Base laws for the formula–circuit gap at four inputs: AIG, MIG and XAG compared class by class

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

3 November 2026 · doi:10.5281/zenodo.23913155

Abstract

For a Boolean function f and a gate basis, let opt(f) be the minimum number of gates of a circuit for f and tree(f) the minimum number of gates of a formula (a circuit in which every gate feeds at most one gate), and call gap(f) = tree(f) − opt(f) the price of forbidding reuse. We compare this price in three bases with free inversion, AND-inverter graphs (AIG), majority-inverter graphs (MIG) and XOR-AND graphs (XAG), on each of the 222 NPN classes of four-input functions. The per-basis values are not new, though not all are published class by class: the size distributions in all three bases have been published before, per-class values only for some of the six columns, and we deposited the per-class values of all three bases earlier as a dataset; we also assume, without having checked, that the two XAG sizes of every class are already in Knuth's book. What we report is how the three gaps relate on the same class. The maximum gap is 6 in AIG, 3 in MIG and 1 in XAG. Adding gates to the basis does not always lower the gap. Every AIG gate is also an MIG gate and an XAG gate; no class pays a gap in XAG without paying in AIG, but 11 classes pay a gap in MIG and in neither of the other two bases. MIG and XAG are not comparable gate for gate, and their gaps are not ordered either: 3 classes pay in XAG and not in MIG. Only one class pays in all three bases, the function E_2 that is true when exactly two of the four inputs are true. We add two smaller results, each stated with its scope: in the optimal AIG cascade for parity the operand-cone intersection at the output gate has $3(n-2)$ gates for $n = 3$ to 12, so its agreement with the gap at $n = 3$ and $n = 4$ does not continue; and over XAG at three inputs every essential variable can be read exactly once by some minimum formula, a fact that fails at four inputs. Every number written in digits from the Abstract to Section 5 is checked by a script against the project's artefacts, and the paper lists, number by number, where each one comes from. The script checks finite data: where a statement is made for every n, it checks the instances it has, and the general case rests on a cited theorem or on algebra written out in the text. Statements without a number are checked only where the provenance table says so.

Introduction

Setting. A function of n inputs is given by its truth table. In all three bases, inputs, constants and inverters are free, and the cost of a circuit is its number of gates: two-input AND gates in AIG; three-input majority gates in MIG, where AND and OR are majorities with a constant input; two-input AND and XOR gates in XAG. With free inversion, every two-input Boolean operation that depends on both inputs is an AND or an XOR with possibly negated inputs and output, and the others are constants or literals, which cost nothing, so XAG size is circuit size over the full binary basis B_2 [Averkov et al. 2025], and AIG size is circuit size over the basis U_2 of the classical literature. opt(f) is the minimum size of a circuit, tree(f) the minimum size of a formula, measured in gates, and gap(f) = tree(f) − opt(f) ≥ 0. All three quantities are invariant under permuting inputs, negating inputs and negating the output (NPN equivalence), so it suffices to work with the 222 NPN classes of four-input functions. A class is named by the smallest truth table in it, written in hexadecimal with input x_0 as the least significant variable.

What is known and what is not. The sizes in each basis are, for the most part, not new, but most are published as distributions, not class by class. The optimal MIG sizes and the minimum MIG formula sizes of the four-input classes were published by Soeken, Amarù, Gaillardon and De Micheli [Soeken et al. 2016], as two marginal

distributions. Over XAG, opt is the circuit complexity C(f) over B_2 and tree is the formula length L(f), and L(f) is published for every class of up to five inputs by Lee, Jiang, Mishchenko and Brayton [Lee et al. 2019]. We have not consulted §7.1.2 of Knuth's book [Knuth 2011]. Until someone does, we work under the hypothesis that it gives both C(f) and L(f) for each four-input class, so that the XAG column of Section 1 is probably already known (see Related work). The distribution of the AIG formula size over all 65,536 four-input functions was published by Madsen, van de Pol and Henriksen [Madsen et al. 2023], as part of a compiler. Our earlier census [Busnello 2026a] gave the AIG and XAG gap distributions at four inputs and deferred the MIG gap; the per-class values in all three bases were deposited as a dataset [Busnello 2026b]. What we did not find published, in the searches described in the Related work section, is the pairing of the three gaps on the same class. That pairing is the subject of this paper. If the hypothesis about Knuth's book holds, what this paper adds is at most the crossing of the XAG column with AIG and MIG in Section 1, and Sections 2 to 5; whether the book also covers part of them has not been checked (see Related work).

