

Adaptive Representation and the Instance Complexity of Multiplication

J. R. Landers

August 26, 2026

Main framing and innovation

The usual $O(n^2)$ statement for multiplying two n -bit integers hides a structural fact: equal bit length does not mean equal arithmetic content. A sparse word such as $1000 \cdots 000_2$ activates few partial products, a dense word such as $1111 \cdots 111_2$ activates many, and some dense words become sparse after exact recoding—for example $111111_2 = 2^6 - 1$.

This note develops that observation into a compact adaptive model. We begin with ordinary binary multiplication, isolate an instance-dependent cost beneath the familiar quadratic envelope, then introduce minimum-weight signed recoding as a cheap exact transformation. The average and worst case remain quadratic, but the constants and, more interestingly, the full distribution of attainable multiplication complexities shift in a meaningful way.

1 From bit strings to variable amounts of work

Let x and y be two nonnegative integers whose ordinary binary representations use at most n bits. Write

$$x = \sum_{i=0}^{n-1} a_i 2^i, \quad y = \sum_{j=0}^{n-1} b_j 2^j,$$

where i and j are bit positions and each coefficient $a_i, b_j \in \{0, 1\}$ is the bit stored at that position. The factor 2^i is not something a digital circuit needs to exponentiate during ordinary multiplication; it is the positional weight assigned to bit location i . A bit placed in location i is interpreted as contributing 2^i when its value is one.

Multiplying the two expansions gives

$$xy = \left(\sum_{i=0}^{n-1} a_i 2^i \right) \left(\sum_{j=0}^{n-1} b_j 2^j \right) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (a_i b_j) 2^{i+j}.$$

Because a_i and b_j are bits, the product $a_i b_j$ is exactly the logical AND $a_i \wedge b_j$. The term 2^{i+j} simply says that this partial product belongs in output column $i + j$. Thus ordinary schoolbook multiplication at the bit level consists of pairwise ANDs, placement into binary columns, and addition of the resulting column contents with carries.

Now define the ordinary Hamming weight

$$H(x) = \sum_{i=0}^{n-1} a_i,$$

so $H(x)$ is the number of one-bits in the binary representation of x . Define $H(y)$ analogously. A partial product is nonzero only when both participating bits are one, so the number of nonzero pairwise products is exactly

$$C_0(x, y) = H(x)H(y).$$

Here $C_0(x, y)$ is an *active partial-product count*: it measures how many of the n^2 possible bit interactions actually carry a one before the additions are resolved.

This immediately separates the input length from the input structure. Since $0 \leq H(x), H(y) \leq n$,

$$0 \leq C_0(x, y) \leq n^2.$$

The upper bound n^2 is the familiar quadratic envelope. But it is attained only by maximally dense operands. A power of two has Hamming weight one, and multiplying two powers of two activates only one partial product, regardless of how many bit positions are available.

First structural observation

For fixed bit length n , multiplication already has an instance geometry:

$$C_0(x, y) = H(x)H(y) \leq n^2.$$

The length n determines the largest possible interaction grid; the actual bit patterns decide how much of that grid is active.

This is not yet an improved multiplication algorithm. A conventional fixed-width hardware multiplier may evaluate a predetermined gate network whether or not many partial products are zero. The point is more basic: the arithmetic instance contains exploitable structure, and an algorithm or architecture capable of skipping zero work could use it.

2 Representation is part of the computation

The previous section measures sparsity in ordinary binary. That choice is natural, but it is not unique. The same integer can sometimes be represented with fewer nonzero digits if negative digits are permitted.

Consider the dense six-bit word

$$63 = 111111_2.$$

Ordinary binary uses six nonzero digits, so $H(63) = 6$. But

$$63 = 64 - 1 = 2^6 - 2^0.$$

If we allow signed binary digits $d_i \in \{-1, 0, 1\}$, this becomes

$$63 = 1 \cdot 2^6 + (-1) \cdot 2^0,$$

which contains only two nonzero digits. Multiplying an integer y by 63 can then be read as

$$63y = (y \ll 6) - y,$$

where $y \ll 6$ denotes shifting the binary representation of y six positions to the left. The shift changes location rather than numerically computing 2^6 , and the dense run of six one-bits has collapsed to one shift and one subtraction.

