Fast and Accurate Weight Loading for Analog Crossbars: A Resource-Consistent Simulation Study

Naman Boggaram

***Abstract*—Loading an analog crossbar requires coordinating shared programming pulses, noisy readback, and repair of cells disturbed by later writes. We study terminal tile-loading time using a public-data-anchored Ag–Cu/silicon device model and compare grouped first writes, disturbance compensation, Kalman-based repair, and selective verification. A frozen-policy evaluation contains 8,640 simulated chip–load observations across three related model variants, two workloads, and four operating regimes. Consistently charging every sensed cell value to a single read lane substantially changes the apparent benefit of the combined method. On a 32-chip confirmation cohort for a 32×32 image target, the combined loader reduces model-operation time from 4.6122 to 4.1452 s relative to an aggressive verification-only baseline, but the totals become approximately 4.6445 and 4.6384 s after separately measured first-load Python controller proxies are added. A safer disturbance-compensated grouping policy reduces operation time by 22.8% relative to the safer grouped baseline and lowers mean true conductance RMS error from 0.44865 to 0.34454 μS. Its comparison gates pass at the point-estimate level in all 12 confirmation image cases, with quality intervals supporting all margins in 10 cases. These results establish a conditional latency–accuracy tradeoff and expose computation, resource accounting, and stopping criteria as essential evaluation dimensions; finite-control and hidden-plant validation remain open.**

***Index Terms*—Analog in-memory computing, crossbar programming, memristor, selective verification, state estimation, loading latency.**

I. Introduction

An analog crossbar implements a matrix through physical conductances. Before a new matrix can be used, a controller must translate its values into device targets, issue programming commands, and determine whether the stored state is usable. Programming is imperfect: pulses have nonlinear, variable effects, measurements are noisy, and a write can perturb previously programmed cells. Consequently, the loading cost is the time of an entire programming-and-verification schedule, including its difficult tail, rather than the average time of an isolated cell.

Shared control creates a tension between first-write precision and scheduling cost. More individualized commands can reduce initial error while increasing the number of pulse groups. Conversely, a coarse shared pulse can leave a few difficult cells that force repeated verification of a column. When sensing is expensive relative to programming, a controller can rationally spend additional pulse groups to avoid readback. The best policy therefore depends on read bandwidth, pulse setup, disturbance, stopping rules, and the computation required to plan each new target.

We investigate this tradeoff computationally, using a surrogate anchored to public pulse trajectories and array data associated with Ag–Cu/silicon crossbars [1]. The hardware literature motivates the array scale and the need for closed-loop programming [2], but does not determine all of the benchmark's timing, uncertainty, or scheduling parameters. We distinguish published evidence, fitted quantities, architectural assumptions, and simulator outputs throughout the study.

The evaluated controller family combines disturbance-compensated grouping (GC), within-load Kalman-based model-predictive repair (KMPC), and selective verification. Aggressive verification (AV) skips some confirmation reads; safer verification (SV) adds margins and observation-staleness limits. These are empirical control heuristics. In particular, a confidence margin used to reuse a measurement does not constitute a statistical certificate of complete-array correctness.

The contributions of this draft are the following:

- A resource-consistent analysis that charges ordinary verification, cleanup, and final acceptance to the same read budget, with operation counts separated from controller computation.
- A frozen-policy comparison demonstrating how a substantial operation-time advantage can nearly disappear against a stronger baseline after fresh-matrix planning is charged.
- A quantified latency–accuracy tradeoff for disturbance-compensated grouping with safer verification, together with stress-case uncertainty and unresolved differential-workload failures.

The empirical contribution is a diagnostic evaluation of implemented policies, not a claim of universal programming superiority. Some runs terminate with residual final-read violations; their times are reported without treating them as successful complete loads. Demonstrating a deployable advantage under a specified whole-tile readiness requirement remains a further step.

II. Background and Related Work

A. Physical Evidence and Model Scope

Yeon *et al.* demonstrate alloyed conducting channels and a 32×32 transistor-less crossbar [1]. Their source workbooks

Working research draft, November 19, 2026. Results are computational and conditional on the declared models. No hardware loading-time measurement or formal whole-array correctness guarantee is claimed.

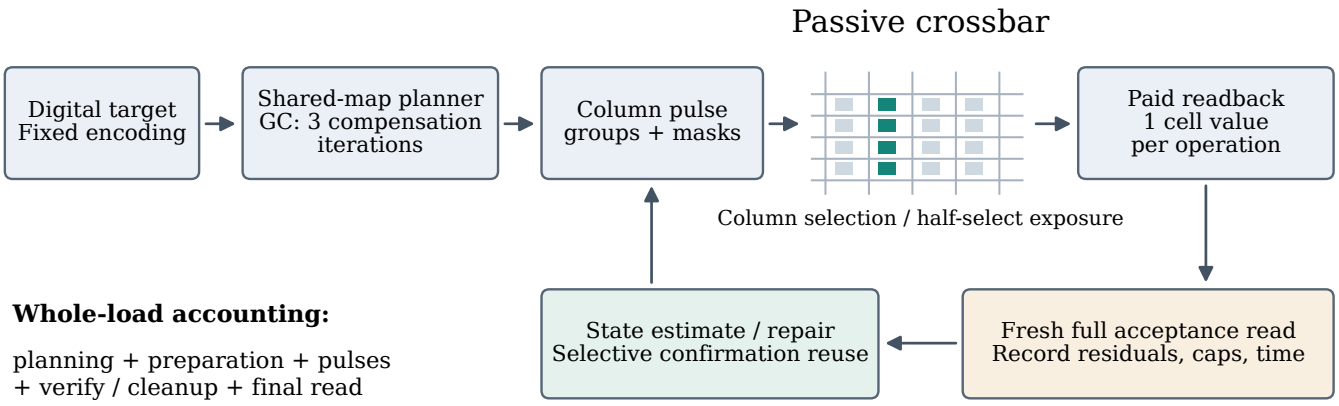

Fig. 1. Controller and accounting boundary. A digital target is mapped to shared pulse groups; readback supports repair and optional confirmation-read reuse. A fresh full acceptance read is retained. The schematic is conceptual and does not imply measured circuitry or simultaneous conversion resources.

supply repeated pulse trajectories and post-tuning array maps. In this project, those records constrain a response surrogate; retention maps are not first-write residual logs, and the trajectory records do not identify all array cells across subsequent targets. Thus, they do not by themselves establish reusable per-cell calibration transfer.