Order of the paper. Sections are ordered by the strength of their support, not by when they were found. Section 1 gives the three gap distributions and their crossing, which rest on the most checks. Section 2 shows that adding gates to the basis does not always lower the gap. Section 3 isolates E_2. Section 4 is about parity at every n, Section 5 about how often a minimum XAG formula must read a variable, and what that does and does not account for.

1. The three gaps on the same class

Over the 222 NPN classes of four-input functions, the three gaps are distributed as follows.

basis	distribution of gap (value: classes)	maximum	classes at the maximum
AIG	{0: 93, 1: 57, 2: 40, 3: 13, 4: 14, 5: 2, 6: 3}	6	`0x1668`, `0x16e9`, `0x6996`
MIG	{0: 168, 1: 50, 2: 3, 3: 1}	3	`0x6996`
XAG	{0: 218, 1: 4}	1	`0x01e9`, `0x0661`, `0x1668`, `0x167e`

The gap of one class in two bases is a difference of two counts of different gates, so comparing gap values across bases compares quantities without a common unit. Whether a class pays a gap at all does not depend on the unit, and the joint pattern of zero and positive gaps is the following.

AIG gap > 0	MIG gap > 0	XAG gap > 0	classes
no	no	no	82
yes	no	no	83
yes	yes	no	42
no	yes	no	11
yes	no	yes	3
yes	yes	yes	1

Two of the eight patterns do not occur: no class pays in XAG without paying in AIG. Of the 129 classes that pay in AIG, 83 pay in AIG only. Comparing gate counts directly, with the caveat about units, the AIG gap exceeds the XAG gap in 127 classes and is never smaller; the AIG gap exceeds the MIG gap in 113 classes and is smaller in 11; the MIG gap exceeds the XAG gap in 54 classes and is smaller in 3.

How each column is supported. The columns do not all rest on the same kind of evidence, and the crossing is only as good as its weakest column.

- *AIG formula size.* Our value for each class comes from an exhaustive dynamic program over formulas. It has three further supports. (i) For each of the 220 classes with at least one gate, that is, all but the constant class `0x0000` and the literal class `0x00ff`, we stored one explicit formula with exactly tree(f) gates,

1,782 gates in all, and these witnesses are re-simulated on all 16 input rows, with the fan-out of every non-output gate recounted, by a checker that imports none of our encoders. (ii) Expanded by the sizes of the NPN orbits, our 222 values give the 16 counts that Madsen, van de Pol and Henriksen publish for the 65,536 functions [Madsen et al. 2023]. Their program and ours share no code, but they implement the same dynamic program, so the agreement guards against the defects the two do not have in common, and it compares histograms, not classes. (iii) The witnesses give tree(c) $\leq$ v(c) for every class c, with v the value in our table; together with the published histogram and the NPN invariance of tree, a descending induction on the size shows that every inequality is an equality, so, given that histogram, the column is determined without any unsatisfiability claim. The histogram rests on the same algorithm as ours, as (ii) says, so (iii) is not independent of (ii): it removes the solver from the argument, not the shared algorithm. A SAT census in formula mode agrees where it finished; it did not finish on the 17 classes with tree $\geq$ 12, and on those classes (iii) is what keeps the solver out of the argument.

- *AIG circuit size.* Our values agree class by class with the catalogue of Krinkin [Krinkin 2026b]. Except for the few classes listed in [Busnello 2026c, §6], their lower-bound side is solver output without retained proofs.
- *MIG.* Our dynamic program agrees with a SAT census in formula mode on 220 of the 222 classes; the two that the SAT runs did not finish, `0x1669` and `0x6996`, are decided by the program alone. Both marginal distributions reproduce the tables of Soeken et al. [Soeken et al. 2016] bin by bin, by class and by function, in 18 bins, and the 3 statements that paper makes about those two classes, which it singles out by name, also agree: 21 comparisons in all. It does not give the formula size of either class. Its bin of formula size 9 holds 2 classes and 34 functions, and the split of those 34 functions into 32 and 2 is ours, from the dynamic program. The two MIG distributions are therefore theirs; the value of each class, the formula size of those two classes included, is ours, since a distribution does not say which class holds which value, and the crossing is not in their paper.
- *XAG.* Our census agrees with the published data noted in the Introduction; its UNSAT side is gated but not proof-certified [Busnello 2026a].