This example suggests a natural optimization variable: not the value of the integer, which must remain unchanged, but the representation through which the multiplication is carried out.

Let a signed binary representation of x be any exact expansion

$$x = \sum_{i=0}^{L_x-1} d_i 2^i, \quad d_i \in \{-1, 0, 1\},$$

where L_x is the number of digit positions used by that representation. Define the signed weight of this particular expansion to be the number of its nonzero digits,

$$w(d) = \#\{i : d_i \neq 0\}.$$

Among all exact signed binary expansions of x , define

$$\mathcal{W}(x) = \min_{\substack{x = \sum_i d_i 2^i \\ d_i \in \{-1, 0, 1\}}} \#\{i : d_i \neq 0\}.$$

Thus $\mathcal{W}(x)$ is the minimum signed-digit weight attainable without changing the integer x .

A classical representation called the *non-adjacent form*, or NAF, realizes this minimum. Its defining property is that no two nonzero digits are adjacent. Operationally, a run such as $01111 \cdots 111$ is replaced by a positive digit at the left edge and a negative digit at the right edge. The recoding can be carried out by a linear pass through the bits, so the cost of finding this minimum-weight signed representation is $O(n)$ for an n -bit input [1, 2].

The important conceptual move is therefore inexpensive: inspect the input, convert it to a structurally cheaper exact form, perform the multiplication there, and interpret the result back in the ordinary integer domain.

3 The adaptive optimization problem

To turn that idea into a cost model, let $\alpha > 0$ denote the constant cost per input bit of inspection and recoding, and let $\beta > 0$ denote the constant cost assigned to each surviving nonzero digit interaction, including its contribution to the accumulation stage at a fixed unit scale. These constants absorb machine-dependent details; the dependence on the input structure is carried by $\mathcal{W}(x)$ and $\mathcal{W}(y)$.

Definition 1: Adaptive sparse multiplication cost

For two operands x and y of at most n bits, define

$$T_n^\star(x, y) = \alpha n + \beta \mathcal{W}(x) \mathcal{W}(y).$$

The first term αn is the linear cost of detecting and constructing the sparse signed representations. The second term counts the pairwise interactions that remain once both operands have been recoded.

This model is deliberately simple. It isolates the question we care about: how much of multiplication's nominal n^2 interaction grid survives after a cheap exact transformation has been chosen? More elaborate models could price additions, carries, memory traffic, or different shifts separately. None of those refinements changes the narrative structure: representation selection has a cost, multiplication in the chosen representation has a cost, and the best path is the one minimizing their sum.

At the broadest level, let $\mathcal{R}$ denote a family of allowed exact representations and let $\mathcal{A}$ denote a family of multiplication procedures compatible with those representations. For a particular pair (x, y) , the ideal adaptive problem is

$$C^\star(x, y) = \min_{R \in \mathcal{R}, A \in \mathcal{A}} \left[C_{\text{detect}}(x, y) + C_R(x, y) + C_A(R(x), R(y)) + C_{R^{-1}} \right].$$

Here R is a chosen representation map, A is the multiplication algorithm used after the map, C_{detect} is the cost of inspecting the operands, C_R is the forward transformation cost, C_A is the transformed multiplication cost, and $C_{R^{-1}}$ is any cost required to return the result to the desired representation. The NAF model above is a tractable special case in which the representation optimization has a known linear-time solution.

That distinction matters. An arbitrarily rich representation family might make multiplication itself very cheap, but finding the best representation could become expensive enough to erase the gain. The useful regime is therefore one in which the transformation is cheap and predictably reduces the active arithmetic.

4 Average case: where a random multiplication lives

We now ask what happens for a typical input pair. To distinguish random inputs from fixed operands, let X and Y be independent random variables, each uniformly distributed over the 2^n integers

$$\{0, 1, \dots, 2^n - 1\}.$$

Equivalently, every one of their n ordinary binary bits is an independent Bernoulli random variable taking the values zero and one with probability $1/2$.

