Parity in majority-inverter graphs: exact values for n ≤ 5, an interval for n = 6, and a conjecture

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

3 November 2026 · doi:10.5281/zenodo.23902529

Abstract

We study `opt(PAR_n)`, the minimum number of majority gates in a majority-inverter graph (MIG) that computes the parity of `n` bits, with inverters and the constant free. The upper bound `opt(PAR_n) ≤ 3⌊n/2⌋` is a known construction; we give a complete proof. Two elementary arguments give `opt(PAR_n) ≥ max(opt(PAR_{n−1}), n)` for `n ≥ 2`. We determine `opt(PAR_n) = 0, 3, 3, 6, 6` for `n = 1, …, 5`: by exhaustive enumeration for `n ≤ 3`, and for `n = 4, 5` by unsatisfiability of the exact-synthesis formulas at every positive size below 6, each with a DRAT proof checked by `drat-trim` and with no symmetry breaking. The values up to `n = 4` were already known from exact synthesis; for them, we provide independent checks: exhaustive enumeration for `n ≤ 3` and checked DRAT certificates for `n = 4`. For `n = 6` the proved interval is `6 ≤ opt(PAR_6) ≤ 9`. A solver run without a proof also excludes six gates, and `7 ≤ opt(PAR_6) ≤ 9` rests on that run. All values fit `opt(PAR_n) = 3⌊n/2⌋`, which we state as a conjecture. Restricting each gate to at most one complemented input, as Soeken et al. and the exact-synthesis tool `percy` do, loses nothing; the observation is theirs, and we write out its proof. `percy` returns the same values for `n = 4, 5`. Finally we record a weighted approach suggested by A. S. Kulikov, which would settle the conjecture for odd `n` only, and the point where the obvious way of proving it stops.

1. Model and conventions

A gate computes `⟨a b c⟩ = ab ∨ ac ∨ bc`. Inverters are free and sit on edges as polarities. The constant-0 node is part of the model and free. The three operands of a gate are distinct nodes. The circuit is a DAG with one output, whose polarity is free. The cost is the number of gates, and `opt(f)` is the minimum cost of an MIG for `f`; since the output polarity is free, `opt(f) = opt(¬f)`. `PAR_n = x_1 ⊕ … ⊕ x_n`.

A gate with the constant among its operands computes an AND or an OR of two literals; we call it binary, and the other gates full. A circuit of binary gates only is an and-inverter graph (AIG). Allowing a repeated operand node would not lower any value below: such a gate computes a constant or one of its operands and can be removed.

2. Results

`n`	`opt(PAR_n)`	lower bound and its status
1	0	trivial
2	3	exhaustive enumeration, two independent routes
3	3	exhaustive enumeration, two independent routes; also Proposition 4
4	6	DRAT proofs, checked, for every size from 1 to 5
5	6	DRAT proofs, checked, for every size from 1 to 5; also Lemma 2 with `n = 4`
6	6 to 9	6 proved (Lemma 2); 7 only by a solver run without a proof
7	7 to 9	7 proved (Proposition 4)
`n ≥ 2`	`≤ 3⌊n/2⌋`	`≥ max(opt(PAR_{n−1}), n)`, proved

The upper bounds are Proposition 1 in every row.

3. The upper bound

Proposition 1. $\mathrm{opt}(\mathrm{PAR}_n) \le 3\lfloor n/2 \rfloor$ for every $n \ge 1$.

Proof. Let a, b, c be three distinct nodes and set $n_1 = \langle a\ b\ c \rangle$, $n_2 = \langle a\ b\ \bar{c} \rangle$, $n_3 = \langle c\ \bar{n}_1\ n_2 \rangle$. We claim $n_3 = a \oplus b \oplus c$. If $a = b$, both n_1 and n_2 equal a, and $n_3 = \langle c\ \bar{a}\ a \rangle = c = a \oplus b \oplus c$. If $a \neq b$, then $n_1 = c$ and $n_2 = \bar{c}$, so $n_3 = \langle c\ \bar{c}\ \bar{c} \rangle = \bar{c} = a \oplus b \oplus c$. The operands of each of the three gates are distinct nodes, so this is a valid three-gate block XOR3. Put $P_1 = x_1$ and $P_{2j+1} = \mathrm{XOR3}(P_{2j-1}, x_{2j}, x_{2j+1})$ for $j \ge 1$; then $P_{2m+1} = \mathrm{PAR}_{2m+1}$ with $3m$ gates. For even $n = 2m$, apply the block once more to P_{2m-1}, x_{2m} and the constant 0, which gives PAR_{2m} with $3(m-1) + 3 = 3m$ gates. In both cases the count is $3\lfloor n/2 \rfloor$. ■