Choi *et al.* describe a stackable system with a memristor computing core and closed-loop conductance programming [2]. The main Methods identify configurable I/O and current-collection circuitry. Physical channel inventory is useful provenance, but does not establish the effective verification conversion rate assumed here. Their stack reconfiguration uses preprogrammed functional layers and is not a measurement of fresh-matrix reload latency on one unchanged array. The single-lane scenario below is an explicit benchmark architecture, not an inferred limitation of that board.

B. Programming and Feedback Alternatives

Schroedter *et al.* study a model-based time-modulated write algorithm for 1R analog crossbars [3]. This is close prior art for using a response model to choose a command. Our analysis emphasizes the interaction of grouped schedules, sensing occupancy, and fresh-target computation. A compatible implementation of their policy has not yet been evaluated in the present harness; the numerical results therefore do not establish superiority over that method.

History-based programming appears in the tri-mode solver of Yu *et al.* [4]. Neural-network inversion of desired conductance updates into pulse times is explored by Yu *et al.* [5]. Such methods motivate stronger alternatives to an ordinary feedback loop. They also require accounting for what history or training is available and what it costs. Our headline KMPC policy estimates state during the current load; it does not learn a persistent sensitivity for each cell across future targets.

Song *et al.* exploit self-limited analog programming for parallel array updates [6]. That device-dependent mechanism illustrates why algorithm comparisons must preserve electrical capabilities. A policy requiring different switching behavior or circuitry should be treated as a separate architecture comparison. The present draft makes no first-of-its-kind claim for model inversion, grouping, estimation, or selective readback individually. A final method claim requires an explicit comparison with the closest compatible published controls.

III. Problem Formulation and Accounting

A. Targets, Encoding, and Readiness

Let $T \in \mathbb{R}^{R\times C}$ be a conductance target on active cell set $\mathcal{A}$, and G the true programmed state. The controller observes noisy readouts Y, not G. For a signed weight represented by two cells,

$$\widehat{w}_i = \frac{G_i^+ - G_i^-}{S}, \qquad e_{w,i} = \frac{e_i^+ - e_i^-}{S}, \tag{1}$$

where S is the fixed conductance scale and $e_i^\pm = G_i^\pm - T_i^\pm$. Placement and encoding are shared across compared policies.

For $M = |\mathcal{A}|$ active cells, the evaluation records

$$E_G = \sqrt{M^{-1}\sum_{i\in\mathcal{A}}(G_i - T_i)^2}, \tag{2}$$

$$q_G = M^{-1}\sum_{i\in\mathcal{A}} \mathbf{1}\{|G_i - T_i| \le \tau\}, \tag{3}$$

$$f_Y = M^{-1}\sum_{i\in\mathcal{A}} \mathbf{1}\{|Y_i^{\mathrm{final}} - T_i| > \tau\}. \tag{4}$$

The reported conductance error is the mean of per-chip E_G, not a pooled cell RMS. The quantity f_Y is a final-read *cell violation fraction*; it is not the fraction of failed tiles. For differential weights, normalized weight RMS is computed separately from (1).

A deployment objective would minimize full loading time subject to an absolute readiness rule, for example a weight-error threshold and a minimum probability of whole-tile success. The archived policies instead run bounded repair and acceptance loops and return residuals. Their terminal times remain informative, but do not establish time-to-success under such an absolute rule. In particular, q_G near one and a small

relative increase in f_Y do not prove that every required cell is acceptable.

B. Resource-Consistent Time

Figure 1 fixes the accounting boundary. For a serial one-lane scenario, operation time is

$$T_{\mathrm{op}} = N_g t_g + N_r t_r, \tag{5}$$

where N_g includes preparation and all executed pulse groups, and N_r includes main verification, cleanup, and every final acceptance pass. The assumed group cost is $t_g = 1.05\,\mu\mathrm{s}$, consisting of a nominal $50\,\mathrm{ns}$ pulse and $1\,\mu\mathrm{s}$ setup. The read cost is $t_r = 1.01\,\mathrm{ms}$, consisting of an assumed $1\,\mathrm{ms}$ read and $10\,\mu\mathrm{s}$ settle. These are scenario costs, not measured board throughput.

The historical engine charged a full acceptance pass by active rows, although ordinary verification was charged by sensed cell values. If N_a^{row} is the historical acceptance-row count and C_a is the active column count, the corrected count is

$$N_r = N_r^{\mathrm{legacy}} + (C_a - 1)N_a^{\mathrm{row}}. \tag{6}$$

Thus, a full 32×32 pass costs 1,024 read operations rather than 32; an 18×4 kernel pass costs 72 rather than 18. Every policy receives the same correction. The calculation preserves the archived execution order and read-noise draws; it does not resimulate elapsed-time-dependent device evolution.

For sensitivity analysis, let $C_{\mathrm{plan},j}$ and $C_{\mathrm{recur},j}$ be separately measured Python CPU proxies for sample j. At proxy multiplier κ and n identical-target loads sharing one plan,

$$\widetilde{T}(n,\kappa) = \overline{T}_{\mathrm{op}} + \kappa\,\mathrm{median}_j\left(C_{\mathrm{recur},j} + \frac{C_{\mathrm{plan},j}}{n}\right). \tag{7}$$

The primary fresh-matrix case is $n = 1$. Equation (7) adds a sensitivity proxy to an operation-time mean; it is neither a per-chip wall-clock measurement nor an embedded-controller prediction. No overlap is credited. Read lanes, bus overlap, and actual preparation would require an executable hardware resource schedule.