For ordinary binary, the Hamming weight therefore satisfies

$$H(X) \sim \text{Binomial}\left(n, \frac{1}{2}\right),$$

which means

$$\mathbb{P}(H(X) = k) = \frac{\binom{n}{k}}{2^n}, \quad k = 0, 1, \dots, n.$$

Its expectation is

$$\mathbb{E}[H(X)] = \frac{n}{2}.$$

Because X and Y are independent, so are $H(X)$ and $H(Y)$, and hence

$$\mathbb{E}[H(X)H(Y)] = \mathbb{E}[H(X)]\mathbb{E}[H(Y)] = \frac{n^2}{4}.$$

Thus a random binary multiplication activates, on average, one quarter of the n^2 possible one-bit interactions in this sparse-work model.

Minimum-weight signed recoding changes the typical density. For NAF, the asymptotic fraction of nonzero digits is $1/3$, so

$$\mathbb{E}[\mathcal{W}(X)] = \frac{n}{3} + O(1).$$

The notation $O(1)$ here means a quantity bounded in magnitude by a constant independent of n ; it records the small boundary effects introduced by the first and last digit positions. Since X and Y are independent and the recoding is applied separately to each operand,

$$\mathbb{E}[\mathcal{W}(X)\mathcal{W}(Y)] = \mathbb{E}[\mathcal{W}(X)]\mathbb{E}[\mathcal{W}(Y)] = \left(\frac{n}{3} + O(1)\right)^2 = \frac{n^2}{9} + O(n).$$

Proposition 1: Average optimized cost

Let X and Y be independent uniformly distributed n -bit operands. Under Definition 1,

$$\mathbb{E}[T_n^\star(X, Y)] = \frac{\beta}{9}n^2 + O(n).$$

Consequently,

$$\mathbb{E}[T_n^\star(X, Y)] = \Theta(n^2),$$

where $\Theta(n^2)$ means that the expected cost is bounded above and below by positive constant multiples of n^2 for sufficiently large n .

Proof. By Definition 1,

$$\mathbb{E}[T_n^\star(X, Y)] = \alpha n + \beta \mathbb{E}[\mathcal{W}(X)\mathcal{W}(Y)].$$

Substituting $\mathbb{E}[\mathcal{W}(X)\mathcal{W}(Y)] = n^2/9 + O(n)$ gives

$$\mathbb{E}[T_n^\star(X, Y)] = \alpha n + \frac{\beta}{9}n^2 + O(n) = \frac{\beta}{9}n^2 + O(n),$$

since the explicit linear recoding term can be absorbed into the final $O(n)$ term. $\square$

The asymptotic exponent has not changed: random multiplication is still quadratic in this model. What has changed is the location of the average within that quadratic envelope. The active interaction scale moves from approximately

$$\frac{n^2}{4} \quad \text{to} \quad \frac{n^2}{9}.$$

Ignoring the lower-order linear recoding cost, this is a reduction by a factor of

$$\frac{n^2/4}{n^2/9} = \frac{9}{4} = 2.25$$

in the expected number of active digit-pair interactions. The point is not that every multiplication becomes 2.25 times faster on every machine, but that the optimized representation exposes substantially less active arithmetic on a typical input.

5 Worst case: the quadratic envelope survives

The optimizer also changes the hardest structural patterns. A NAF representation contains no adjacent nonzero digits. An n -bit integer may require at most $n + 1$ NAF digit positions because a carry can create one additional leading digit. Among $n + 1$ positions with no adjacent nonzero entries, at most

$$\left\lceil \frac{n+1}{2} \right\rceil$$

can be nonzero. Therefore every n -bit integer x satisfies

$$\mathcal{W}(x) \leq \left\lceil \frac{n+1}{2} \right\rceil.$$

The same bound holds for y , and Definition 1 gives

$$T_n^\star(x, y) \leq \alpha n + \beta \left\lceil \frac{n+1}{2} \right\rceil^2.$$

Hence

$$\max_{x,y} T_n^\star(x, y) = \Theta(n^2).$$

This is an important limitation, not a defect in the formulation. A cheap change of representation has improved the constant and altered which patterns are difficult, but it has not removed the possibility of operands whose sparse signed forms still contain linearly many nonzero digits. Two such operands still generate quadratically many pairwise interactions.

