# PPU Version A — Explained

## What the noise source is, where the weights come from, and when each of them changes

Five questions, answered step by step, with the real circuits
Probabilistic Processing Unit · 16-spin Ising / QUBO solver · TSMC 65 nm GP · standard CMOS

Prepared 9 Sep 2026 · minimum font size 12 pt
Circuit slides: PPU_integer_factorization_test (1).pptx (9 Aug 2026). Facts: ANALOG_SPEC.md, ppu_noise_therm2.scs, ppu_sequencer.v, ppu_wlatch.v, post-layout measurements (26–27 Aug 2026).

# THE FIVE QUESTIONS

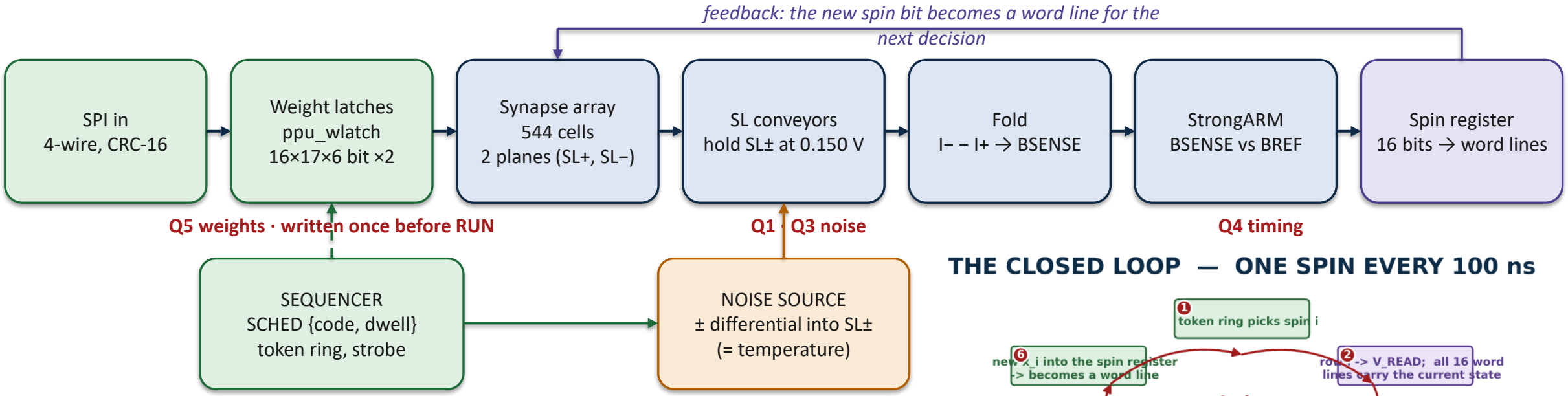
*and where each one is answered in this deck*

| # | Question (as asked) | Short answer | Slides |
|---|---|---|---|
| 1 | Which noise source did we use? | Two, in sequence: an LFSR-steered current DAC (as presented on 9 Aug) → amplified THERMAL noise (taped out 26 Aug). The switch was decided on 9 Aug. | 9–19 |
| 2 | We built the chip in TSMC 65 nm — so a test ASIC before eFLASH? | Yes. Standard-CMOS array; weights in 6-bit latches, not floating gates; entropy from a separate thermal-noise block, not from RTN. | 4–6 |
| 3 | Which noise sources exactly? | LFSR DAC: 6 constant-sum legs, 10.2 µA total. Thermal chain: a differential pair's own kT noise → 3 gain stages (≈1340×) → 6-bit constant-sum steerer, up to ~91 µA at the decision node. | 10–19 |
| 4 | Is the noise refreshed at every spin decision? | The random VALUE: yes, every slot (LFSR redraw / physical noise). The AMPLITUDE (temperature): no — it steps per SCHED entry, held for `dwell` supercycles. | 20–22 |
| 5 | Are the weights loaded once? From where? | Once, before RUN, then held. PC/Python → PCIe → Zynq 7030 → 4-wire SPI (CRC-16, 544-weight burst) → ppu_wlatch (16×17×6 bit ×2 planes) → 544 synapse cells. | 23–25 |

Every number is traced to a file or a measurement. Where the 9-Aug presentation and the taped-out chip differ, both are shown and dated.

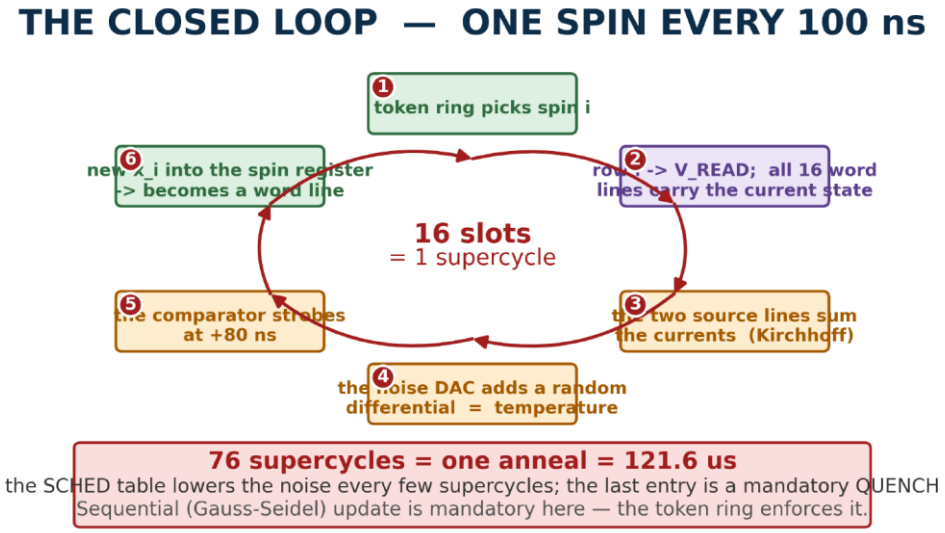
# THE MACHINE ON ONE PAGE — WHERE EACH QUESTION LIVES

*Read left to right: a problem goes in once, currents are compared, one spin bit comes out per slot and feeds straight back*




Three things move at three different speeds:

- the WEIGHTS are written once and never change during a run (Q5)
- the NOISE VALUE is new at every slot; its AMPLITUDE steps with the schedule (Q1, Q3, Q4)
- one SPIN BIT is rewritten per slot and immediately becomes an input for the next decision




*deck: THE CLOSED LOOP — one spin every 100 ns (16 slots = 1 supercycle)*

# Q2 · WHAT AN eFLASH CHIP WOULD HAVE BEEN — AND WHAT VERSION A IS INSTEAD

*The two things eFLASH would have provided, and how each was replaced with standard CMOS*

**PART A — the chip**

| the eFLASH plan (PDK not available) | Version A as built (standard CMOS, taped out 26 Aug) |
|---|---|
| WEIGHT = charge on a floating gate<br>analog, non-volatile, programmed by ISPP<br>one cell = one weight, current set by stored charge | WEIGHT = a 6-bit code in a latch (ppu_wlatch)<br>digital, loaded over SPI, held for the run<br>six binary-weighted current legs turn the code into a current (ppu_synapse6) |
| ENTROPY = random telegraph noise (RTN) of the flash cell itself<br>"free" randomness inside every weight cell | ENTROPY = a separate block: the thermal noise of a transistor pair, amplified ≈1340×<br>and steered into the shared wires (ppu_noise_therm2)<br>one source for the whole array, amplitude set by a 6-bit code |

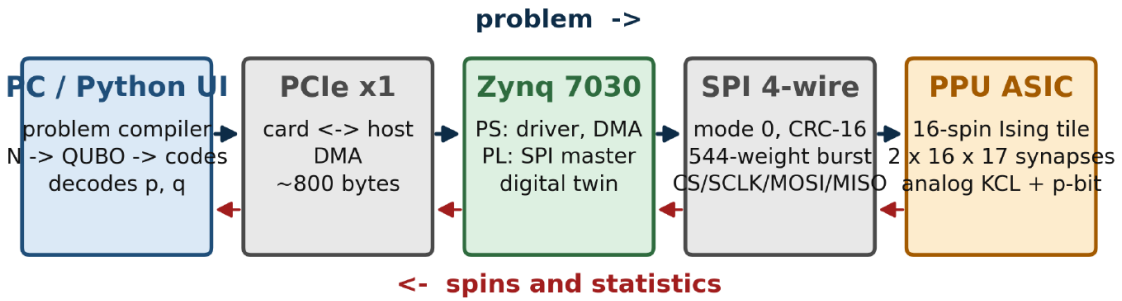
**ANALOG_SPEC §0 (decision dated 9 Aug 2026): "Because the eFLASH PDK is not ready, the bit array is built from standard CMOS transistors. The entropy source is not RTN (no eFLASH) and not an LFSR (true randomness is wanted): the transistors' own thermal noise."**

# Q2 · WHERE THE CHIP SITS, AND WHAT IS ON THE DIE

*Left: the board context as presented. Right: floor-plan-level block diagram of the standard-CMOS die (drawn)*