IV. Device Surrogate and Loading Policies

A. Response and Observation Models

The mean pulse response follows the archived fitted model. For conductance g, positive strength s, gain k_c, and RESET count n since the polarity switch,

$$\Delta g_+ = s k_c k_+ (G_s - g), \tag{8}$$

$$\Delta g_- = -s k_c \left(1 - e^{-n/\tau_d}\right)\left[a_d e^{b_d(g-15)} + c_d\right]. \tag{9}$$

Conductance is expressed in μS. Additive write noise and multiplicative cell/load variation perturb this response, with a conductance floor of $0.5\,\mu$S. The reference v2 model omits the RESET-onset factor; a steep variant increases the RESET exponential prefactor. These variants share a modeling lineage and are not independent device technologies.

The observation model includes approximate line-resistance voltage drop, level-dependent Gaussian read noise, and quantization with step $0.0503\,\mu$S. Nominal read SD is $0.30\,\mu$S at the reference level; the array read model differs from the trajectory-fit noise parameter. Preparation samples a starting state around the fitted g_{start} with SD $0.30\,\mu$S and charges 50 pulse groups. This is a modeled initialization protocol, not a measured reset sequence. Partially selected cells receive a deterministic mean response scaled by assumed factor h and command strength; $h = 0.003$ in the base scenario. This factor is not a measured voltage-to-response law.

Figure 2 plots public trajectories and the frozen mean-map replay, without refitting. The saved validation record reports held-out trajectory-cycle RMS statistic $0.79695\,\mu$S, inside its simulated 95% predictive interval $[0.73147, 0.89783]\,\mu$S. However, the held-out mean-trajectory statistic $0.54186\,\mu$S exceeds its interval $[0.34423, 0.48418]\,\mu$S, and held-out spread $0.67479\,\mu$S is below $[0.72455, 0.90019]\,\mu$S [7]. These mismatches preclude describing the surrogate as a fully validated digital twin.

B. Grouped First Writes and Disturbance Compensation

P2 uses a shared response map to assign first-write doses to groups, then applies the reference feedback and cleanup loop. It uses $K = 8$ first-write codes for the image and $K = 2$ for the kernel workload. P1, the earlier ordinary-feedback baseline, is an approximation to the published programming practice, not an exact reproduction of unavailable loop details. We retain P2 as the stronger main reference; historical P1 comparisons do not determine the paper's headline.

GC sorts targets within each column and polarity and partitions them into at most K contiguous groups. For sorted effective targets $u_1, \ldots, u_m$ and attainable mean levels $\mathcal{L}$, let

$$C(a,b) = \min_{\ell \in \mathcal{L}} \sum_{i=a}^{b} c(\ell, u_i), \tag{10}$$

$$D(k,b) = \min_{a \le b}\{D(k-1, a-1) + C(a,b)\}. \tag{11}$$

The image policy uses squared error $c(\ell, u) = (\ell - u)^2$. A kernel variant uses an approximate first-read miss-probability cost. The search grid has 1,500 positive dose values up to 80 standard-pulse equivalents. Dynamic programming is exact for this restricted grid and contiguous-partition objective, not for full loading latency.

Starting from $T_{\mathrm{eff}}^{(0)} = T$, the planner computes codes $c^{(q)}$, replays the entire first-write schedule through a noise-free shared model F, and updates written targets by

$$T_{\mathrm{eff}}^{(q+1)} = T_{\mathrm{eff}}^{(q)} + T - F(c^{(q)}). \tag{12}$$

Three iterations precede final code assignment. GC uses $K = 16$ for the image and $K = 4$ for kernels, and writes all first-write columns before verification. The image write band is $0.6\,\mu$S around the prepared mean. Consequently, GC versus P2 changes code budget, grouping objective, compensation, write band, and schedule together. It is a policy comparison, not an isolated causal ablation of disturbance compensation.

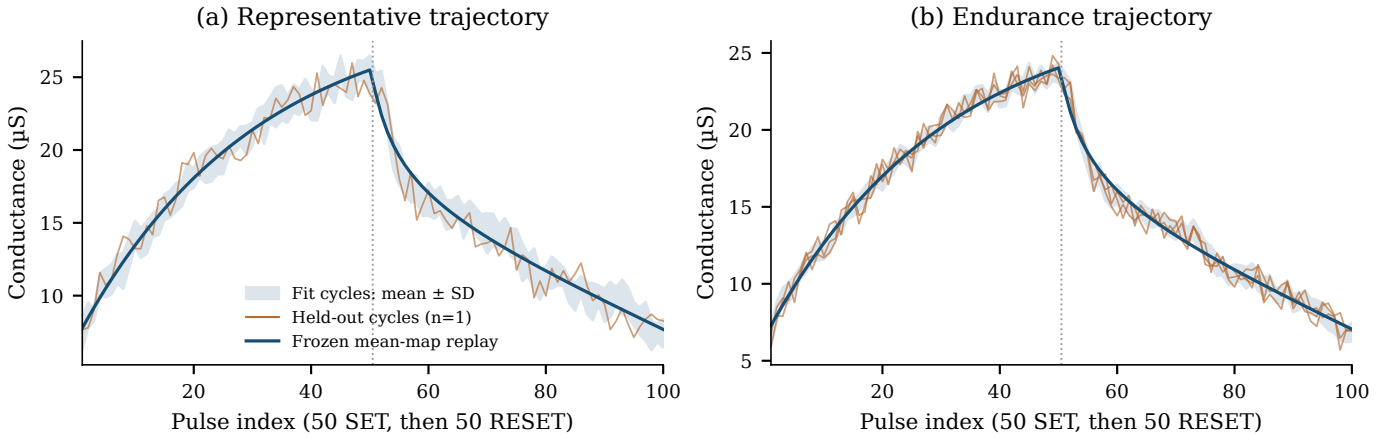

Fig. 2. Public-data model anchor. Gray bands show the fit-cycle mean ± one SD; orange curves show the held-out cycles (one representative and three endurance cycles); blue curves replay each frozen v3 mean map from its fitted start. The public records contain 100 post-pulse reads per cycle; pulse zero is absent. These descriptive overlays are not predictive confidence bands or new device measurements. Data: Yeon *et al.* [1], processed by the project.