The contrast with ordinary binary is nevertheless sharp. The raw active-interaction envelope is n^2 , attained when both words are all ones. Under minimum-weight signed recoding, the corresponding envelope is approximately

$$\frac{n^2}{4}.$$

Moreover, the all-ones word is no longer hard at all: $111 \cdots 111_2 = 2^n - 1$ becomes a two-term signed expression. The hard cases migrate toward patterns whose nonzero signed digits are already well separated, such as alternating structures that cannot be compressed into long runs.

Proposition 2: Worst-case optimized envelope

For every pair of n -bit operands x, y ,

$$T_n^\star(x, y) \leq \alpha n + \beta \left\lceil \frac{n+1}{2} \right\rceil^2.$$

Thus the adaptive scheme preserves the worst-case order $\Theta(n^2)$ while reducing the maximum possible density of active signed-digit interactions.

The average and worst case now tell complementary stories. The worst case says the quadratic exponent survives. The average case says most random inputs sit much lower inside that envelope. To see the structure between those two summaries, we need the full distribution.

6 The full distribution of instance complexities

For a fixed bit length n , every integer $x \in \{0, 1, \dots, 2^n - 1\}$ has a definite optimized weight $\mathcal{W}(x)$. Define

$$p_n(k) = \mathbb{P}(\mathcal{W}(X) = k) = 2^{-n} |\{x \in \{0, \dots, 2^n - 1\} : \mathcal{W}(x) = k\}|.$$

Here $|S|$ denotes the number of elements in a finite set S . Thus $p_n(k)$ is simply the fraction of all n -bit inputs whose minimum signed-digit weight equals k . The values $p_n(k)$ form a probability mass function: each is nonnegative and

$$\sum_k p_n(k) = 1.$$

Because the optimized active multiplication term is $\mathcal{W}(X)\mathcal{W}(Y)$, its distribution is obtained by multiplying two independent random weights. Let

$$R = \mathcal{W}(X)\mathcal{W}(Y)$$

be the random number of active signed-digit pairs before multiplication-specific constant factors are applied. Then for any nonnegative integer r ,

$$\mathbb{P}(R = r) = \sum_{\substack{k, \ell \geq 0 \\ k\ell = r}} p_n(k)p_n(\ell).$$

The sum runs over all ordered factor pairs (k, ℓ) whose product is r .

Proposition 3: Exact input-induced complexity law

For independent uniformly sampled n -bit operands X and Y , the optimized cost of Definition 1 has probability mass function

$$\mathbb{P}(T_n^\star(X, Y) = \alpha n + \beta r) = \sum_{\substack{k, \ell \geq 0 \\ k\ell = r}} p_n(k)p_n(\ell).$$

In words: the full multiplication-complexity distribution is the product distribution of two independent optimized representation weights, shifted by the deterministic recoding cost αn and scaled by β .

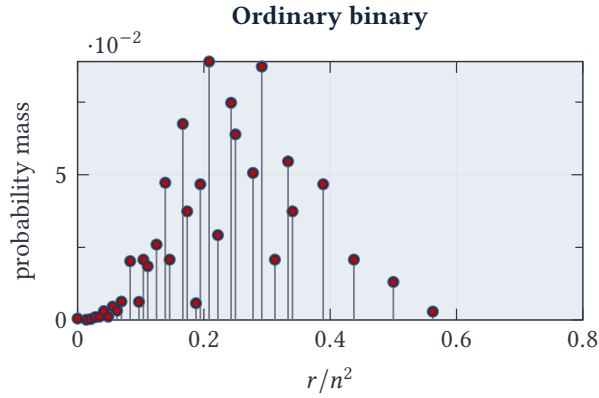


Figure 1: Ordinary binary, $n = 12$. The horizontal coordinate is the active-pair count r , normalized by n^2 . Most mass lies around the binary mean $n^2/4$.