**PART A — the chip**

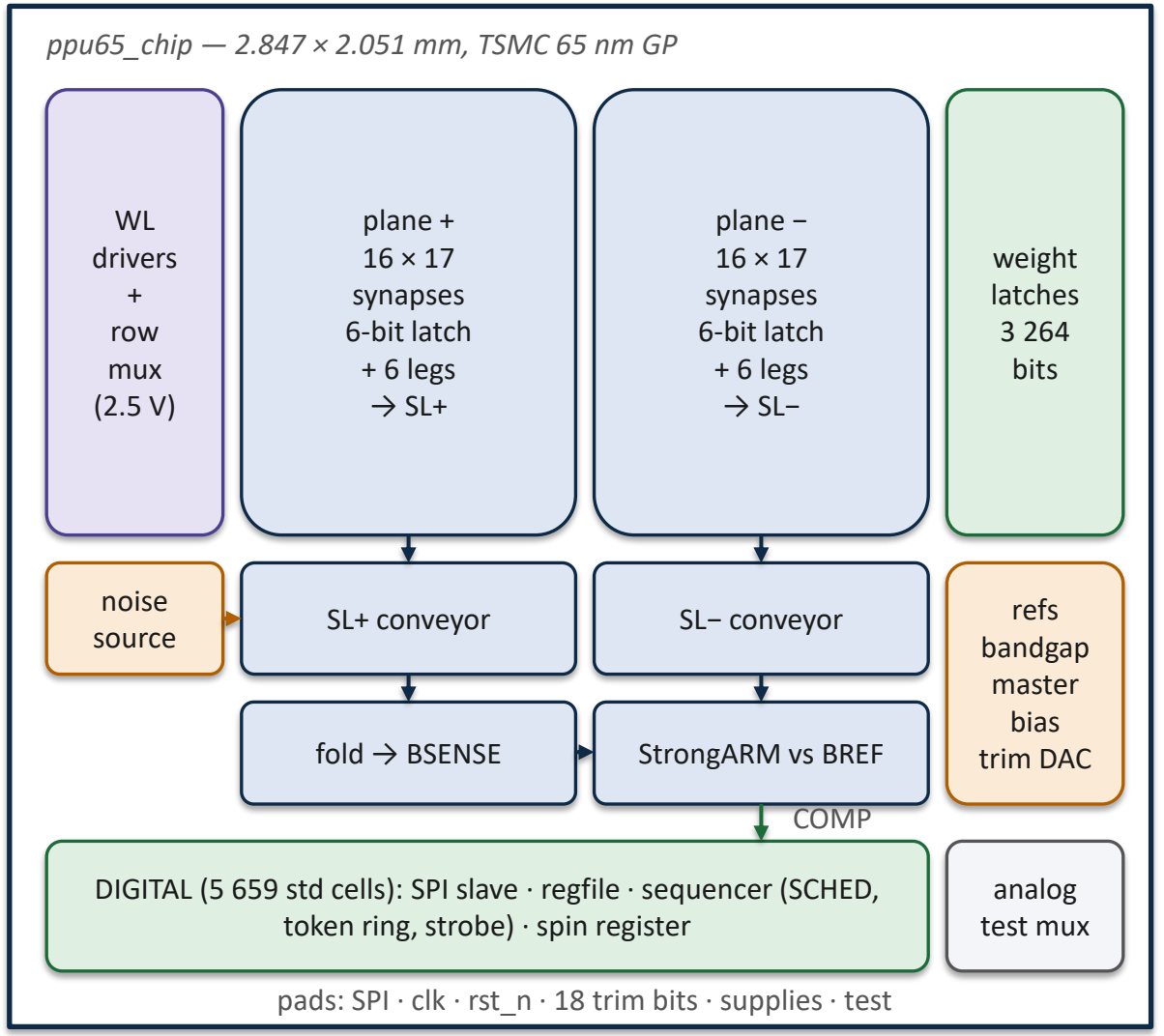
## WHERE THE CHIP SITS




**DIVISION OF LABOUR**

PCB, FPGA firmware, PCIe/DMA and the Python UI are built with the Antigravity toolchain.
The ASIC — array, SPI slave, register map, FSMs, timing, references — is the work reported here.
The interface contract between the two is the ICD + regmap.yaml golden source.

*deck: PC → PCIe → Zynq 7030 → SPI → PPU ASIC; only spins and statistics come back*




*no eFLASH cell anywhere: every weight is a latch + current legs*

# Q2 — ANSWER: A STANDARD-CMOS TEST ASIC, BUILT BEFORE THE eFLASH CELL EXISTS

*Yes: TSMC 65 nm GP, no eFLASH, weights in latches, entropy from a thermal-noise block*

**PART A — the chip**

- Technology: TSMC CRN65GPLUS (65 nm GP), 9 metal layers, wire-bond, die 2.847 × 2.051 mm
- 16-spin Ising / QUBO tile: 544 synapses = 2 planes (SL+, SL−) × 16 rows × 17 columns (16 couplings + 1 bias column)
- What "standard CMOS" changes, in practice:
  - weights are DIGITAL — a 6-bit code per synapse in ppu_wlatch, realised as six binary-weighted current legs (1:2:4:8:16:32); loaded once over SPI (Q5)
  - entropy cannot come from RTN of a floating gate (there is none) — a separate noise block had to be designed (Q1, Q3)
  - everything else — the shared-wire summation, the conveyors, the fold, the StrongARM comparator, the sequencer — is the architecture that a later eFLASH version keeps
- What it proves: the annealer architecture works with measured device models before the eFLASH weight cell exists; 3 test problems (ferro4, af4, factorization N = 77) compiled onto the same silicon

**So: a fully working standard-CMOS annealer, built to prove the architecture before the eFLASH weight cell exists.**

# WHY AN ANNEALER NEEDS NOISE AT ALL

*The score is a landscape; the chip rolls downhill; noise is the shaking that gets it out of the wrong valley*

**PART B — the noise**


- A greedy machine only ever rolls downhill. It stops in the first valley it meets — usually the wrong one (the deck's "why a deterministic machine is not enough").
- Noise plays the role of TEMPERATURE: it occasionally pushes a decision the "wrong" way, which is exactly what lets the state climb out of a shallow valley.
- Annealing = start HOT (explore widely), COOL gradually (commit), then QUENCH (freeze). The noise amplitude is the temperature knob; the SCHED table is the cooling programme.
- Measured on this problem (N = 77): with too little noise the success rate is 20 %; at A_max ≈ 40 µA and above it is 44–48 % per anneal (deck p59).

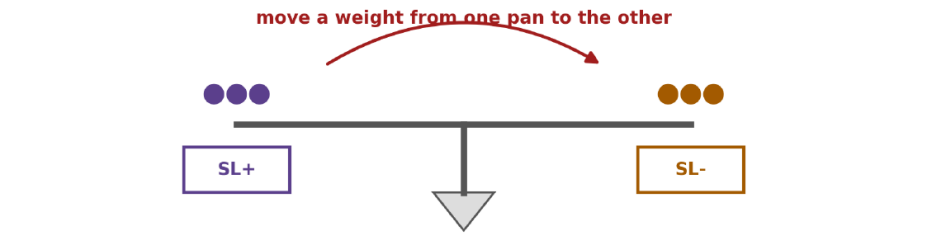
**No noise → no annealing. That is why the question "which noise source?" matters as much as the weights.**

# WHAT THE NOISE MUST DO — AND WHAT IT MUST NOT

*One rule shared by BOTH noise sources: move current between the two wires, never add current to them*



## WHAT THE NOISE HAS TO DO — AND WHAT IT MUST NOT




**the total must not change**

if the noise added current, it would look like extra weight and shift every decision.
If it only MOVES current between the two wires, the sum is untouched
and only the difference — the thing being judged — is randomised.

So the noise source is built as a set of legs that always conduct, and are merely steered left or right.

**That is why it is called a constant-sum steering DAC.**



*deck: a weight moved from one pan to the other — the total never changes*

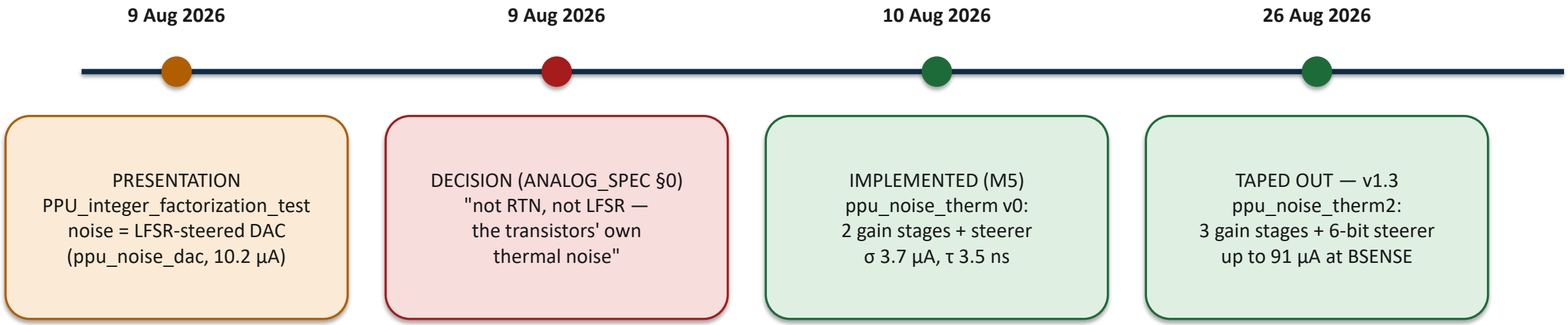
- The decision compares two currents: I(SL+) against I(SL−). Only their DIFFERENCE matters.
- If the noise ADDED current to a wire, it would look like extra weight and shift every decision in one direction — an offset, not a temperature.
- So the noise source is built as legs that ALWAYS conduct and are merely STEERED left or right: the sum I(SL+) + I(SL−) is untouched; only the difference is randomised.
- This is called a constant-sum steering DAC. The LFSR version and the thermal version both obey it — they differ only in WHAT decides the steering: a digital bit, or an analog noise voltage.
- Measured (LFSR DAC): total 10.2254 µA, constant to 0.003 % across every pattern. Measured (thermal chain, noiseless run): the DC level of BSENSE is 0.65606 V at every NCODE — the common mode does not move with the code.