C. State Estimation and Selective Verification

KMPC tracks per-cell mean μ and variance P during a load, predicting selected and half-selected changes. For linearized read gain H, measurement variance R_y, and predicted read $\widehat{y}$, its scalar update is

$$K_f = \frac{PH}{H^2P + R_y}, \tag{13}$$

$$\mu^+ = \mu + K_f(y - \widehat{y}), \tag{14}$$

$$P^+ = (1 - K_f H)P. \tag{15}$$

The repair dose is obtained by inversion of the shared response map at the posterior mean. The controller knows the matched response and uncertainty parameters but not the true simulated conductances or cell-specific gains. Current repairs allow effectively continuous strength values rounded numerically; finite shared waveform validation is incomplete.

Selective verification rereads recently pulsed cells and may reuse cached observations to avoid a full-column confirmation. A cached value v_i can be reused if

$$|v_i - T_i| \le \tau - z\sigma_i - c_h m_i, \qquad n_i \le n_{\max}, \tag{16}$$

where m_i is accumulated model-predicted disturbance magnitude and n_i counts half-select events since its last read. A final fresh full acceptance read remains mandatory. Equation (16) is an empirical rule: Gaussian margins, multiple comparisons, correlated disturbance, and model error are not converted into a whole-tile probability bound.

For images, AV uses $z = 0.5$, $c_h = 0$, and no event limit. Frozen SV settings are $(z, c_h, n_{\max}) = (2, 4, 2)$ for P2 and $(1, 0, 2)$ for GC. These are different frozen operating points, so "safer" does not imply an identical formal risk budget across methods. The kernel P2 SV setting is $(0.25, 2, 8)$; no frozen GC SV kernel setting was available, and none was selected using confirmation results.

Algorithm 1 Evaluated GC with selective verification (outline)

Require: Target T, active mask, shared model, tolerance τ
Compute GC codes using (11) and three updates (12)
Prepare array; charge preparation groups
Execute grouped first writes over all active columns
for each active column **do**
  Read column; update cached observations
  while repair is required and cycle cap is not reached **do**
    Apply grouped repairs; track half-select exposure
    Read pulsed cells and update observations
    Skip confirmation only if (16) passes
  end while
end for
Perform a fresh full-array acceptance read
Run at most two cleanup rounds, each followed by acceptance
Return time, read violations, and cap counts
Evaluator separately records hidden true-state metrics

V. Experimental Protocol

A. Workloads, Cohorts, and Regimes

The image workload contains 1,024 conductance targets, using $T = 2.5 + 0.044 M_0^{\mathsf{T}}\ \mu$S from the initial public image map. The factor $0.044\,\mu$S per pixel is an approximate project conversion. The kernel workload contains 36 normalized signed weights from four textbook 3×3 kernels, mapped to 72 physical conductances on an 18×4 region with $S = 11.22\,\mu$S in (1). Kernel coefficients are project assumptions, not recovered measured programming targets. Figure 3 visualizes the target specifications, not post-programmed results.

The saved stage-one protocol fixes 16 historical chips (IDs 2000–2015), 32 confirmation chips (4000–4031), load seed 11, and all policies before the reported runs [8]. Across three model variants, two targets, four regimes, and the available

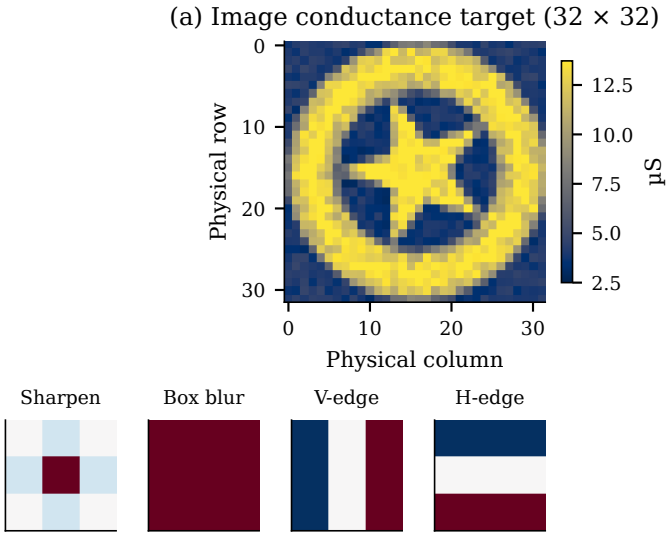

Fig. 3. Workload specifications. (a) Conductance target derived from the public initial image map. (b) Normalized textbook kernel weights; blue/red denote opposite signs with a fixed $[-1, 1]$ color scale. Image loading evaluates conductance accuracy; differential kernel loading also evaluates encoded-weight error.

policies, the evaluation contains 360 cohort-specific configurations and 8,640 chip–load observations. The confirmation chips test frozen settings; now that their results have been examined, they are not an untouched reserve for later tuning.

Base parameters are $\tau = 1.25\,\mu$S, reference read SD $0.30\,\mu$S, half-select factor 0.003, and stable log-gain SD 0.02. Stress conditions change one parameter at a time: $h = 0.01$, $\tau = 0.75\,\mu$S, or read SD $0.60\,\mu$S. The repair cap is 50 iterations per column invocation, with at most two cleanup rounds and a final acceptance pass. Read-violation fractions and cap counts remain in the records; no success-only filtering is applied.