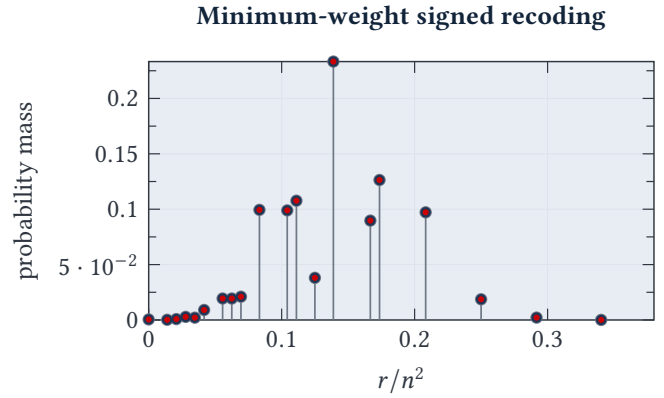


Figure 2: Minimum-weight signed recoding, $n = 12$. The exact finite-input distribution moves left and becomes more concentrated.

Proof. The event $\mathcal{W}(X)\mathcal{W}(Y) = r$ can occur only when $\mathcal{W}(X) = k$ and $\mathcal{W}(Y) = \ell$ for some factor pair $k\ell = r$. Those cases are disjoint. Independence gives

$$\mathbb{P}(\mathcal{W}(X) = k, \mathcal{W}(Y) = \ell) = p_n(k)p_n(\ell),$$

and summing over all factor pairs yields the stated law. The affine transformation $\alpha n + \beta r$ converts active-pair count into the modeled total cost. $\square$

For ordinary binary, the analogous distribution is explicit because $H(X)$ is binomial:

$$\mathbb{P}(H(X) = k) = \frac{\binom{n}{k}}{2^n}.$$

Therefore the raw binary active-interaction count $C_0(X, Y) = H(X)H(Y)$ satisfies

$$\mathbb{P}(C_0(X, Y) = r) = \frac{1}{4^n} \sum_{\substack{k\ell=r \\ 0 \leq k, \ell \leq n}} \binom{n}{k} \binom{n}{\ell}.$$

This is the exact finite- n distribution, not an asymptotic approximation. It says that each possible multiplication complexity is weighted by the number of operand pairs whose Hamming weights multiply to that value.

These two plots show the shape of the idea directly. They are exact discrete laws for $n = 12$, obtained by counting all possible operand weights and then forming the corresponding product distributions; no Monte Carlo sampling is involved. Ordinary binary concentrates around an active density of about $1/4$, while minimum-weight signed recoding concentrates around about $1/9$.

The concentration can also be stated asymptotically. Since $\mathcal{W}(X)/n$ converges in probability to $1/3$, the normalized active cost satisfies

$$\frac{T_n^*(X, Y) - \alpha n}{\beta n^2} = \frac{\mathcal{W}(X)}{n} \frac{\mathcal{W}(Y)}{n} \xrightarrow{P} \frac{1}{9}.$$

The symbol $\xrightarrow{P}$ means convergence in probability: for any fixed tolerance $\varepsilon > 0$, the probability that the normalized quantity differs from $1/9$ by more than ε tends to zero as n grows. In plain language, large random multiplications increasingly cluster around a predictable fraction of the quadratic envelope.

This is the distributional view that average-case notation hides. Instead of assigning all n -bit multiplications one number, we assign every input pair (x, y) its own optimized cost and ask how those costs are arranged over the 2^{2n} possible pairs.

7 From sparse recoding to adaptive multiplication

The preceding results can be viewed geometrically over the input space. There are 2^n possible n -bit values for each operand and therefore 2^{2n} ordered pairs (x, y) . The map

$$(x, y) \mapsto T_n^\star(x, y)$$

assigns a cost to every point in this discrete space. The average case is the mean height of that landscape, the worst case is its highest point, and Proposition 3 tells us how much of the landscape lies at each attainable height.