# Q1 · TWO NOISE SOURCES IN THE HISTORY OF THIS CHIP — A TIMELINE

*The presentation and the taped-out silicon describe different noise sources; the change was decided the day the deck was written*

**PART B — the noise**

Why it matters for the questions: if you read the deck, the noise source is a Linear Feedback Shift Register driving six current legs. If you look at the silicon, the noise source is the amplified thermal noise of a transistor pair. Both are real design states of Version A — one presented, one manufactured.




**Proof the LFSR was not manufactured: ppu_lfsr.v exists in digital/rtl/ but is not instantiated in ppu_digital_top; the placed-and-routed netlist (38 404 cells) contains 0 LFSR cells.**

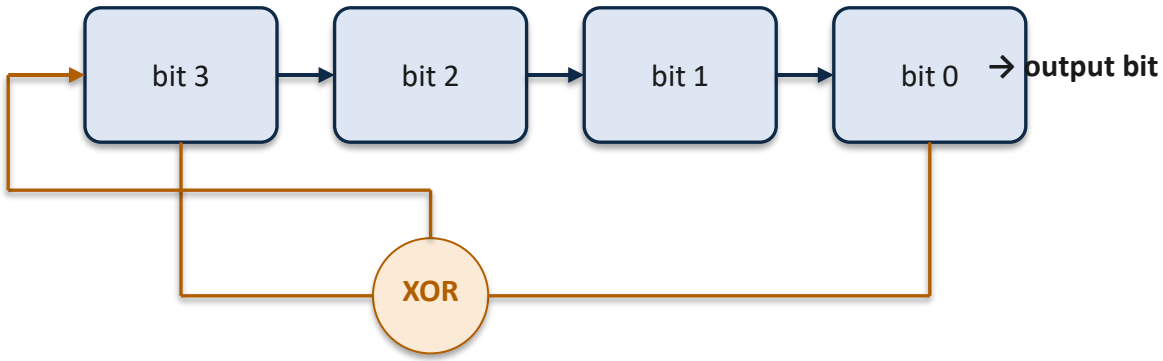
Sources: deck file date 9 Aug 20:05; git c976dbf / fcdd21c (10–11 Aug, M5 noise path); analog/netlists/ppu_noise_therm2.scs; DELIVERY.md (26 Aug).

# NOISE SOURCE #1 · WHAT AN LFSR IS

*Linear Feedback Shift Register — a digital circuit that produces a random-LOOKING but fully deterministic bit sequence*

**PART B — the noise**

*clock → every bit shifts one place; the vacated bit = XOR of chosen taps*




*example 4-bit Galois form; the chip's RTL (ppu_lfsr.v) is a 16-bit Galois LFSR*

- Deterministic: the same seed always gives the same sequence. Useful for tests — repeatable — but not "random" in the physical sense.
- Periodic: an n-bit LFSR repeats after at most $2^n - 1$ steps (65 535 for 16 bits).
- Linearly related bits: consecutive outputs are XOR combinations of earlier ones — correlations a physical noise source does not have.
- Cheap and synthesisable: a handful of flip-flops and XOR gates. That is why it was the first choice.
- In the deck: one LFSR bit per noise leg, six legs, pattern redrawn every slot; leg size (VBN) set by the SCHED code = temperature.

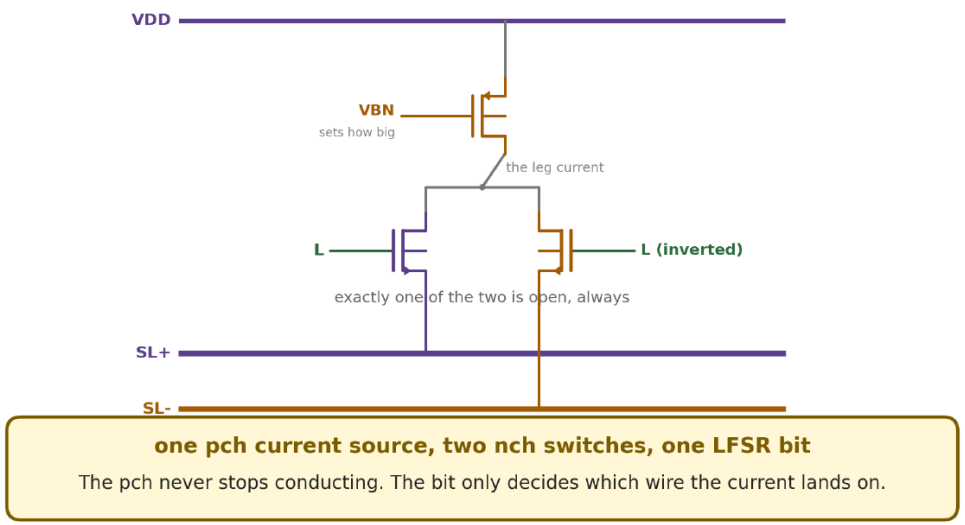
**Pseudo-random ≠ random. That distinction is the reason the LFSR was replaced (slide 13).**

# NOISE SOURCE #1 · ONE LEG OF THE LFSR DAC — ppu_noise_leg

*One pch current source, two nch switches, one LFSR bit: the current never stops, the bit only decides which wire it lands on*

**PART B — the noise**

## ONE LEG — ppu_noise_leg






*deck: exactly one of the two switches is open, always*

- VBN sets the size of the leg current — that is the temperature knob (SCHED code → VBN).
- L and L-inverted are one LFSR bit and its complement: exactly one nch switch conducts, so the leg current lands on SL+ or on SL−, never nowhere.
- Because the pch source never turns off, the SUM of what reaches the two wires is constant — the rule from slide 8.
- The leg does not know or care about the weights; it only adds a random ± push to the difference the comparator will judge.

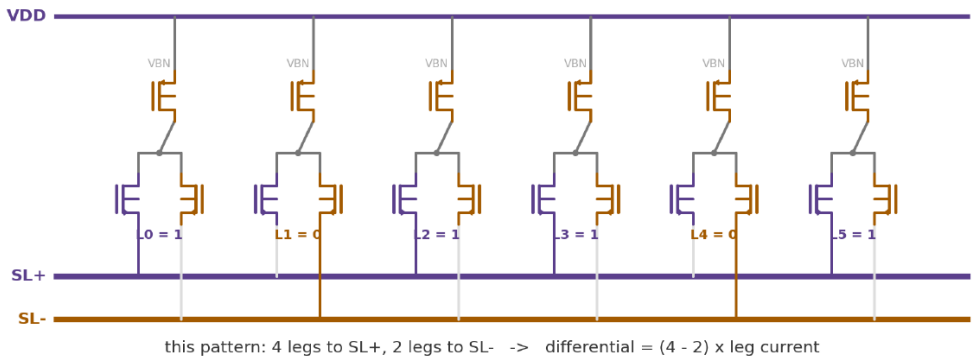
**One leg = one random push of fixed size. Six legs (next slide) give seven possible pushes.**

# NOISE SOURCE #1 · SIX LEGS TOGETHER — ppu_noise_dac

*k legs to SL+ and 6−k to SL− → differential = (2k − 6) × leg current: seven pushes −6, −4, −2, 0, +2, +4, +6*

**PART B — the noise**

## SIX LEGS TOGETHER — ppu_noise_dac




*deck: this pattern — 4 legs to SL+, 2 to SL− → +2 legs of push*

- Six independent LFSR bits → 64 patterns, but only the COUNT of ones matters: seven distinct pushes, binomially distributed (0 is the most likely, ±6 the rarest).
- "The LFSR redraws the pattern every slot, so the push is random and fresh each decision" (deck p77) → answers Q4 for this version: yes, per slot.
- Measured: 10.2254 µA total, constant to 0.003 % across every pattern — the constant-sum property holds.
- Amplitude range as built: up to ±10.2 µA differential. That number is the limit found by the factorization problem (next slide).

**Version A's LFSR DAC: 6 legs · 7 pushes · 10.2 µA full scale · redrawn every slot.**

# WHY THE LFSR DAC WAS DROPPED — TWO INDEPENDENT REASONS

*One is about the KIND of randomness, the other about the AMOUNT*

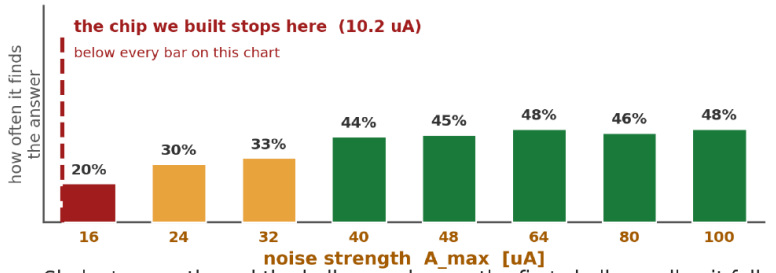
**PART B — the noise**

REASON 1 — it is not truly random

- An LFSR is a fixed arithmetic rule: same seed → same sequence, period $2^n - 1$, bits linearly related.
- For an annealer this means the "temperature" is a replayable script, not a physical bath: two runs with the same seed take the same path, and consecutive pushes are correlated by construction.
- ANALOG_SPEC §0, 9 Aug 2026: "not an LFSR (true randomness is wanted)".
- The replacement uses a physical process — the thermal motion of electrons in a transistor — which no seed can replay.