B. Statistics and Comparison Gates

Operation-time ratios are ratios of arithmetic means on paired chip IDs. The archived analysis uses 2,000 chip-level bootstrap resamples, seed 20261117, and the 2.5th/97.5th percentiles. The quality comparison allows a final-read cell-violation increase of at most one percentage point, mean per-chip true RMS ratio at most 1.05, and true in-band fraction decrease at most 0.1 percentage point. A time win requires ratio ≤ 0.95, bootstrap upper endpoint below one, and passing quality point gates.

The stronger interval check requires each quality interval to lie within its respective margin. These are marginal diagnostic intervals, without familywise adjustment across metrics or scenarios. Pairing controls chip identity; the batch-seeded simulator does not guarantee identical exogenous write/read noise paths across different policies or cohort sizes. Results at one load seed consequently do not quantify full load-to-load stochastic variation.

CPU measurements use six historical chips (2000–2005), batch size one, and separate single-thread execution. Physical-plant simulation time is excluded, but model evaluations inside GC planning remain charged to controller computation. We reuse those recorded proxies rather than infer hardware latency. The prior reproduction record checked 70 configurations and 1,005 numerical/categorical comparisons with no failed comparisons; the stage-one record reconciled 180 historical checks and preserved 1,432 archive files [8], [9]. For this draft, figures and tables were regenerated from the saved records; the 8,640 simulation observations were not rerun.

VI. Results

A. Read Accounting and the Stronger Baseline

Table I and Fig. 4 summarize the reference v3 image case on 32 confirmation chips. Ordinary P2 requires 7.9632 s under consistent one-lane accounting. AV alone reduces that to 4.6122 s, showing that a strong verification policy accounts for much of the attainable operation-time gain. The combined GC+KMPC+AV method reaches 4.1452 s; relative to P2+AV its operation-time ratio is 0.898751, with paired-bootstrap interval $[0.893780, 0.904236]$.

The combined policy saves $466.98\,\mathrm{ms}$ of modeled operations relative to P2+AV, but its additional first-load controller proxy is $460.87\,\mathrm{ms}$. The resulting totals are 4.6384 versus 4.6445 s, a difference of approximately $6.1\,\mathrm{ms}$. This difference is too small to support a practical timing advantage from a Python-to-hardware proxy. The appropriate base-case interpretation is an approximate latency tie with lower simulated true RMS for the combined policy, 0.35766 versus 0.45765 μS.

The combined loader also spends more pulse groups: approximately 3,836 versus 2,255 for P2+AV, while reducing read operations from approximately 4,564 to 4,100. Under (5), one read costs about 962 pulse groups. The benefit is therefore dominated by saved reads in this cost scenario. Reducing read cost or increasing pulse setup can change the ranking even if the schedule and final errors are unchanged.

B. Safer Grouping: Accuracy and Stress Results

GC+SV requires 4.4276 s versus 5.7336 s for P2+SV: ratio 0.7722 and bootstrap interval $[0.7625, 0.7824]$. Its true RMS is 0.34454 rather than 0.44865 μS, a 23.2% reduction. With first-load CPU proxies, the time ratio becomes approximately 0.8570, corresponding to a 14.3% reduction. The comparison is against the frozen safer baseline; it does not identify the fastest admissible policy at every possible absolute quality threshold.

At the same condition, GC+SV totals 4.9715 s, approximately 7.0% slower than P2+AV after computation. It achieves lower true error and a higher true in-band fraction. Thus, its utility depends on the required accuracy and robustness operating point. The fact that AV and SV both pass some relative comparison gates does not make their stored-matrix quality identical.

Across the 12 confirmation image cases, GC+SV passes all quality point gates and the time-win rule versus P2+SV. Quality intervals support all margins in 10 cases; the v3 and steep tight-tolerance cases remain uncertain (Fig. 5). All three

TABLE I

BASE V3 IMAGE WORKLOAD, 32 CONFIRMATION CHIPS. TIMES ARE SCENARIO ESTIMATES; THE COMBINED COLUMN ADDS THE SEPARATELY MEASURED FIRST-LOAD CPU PROXY. TRUE METRICS USE EVALUATOR-ONLY SIMULATED STATES. AV: AGGRESSIVE VERIFICATION; SV: SAFER VERIFICATION. FINAL-READ VIOLATIONS ARE CELL FRACTIONS, NOT FAILED-TILE RATES.

Policy	Model time (s)	+ CPU proxy (s)	True RMS (μS)	True in band (%)	Final-read violations (%)
P2	7.9632	7.9938	0.45087	99.8291	1.8005
P2 + AV	4.6122	4.6445	0.45765	99.8474	2.0996
P2 + SV	5.7336	5.8008	0.44865	99.9512	1.7609
GC + SV	4.4276	4.9715	0.34454	99.9939	1.0681
GC + KMPC + AV	4.1452	4.6384	0.35766	99.9420	1.4282

