau$, synthesizes Pavlogiannis's exact $(2m + \tau(G) + 1)$-gate CNOT circuit (Steps (1)–(3)), and simulates every CNOT gate in-place over $\mathbb{F}_2$ to confirm exact matrix equality with B_G (Fig. 2). Certificate `C12` runs the full pipeline end-to-end on the Lemma 2 image $G(K_2)$ ($n = 8, m = 24, \tau = 6$), simulating a 55-gate CNOT circuit in $\mathrm{GL}(33, 2)$.

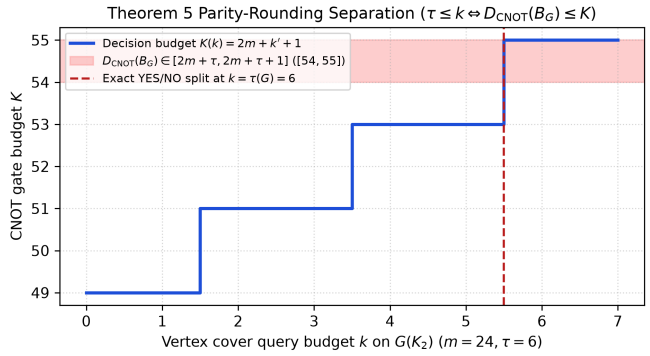

Figure 3: Theorem 5 parity-rounding decision budget $K(k) = 2m + k' + 1$ on $G(K_2)$ ($m = 24, \tau = 6$), cleanly separating YES ($k \ge 6$) and NO ($k \le 5$) instances around the Lemma 4 band $[54, 55]$.

Mode 4 (`free_resources_and_parity_thm5`, `C13`–`C16`). Verifies Theorem 5's even-τ parity-rounding decision equivalence ($\tau(G) \le k \iff D_{\mathrm{CNOT}}(B_G) \le 2m + k' + 1$, Fig. 3) on YES and NO instances, and executes Pavlogiannis's syntactic translation from CNOT circuits augmented with free SWAP gates and clean ancillas (0-control no-op, 0-target copy, SWAP wire relabeling) to an XOR circuit of at most #CNOT gates, confirming exact wire-by-wire output agreement over $\mathbb{F}_2$ and compliance with the $2m + \tau(G)$ lower bound.

4 Certificate Suite and Adversarial Testing

Table 2 summarizes all 16 canonical certificates (`C01`–`C16`). Our test suite (`tests/test_verifier.py`) executes 36 tests in under 0.25 s: all 16 certificates, the AST zero-float audit, a `C13` zero-control/zero-target reduction test (11 CNOTs translating to 9 XOR gates $\le K = 9$), and 18 adversarial forgery-rejection tests covering schema/mode tampering, `bool`/float coercion, forged $\tau(H)/\tau(G)/D_{\mathrm{XOR}}$/CNOT counts, incomplete or out-of-range XOR circuits, non-Hamiltonian and non-covering vertex sets in Mode 3, empty/incomplete circuits, odd-τ rejection, and flipped decisions in Mode 4, unknown gate ops, self-loops, and duplicate edges.

5 Reproducibility and Disclosure

All files are SHA-256 hashed in `MANIFEST.sha256` and archived at https://github.com/lluiseriksson/optimal-cnot-hardness-certificates (`v1.0.0`). Independent reproduction requires only Lean 4.35.0 and Python 3.10+:

```
cd lean && lake build && cd ..
python scripts/generate_certificates.py
python -m src.cnot_hardness_verifier
certificates/
pytest -q
```

Table 2: Certificate inventory (`CNOT-HARD-CERT-1.0`, 16/16 verified).

ID	M	Graph / Instance	Exact Certified Witness
C01	1	$H = P_3 \to G(12)$	$\tau(H) = 1, \tau(G) = 8$ even, Ham path
C02	1	$H = K_3 \to G(12)$	$\tau(H) = 2, \tau(G) = 10$ even, Ham path
C03	1	$H = C_4 \to G(16)$	$\tau(H) = 2, \tau(G) = 12$ (exact B&B)
C04	1	$H = K_{1,3} \to G(16)$	$\tau(H) = 1, \tau(G) = 10$ (exact B&B)
C05	2	P_4, 5-gate XOR	Optimal Case (a) elimination, $D = 5$
C06	2	P_4, 6-gate XOR	Case (b) merge elimination, $D = 5$
C07	2	C_5, 8-gate XOR	Full telescoping elimination, $D = 8$
C08	2	K_4, 9-gate XOR	Full telescoping elimination, $D = 9$
C09	3	P_4, GL(8, 2)	$B_G^2 = I$, exact 9-gate CNOT sim.
C10	3	$C_4 + e$, GL(10, 2)	$B_G^2 = I$, exact 13-gate CNOT sim.
C11	3	K_4, GL(11, 2)	$B_G^2 = I$, exact 16-gate CNOT sim.
C12	3	$G(K_2)$, GL(33, 2)	$B_G^2 = I$, exact 55-gate CNOT sim.
C13	4	P_4, $k = 2$, anc+SWAP	YES ($K = 9$), XOR trans. exact
C14	4	P_4, $k = 1 \to k' = 0$	NO ($K = 7 < 8$), parity rounded
C15	4	$C_4 + e$, $k = 3$, SWAP	YES ($K = 13$), XOR trans. exact
C16	4	$G(K_2)$, $k = 5 \to k' = 4$	NO ($K = 53 < 54$), full pipeline

AI-Assisted Formalization Disclosure. This verification artifact was authored by Lluis Eriksson using Google Antigravity for Lean 4 proof engineering, zero-float $\mathbb{F}_2$ Python verification scripting, test design, and manuscript preparation. Every verified claim is scoped strictly to the 18 Lean 4.35.0 theorems and 16 exact $\mathbb{F}_2$ certificates (`C01`–`C16`).

References

[1] A. Pavlogiannis, *On the Hardness of Optimal CNOT Circuits*, arXiv:2611.34138v1, November 2026.

[2] J. Boyar, P. Matthews, and R. Peralta, *Logic Minimization Techniques with Applications to Cryptology*, Journal of Cryptology **26** (2013), 280–312.