REASON 2 — it is too weak for hard problems

**THE NOISE KNOB: THE VERSION A CHIP CANNOT SHAKE HARD ENOUGH**




Shake too gently and the ball never leaves the first shallow valley it falls into.
This problem needs about 40 uA; the Version A silicon was built with only 10.2 uA,
because the earlier test problems were solved without any shaking at all.

**Factorization is the first problem that actually needs the noise — and it found the limit.**



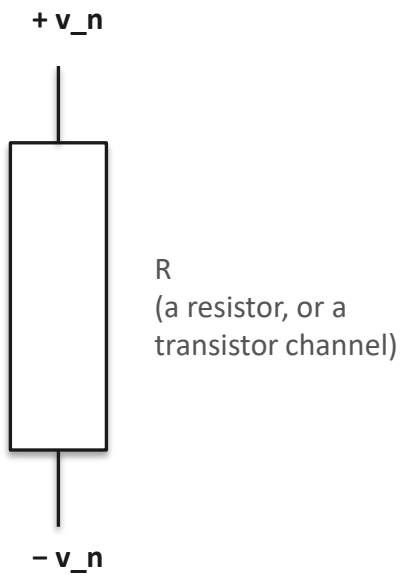
*deck: factorization needs A_max ≈ 40 µA; Version A's LFSR DAC stops at 10.2 µA — below every bar*

**Result: the thermal chain was specified to keep the SAME 6-bit amplitude control and the SAME cold floor, but with true randomness and a full scale ≥ 40 µA.**

# NOISE SOURCE #2 · WHAT THERMAL NOISE IS

*Every resistor and every transistor channel produces a tiny random voltage from the thermal motion of its electrons — no seed, no period, no script*

**PART B — the noise**




**v_n(t): a random waveform**

Johnson–Nyquist: $v_n^2 = 4\,k\,T\,R\,\Delta f$

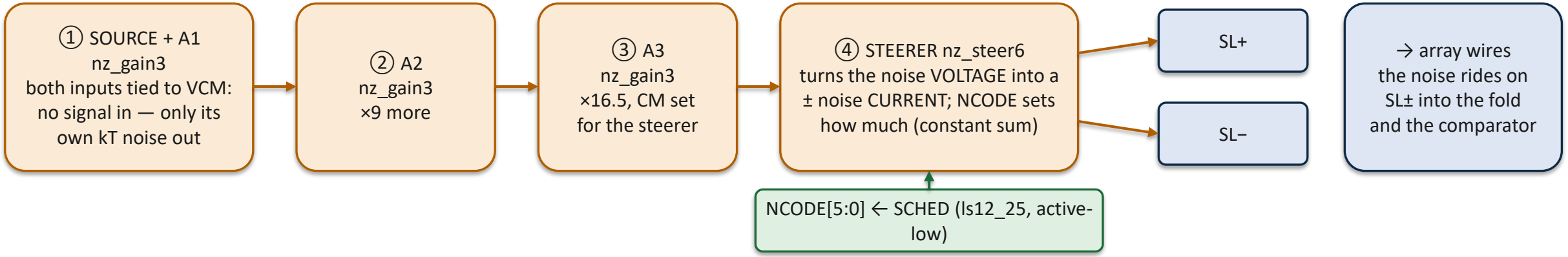
k = Boltzmann's constant · T = temperature · R = resistance · Δf = bandwidth

- Truly random: it comes from the thermal agitation of electrons — a physical process; no two runs are alike and nothing can replay it.
- White: flat spectrum far beyond the chip's bandwidth, so the amplifier's own band shaping decides its time constant (τ).
- Tiny: a few µV at the source. A decision needs tens of mV at BSENSE → it must be amplified by ~1000×. That is what the three gain stages do.
- Gaussian: a bell-shaped distribution — exactly the kind of "temperature" a Boltzmann-style annealer assumes (the p-bit sigmoid, slide 19).
- Its level does not depend on any code: a fixed physical amount that is then scaled by the steerer. So the "temperature knob" is a gain setting, not a change of the source.

# NOISE SOURCE #2 · THE THERMAL CHAIN, BLOCK BY BLOCK — ppu_noise_therm2 v1.3

*What each block does, in plain words (drawn from analog/netlists/ppu_noise_therm2.scs)*





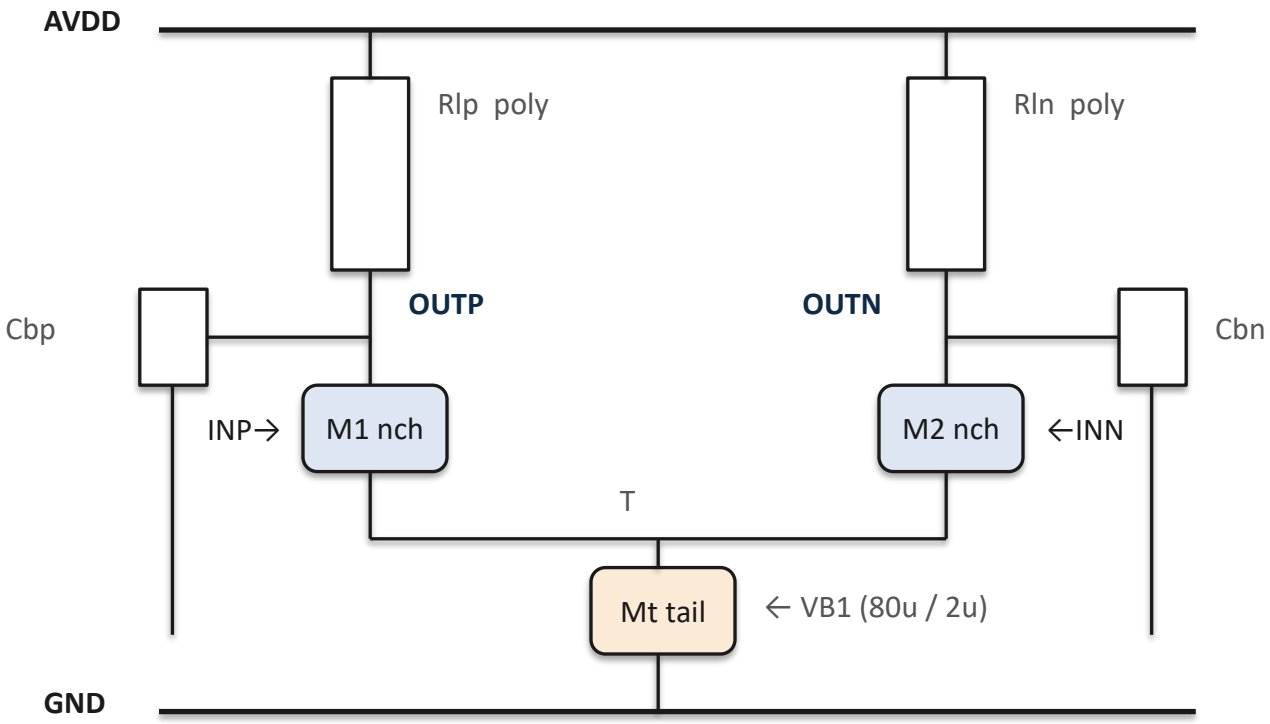

*Gain budget (design note in the netlist): 9 × 9 × 16.5 ≈ 1340×;  target σ at A3 ≈ 60 mV;  band shaping ≈ 4–6 MHz*

| Block | Cell | Inside it | Job |
|---|---|---|---|
| ① source + A1 | nz_gain3, rl = 200.5 µm poly | nch_25 pair 10u/0.28u, poly loads ≈ 76 k, 440 fF, long-L tail 80u/2u (VB1) | its own thermal noise is the entropy; first ×9 |
| ② A2 | nz_gain3, rl = 283.0 µm | same pair, loads ≈ 107 k, 290 fF | ×9 more, band shaping |
| ③ A3 | nz_gain3, rl = 344.0 µm | same pair, loads ≈ 130 k, 200 fF; output CM ≈ 0.55 V | ×16.5; sets the DC level the steerer needs |
| ④ steerer | nz_steer6 | pch_25 tails 80u/2u (VBT, fixed) · pch pair 80u/0.28u · 5 binary-weighted poly degeneration branches + 1 full-gm switch, pch switches 40u/0.28u (NC0B…NC5B) | noise voltage → ± current into SL±; code sets effective gm; sum constant |

# NOISE SOURCE #2 · INSIDE ONE GAIN STAGE — nz_gain3

*A differential pair with poly-resistor loads and a band-shaping capacitor; three of these in a row make ≈1340×*