With c the constant, the block becomes $n_1 = a \wedge b$, $n_2 = a \vee b$, $n_3 = \neg n_1 \wedge n_2$: three binary gates, the usual AIG gadget for XOR2. The construction was also simulated on all 2^n input rows for every n from 2 to 13.

4. Elementary lower bounds

Lemma 2 (restriction). $\mathrm{opt}(\mathrm{PAR}_{n-1}) \le \mathrm{opt}(\mathrm{PAR}_n)$ for $n \ge 2$.

Proof. Substitute the constant 0 for x_n in an optimal MIG for PAR_n. Polarities carry over and the constant is a node of the model, so each gate is still a majority gate, unless two of its operands have become the same node; such a gate computes a constant or one of its operands. Going through the gates in topological order, we remove each such gate and let its readers read the node it computes, which may in turn give a later gate a repeated operand. The result computes PAR_{n-1} with at most as many gates. ■

Lemma 3 (fan-out). Let $n \ge 2$ and let C be an MIG of minimum size for PAR_n. Every input is an operand of at least two gates, and every gate other than the output is an operand of at least one gate.

Proof. A non-output gate that feeds nothing can be deleted. Every input feeds some gate, because PAR_n depends on every input. Suppose the input x feeds exactly one gate $h = \langle x^{\alpha}, u, v \rangle$, where u and v are the other two operands with their polarities. Since u and v come before h in the DAG and every path from x passes through h, neither u nor v depends on x. Fix the other $n-1$ inputs to some assignment a. If $u(a) = v(a)$, then h takes that value whatever x is, so the output does not change when x changes, but PAR_n does. Hence $u(a) \neq v(a)$ for every a, that is, $v = \neg u$ as functions, and then $h = x^{\alpha}$. The output is not h, because PAR_n is not a literal when $n \ge 2$. Let every reader of h read x instead, with the polarity adjusted, and delete h. No reader of h already read x, because x fed only h, so operands stay distinct, and C loses a gate: a contradiction. ■

Proposition 4. For $n \ge 2$, every MIG for PAR_n with F full and B binary gates has $2F + B \ge 2n - 1$, and $\mathrm{opt}(\mathrm{PAR}_n) \ge n$.

This bound is elementary, and we include it for completeness.

Proof. Take a circuit of minimum size, with $k = F + B$ gates. Its $3k$ operand slots are filled by inputs, by gates and by the constant. By Lemma 3 the inputs fill at least $2n$ slots and the gates at least $k - 1$, so $3k \ge 2n + k - 1$, which gives $k \ge n - 1/2$ and so $k \ge n$. For the weighted statement, note that a binary gate reads the constant exactly once and a full gate never does, so the constant fills exactly B slots and $3F + 2B \ge 2n + F + B - 1$, which is $2F + B \ge 2n - 1$. This holds for a circuit that minimises $2F + B$ instead of k, because the two deletions in the proof of Lemma 3 remove one gate and do not change the type of any other gate; and the minimum of $2F + B$ bounds every circuit. ■

The gate-count bound in Proposition 4 is tight at $n = 3$. It gives $\mathrm{opt}(\mathrm{PAR}_7) \ge 7$, one more than what the exact values give through restriction. Beyond that, what is proved grows like n, and the construction like $3n/2$.