2. Adding gates does not always lower the gap

Every AIG gate is also an MIG gate, a majority with a constant input, and an XAG gate. A natural reading of the table in Section 1 is that adding gates to a basis leaves less room for a gap. For the inclusion of AIG in XAG this holds on every class: no class pays in XAG without paying in AIG (Section 1). For the inclusion of AIG in MIG it fails.

MIG pays where AIG and XAG do not. In 11 classes the gap is 0 in AIG and in XAG and positive in MIG, and in all 11 the triple of gaps (AIG, MIG, XAG) is (0, 1, 0). They are exactly the 11 classes in which the AIG gap is smaller than the MIG gap (Section 1). We have no explanation for them to offer.

XAG pays where MIG does not. In 3 classes the gap is 0 in MIG and positive in XAG: `0x01e9`, `0x0661` and `0x167e`. They are three of the four classes with XAG gap 1. The clearest witness is `0x0661`: its (opt, tree, gap) is (9, 11, 2) in AIG, (5, 5, 0) in MIG and (6, 7, 1) in XAG. These three are not a failure of monotonicity, because MIG and XAG are not comparable gate for gate; together with the classes of the previous paragraph, they show that neither of the two gaps bounds the other.

Together these are 14 classes, of two different kinds. The 11 are failures of the implication «no gap in the smaller basis implies no gap in the larger one» for the one pair, AIG and MIG, where it fails; the 3 are not failures of that implication, since there is no chain AIG $\rightarrow$ MIG $\rightarrow$ XAG, and with the first kind they rule out any order between the MIG gap and the XAG gap. We offer no rule in place of monotonicity. The result is negative and is stated as a count.

3. E_2 and what symmetry does not explain

Let E_2 be the four-input function that is true exactly when two of its inputs are true. Its truth table is `0x1668`, which is already the smallest table in its NPN class. Its (opt, tree, gap) is (9, 15, 6) in AIG, (6, 8, 2) in MIG and (6, 7, 1) in XAG.

E_2 is the only class of the 222 with a positive gap in all three bases (Section 1, last row of the pattern table). In XAG it is one of four classes of gap 1, and the four have the same XAG sizes, (opt, tree) = (6, 7), so XAG itself does not set E_2 apart. A property from outside XAG does: it is the only one of the four that contains a symmetric function.

The 32 symmetric functions of four inputs fall into 10 NPN classes. Over these 10 classes the AIG gap is distributed as {0: 4, 2: 2, 4: 1, 5: 1, 6: 2}, the MIG gap as {0: 5, 1: 2, 2: 2, 3: 1} and the XAG gap as {0: 9, 1: 1}.

Ten classes are too few to support a law beyond this census, and we draw none. The section exists to exhibit E_2 as the only class that pays in all three bases studied, not to derive the gap from symmetry. We did not find a theorem or a census that relates symmetry to the gap between circuits and formulas (see Related work).

4. The sharing term of parity

Krinkin's withdrawn note on the unit gap [Krinkin 2026a, Corollary 6] bounds a sharing term $s = |D(g_1) \cap D(g_2)|$, the number of gates common to the cones of the two operands g_1, g_2 of the output gate of an optimal AIG, by 1. Let r be the number of gates of fan-out at least 2. Take the cascade for the parity $\oplus_n$ of n inputs, with $u = \oplus_{n-1}$ computed by the same cascade, $v = x_n$, $g_1 = u \wedge \neg v$, $g_2 = \neg u \wedge v$ and output $\neg(\neg g_1 \wedge \neg g_2)$. It has $3(n-1)$ gates, which is optimal: the U_2 complexity of parity is exactly $3(n-1)$ [Jukna 2012, Theorem 1.29; Kombarov 2013, Theorem 1], by the gate-elimination method of Schnorr [Schnorr 1974].