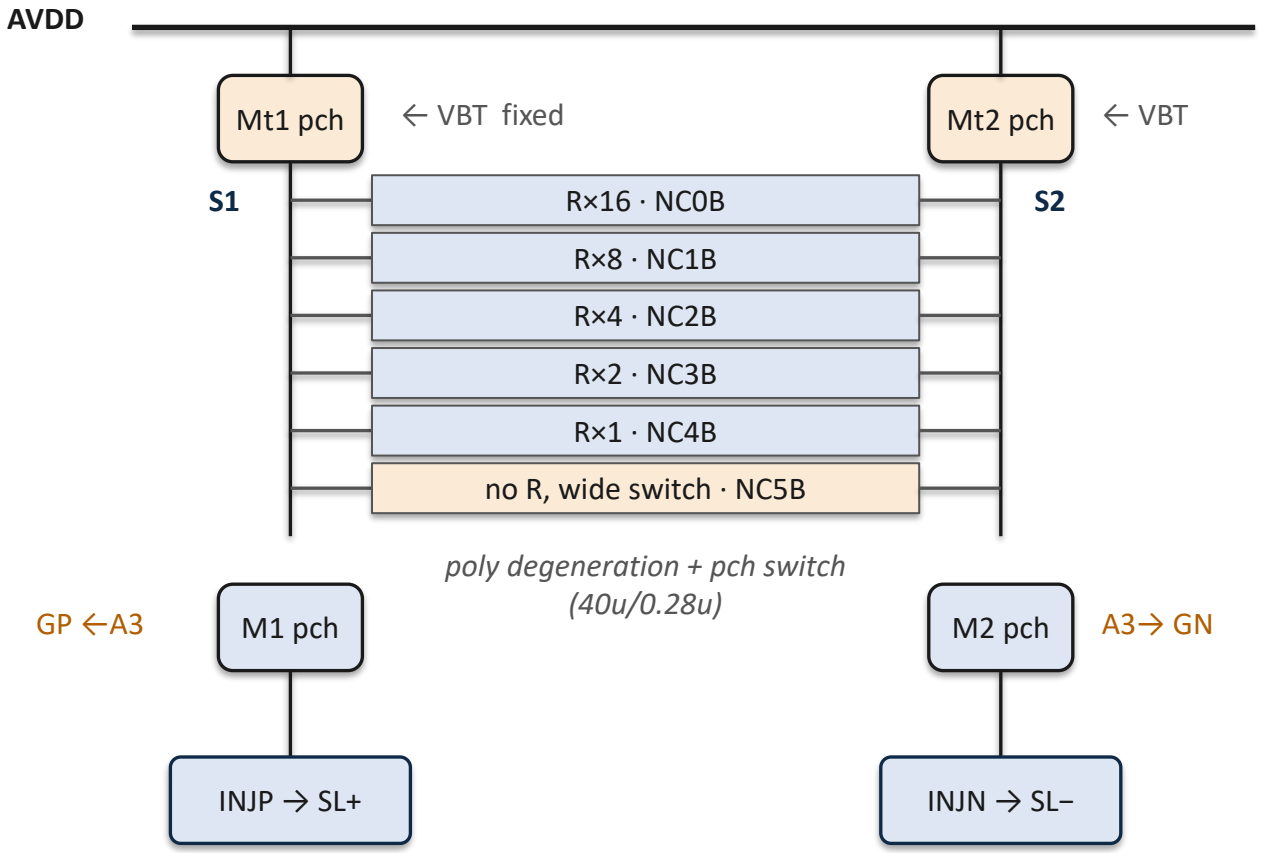

*Drawn from the netlist: M1/M2 nch_25 10 µm/0.28 µm · loads rppolywo 2 µm wide, length 200.5 / 283.0 / 344.0 µm (≈ 76 k / 107 k / 130 k) · Cb 440 / 290 / 200 fF · tail nch_25 80 µm/2 µm*

- In A1 both gates sit at the same DC voltage (VCM): there is NO input signal. What appears at OUTP−OUTN is purely the stage's own thermal noise (pair + loads), amplified by its gain. That is the entropy source.
- A2 and A3 take that differential noise and multiply it: ≈ 9 × 9 × 16.5 ≈ 1340× in total → ≈ 60 mV rms at A3.
- The load capacitors shape the band (≈ 4–6 MHz). This sets the noise's time constant τ — which is why consecutive slots can be correlated at some codes (slide 21).
- Poly loads, not ideal resistors: I × R tracks the master-bias current, so the DC levels stay put over process. A silent bug here (ideal 60 k in all three stages) was caught on 12 Aug and fixed.
- The long-L tail keeps the stage's common mode where the next block needs it; A3's output CM must be ≈ 0.55 V for the pch steerer to work.

# NOISE SOURCE #2 · INSIDE THE STEERER — nz_steer6, WHERE THE 6-BIT CODE ACTS

*Fixed tail currents; the code changes how strongly the noise voltage steers them — so the SUM never changes*

**PART B — the noise**




*Drawn from the netlist: tails pch_25 80 µm/2 µm on VBT · pair pch_25 80 µm/0.28 µm · branches Rd0…Rd4 = poly 84.66 / 42.33 / 21.16 / 10.58 / 5.29 µm (16:8:4:2:1) each with a pch_25 40 µm/0.28 µm switch · Ms5 200 µm wide, no resistor*

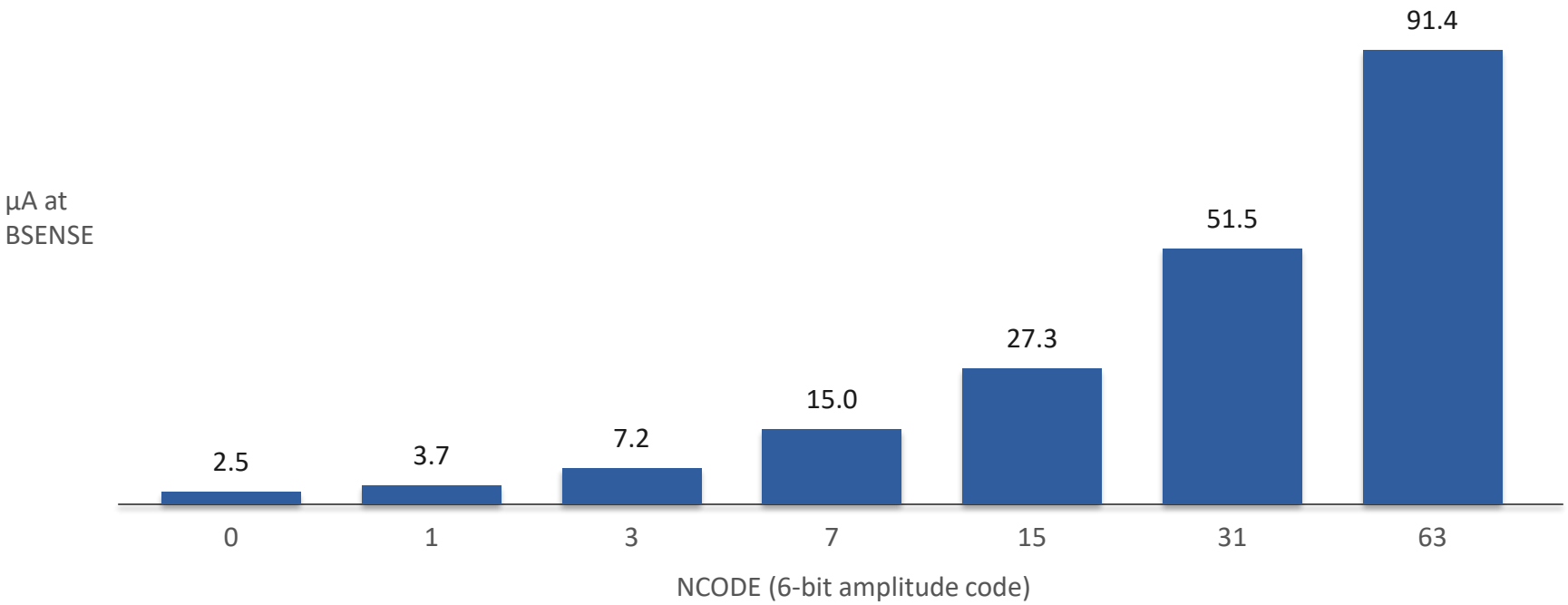
- The two tails Mt1/Mt2 carry FIXED currents (VBT). Whatever the code, the total current that reaches SL+ + SL− is the same → the common mode is code-independent (measured: BSENSE DC 0.65606 V at every code).
- M1/M2 are driven by A3's noise voltage. With the bridge OPEN they are two separate source-followers and the noise barely steers anything — that is the cold floor.
- Closing a bridge branch couples S1 and S2 through a resistor: the pair now works as a degenerated differential pair, and the noise voltage steers current from one side to the other. Smaller R → more steering. The five resistors are binary-weighted (16:8:4:2:1); the sixth branch is a wide switch with no resistor = full gm.
- So NCODE is a transconductance setting: it scales how much of the (fixed-size) noise voltage becomes a ± current. Six bits → 64 amplitude steps, ≈ 1.8× per bit.
- Polarity: the switches are pch, conducting at gate = 0 V. The chip's NCODE goes through ls12_25 whose complementary output drives NCkB. NCODE = 0 → all NCkB = 2.5 V → all branches open → MINIMUM noise (verified 16.6× ratio min→max).