5. Exact values

n = 2, 3. A depth-first search over gate sequences finds no circuit with two gates for PAR_2 or PAR_3, and exactly 192 labelled optimal circuits of three gates for each. Here a labelled circuit is a sequence of gates, each choosing a set of three earlier nodes and one of eight polarity patterns, every gate but the last read by a later gate, the last one being the output. An all-solutions SAT run on the encoder described below gives the same counts. Under the symmetries of the problem (reordering gates, self-duality $\langle \bar{a}\ \bar{b}\ \bar{c}\rangle = \neg\langle a\ b\ c\rangle$, permuting and negating inputs, output polarity), the optimal circuits fall into 4 classes for PAR_2 and into 2 for PAR_3. The two classes of PAR_3, with 96 labelled circuits each, are A3 = $\langle\langle x_1\ x_2\ \bar{x}_3\rangle\ \langle x_1\ x_3\ \bar{x}_2\rangle\ \bar{x}_1\rangle$ and B3 = $\langle\langle\langle x_1\ x_2\ \bar{x}_3\rangle\ x_3\ \bar{x}_1\rangle\ \langle x_1\ x_2\ \bar{x}_3\rangle\ \bar{x}_2\rangle$.

Theorem 5. opt(PAR_4) = opt(PAR_5) = 6.

Proof. The upper bounds are Proposition 1. Let E_n(k) be the CNF formula "an MIG with exactly k gates computes PAR_n". It has one selector variable per gate and per choice of three distinct earlier nodes with a polarity pattern, exactly one selector per gate, one value variable per gate and truth-table row tied to the selected majority, and the output at gate k with a free polarity. It also requires that no two gates select the same signed triple and that every gate before the last feeds a later gate. Both requirements hold in every circuit of minimum size m once its output is placed last, so such a circuit is a model of E_n(m). For n = 4 and for n = 5, E_n(k) is unsatisfiable for each k from 1 to 5, and each refutation is a DRAT proof checked by drat-trim. A circuit with no gates outputs a constant or a literal, and neither PAR_4 nor PAR_5 is one. So no circuit with at most 5 gates exists, and opt ≥ 6. ■

The encoding needs the exact size m, which is why each k ≤ 5 is excluded separately. No symmetry breaking is used. For n = 5 the bound has a second route, Lemma 2 applied to n = 4. The hardest formulas are those at k = 5. For PAR_4 the formula has 6,094,196 bytes; kissat 4.0.4 refuted it in 193.7 s with a proof of 221,498,877 bytes, checked in 222.1 s. For PAR_5 the formula has 14,584,525 bytes, refuted in 318.5 s with a proof of 416,229,826 bytes, checked in 392.9 s (wall-clock times on a laptop running other work). At k = 6 both formulas are satisfiable, and the decoded six-gate circuits compute the right function on all 16 and all 32 rows. As a negative control of the checker, each proof for k = 5 is rejected against the satisfiable formula for k = 6.

A second tool. The exact MIG synthesis of percy (EPFL) also returns six gates for PAR_4 and for PAR_5. We read the returned chains and evaluated them ourselves, because percy's own simulation misreads chains that use the constant. Both tools do exact synthesis by SAT, so they share the method; the formulation, the encoder and the solver are independent. percy searches only circuits in which each gate has at most one complemented operand. This is the normal form of Soeken et al. [5, Sec. III-B], who note that it affects neither universality nor size, since inverters can be moved toward the outputs; it is the inverter-propagation axiom Ω.I of the MIG algebra of Amarù, Gaillardon and De Micheli [13] (also in [2]) applied gate by gate. We write the proof out as Lemma 6, because it carries the minimum of percy over to our model, if its search over that space is complete, which we did not audit.

Lemma 6 (one complemented operand; [5, 13]). Every MIG with k gates can be turned into an MIG with k gates for the same function in which every gate has at most one complemented operand.

Proof. We go through the gates in topological order and replace each node v by a node v∗ and a bit s(v) with v = v∗ ⊕ s(v); for inputs and the constant, v∗ = v and s(v) = 0. Let g = ⟨u_1^{p_1}, u_2^{p_2}, u_3^{p_3}⟩, where u^p is u complemented when p = 1. Then u_i^{p_i} = (u_i∗)^{q_i} with q_i = p_i ⊕ s(u_i). If at most one q_i is 1, put g∗ = ⟨(u_i∗)^{q_i}⟩ and s(g) = 0. Otherwise at most one 1 − q_i is 1, and self-duality gives g = ¬⟨(u_i∗)^{1−q_i}⟩; put g∗ = ⟨(u_i∗)^{1−q_i}⟩ and s(g) = 1. The operand nodes of g∗ are those of g, so they stay distinct, and the number of gates does not change. If the circuit outputs o ⊕ p_out, it now outputs o∗ ⊕ p_out ⊕ s(o): the new output polarity is p_out ⊕ s(o), which is free. ■