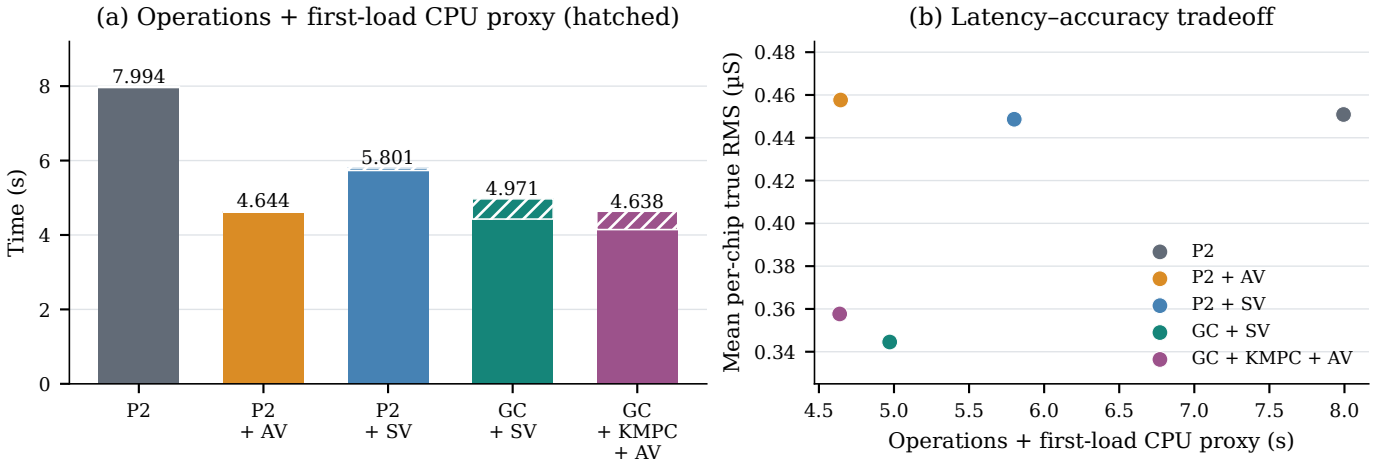

Fig. 4. Base v3 image comparison. (a) Consistent model-operation time plus the median first-load CPU proxy; hatched segments are computation. (b) Mean true RMS versus the combined sensitivity estimate, where lower values on both axes are desirable. Each operation/error mean uses 32 chips; CPU proxies use six separate historical chip runs. These are point estimates, not hardware timing measurements.

base-case operation-time ratios lie between 0.772 and 0.801. Much larger gains in tight/noisy settings partly reflect slow or capped baseline loops, so they are not used as the headline performance claim.

The aggressive combined method does not inherit the safer policy's stress behavior: against P2+SV, it fails the stricter quality comparison in the new v3 high-disturbance and tight-tolerance cases. Combining mechanisms is therefore not a uniform robustness improvement. Because the policies change several controls together, the present comparison also does not isolate each component's independent causal contribution.

C. Tail Behavior and Differential Kernels

Figure 6 shows all terminal times for the base workloads, including runs with residual read violations. For images, the empirical model-time 95th percentiles are 8.6432 s for P2, 4.7131 s for P2+AV, 4.5358 s for GC+SV, and 4.1491 s for the combined policy. No base v3 run for these policies hits the repair cap. These are small-cohort empirical quantiles; they are not population tail guarantees or success-conditioned completion-time estimates.

For base v3 kernels, P2 takes 0.33530 s and P2+SV takes 0.29405 s. Their normalized encoded-weight RMS values are 0.04427 and 0.04347, respectively. GC+AV takes 0.29601 s with weight RMS 0.04248. These values characterize the specified differential mapping; image conductance improvements do not substitute for signed-weight validation.

P2+SV passes quality point gates relative to P2 in 11 of the 12 confirmation kernel cases. In the v2 tight-tolerance case, its final-read cell violations increase by 1.171875 percentage points, its true RMS ratio is 1.06418, and its true in-band fraction decreases by 0.30382 percentage points, exceeding the allowed margins. The historical kernel cohort also contains a mismatch. Therefore, the image findings do not establish generalization to arbitrary signed-weight workloads.

D. Computation, Reuse, and Calibration

Figure 7 separates two deployment assumptions. At $n = 1$, each fresh matrix pays its own target-dependent plan. At $n = 16$ and the recorded proxy scale, the combined policy's per-load estimate becomes approximately 4.282 s versus 4.636 s for P2+AV. This conditional benefit applies only when identical targets can reuse a plan; it cannot be credited to a stream of unseen matrices. Larger proxy multipliers make GC increasingly expensive.

Reusable calibration is a separate hypothesis from within-load state estimation. In an ideal additive model, let stable cell bias b have variance V_b, and a noisy reference estimate $\widehat{b} = b + \epsilon$ have independent noise variance V_{ref}. A shrinkage correction $\alpha\widehat{b}$ leaves

$$\mathrm{Var}(b - \alpha\widehat{b}) = (1-\alpha)^2 V_b + \alpha^2 V_{\mathrm{ref}}, \tag{17}$$