# NOISE SOURCE #2 · HOW LOUD EACH CODE IS — MEASURED AT THE DECISION NODE

*Post-layout, real bias chain, static operating point; 3.6 mV at BSENSE = 1 µA of differential array current*

**PART B — the noise**

**σ of the noise at BSENSE, in µA equivalent, per NCODE**




- Monotonic, ≈ 1.8× per bit — the binary-weighted degeneration ladder does what it should.
- Cold floor (NCODE 0): 2.5 µA ≈ ⅛ of one weight unit — below the smallest field in the example deck, so a cold decision is close to deterministic.
- The factorization requirement (≥ 40 µA) is met between NCODE 15 and 31 — the limit of the LFSR DAC (10.2 µA) is passed at NCODE ≈ 7.
- Caveat (27 Aug): these are STATIC values. Measured in the tile at the decision instant, σ is ≈ 2.4× lower at NCODE 0 (5.0 mV, N = 24 seeds). Use the trend for the schedule, not the absolute numbers for a margin budget.

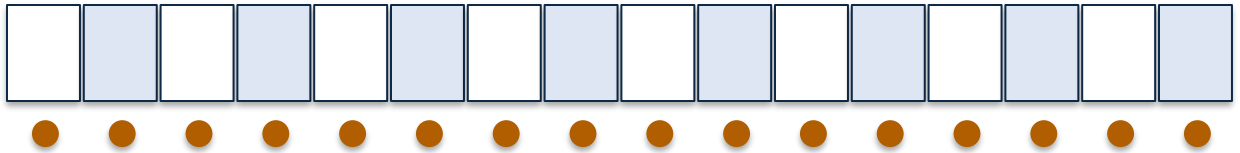
| NCODE | 0 | 1 | 3 | 7 | 15 | 31 | 63 |
|---|---|---|---|---|---|---|---|
| σ(BSENSE) mV | 9.1 | 13.2 | 26.0 | 54.0 | 98.2 | 185.4 | 328.9 |
| µA equivalent | 2.5 | 3.7 | 7.2 | 15.0 | 27.3 | 51.5 | 91.4 |
| in weight units (7.42 µA) | 0.34 | 0.49 | 0.97 | 2.02 | 3.67 | 6.94 | 12.3 |

# Q4 · WHEN THE NOISE CHANGES — EVERY SLOT vs EVERY SCHED STEP

*Two different clocks: the random VALUE is new in every slot; the AMPLITUDE (temperature) steps with the schedule*

**PART C — timing**

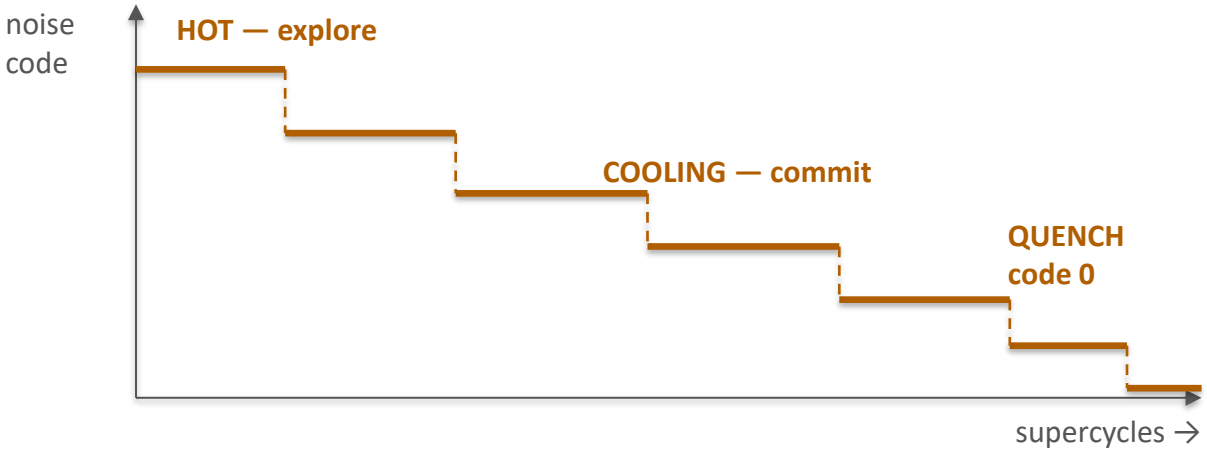
**one supercycle = 16 slots (one decision per slot)**



● = a fresh random push in every slot (LFSR redraw / physical noise)

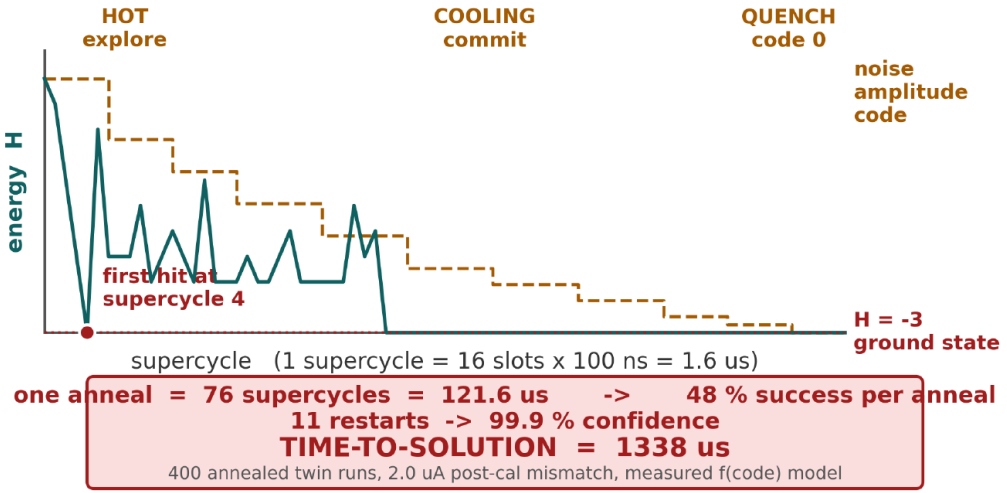
spin 0 · spin 1 · … · spin 15 → each new bit becomes a word line before the next decision (Gauss–Seidel)

**SCHED = {noise code, dwell}: the code holds for `dwell` supercycles, then steps**




*a step = one SCHED entry, `dwell` × 16 slots wide — the random push is renewed inside every one of those slots*

## THE ANNEAL, SUPERCYCLE BY SUPERCYCLE



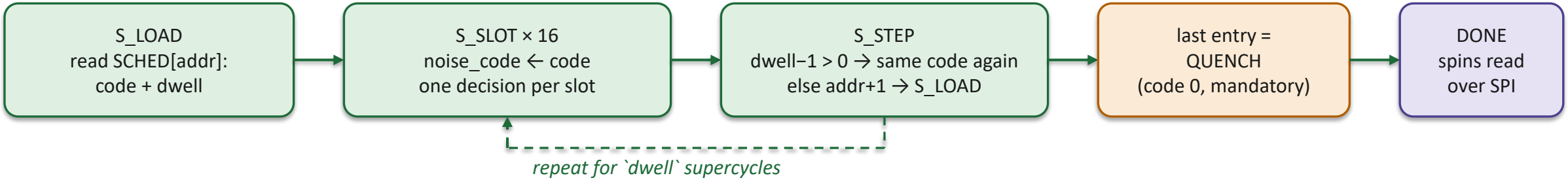

*deck: the anneal supercycle by supercycle — 76 supercycles = 121.6 µs, staircase from HOT to QUENCH*

**Value: every slot. Temperature: every SCHED step.**

# Q4 · HOW THE SEQUENCER STEPS THE TEMPERATURE — AND ONE MEASURED SUBTLETY

*ppu_sequencer.v: sched_data = {noise_code[5:0], dwell[9:0]} · consecutive slots are not perfectly independent at hot codes*

**PART C — timing**




So the amplitude changes only when the sequencer advances to the next SCHED entry — every `dwell` × 16 slots. Inside a dwell the code is constant, but the noise VALUE is new each slot: a thermal source produces a fresh sample continuously; an LFSR would have been redrawn each slot.

| NCODE | τ (ns) | r at 120 ns | consecutive-slot correlation | meaning |
|---|---|---|---|---|
| 3 | 31.8 | 0.023 | 2.3 % | fresh — decisions independent |
| 7 | 32.5 (8 seeds: 34.5 ± 0.9) | 0.025 | 2.5 % | fresh |
| 15 | 35.2 | 0.033 | 3.3 % | fresh |
| 31 | 82.6 | 0.234 | 23.4 % | partly shared between neighbouring slots |
| 63 | 72.1 (8 seeds: 63–140) | 0.189 | 18.9 % | partly shared — open item |

**Subtlety (measured 26–27 Aug): at the hottest codes τ rises to ≈ 80 ns, so at a 120 ns slot about ⅕ of a push carries over to the next decision. At cold/mid codes the pushes are independent. Fix candidates are a v2 design item.**

# Q4 — ANSWER: IS THE NOISE REFRESHED AT EVERY SPIN DECISION?

*Yes for the random value; no for the amplitude — and that separation is what makes it an anneal rather than a random walk*

**PART C — timing**

| | Presentation (LFSR DAC, 9 Aug) | Taped-out chip (thermal, 26 Aug) |
|---|---|---|
| Random value per decision? | YES — "the LFSR redraws the pattern every slot" | YES — a physical noise process is fresh at every strobe by nature |
| Independence between consecutive slots | by construction of the LFSR sequence (but linearly related bits) | measured: 2–3 % at NCODE ≤ 15; 19–23 % at NCODE 31/63 (τ ≈ 80 ns vs 120 ns slot) |
| Amplitude / temperature per decision? | NO — VBN follows the SCHED code | NO — NCODE follows the SCHED code, held for `dwell` supercycles |
| Order of updates | sequential (token ring) | sequential (token ring) — Gauss–Seidel |

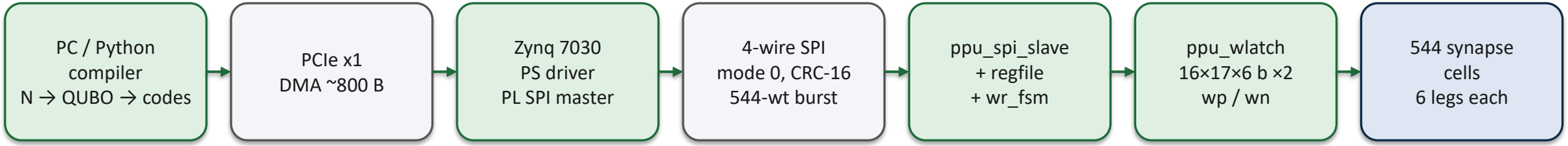
- Worked example in the deck: 76 supercycles = 121.6 µs at 100 ns/slot, staircase from HOT (explore) → COOLING (commit) → QUENCH (code 0).
- A new random push every slot + the same temperature for a whole dwell = simulated annealing in hardware. If the amplitude also changed every slot it would be a random walk with no cooling programme.