n = 6. Lemma 2 with Theorem 5, and also Proposition 4, give opt(PAR_6) ≥ 6. The formula E_6(6), with 5,825 variables and 4,149,291 clauses, was found unsatisfiable by kissat 4.0.4 after 239,164 s of CPU time (66.4 hours), without symmetry breaking and without a proof: the run that was writing a proof was lost. So opt(PAR_6) ≥ 7 is a solver result, not a certificate. At the other end, E_6(9) is satisfiable; the decoded nine-gate circuit, which has seven full and two binary gates, computes PAR_6 on all 64 rows. The runs on E_6(7) and E_6(8) were stopped after 13 days and 3 hours of process time each, without a verdict. Unsatisfiability at k = 6 says nothing about

k = 7 or k = 8, so the value 9 needs both of those formulas to be unsatisfiable. A run of percy on PAR_6 had not returned when this note was written. In the AIG basis, for some functions of three inputs and three outputs, Averkov et al. write that "proving that there is no smaller circuit is already a difficult task for the state-of-the-art SAT solvers" [12].

6. A weighted measure, and where the obvious proof stops

The direction in this section came out of an exchange with A. S. Kulikov, who suggested an induction over the variables that gives different weights to full gates and to the binary gates they become after constants are substituted, aiming at a bound around 3n/2.

Let μ(C) = 2F + B. The construction of Proposition 1 has μ = 3(n − 1) for odd n. We conjecture

(H1) μ_min(PAR_n) = 3(n − 1), the minimum taken over all MIGs for PAR_n.

Every gate adds at most 2 to μ, so H1 would give opt(PAR_n) ≥ ⌈3(n − 1)/2⌉. For n = 2m + 1 this is 3m, which is Conjecture C1 below; for n = 2m it is 3m − 1, one less than the construction. So H1 settles C1 for odd n only.

Why these weights. Write μ_α = αF + B and ask for a step that fixes one variable and lowers μ_α by δ. The step must not overshoot the XOR3 chain and must return the gate bound the chain attains at odd n, which forces δ = 3α/2 with α ≥ 1. It must hold on AIGs, where parity of n bits needs 3(n − 1) gates (Schnorr [7], as quoted in [9]), which forces δ ≤ 3 and so α ≤ 2. And it must hold on the class A3 of optimal circuits for PAR_3, where every restriction leaves a circuit for PAR_2, which has at least three gates; so the drop is at most 3α − 3, which forces α ≥ 2. Hence α = 2 and δ = 3.

Status of H1. It holds at n = 2 and n = 3, by exhaustive enumeration: no circuit for PAR_3 has μ ≤ 5. It holds at n = 4. There Theorem 5 excludes 5 gates or fewer, and μ = k + F ≥ k excludes 9 gates or more, so it remains to exclude μ ≤ 8 with 6, 7 or 8 gates. The formulas below keep the two requirements of E_n(k) from Section 5, and they lose nothing here: take a circuit for PAR_4 with μ ≤ 8 and the fewest gates. Deleting a gate that feeds nothing, or merging two gates with the same signed triple and removing any gate left with a repeated operand as in Lemma 2, gives a circuit for PAR_4 with fewer gates and no new full gate, so with μ ≤ 8; hence the chosen circuit has neither, and it is a model of the formula for its size. With 6 gates and at most 2 full ones the formula is unsatisfiable, with a DRAT proof of 9,989,488,865 bytes checked in 14,852 s. With 7 gates and at most 1 full one it is unsatisfiable, certified in 80 parts: the formula is split by the choice of the first gate, the 80 choices form a clause of the formula itself, so the split needs no separate proof, and each of the 80 parts has its own DRAT proof, checked. With 8 gates, μ ≤ 8 means no full gate, that is, an AIG with 8 gates, which Schnorr's bound excludes. The bound is attained: 61,440 of the labelled optimal circuits of PAR_4 have 3 full gates and μ = 9. At n = 5, every optimal circuit has μ = 12 = 3(n − 1), but that does not prove H1, because μ_min ranges over circuits of every size.

