How Much of a Small Chip Actually Computes?

Physical-implementation overhead in tiny arithmetic blocks, measured on an open 130nm flow

Preprint (draft). Subomi Olagoke.

Abstract

A synthesized netlist is logic. A layout that a foundry could actually make is logic plus all the machinery that lets logic survive being manufactured: antenna diodes that bleed off etch charge before it destroys a gate, well taps that keep the substrate from latching up, buffers inserted to close timing and fix hold, and fill that pads the die to a target density. This overhead is routinely summarized away into a single area number, and I have not found it tracked as a block scales. Taking one operation, a signed int8 multiply-accumulate, through the open SkyWater 130nm flow at lane counts from 1 to 64, I decompose every placed layout into logic and infrastructure and watch the split move. The arithmetic share does not hold and does not improve with size: it falls from 59% of the placed cells at one lane to 48% at sixty-four, crossing below half along the way, so a large multiply-accumulate turns out to be mostly not multiply-accumulate. The overhead does not amortize, because it outgrows the logic, and its pieces scale against different quantities: well taps against die area, antenna diodes against routed wirelength, and the wirelength itself grows faster than the cell count, which is the thing that erodes the logic share.

1. Why measure this

Two numbers describe a hardware block in almost every writeup: its area and its clock. Both are outputs of physical implementation, and both hide what physical implementation actually did. Between a gate-level netlist and a GDSII the tools add a surprising amount of matter that is not logic, and they add it for physical reasons that have nothing to do with what the block computes. A reader is told the block is, say, a tenth of a square millimetre, and is left to assume that tenth is arithmetic. It is not.

The gap is easy to see once at any single design point. In an earlier note I took this same multiply-accumulate to a layout and found that of roughly 7,600 placed standard cells, only about half did any arithmetic; the rest were 1,759 antenna diodes, 1,440 well taps, and a few hundred buffers, with thousands of fill cells beyond that. One point is an anecdote. The question this note asks is

whether it is a curve: as the block grows, does the overhead shrink as a share, hold constant, or grow, and do its parts move together or separately?

2. The block and the flow

The device under test is `mak`, a signed int8 multiply-accumulate lane: it multiplies LANES activation and weight bytes, sums the products through an adder tree, and accumulates the result across cycles into a fixed 32-bit register. It is the atomic operation of a small quantized matmul, and it is parameterized on LANES, which is the only thing swept here. The accumulator width is held at 32 bits throughout, so the sequential state is deliberately constant while the combinational arithmetic grows.

Every point uses the identical flow and identical targets: the open SkyWater 130nm PDK (sky130A), synthesized with Yosys and placed and routed with OpenROAD under OpenLane, at a fixed 40 ns clock, a fixed 40% core utilization target, with heuristic antenna-diode insertion enabled. Only LANES changes, across 1, 2, 4, 8, 16, 32 and 64. Each run is signed off with design-rule and layout-versus- schematic checks, and each emits a machine-readable metrics file from which the cell census, wirelength, die area and worst-corner timing slack are read. Nothing below is hand-counted.

3. Results

For each lane count the placed layout is decomposed into logic (the combinational arithmetic, its inverters, and the 32 accumulator flip-flops) and the machinery around it, antenna diodes, well taps, and timing and hold buffers, alongside die area, routed wirelength and worst-corner setup slack. All seven runs sign off clean on design-rule and layout-versus-schematic checks.

LANES	logic	diodes	taps	buffers	total cells	logic %	die (μm^2)	wire (mm)
1	619	80	220	155	1,054	58.7%	20,862	13.9
2	1,076	207	384	192	1,838	58.5%	34,222	23.4
4	2,005	792	731	275	3,783	53.0%	61,190	48.1
8	3,979	1,710	1,440	466	7,571	52.6%	115,201	105.1
16	7,859	3,427	2,856	796	14,924	52.7%	219,286	232.0
32	16,400	8,974	6,050	1,511	32,942	49.8%	448,891	543.3

LANES	logic	diodes	taps	buffers	total cells	logic %	die (μm²)	wire (mm)
64	31,827	20,073	11,795	3,023	66,726	47.7%	865,429	1,276.4