**Value: every slot. Temperature: every SCHED entry. Weights: never during a run.**

# Q5 · HOW A PROBLEM GETS INTO THE CHIP, AND WHERE IT STAYS

*Top: the write path, host to latch (drawn from the deck's board slide and the RTL). Bottom: the two tables that ARE the problem, as presented*

**PART D — the weights**




*written ONCE before RUN — then only the 16-bit spin register moves; readback = spin register only (SPI → PCIe → Python decodes p, q)*

## SO A WHOLE PROBLEM IS JUST TWO TABLES OF SMALL NUMBERS

**table for wire +**

| | p1 | p2 | q1 | q2 | q3 | w11 | w12 | w13 |
|---|---|---|---|---|---|---|---|---|
| p1 | | | 6 | 3 | 3 | | | |
| p2 | | | 3 | 6 | 3 | 3 | | |
| q1 | 6 | 3 | | | | | | |
| q2 | 3 | 6 | | | | 3 | | |
| q3 | 3 | 3 | | | | | 3 | |
| w11 | | 3 | | 3 | | | | |
| w12 | | | | | 3 | | | |
| w13 | | | | | | | | |

(the top-left corner of the real 16 x 17 table)

**table for wire -**

| | p1 | p2 | q1 | q2 | q3 | w11 | w12 | w13 |
|---|---|---|---|---|---|---|---|---|
| p1 | | | | | | 6 | 6 | 6 |
| p2 | | | | | | | | |
| q1 | | | | | | 6 | | |
| q2 | | | | | | | 6 | |
| q3 | | | | | | | | 6 |
| w11 | 6 | | 6 | | | | | |
| w12 | 6 | | | 6 | | | | |
| w13 | 6 | | | | 6 | | | |

(the top-left corner of the real 16 x 17 table)

Each square is one cell of the chip; the number in it is that cell's code.
Empty means code 0 — that cell contributes nothing.

**Loading a problem into the chip = writing these numbers in. Nothing else changes.**

*the same silicon solves any problem — only the tables differ*

row label = the spin this row decides  column label = the other spin it is coupled to



*deck: plane + and plane − tables (top-left of the real 16×17)*

## THE EXTRA 17th COLUMN: EACH SPIN'S OWN PREFERENCE

sixteen columns are the couplings to the other spins. The seventeenth is the bias a_i.

| spin | bias a_i | code on + | code on - | difference |
|---|---|---|---|---|
| **p1** | +1 | 9 (59.2 uA) | 8 (52.7 uA) | **+6.5 uA** |
| **w11** | +5 | 15 (97.7 uA) | 8 (52.7 uA) | **+45.0 uA** |
| **c3_0** | +9 | 21 (135.6 uA) | 8 (52.7 uA) | **+82.9 uA** |

**why is there a big number on BOTH sides?**

both planes carry the same fixed pedestal (code 8 = 52.7 uA) so that neither wire is ever empty.
Since only the DIFFERENCE matters, the pedestal cancels exactly and changes nothing.

**What survives is precisely the bias a_i.**

So the two tables together hold the whole problem: every coupling Q_ij and every bias a_i.



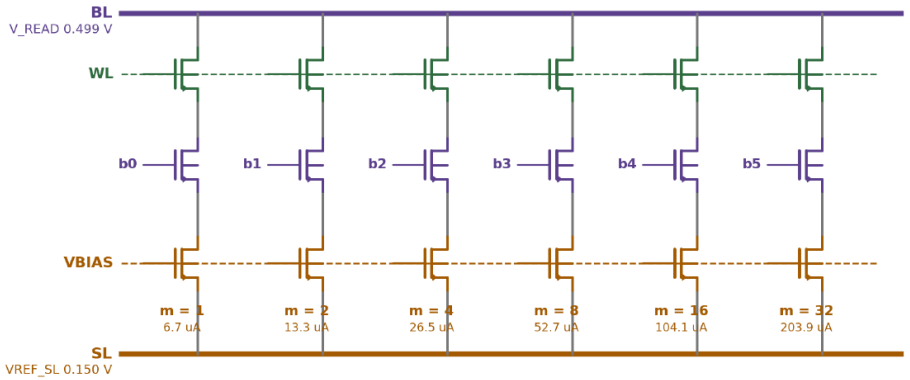
*deck: the 17th column — the bias a_i on a cancelling pedestal (code 8 on both planes)*

# Q5 · HOW A STORED WEIGHT BECOMES A CURRENT — ppu_synapse6

*Left: the real cell schematic. Right: the six doubling legs — choosing which legs are on IS the weight*

**PART D — the weights**

## BLOCK 1 — ppu_synapse6 : ONE WEIGHT BECOMES A CURRENT



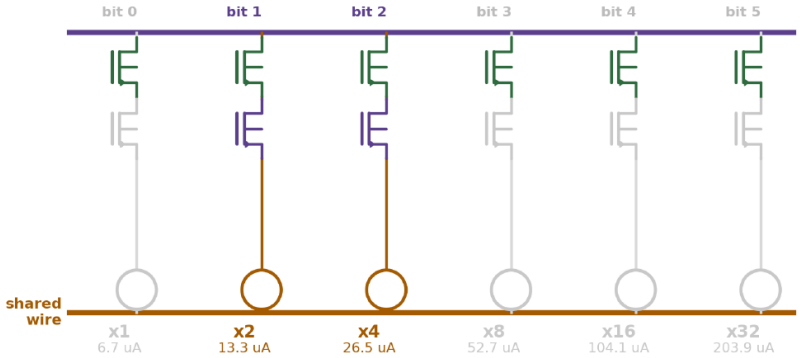

WL = spin bit x_j  b0..b5 = the 6-bit weight code  VBIAS sets the size of one unit

Three nch in series per leg: spin switch (1u/0.06u), weight-bit switch (1u/0.06u), current source (4u/1.0u).
Leg k is built m = 2^k times, so the six legs give 1 : 2 : 4 : 8 : 16 : 32 -> any code 0...63.
**cell current = x_j x code x 6.68 uA  (18 transistors, 544 cells on the chip)**

problem -> silicon - 2 / 9

*three nch per leg: spin switch (WL) · weight-bit switch (b_k) · current source (VBIAS); legs built ×1 … ×32*

## SIX LEGS WITH DOUBLING CURRENTS = ANY WEIGHT 0...63




**this example**
**code 6**
= 4 + 2
legs 2 and 1 are on
**39.9 uA**
flows into the wire

Six switchable currents in the ratio 1 : 2 : 4 : 8 : 16 : 32 can make ANY whole number of units from 0 to 63.
Choosing which legs are on IS writing the weight. That is what "6-bit weight" means.

**One cell = one number from the matrix, stored as six switches.**

how one cell holds a weight - 2 / 4

*code 6 = legs 2 and 1 on → 39.9 µA into the shared wire*

- Each leg is an AND of two switches — the spin bit x_j on the word line and one weight bit b_k from the latch — above a current source set by VBIAS. Cell current = x_j × code × one unit (6.68 µA measured; 7.42 µA post-layout).
- All cells of a row pour into the same wire (SL): Kirchhoff does the sum. No multiplier, no adder — 18 transistors per cell, 544 cells.