Where one-variable elimination stops. Fixing a variable turns each full gate that reads it into a binary gate and removes each binary gate that reads it. On class B3 of optimal circuits for PAR_3, each of the six restrictions x_j := c, simplified syntactically, leaves one full and two binary gates in all 96 circuits of the class: a drop of 2 in μ, not 3. The restricted circuit computes XOR2 with μ = 4, while μ_min(PAR_2) = 3; only resynthesis recovers the missing unit. So a proof of H1 by one-variable elimination has to use more than syntactic simplification.

7. Conjecture C1 and its evidence

Conjecture C1. opt(PAR_n) = 3⌊n/2⌋ for every n ≥ 1.

C1 holds for n ≤ 5 and is not refuted at n = 6. By Lemma 2 and Proposition 1, opt(PAR_{2m}) ≤ opt(PAR_{2m+1}) ≤ 3m, so C1 at 2m implies C1 at 2m + 1: the conjecture is equivalent to its even case, opt(PAR_{2m}) = 3m, which in particular makes parity of 2m bits cost as much as parity of 2m + 1 bits.

The structure of optimal circuits supports a stronger form. Conjecture C2: the gates of every optimal MIG for PAR_n split into disjoint blocks of three gates such that each block, read as a circuit whose inputs are the nodes outside the block that it reads, is an optimal circuit for XOR3 (classes A3, B3) or for XOR2 (the four classes of PAR_2), up to the polarities on its edges; sharing occurs only inside a block, as in B3; and each variable, and each

block output other than the circuit output, is read by the gates of exactly one block, so the blocks form a tree. There is one XOR2 block for even n and none for odd n. For n = 4 and n = 5 we enumerated all labelled optimal circuits with an all-solutions SAT run, split by the choice of the first gate and with each model simulated again; these enumerations carry no proofs. For n = 4 there are 368,640 of them, in 28 classes, each with 3 or 4 full gates (61,440 and 307,200), as C2 predicts. For n = 5 there are again 368,640, in 8 classes, all with 6 full gates, as C2 predicts. In both censuses, fixing the best pair of variables removes exactly 3 gates in every circuit, which is the two-variable step that would give C1 by induction if it held for every n. The nine-gate circuit found for PAR_6 is also a tree of blocks: one XOR2 and two XOR3.

8. Related work

The three-gate XOR3 block is the sum output of the three-gate full adder in Example 1 of Soeken et al. [5], and Table 4.5 of Aalam's thesis [1] gives the counts 3⌈n/2⌉ for even n and 3(⌈n/2⌉ − 1) for odd n, which equal 3⌊n/2⌋; Proposition 1 is not new. The values for n ≤ 4 are not new either: the mig_npn database of the mockturtle library, described in its source as pre-computed size-optimum MIGs with up to four variables, gives 3, 3 and 6 for the classes of XOR2, XOR3 and four-variable parity; Fig. 4 of [5] shows a size-optimum three-gate MIG for x_1 ⊕ x_2 ⊕ x_3; and Soeken et al. [4] report the distribution of optimal sizes over the 222 NPN classes of four-variable functions, which our encoder reproduces. What Section 5 adds for n ≤ 4 is an independent check of each lower bound: exhaustive enumeration for n = 2, 3, and checked DRAT certificates for n = 4. For n = 5 we found no earlier exact value; the databases stop at four variables. Chu et al. [2] write that the optimal MIG of 4-input parity, obtained by exact synthesis, "requires nine MAJ operations". The expression they display has nine MAJ occurrences but admits a six-gate DAG implementation, by sharing subexpressions up to complementation; nine is therefore not a lower bound in our model. The exact synthesis tool percy is from the same group as [4, 5].

For parity over AIGs, and more generally over U_2, the exact complexity 3(n − 1) is Schnorr's [7] (we cite it as quoted in [9]). Carmosino, Dang and Jackman [10] describe all optimal AIGs for parity as trees of n − 1 XOR2 gadgets, which is the model for C2 here; Kombarov [11] studies the model with zero-cost inverters, and his survey [3] covers other bases. Podolskaya [8] gives the exact value ⌊(n + 1)/2⌋ for parity in an infinite basis: all functions whose set of ones is an antichain of the cube. We found no lower bound for parity over {MAJ3, ¬} beyond the elementary ones above.