The count. In this cascade $s = 3(n-2)$ and $r = n-2$, so $s = 3r$, for every n from 3 to 12, where the cascades were built and simulated on all 2^n rows. The structure of optimal AIGs for parity established by Carmosino, Dang and Jackman [Carmosino et al. 2026, Theorem 37] makes the same counts hold in every optimal AIG of $\oplus_n$. The step is short. By their theorem, every optimal circuit for $\oplus_n$ with free negation splits into $n-1$ blocks of three gates, each computing the XOR of its two input wires, and the blocks contract to a binary tree whose leaves are the inputs. The output gate is the top of the root block, and its two operands g_1 and g_2 are the two bottom gates of that block; both read the block's two input wires, which head disjoint subtrees with a and b leaves, $a + b = n$. So $D(g_1) \cap D(g_2)$ is the union of the two subtrees, which have $3(a-1) + 3(b-1) = 3(n-2)$ gates, three per block, and the gates of fan-out at least 2 are exactly the top gates of the $n-2$ non-root blocks, each read by the two bottom gates of its parent block. The translation of their circuits to AIG and these counts are ours, derived from their theorem, and have not been read by a person.

Against the gap. Khrapchenko's bound [Khrapchenko 1971] gives $\text{tree}(\oplus_n) \geq n^2 - 1$ gates, and the cascade gives $\text{opt}(\oplus_n) \leq 3(n-1)$, so $\text{gap}(\oplus_n) \geq (n-1)(n-2)$. At $n = 3$ and $n = 4$ the sharing term equals the gap, 3 and 6, measured on circuits whose optimality rests on DRAT proofs [Busnello 2026a]. From $n = 5$ on, $s = 3(n-2) < (n-1)(n-2)$, so s is below the gap, with $s/\text{gap} \leq 3/(n-1)$. The equality at three and four inputs is a coincidence of small arity: s and r grow linearly and the gap at least quadratically.

This answers a question about Corollary 6, not about the unit gap theorem of the same note under its hypothesis gap = 1: parity has gap at least 3 from $n = 3$ on, so parity alone says nothing about that case.

5. Reading a variable once over XAG, at three inputs

Let $m^*(F, y)$ be the least number of times a variable y is read by a minimum formula for F, over all minimum formulas.

At three inputs. Over XAG, m*(F, y) = 1 for all 720 pairs (F, y) of a three-input function F and a variable y essential in F. Over AIG the same holds in 390 pairs and $m = 2$* in the other 330. So at three inputs, for each essential variable y, some minimum XAG formula reads y once, while over AIG, in the other pairs, every minimum formula reads y at least twice. The statement is per variable. It does not say that one minimum formula reads every variable once: of the 218 three-input functions that depend on all three inputs, 104, the majority function among them, have no minimum XAG formula that reads all three once.

The three-input statement is one of three conditions behind a measured fact about substitution, within the scope of that measurement, and the other two were measured, not derived from it. The sweep has 8,192 one-step substitutions with output of four inputs, in two regimes: $z(x_0, x_1, x_2)$ with x_0 replaced by $g(x_0, x_3)$, and $z(x_0, x_1)$

with x_0 replaced by $g(x_0, x_2, x_3)$, with opt_XAG(g) ≤ 4. Over XAG the gap of the result, measured against the XAG atlas, equals the gap of z in all 4,360 non-degenerate constructions; in every one of them, writing f for the result and m* = m*(z, x_0), the two general bounds opt(f) ≤ opt(z) + opt(g) and tree(f) ≤ tree(z) + m·*tree(g), read over XAG, are attained, g has no gap and m* = 1, so the change is zero. Of these three conditions only the single reading of x_0 comes from the statement above, and it bounds the formula side only from above; that both bounds are attained and that g has no gap were measured on each construction. The functions z have at most three inputs, and the results have four. With the 16,384 constructions that place the new gate beside z instead of inside it, none of the 24,576 constructions of the sweep reaches any of the four classes of XAG gap 1. No construction of the sweep creates an XAG gap. Outside its two regimes one can: with $z = x_0$ and $g = E_2$ the result is E_2, of XAG gap 1 (Section 3). Another explanation, that the XAG gap is the AIG gap rescaled by the price of an XOR, is false: 31 classes have AIG gap at least 3 and XAG gap 0.