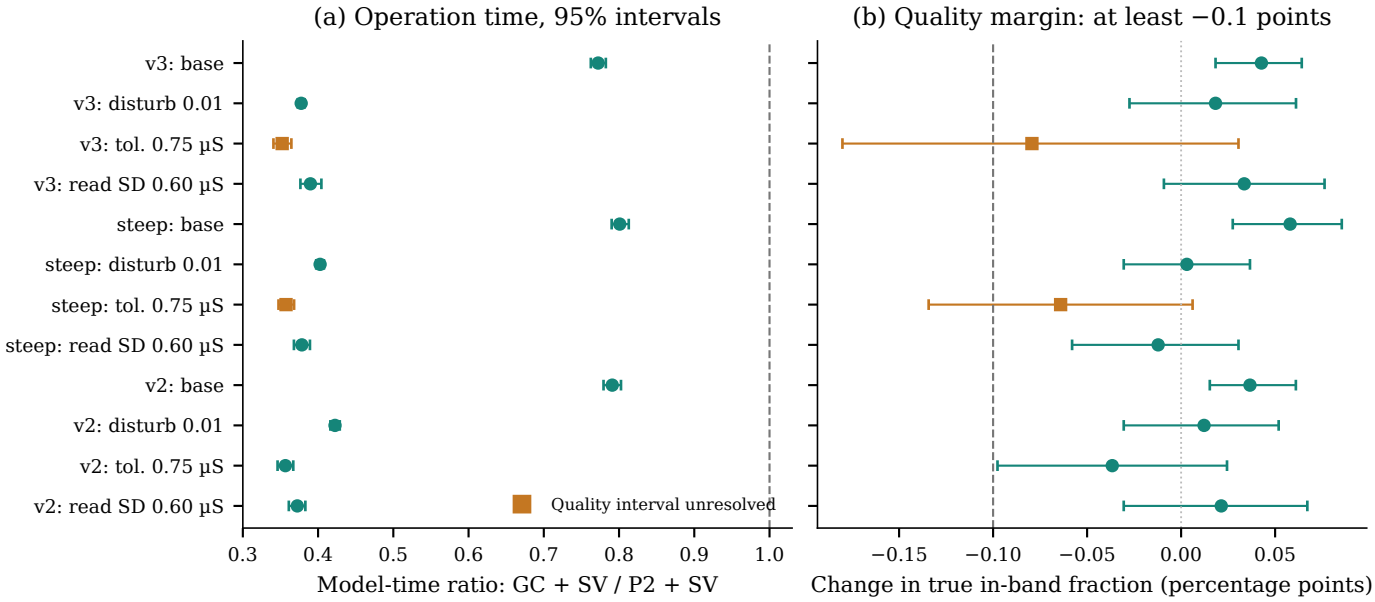

Fig. 5. GC+SV versus P2+SV on all 12 confirmation image cases. (a) Mean operation-time ratios. (b) Change in mean true in-band fraction; the dashed line is its allowed deterioration margin. Bars are paired-bootstrap 95% intervals over 32 chips. Squares flag cases in which at least one quality interval does not support its margin, despite passing point gates. The v3 tight-tolerance case also has an unresolved final-read violation margin. Computation is excluded from the time ratios.

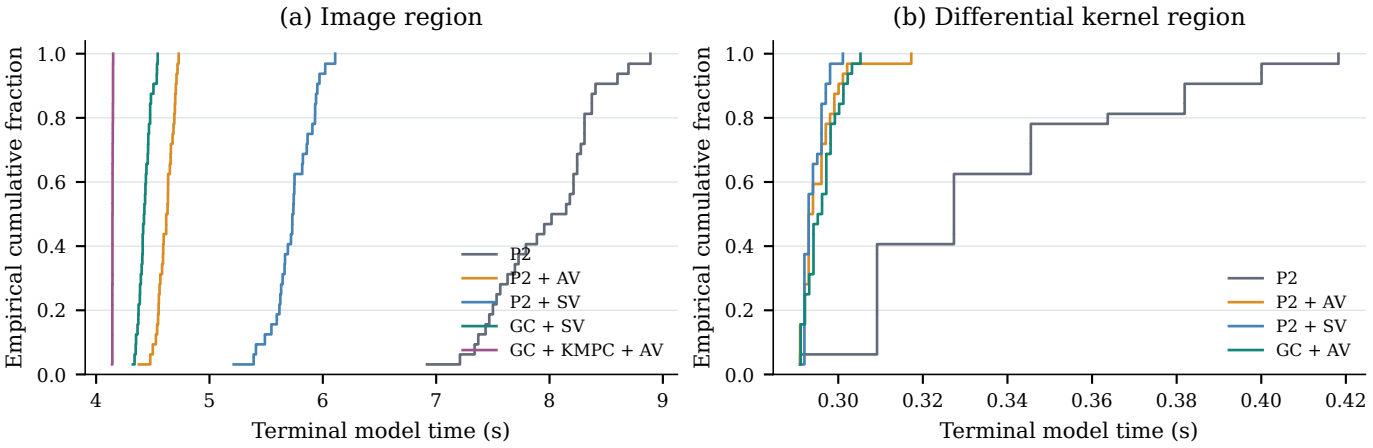

Fig. 6. Empirical terminal model-time distributions over 32 confirmation chips at the base v3 condition. (a) Image; (b) 72-cell differential kernel region. All recorded runs remain in the distributions. Different method availability reflects frozen policies; a GC+SV kernel policy was not evaluated. The curves do not imply all-cell acceptance or include CPU proxies.

minimized at $\alpha^* = V_b/(V_b + V_{\mathrm{ref}})$. Full correction helps squared error only if $V_b > V_{\mathrm{ref}}$. This elementary calculation explains why noisy calibration can lose; it does not establish scheduling or loading-time gains. The existing base-variability calibration sweeps do not justify making persistent calibration the primary solution.

VII. Limitations and Required Validation

The strongest remaining uncertainties concern realizable controls and model knowledge. Repair strengths are effectively continuous; finite DAC/width choices and shared-driver constraints have not been fully imposed on every compared policy. The controller and plant remain matched in response family and relevant parameters. Testing new chips within the same surrogate does not test hidden structural mismatch, incorrect half-select response, correlated noise, or state-dependent relaxation.

The one-lane recosting is consistent but architectural. It preserves archived batching/cleanup behavior, and does not model retention as a function of the enlarged elapsed time. Preparation counts do not verify electrical feasibility. Most importantly, relative quality gates do not replace an application-level absolute requirement, and a nonzero final-read violation fraction prevents interpreting every terminal run as a successful complete load. Future evaluation must report tile failures, timeout probability, retries-to-success, and weight-error thresholds