What this note does not claim

- C1, H1 for n ≥ 5, and C2 are conjectures. The censuses for n = 4, 5 confirm predictions of C2; they do not prove it, and they carry no proofs.
- opt(PAR_6) ≥ 7 is a solver verdict without a proof. What is proved is 6 ≤ opt(PAR_6) ≤ 9; the interval from 7 to 9 rests on that verdict.
- Agreement with percy at n = 4, 5 is agreement between two exact-synthesis tools, not two independent proofs. Lemma 6 shows that percy's space is large enough; we did not audit the rest of its encoding.
- The construction and the values for n ≤ 4 are known (Section 8), and so is the normal form of Lemma 6 [5]. Proposition 4 is elementary and may be folklore.
- No proof of C1 for odd n is claimed: Section 6 only shows which weights a one-variable induction would need, and where syntactic simplification falls short.

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 DRAT proofs checked by drat-trim, the exhaustive enumerations and the proofs written out in the text, not on the models.

Data availability

The encoder, the CNF hashes, the DRAT records and the checker outputs for PAR_4 and PAR_5 are in the dataset [6]. The data behind every other number are in the dataset [D]: the logs of the enumerations, of the percy runs

on PAR_4 and PAR_5 and of the solver runs on PAR_6; the drat-trim outputs of the cells of Section 6, one per part for the split cell; the branch files of the censuses; the optimal size of each four-variable NPN class, with its comparison with the counts of [4]; the scripts that produced them, except the scripts that launched the long runs, whose commands and exit codes are in the logs, and our decoder of the third-party mig_npn file, of which only the output is included (N16); and the output of checks run from those scripts when the dataset was assembled. The table below names, for each number, the file in [D] or [6] that holds it. Formulas and proofs are not stored because of their size: the formulas regenerate byte for byte from the encoder, whose sha256 values are pinned, and the proofs regenerate by running kissat 4.0.4 again; the checker outputs are kept. Third-party sources are not included.

Provenance by number

id	numbers as stated	artefact	status
N01	n = 2 to 13	[D] logs/h_constr.txt, checks/h_constr.out	proof and simulation
N02	192 and 192 labelled optima, 0 with two gates; 4 and 2 classes; A3 and B3, 96 each	[D] logs/gate_contagens.txt, logs/d_classes.txt, checks/gate_contagens.out, checks/d_classes.out	exhaustive enumeration
N03	k from 1 to 5 refuted with checked proofs, n = 4 and n = 5; witnesses on 16 and 32 rows	[6] logs/par4_drat.log, logs/par5_drat.log	DRAT, checked
N04	6,094,196; 193.7; 221,498,877; 222.1; 14,584,525; 318.5; 416,229,826; 392.9	[6] same two logs, lines [par4 k=5] and [par5 k=5]	measured
N05	percy: 6 gates for PAR_4 and for PAR_5	[D] logs/percy_par4_par5.log; chains evaluated in checks/avalia_mig_percy.out	second tool
N06	5,825 and 4,149,291; 239,164 s, 66.4 h	[D] logs/par6_cnf_regen.txt, logs/par6_k6_unsat.log	solver only
N07	9 gates, 7 full and 2 binary, 64 rows	[D] logs/par6_k9_model.txt, checks/decodifica_par6_k9.out	simulated witness
N08	13 days and 3 hours each, no verdict	[D] logs/par6_k7k8_stopped.txt	stopped
N09	no circuit for PAR_3 with $\mu \leq 5$	[D] logs/mu3_enum.txt	exhaustive enumeration
N10	9,989,488,865 bytes, 14,852 s	[D] logs/h1_n4_k6_f2.log, logs/h1_n4_k6_f2.drattrim.txt, checks/confere_celula_h1_n4_k6_f2.out	DRAT, checked
N11	80 parts, 80 checked proofs	[D] logs/cubos_k7/, data/cubos_h1_n4_k7_f1.csv, checks/cubos_gate.out, checks/conta_cubos.out	DRAT, checked
N12	61,440 with 3 full gates	[D] data/ramos_n4_k6/, summed in checks/census_n4.out	enumeration without proofs
N13	class B3: 96 circuits, 1 full and 2 binary gates after each of 6 restrictions	[D] logs/k_pesos.txt, checks/k_pesos.out	exhaustive
N14	n = 4: 368,640, 28 classes, 61,440 and 307,200; n = 5: 368,640, 8 classes, 6 full	[D] data/ramos_n4_k6/, data/ramos_n5_k6/, summed in checks/census_n4.out and checks/census_n5.out	enumeration without proofs
N15	drop of exactly 3 in every circuit, n = 4 and n = 5	same	enumeration without proofs