At four inputs it fails. Over the 261,120 pairs (F, y) of a four-input function and an essential variable, m*_XAG is 1 in 206,624 pairs and 2 in 54,496, and never more. In NPN terms, 86 classes have some variable with m* = 2: 85 of gap 0, and `0x167e`, of gap 1. So «for each essential variable, some minimum XAG formula reads it once» is true at three inputs and false at four. Even at three inputs it does not explain on its own why the XAG gap of the four-input classes is small: it is one of the three conditions of the sweep above, and it does not hold at four inputs. The four-input count comes from one exhaustive dynamic program over formulas, whose formula sizes agree on all 65,534 non-constant functions with our earlier exhaustive XAG formula table, a separate program over the same layered scheme; the agreement guards against slips of implementation, not against a shared error of method. It has had no human review, and its outside checks are reviews by language models only; we report it as a measured limit of the three-input statement, not as a theorem, and the mechanism at four inputs remains open.

Related work

Our own earlier work. The AIG and XAG gap distributions at four inputs were in our census of the refuted unit gap [Busnello 2026a, Section 5], where the MIG gap was deferred; the per-class values in the three bases were deposited as a dataset [Busnello 2026b]. The checks of the AIG formula size against Madsen et al., and their limits, are discussed in [Busnello 2026c, Section 4].

Per-basis tables. For MIG, both marginal distributions at four inputs are in Soeken et al. [Soeken et al. 2016], computed with an SMT solver; the extension [Soeken et al. 2017] develops the method. For XAG, the formula side for every class of up to five inputs is the Lflib5 library of Lee et al. [Lee et al. 2019], who describe their table as coherent with Knuth's data [Knuth 2011, §7.1.2]. We have not consulted Knuth's book itself, and no argument of this paper rests on it. We work under the hypothesis that its §7.1.2 gives both C(f) and L(f) over B_2 for each four-input class, so that the XAG column of Section 1 is probably in it. If the hypothesis holds, what remains of this paper is at most the crossing of that column with AIG and MIG, and Sections 2 to 5. Whether the book says anything about how often a minimum formula must read a variable, the subject of Section 5, we do not know, and we cannot exclude it until the book is read. The identification of XAG with B_2 under free inversion is stated, for instance, in [Averkov et al. 2025]. For AIG, the circuit sizes are in Krinkin's catalogue [Krinkin 2026b] and the formula-size histogram in Madsen et al. [Madsen et al. 2023], whose minimizer is part of the Flix compiler; we found that table only after our own was complete, and it then became the check in Section 1 (ii).

Parity. The classical parity bounds used in Section 4 are the U_2 complexity [Schnorr 1974; Jukna 2012; Kombarov 2013], Khrapchenko's formula bound [Khrapchenko 1971], attained at $n = 2^k$ [Tarui 2010]. The sharing term s is Krinkin's [Krinkin 2026a]. We did not find a standard name for it or a published value for parity. The closest phrases we found in the literature, such as shared cones and multiple-fanout counts in technology mapping, measure other things.

Symmetry. We did not find a theorem, folklore statement or census relating symmetry to the gap between circuits and formulas. The nearest work we found relates autosymmetry to multiplicative complexity [Bernasconi et al. 2020], which is a different measure.

How we searched. We searched for the pairing of gaps across bases, for symmetry against the gap, and for the sharing term on several occasions between September and November 2026, with general and academic search

tools, including eight query angles over a corpus of peer-reviewed papers on 1 November 2026, whose queries and returned titles are available from the author. None of the searches returned a per-class comparison of the three gaps. A negative search is a statement about the search. One table this paper relies on, Madsen et al., sat outside the areas we had searched for months.

What this paper does not claim

- It does not claim the per-basis values as new. Their distributions are published in all three bases. Class by class, the AIG circuit sizes [Krinkin 2026b] and the XAG formula sizes [Lee et al. 2019] are published; the other per-class values, the MIG ones among them, are ours and were deposited before this paper [Busnello 2026b], and for the AIG formula sizes Section 1 shows that the published histogram together with our witnesses determines them. What this paper analyses is their crossing.
- It does not claim to be the first to compare the gaps across bases. We did not find such a comparison; that is all.
- It does not claim a law relating the gap to the expressiveness of the basis (Section 2 refutes the obvious one), or relating the gap to symmetry (Section 3, ten classes).
- It does not claim that the agreement with Madsen et al. is independent evidence in the strong sense: the two computations share an algorithm.
- It does not claim certified lower bounds for opt beyond the classes for which proofs were retained.
- It does not claim that the XAG mechanism of Section 5 extends beyond three inputs. At four inputs it fails.
- It does not claim the XAG column of Section 1. We take it to be probably in §7.1.2 of Knuth's book, which we have not consulted; for the same reason we cannot exclude that the book also covers the subject of Section 5.
- It does not claim anything at five inputs or more, except the parity statements of Section 4.