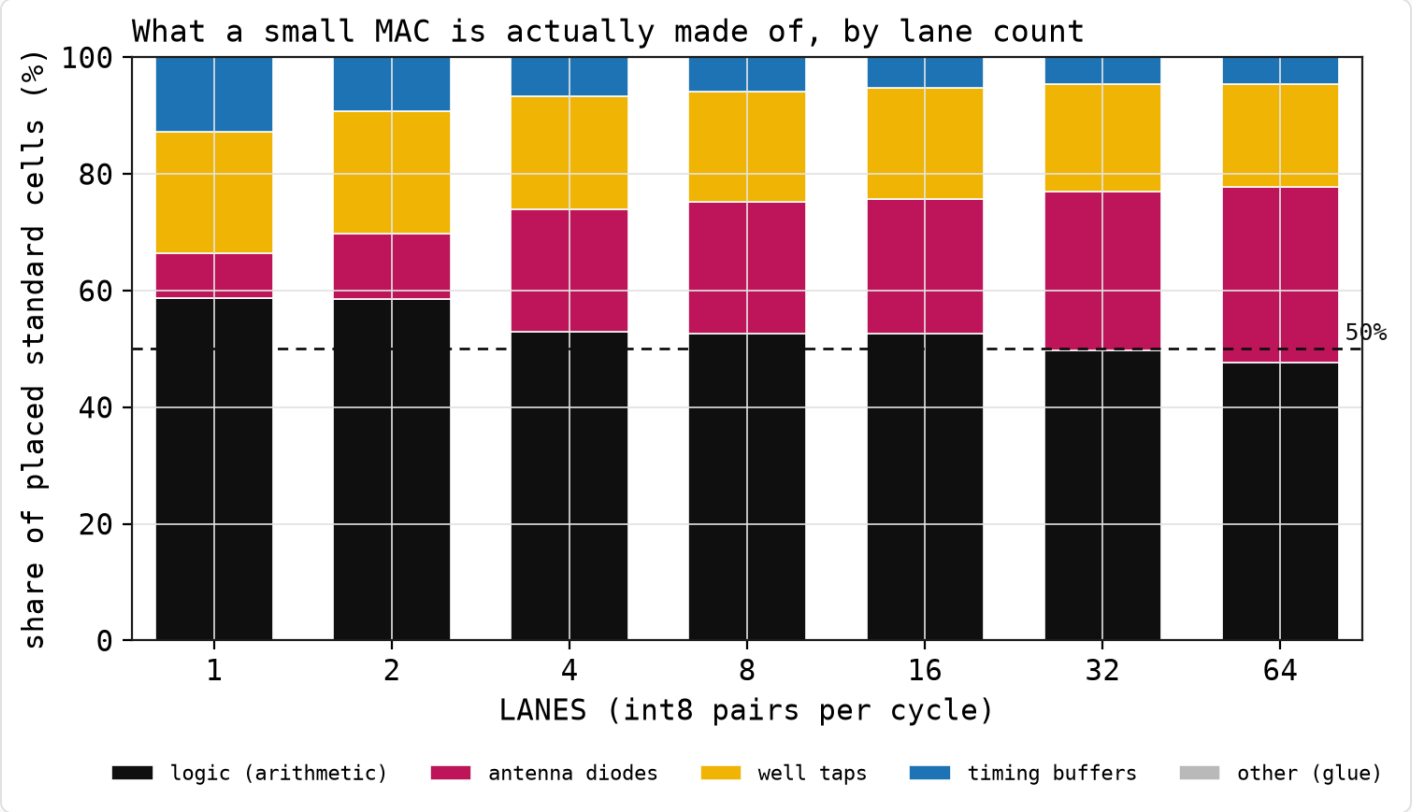

Figure 1. The composition of the placed layout as a share of standard cells, by lane count. The black band is logic; it starts above half and slides beneath the fifty-percent line as the antenna diodes (in pink) claim a growing share.