# Q5 — ANSWER: WEIGHTS ARE LOADED ONCE, HELD IN LATCHES, WRITTEN OVER SPI

*"Loading a problem into the chip = writing these numbers in. Nothing else changes."*

**PART D — the weights**

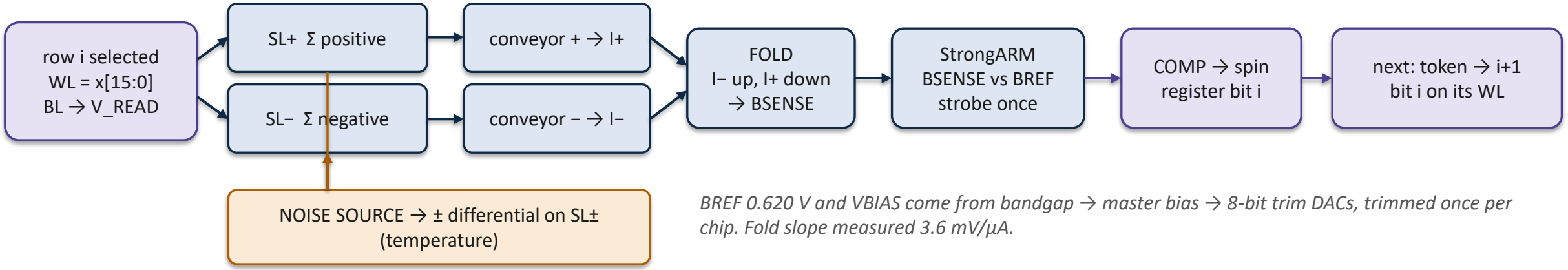
- Loaded ONCE before RUN and never touched during the anneal — only the 16 spin bits change.
- Where they come from: PC / Python compiler (N → QUBO → 6-bit codes) → PCIe x1 DMA (~800 B) → Zynq 7030 (PS driver, PL SPI master) → 4-wire SPI, mode 0, CRC-16, 544-weight burst → PPU SPI slave.
- Where they live on chip: ppu_wlatch — 16 rows × 17 columns × 6 bits × 2 planes = 3 264 latch bits. Plane + (wp) feeds SL+, plane − (wn) feeds SL−; column 17 is the bias a_i on a cancelling pedestal (code 8 = 52.7 µA on both planes).
- Write protocol (RTL): jwr_stb with jaddr_i, jaddr_j, jplane_p[5:0], jplane_m[5:0] for couplings; bwr_stb with bias_idx, bias_code[11:0] (6 + 6 bits) for biases.
- Read back: the 16-bit spin register only — SPI → PCIe → Python decodes p, q.
- Why once is enough: the weights ARE the problem; the annealer's job is to search the spins, not to learn the weights. A different problem = a different pair of 16×17 tables, same silicon.

**The same silicon solves any problem — only the two 16×17 tables differ. Nothing in the chip is specific to factorization.**

# EVERYTHING IN ONE SLOT — WHERE WEIGHTS, NOISE AND SPINS MEET

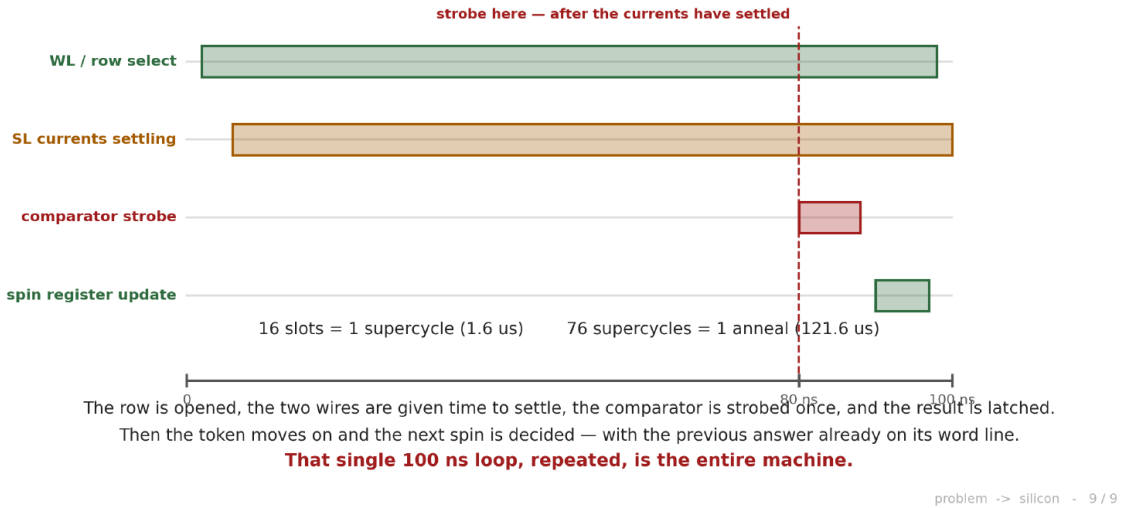
*Top: the decision chain (drawn). Bottom: the 100 ns slot timeline as presented*

**PART E — together**




*BREF 0.620 V and VBIAS come from bandgap → master bias → 8-bit trim DACs, trimmed once per chip. Fold slope measured 3.6 mV/µA.*

**BLOCK 8 — ONE 100 ns SLOT : THE LOOP CLOSED IN TIME**




*deck: WL/row select → SL settling → comparator strobe (+80 ns) → spin register update, in a 100 ns slot*

# SUMMARY — THE FIVE ANSWERS IN ONE TABLE

| Item | Presentation (9 Aug) | Taped-out chip (26 Aug) | Source |
|---|---|---|---|
| Technology | TSMC 65 nm GP, standard CMOS | same — CRN65GPLUS, 9M, 2.847 × 2.051 mm | deck p1; DELIVERY.md |
| Pre-eFLASH test ASIC? | yes | yes — "eFLASH PDK not ready" | ANALOG_SPEC §0 |
| Noise source | LFSR-steered constant-sum DAC, 6 legs | thermal noise of a transistor pair → 3 gain stages (≈1340×) → 6-bit constant-sum steerer | deck p75–77; ppu_noise_therm2.scs |
| Noise full scale | 10.2 µA (too small: needs ≥ 40) | ~91 µA equiv. at the decision node (static) | deck p59; sigma_bsense_dac.txt |
| Fresh value per spin? | yes (LFSR redraws every slot) | yes (physical); 2–3 % slot correlation at cold/mid codes, 19–23 % at hot codes | deck p77; noise_tau.txt |
| Temperature per spin? | no — per SCHED code | no — NCODE per SCHED entry × dwell | ppu_sequencer.v |
| Weights loaded | once, SPI burst, before RUN | same | deck p61; ppu_wlatch.v |
| Weight storage | 6-bit latches, 2 planes, 16×17 | same — 3 264 bits | ppu_wlatch.v |
| Unit current | 6.68 µA | 7.42 µA post-layout | deck p67; hwcal.py |
| Slot | 100 ns, strobe +80 ns | 12 cycles: 240 ns @ 50 MHz / 120 ns @ 100 MHz — clock undecided | ppu_sequencer.v |

Repository: github.com/ozgurkazanci/p-ASIC — docs/spec/ANALOG_SPEC.md, TAPEOUT_OPEN_ITEMS.md; analog/sim/*.txt hold the measurements quoted above.

# GLOSSARY — THE WORDS USED IN THIS DECK

| Term | Meaning here |
|---|---|
| LFSR | Linear Feedback Shift Register — shift register with XOR feedback; deterministic, periodic pseudo-random bits |
| thermal noise | random voltage from the thermal motion of electrons in a resistor or transistor channel (Johnson–Nyquist); truly random |
| constant-sum steering | the noise moves current between SL+ and SL− without changing their sum — a temperature, not an offset |
| SL+ / SL− | the two shared source lines; every cell of a plane pours its current into one of them (Kirchhoff sum) |
| BL / WL | bit line (row drive, V_READ) / word line (= the spin bit x_j that switches a cell on) |
| conveyor | ppu_vg_conveyor2 — holds SL at 0.150 V and copies its current out (I+, I−) |
| fold / BSENSE | two mirrors on one node: I− pushes it up, I+ pulls it down; its voltage is the decision variable (3.6 mV per µA) |
| BREF / StrongARM | the comparator's reference (0.620 V, trimmed) / the latched comparator strobed once per slot |
| NCODE / NCkB | 6-bit noise amplitude code from the SCHED table / its active-low, level-shifted form driving the steerer switches |
| SCHED · dwell · QUENCH | the annealing programme: a list of {code, dwell}; dwell = how many supercycles a code is held; final entry = code 0 |
| slot · supercycle | one spin decision (100 ns in the deck; 12 clock cycles as taped out) · 16 slots, one per spin |
| p-bit sigmoid | P(spin = 1) as a smooth function of the local field — the signature of a probabilistic bit |

# SOURCES AND DATES — WHERE EACH CLAIM COMES FROM

| What | File / measurement | Date |
|---|---|---|
| Circuit slides (synapse, legs, conveyor, fold, comparator, closed loop, slot, anneal, board) | PPU_integer_factorization_test (1).pptx / .pdf, pages 46–80 | 9 Aug 2026 |
| Decision: standard CMOS, thermal entropy, no LFSR | docs/spec/ANALOG_SPEC.md §0 | 9 Aug 2026 |
| Thermal chain topology (gain stages, steerer, device sizes) | analog/netlists/ppu_noise_therm2.scs (v1.1 header, shipped as v1.3) | 10–26 Aug 2026 |
| σ vs NCODE at the decision node; τ and slot correlation | analog/sim/sigma_bsense_dac.txt; analog/sim/noise_tau.txt | 26–27 Aug 2026 |
| p-bit sigmoid (hot/cold) | ANALOG_SPEC.md "M2 SONUCU"; analog/tb/tb_pbit_sigmoid.scs | Aug 2026 |
| Sequencer: SCHED {code, dwell}, quench, slot length | digital/rtl/ppu_sequencer.v | Aug 2026 |
| Weight latches and write protocol | digital/rtl/ppu_wlatch.v; digital/rtl/README.md | Aug 2026 |
| LFSR not shipped | digital/rtl/ppu_lfsr.v not instantiated; P&R netlist 0 LFSR cells | 26 Aug 2026 |
| Die size, delivery checksums | docs/spec/DELIVERY.md; docs/spec/SUBMISSION_DOSSIER.md | 26–27 Aug 2026 |
| This deck | generated 9 Sep 2026, all text ≥ 12 pt, rendered and checked in PowerPoint | 9 Sep 2026 |