Data availability

The per-class circuit sizes, formula sizes and gaps in the three bases are in the atlas dataset [Busnello 2026b], file `atlas_n4_three_bases.csv`. The DRAT proofs for parity at three inputs are in [Busnello 2026d], folder `certificates/`. The data behind every other number are in the dataset [D]: the three per-basis files, whose values coincide with the atlas class by class, the MIG formula sizes with the SAT comparison, the formula witnesses, the comparisons with the published tables of Soeken et al. and Madsen et al., the tables of Sections 3 to 5, the drat-trim record of the parity proofs at four inputs, the scripts that produced the tables, and the output of checks run from those scripts when the dataset was assembled. The table below names, for each number, the file that holds it. Third-party sources are not included; the published counts that the comparisons use are, with their source.

Provenance by number

Every number written in digits from the Abstract to Section 5 has a row here, apart from section, theorem and citation numbers, dates, and the formulas in n that define a quantity. Each row names the files that hold its numbers, in the atlas dataset [Busnello 2026b] or in the dataset [D], and the check that recomputes them. The script `code/recount.py` of [D] recomputes every row from the files in [D] and prints its numbers in the order stated here; [D] was assembled only after that output matched this table row by row, and its file `checks/recount_plant.out` shows two planted errors that break the match. Before deposit, the same computation was run on this paper and fails if a number in the text disagrees with its row. What it recomputes is finite. For N18 and N19 it checks the instances $n = 3$ to 12 in the artefact; the statements for every n rest on the theorems cited in Section 4 and on the algebra written there, which the script does not prove, and the last column of those two rows says so.

#	numbers as stated	section	artefact	check
N01	222 classes	Abstract, 1	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/atlas_n4.csv`, `data/npn4_tree_mig_dp.csv`, `data/npn4_xag_gap.csv`	[D] `checks/recount.out`
N02	AIG gap {0: 93, 1: 57, 2: 40, 3: 13, 4: 14, 5: 2, 6: 3}, maximum 6 at `0x1668`, `0x16e9`, `0x6996`	Abstract, 1	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/atlas_n4.csv`	[D] `checks/recount.out`
N03	MIG gap {0: 168, 1: 50, 2: 3, 3: 1}, maximum 3 at `0x6996`	Abstract, 1	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/npn4_tree_mig_dp.csv`	[D] `checks/recount.out`, `checks/tree_mig_dp.out`
N04	XAG gap {0: 218, 1: 4}, maximum 1 at `0x01e9`, `0x0661`, `0x1668`, `0x167e`	Abstract, 1	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/npn4_xag_gap.csv`	[D] `checks/recount.out`
N05	zero/positive patterns 82, 83, 42, 11, 3, 1; 129 classes pay in AIG	1	[Busnello 2026b] `atlas_n4_three_bases.csv`, the three gap columns joined	[D] `checks/recount.out`
N06	AIG > XAG in 127 and AIG < XAG in 0; AIG > MIG in 113 and AIG < MIG in 11; MIG > XAG in 54 and MIG < XAG in 3	1	[Busnello 2026b] `atlas_n4_three_bases.csv`, the three gap columns joined	[D] `checks/recount.out`
N07	220 witnesses, 1,782 gates, 16 rows each; none for `0x0000` and `0x00ff`, the classes without a gate	1	[D] `data/tree_witnesses_n4.jsonl`, `data/atlas_n4.csv`	[D] `checks/verify_tree_witnesses.out`, `checks/verify_tree_witnesses_plant.out`, `checks/recount.out`
N08	16 counts over 65,536 functions	1	[D] `data/madsen_tree_n4.capture.txt`, `data/class_invariants_n4.csv`	[D] `checks/recount.out`, `checks/madsen_tree_n4.out`, `checks/class_invariants.out`
N09	17 classes with tree 12 or more	1	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/atlas_n4.csv`	[D] `checks/recount.out`
N10	MIG program and SAT agree on 220 of 222; SAT unfinished on `0x1669`, `0x6996`	1	[D] `data/npn4_tree_mig_dp.csv`	[D] `checks/recount.out`, `checks/tree_mig_dp.out`
N11	21 comparisons with Soeken et al.: 18 bins and 3 statements on named classes; bin of formula size 9 with 2 classes and 34 functions, split 32 and 2 by our program	1	[D] `data/prior_art_mig_n4.csv`, `data/npn4_tree_mig_dp.csv`	[D] `checks/recount.out`, `checks/prior_art_mig_n4.out`
N12	11 classes with gaps (0, 1, 0), the same 11 with AIG gap below MIG gap; 3 classes `0x01e9`, `0x0661`, `0x167e`; 14 in all	Abstract, 2	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/base_gap_monotonicity.csv`	[D] `checks/recount.out`, `checks/base_gap_monotonicity.out`
N13	`0x0661`: (9, 11, 2) AIG, (5, 5, 0) MIG, (6, 7, 1) XAG	2	[Busnello 2026b] `atlas_n4_three_bases.csv`, row `0x0661`	[D] `checks/recount.out`
N14	$E_2 =$ `0x1668`: (9, 15, 6) AIG, (6, 8, 2) MIG, (6, 7, 1) XAG; four classes of XAG gap 1, all (6, 7)	3	[Busnello 2026b] `atlas_n4_three_bases.csv`, the three gap columns joined	[D] `checks/recount.out`