Minimum-weight signed recoding deforms this landscape nonuniformly. A dense run of ones can move dramatically downward because it becomes a short difference of powers of two; a pattern that is already alternating may move very little. The optimizer therefore does not remove a fixed amount of work from every multiplication. It recognizes particular structural redundancies and compresses the inputs that contain them.

This suggests the more general adaptive problem. Let $\mathcal{R}$ be a family of exact representations and $\mathcal{A}$ a family of multiplication algorithms compatible with them. For a particular operand pair (x, y) , define

$$C^\star(x, y) = \min_{R \in \mathcal{R}, A \in \mathcal{A}} C(x, y; R, A),$$

where $C(x, y; R, A)$ is the complete cost of inspecting the operands, transforming them with R , multiplying with A , and returning the result to the desired representation. If t denotes an attainable cost value, then uniform sampling over all input pairs induces the distribution

$$\mathbb{P}(C^\star = t) = \frac{1}{2^{2n}} |\{(x, y) : C^\star(x, y) = t\}|.$$

This is the representation-agnostic version of Proposition 3. Once an optimizer is specified, the object of study is no longer merely one asymptotic bound, but the full distribution of cheapest exact computational paths.

There is an unavoidable tradeoff. Enlarging $\mathcal{R}$ and $\mathcal{A}$ may make the multiplication stage cheaper, but deciding which path to use also costs time. The useful question is therefore

Can a simpler representation be recognized and exploited more cheaply than multiplying directly?

NAF provides a clean affirmative example of the right asymptotic shape: its structural detection and recoding cost is linear in n , while the active multiplication term remains quadratic. A richer optimizer would need to preserve the same imbalance between cheap recognition and expensive work avoided.

8 Interpretation

The ordinary statement $O(n^2)$ describes how the size of the multiplication problem scales with bit length. It does not say that every pair of n -bit inputs contains the same amount of arithmetic structure. Once the bit patterns are made explicit, three layers appear. First, ordinary Hamming weight determines how many partial products are nonzero. Second, representation can expose additional sparsity that ordinary binary hides. Third, selecting that representation is itself an optimization whose cost must be counted.

The NAF model makes all three layers visible without making the selection problem difficult. A linear-time exact recoding changes the typical nonzero digit density from $1/2$ to $1/3$. In the pairwise interaction model this moves the mean active term from $n^2/4$ to $n^2/9$. The worst-case order remains $\Theta(n^2)$, but its envelope is lower and the inputs that approach it are different. Between the easy and hard extremes lies the discrete distribution derived in Proposition 3.

Closing view

Multiplication has both a length complexity and an instance geometry. The bit length n fixes the size of the available interaction space; the arrangement and representation of the digits determine where a particular multiplication lies inside it. A cheap exact recoding can therefore be understood as a deformation of the complexity distribution over input space. In this view, representation choice is not cosmetic notation. It is part of the algorithm.

Scope

The cost model used here counts active signed partial products together with linear recoding overhead. It is therefore best read as an algorithmic or architecture-aware complexity measure: it describes systems capable of skipping work associated with zero digits or exploiting sparse decompositions. A fixed hardware multiplier may traverse essentially the same physical circuit regardless of operand sparsity, so wall-clock latency need not equal $T_n^*(x, y)$. Likewise, NAF optimality and its asymptotic one-third nonzero density are classical facts. The contribution of this note is the compact adaptive-optimization framing, the explicit separation between input length and input structure, and the induced distributional view of multiplication complexity.

References

- [1] G. W. Reitwiesner, "Binary Arithmetic," *Advances in Computers*, vol. 1, pp. 231–308, 1960. [doi:10.1016/S0065-2458\(08\)60610-5](https://doi.org/10.1016/S0065-2458(08)60610-5).
- [2] A. D. Booth, "A signed binary multiplication technique," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, no. 2, pp. 236–240, 1951.
- [3] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, 1998.