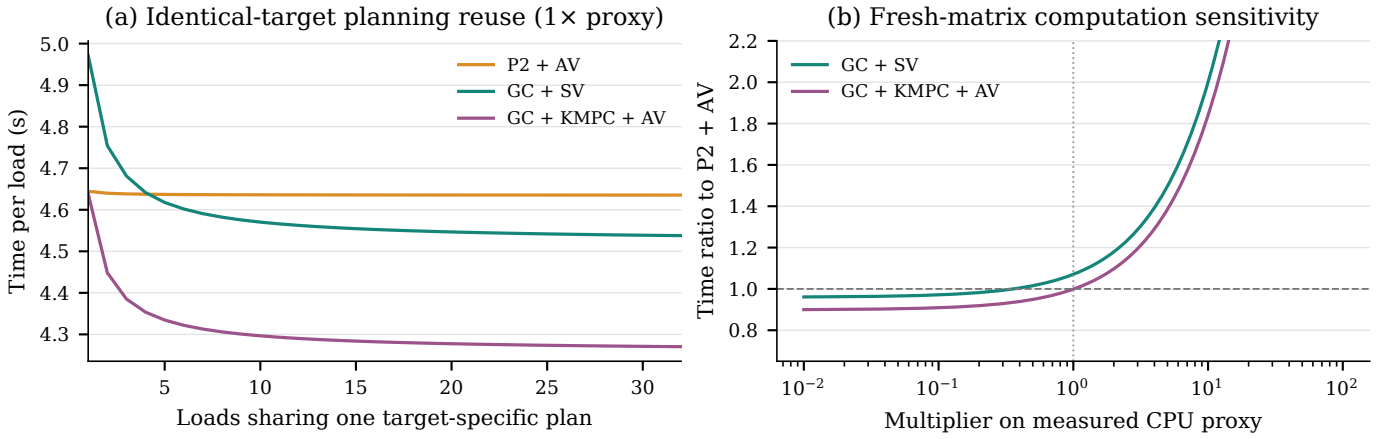

Fig. 7. Computation sensitivity at the base v3 image condition. (a) Equation (7) for identical-target plan reuse, using the median of adjusted sample costs. (b) Fresh-target time ratio versus a multiplier on the measured Python CPU proxy, with P2+AV as reference. Operations and read noise are held fixed; the sweep is a cost sensitivity calculation, not an embedded implementation or new device experiment.

under one explicit readiness rule.

The evaluation has one load seed, two fixed target families, and 32 confirmation chips. Policy development already examined these results; a new final reserve is required after further optimization. Confidence intervals are marginal, CPU proxies lack a hardware calibration, and the strongest close published model-based and learned controls are not yet implemented under identical resources. These gaps limit the draft's scope to a diagnostic conditional study.

The next experiment should jointly impose finite shared controls and reduce fresh-target planning cost while retaining P2+AV and P2+SV as comparators. If a meaningful advantage survives, the method can be frozen before hidden-plant mismatch tests and held-out signed-weight matrices. If only plan reuse supports the gain, that reuse regime should become the stated use case. If a simpler verification policy matches the result, the contribution should center on the benchmark and the demonstrated accounting tradeoff.

VIII. Conclusion

Crossbar loading must be assessed as a complete scheduled operation with paid sensing and target-dependent computation. In the evaluated surrogate, consistent read accounting and a stronger verification baseline reduce the combined loader's base fresh-target timing advantage to an approximate tie after controller proxies are included. Disturbance-compensated grouping with safer verification offers a lower-error operating point and a conditional time benefit relative to its safer reference, with two unresolved image quality intervals and a differential-kernel stress failure. The results support further focused work on the latency–accuracy tradeoff; they do not establish a hardware speedup or a general whole-array acceptance guarantee.

Artifact and Data Provenance

The accompanying package contains the LaTeX source, bibliography, seven vector figures, their PNG previews, copied per-chip and policy records, an evidence map, and a figure/table regeneration script. Public trajectory extracts and the frozen split/model records support the descriptive model figure. Running the script reconstructs manuscript plots and tables from saved data and checks cost arithmetic; it does not rerun or independently validate the device simulation. Local provenance and hashes are listed in the package README and evidence manifest.

References

[1] H. Yeon, P. Lin, C. Choi *et al.*, "Alloying conducting channels for reliable neuromorphic computing," *Nature Nanotechnology*, vol. 15, pp. 574–579, 2020, doi: 10.1038/s41565-020-0694-5.

[2] C. Choi, H. Kim, J.-H. Kang *et al.*, "Reconfigurable heterogeneous integration using stackable chips with embedded artificial intelligence," *Nature Electronics*, vol. 5, pp. 386–393, 2022, doi: 10.1038/s41928-022-00778-y.

[3] R. Schroedter, A. S. Demirkol, I. Messaris, C. Bruchatz, E. Mgeladze, S. Slesazeck, T. Mikolajick, and R. Tetzlaff, "Model-based time-modulated write algorithm for 1R analog memristive crossbar arrays," *Advanced Electronic Materials*, p. e00777, 2026, doi: 10.1002/aelm.202500777.

[4] L. Yu, T. Zhang, Y. Han *et al.*, "High-concurrency tri-mode memristor-based ordinary differential equation solver," *Nature Communications*, vol. 17, p. 1373, 2026, doi: 10.1038/s41467-025-68122-4.

[5] Z. Yu, M.-J. Yang, J. Finkbeiner, S. Siegel, J. P. Strachan, and E. Neftci, "The ouroboros of memristors: Neural networks facilitating memristor programming," in *Proc. IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2024, pp. 398–402, doi: 10.1109/AICAS59952.2024.10595913.

[6] H. Song *et al.*, "Parallel operation of self-limited analog programming for fast array-level weight programming and update," *Advanced Intelligent Systems*, vol. 2, p. 2000014, 2020, doi: 10.1002/aisy.202000014.

[7] "Crossbar project: Frozen v3 device model, split seed 20261117, and validation report," 2026, unpublished project records; copied as data/frozen_model_v3.json, data/split_seed20261117.json, and data/model_validation_v3.json in the accompanying draft package.

[8] "Crossbar project: Publication stage 1—consistent read costing and safer verification," 2026, unpublished computational evaluation, Nov. 18. Protocol, per-chip records and analysis included in the accompanying draft package; no hardware measurements.

[9] "Crossbar project: Archived-result reproduction and verification record," 2026, unpublished computational verification, Nov. 18; 70 configurations and 1,005 checked comparisons. Provenance recorded in the accompanying evidence manifest.