id	numbers as stated	artefact	status
N16	Aalam, Table 4.5, rows for even and odd n; Chu, "nine MAJ", its expression rebuilt as a six-gate DAG; mig_npn, 3, 3 and 6 for XOR2, XOR3 and parity, 222 classes	Aalam [1]; Chu et al. [2], with the six-gate DAG checked in [6] `checks/chu_colapso.out`; the `mig_npn` database of mockturtle, decoded in [D] `logs/mockturtle_redecodifica.txt`	literature
N17	formulas of third parties (Schnorr as quoted by Carmosino et al., Podolskaya and the definition of its basis, the normal form of Soeken et al., Ω.I of Amarù et al. through its DOI) and the identifiers in the references	the cited works: [5] for the normal form, [7] as quoted in [9], [8], [2] and [13] for Ω.I; the identifiers as printed by each source	literature
N18	distribution of optimal sizes over the 222 NPN classes of four-variable functions, sizes 0 to 7, 65,536 functions	[D] `data/npn4_mig.csv`, `data/prior_art_mig_n4.csv`, `checks/npn4_soeken.out`	comparison with published counts

References

1. U. Aalam. *Inversion optimization in majority inverter graph with minority operations: a novel approach.* MS (Research) thesis, IIT Indore, 2020. https://dspace.iiti.ac.in/handle/123456789/2696
2. Z. Chu, M. Soeken, Y. Xia, L. Wang, G. De Micheli. Structural rewriting in XOR-majority graphs. *ASP-DAC 2019*, pp. 663–668, doi:10.1145/3287624.3287671.
3. Yu. A. Kombarov. Complexity and structure of circuits for parity functions. *J. Math. Sci.* 233:95–99, 2018, doi:10.1007/s10958-018-3926-6. Translated from *Fundam. Prikl. Mat.* 20(6):147–153, 2015.
4. M. Soeken, L. G. Amarù, P.-E. Gaillardon, G. De Micheli. Optimizing majority-inverter graphs with functional hashing. *DATE 2016*, pp. 1030–1035.
5. M. Soeken, L. G. Amarù, P.-E. Gaillardon, G. De Micheli. Exact synthesis of majority-inverter graphs and its applications. *IEEE TCAD* 36(11):1842–1855, 2017.
6. L. A. Busnello. Exact majority-inverter graph size of SUM_5 and PAR_5: encoders, CNF hashes, DRAT records and checks. Dataset, *Zenodo*, 2026, doi:10.5281/zenodo.23382607.
7. C. P. Schnorr. Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen. *Computing* 13:155–171,
   1974. doi:10.1007/BF02246615.
8. O. Podolskaya. On circuit complexity of parity and majority functions in antichain basis. arXiv:1410.2456, 2014.
9. M. Carmosino, N. Dang, T. Jackman. Constructive separations from gate elimination. arXiv:2604.23958, 2026.
10. M. Carmosino, N. Dang, T. Jackman. Simple circuit extensions for XOR in PTIME. *STACS 2026*, LIPIcs 364, 23:1–23:20, 2026.
11. Yu. A. Kombarov. On minimal circuits for linear functions over some bases. *Discrete Math. Appl.* 23(1):39–51, 2013, doi:10.1515/dma-2013-002. Russian original: *Diskretnaya Matematika* 25(1):33–44, 2013.
12. D. Averkov, T. Belova, G. Emdin, M. Goncharov, V. Krivogornitsyna, A. S. Kulikov et al. Cirbo: a new tool for Boolean circuit analysis and synthesis. *AAAI 2025*.
13. L. G. Amarù, P.-E. Gaillardon, G. De Micheli. Majority-inverter graph: a novel data-structure and algorithms for efficient logic optimization. *DAC 2014*, pp. 1–6, doi:10.1145/2593069.2593158.

[D] L. A. Busnello. Parity in majority-inverter graphs: enumerations, censuses, cell certificates and solver logs. Dataset, *Zenodo*, 2026, doi:10.5281/zenodo.23900926.