#	numbers as stated	section	artefact	check
N15	32 symmetric functions in 10 classes	3	[D] `data/symmetry_vs_gap_n4.csv`	[D] `checks/recount.out`, `checks/symmetry_vs_gap_n4.out`
N16	AIG gap on symmetric classes {0: 4, 2: 2, 4: 1, 5: 1, 6: 2}	3	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/symmetry_vs_gap_n4.csv`	[D] `checks/recount.out`
N17	MIG gap on symmetric classes {0: 5, 1: 2, 2: 2, 3: 1}; XAG {0: 9, 1: 1}	3	[Busnello 2026b] `atlas_n4_three_bases.csv`; [D] `data/symmetry_vs_gap_n4.csv`	[D] `checks/recount.out`
N18	$s = 3(n-2)$, $r = n-2$, $n = 3$ to 12	Abstract, 4	[D] `data/sharing_term_parity_n.csv`	[D] `checks/recount.out`, `checks/sharing_term_parity_n.out`; finite check $n = 3$ to 12; every optimal AIG and every n by [Carmosino et al. 2026, Theorem 37] and the counting of Section 4, not checked by the script
N19	$s = \text{gap} = 3$ at $n = 3$ and 6 at $n = 4$, DRAT; s below $(n-1)(n-2)$ from $n = 5$	4	[D] `data/sharing_term_parity_n.csv`	[D] `checks/recount.out`, `checks/sharing_term_parity_n.out`; DRAT proofs at $n = 3$ in [Busnello 2026d] `certificates/`; at $n = 4$, the drat-trim record of each proof in [D] `logs/verify_xor4.log`; finite check $n = 3$ to 12; from $n = 5$ on by the algebra $3(n-2) < (n-1)(n-2)$ for $n > 4$ and the cited bounds, not checked by the script
N20	720 pairs; XAG $m^* = 1$ in all; AIG $m^* = 1$ in 390 and 2 in 330; 104 of the 218 functions of three essential inputs without a minimum XAG formula reading all three once	Abstract, 5	[D] `data/mstar_xag_n4.capture.txt` (XAG line only); all four counts recomputed by the dynamic program in `code/recount.py`	[D] `checks/recount.out`
N21	8,192 constructions, opt_XAG(g) at most 4; 4,360 non-degenerate, all with both bounds attained, g without gap and $m^* = 1$; 16,384 beside; 24,576 in all; four classes of gap 1 not reached	5	[D] `data/subst_cost_law_xag.csv`, `data/subst_cost_law_xag_beside.csv`, `data/npn4_xag_gap.csv`	[D] `checks/recount.out`, `checks/subst_cost_law_xag.out`
N22	31 classes with AIG gap at least 3 and XAG gap 0	5	[Busnello 2026b] `atlas_n4_three_bases.csv`, the AIG and XAG gap columns	[D] `checks/recount.out`
N23	261,120 pairs: 206,624 with $m^* = 1$, 54,496 with $m^* = 2$; 86 classes, 85 of gap 0; 65,534 functions	5	[D] `data/mstar_xag_n4.capture.txt`, `data/mstar_xag_n4.csv`	[D] `checks/recount.out`, `checks/mstar_xag_n4.out`

References

- [Averkov et al. 2025] 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.