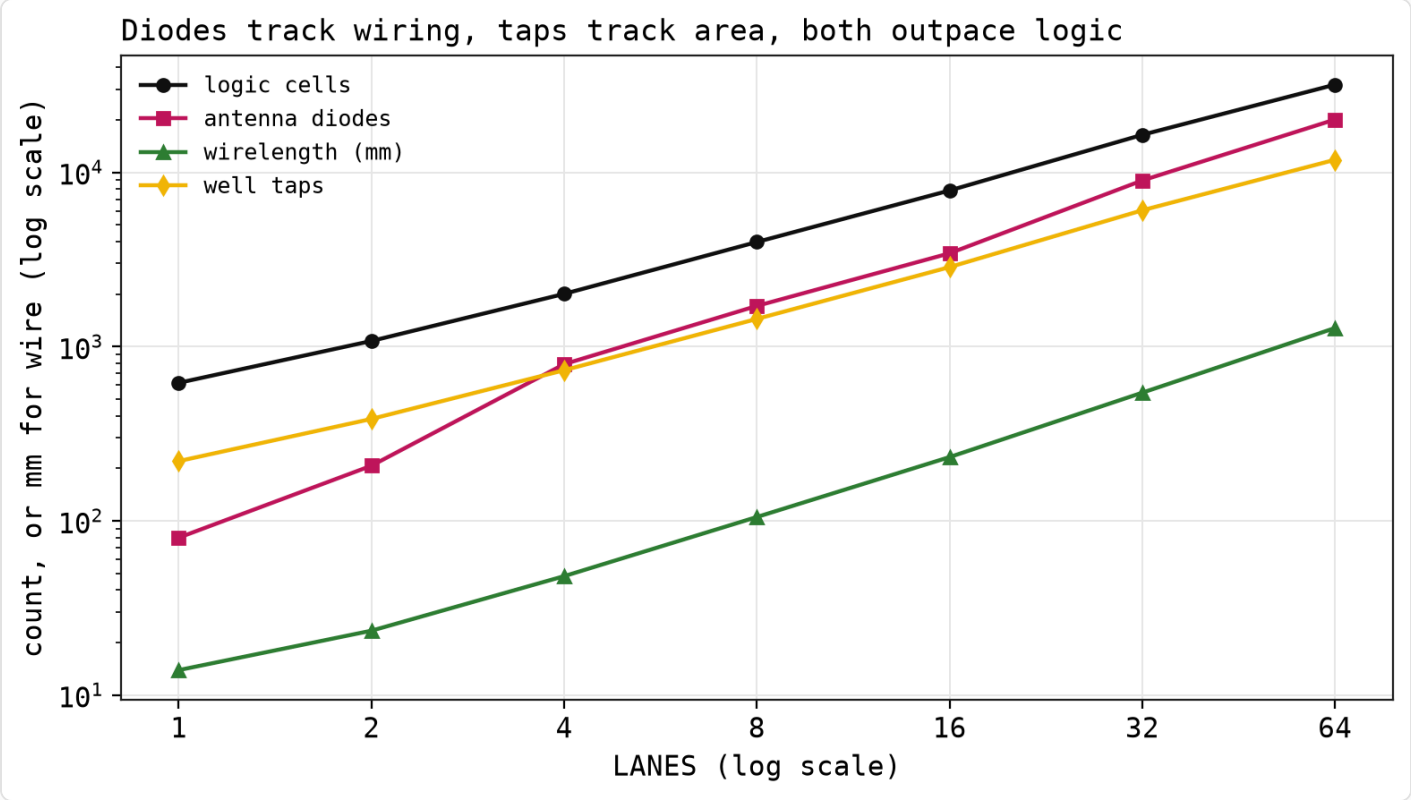

Figure 2. Logic, antenna diodes, well taps and routed wirelength against lane count, log-log. The diode curve runs parallel to the wire curve, not the logic curve, and both are steeper than logic.

Three things fall out:

1. **The arithmetic share falls, and crosses below half.** Logic is 58.7% of the placed cells at one lane and 47.7% at sixty-four, sliding past 50% somewhere between sixteen and thirty-two lanes (Figure 1). Bigger does not mean more logic. The overhead is not a fixed tax that a larger block dilutes; it grows faster than the arithmetic it exists to protect.
2. **The pieces scale against different quantities, and only one of them is logic.** Well taps hold at roughly a fifth of the cells across the whole range, because they sit on a fixed placement grid and so track die area rather than gates. Timing buffers stay small. The accumulator is thirty-two flip-flops at every size, a constant that is three percent of the cells at one lane and rounding error by sixty-four. The antenna diodes are the mover: they hold at about sixteen per millimetre of routed wire across the entire sweep, so they track wiring, not logic (Figure 2), and their share climbs from eight percent of the cells to thirty.
3. **Wiring is the engine of the decline.** Routed length grows super-linearly, from 14 mm at one lane to 1.28 metres at sixty-four, outpacing the cell count, so the metal spent per logic cell rises from 22 to 40 μm as the datapath widens and its nets reach further. Because the diodes track that wire, they outpace the logic, and the logic share erodes. The knob that moves the whole result is not the arithmetic, it is how much metal it takes to connect it.

4. Discussion

The reason any of this matters is that it changes how a small hardware result should be read and reported. "A 0.1 mm² MAC" invites the assumption that the MAC is 0.1 mm² of MAC. At small sizes it is not close, and the difference is not noise, it is structural and predictable. The infrastructure is not waste and it is not optional; a layout without the diodes fails fabrication, and one without the taps latches up. But it is also not the thing the block is for, and lumping it into a single area figure quietly overstates how much silicon the arithmetic actually costs.

There is a practical corollary for anyone estimating an accelerator from a tile. An accelerator is thousands of these, so the per-tile overhead is paid thousands of times, and the direction the fraction moves is a real design lever rather than a rounding error. The comfortable assumption is that a wider tile amortizes the overhead. The data points the other way: a wider tile buys a worse logic fraction, because the wiring, and the antenna diodes that track it, grow faster than the multipliers do. So on this axis the overhead is not only paid many times over, it is a growing share of each tile as the tile widens, which is a quiet argument for many small tiles rather than a few large ones. That is one process and one block shape talking, but it is the opposite of the intuition, and the intuition is what usually goes unstated in a napkin estimate.

5. Limitations

This is one process, one flow, one block shape, and open-flow defaults. A different PDK, a different utilization target, or a commercial flow with more aggressive optimization would move the absolute numbers, and possibly the trend. The block is combinational-heavy with a single small register, so the constant sequential cost is unusually visible; a register-heavy block would tell a different story. The antenna and tap counts are artifacts of a particular diode-insertion heuristic and a particular tap spacing, both configurable. And nothing here is fabricated; these are signed-off layouts, not measured silicon. What the note claims is narrow and, within that, exact: on this flow, at these targets, the composition of a small MAC is what the numbers say, and it is not a constant.

6. Conclusion

The headline number on a small hardware block, its area, is mostly not the thing you think it is measuring. A large share of a tiny layout is machinery for surviving fabrication rather than machinery for computing, that share moves predictably with size, and its parts scale against logic, wiring and area separately. It costs nothing to report the split alongside the total, and it is more honest than letting one area figure imply that all of it computes.