- [Bernasconi et al. 2020] A. Bernasconi, S. Cimato, V. Ciriani, M. C. Molteni. *Multiplicative Complexity of Autosymmetric Functions: Theory and Applications to Security.* DAC 2020.
- [Busnello 2026a] L. A. Busnello. *The Unit Gap Is Base-Dependent: A Mechanically Verified Refutation and a Cross-Base Census.* Zenodo, v2.1, doi:10.5281/zenodo.23328820 (concept doi:10.5281/zenodo.23226733).
- [Busnello 2026b] L. A. Busnello. *Atlas of the 222 NPN classes of 4-variable Boolean functions in three bases (AIG, MIG, XAG): circuit size, formula size, gap and structural invariants.* Zenodo dataset, v1.1, doi:10.5281/zenodo.23353481 (v1.0: doi:10.5281/zenodo.23329555).
- [Busnello 2026c] L. A. Busnello. *Auditing published circuit tables against exact computation.* Zenodo, v1.2, doi:10.5281/zenodo.23871818 (concept doi:10.5281/zenodo.23642615).
- [Busnello 2026d] L. A. Busnello. *unit-gap-artefacts: machine-checkable artefacts for refutations of Theorems 2 and 7 of arXiv:2603.08033.* Zenodo, v1.2.0, doi:10.5281/zenodo.23597764 (concept doi:10.5281/zenodo.21630762).
- [Carmosino et al. 2026] 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.
- [Jukna 2012] S. Jukna. *Boolean Function Complexity: Advances and Frontiers.* Springer, 2012. Theorem numbers as in the author's online version.
- [Khrapchenko 1971] V. M. Khrapchenko. Complexity of the realization of a linear function in the class of Π-circuits. *Mathematical Notes* 9(1) (1971) 21–23.
- [Knuth 2011] D. E. Knuth. *The Art of Computer Programming, Volume 4A.* Addison-Wesley, 2011. Section 7.1.2.
- [Kombarov 2013] Yu. A. Kombarov. On minimal circuits for linear functions in some bases (in Russian). *Diskretnaya Matematika* 25(1) (2013) 33–44. doi:10.4213/dm1219.
- [Krinkin 2026a] K. Krinkin. *The Unit Gap: How Sharing Works in Boolean Circuits.* arXiv:2603.08033v2, withdrawn.
- [Krinkin 2026b] K. Krinkin. *A Simple Constructive Bound on Circuit Size Change Under Truth Table Perturbation.* arXiv:2603.09379.
- [Lee et al. 2019] S.-Y. Lee, J.-H. R. Jiang, A. Mishchenko, R. Brayton. *Enumeration of Minimum Fanout-Free Circuit Structures.* IWLS 2019.
- [Madsen et al. 2023] 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, 2023. doi:10.1145/3622816.
- [Schnorr 1974] C.-P. Schnorr. Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen. *Computing* 13(2) (1974) 155–171.
- [Soeken et al. 2016] M. Soeken, L. G. Amarù, P.-E. Gaillardon, G. De Micheli. *Optimizing Majority-Inverter Graphs With Functional Hashing.* DATE 2016.
- [Soeken et al. 2017] M. Soeken, L. G. Amarù, P.-E. Gaillardon, G. De Micheli. Exact synthesis of majority-inverter graphs and its applications. *IEEE TCAD* 36(11) (2017) 1842–1855.
- [Tarui 2010] J. Tarui. Smallest formulas for the parity of 2^k variables are essentially unique. *Theoretical Computer Science* 411 (2010) 2623–2627. doi:10.1016/j.tcs.2010.03.022.
- [D] L. A. Busnello. *Base laws at four inputs in AIG, MIG and XAG: data, scripts and checks.* Zenodo dataset, doi:10.5281/zenodo.23901760.

Use of generative AI

This work used AI models extensively, under the author's direction: Anthropic's Claude for the analysis, code and drafting of this paper, and models of other families for read-only review of the results it cites. The author is responsible for all claims. Correctness rests on the artefacts, the scripts that recompute them, and the cited results, not on the models.