Proceedings

2026 IEEE 44th International Conference on Computer Design

ICCD 2026

16-18 November 2026

Hong Kong SAR, China

IEEE COMPUTER SOCIETY CONFERENCE PUBLISHING SERVICES

Los Alamitos, California
Washington • Tokyo

IEEE COMPUTER SOCIETY

Copyright © 2026 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved.

Copyright and Reprint Permissions: Abstracting is permitted with credit to the source. Libraries may photocopy beyond the limits of US copyright law, for private use of patrons, those articles in this volume that carry a code at the bottom of the first page, provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

Other copying, reprint, or republication requests should be addressed to: IEEE Copyrights Manager, IEEE Service Center, 445 Hoes Lane, P.O. Box 133, Piscataway, NJ 08855-1331.

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society, or the Institute of Electrical and Electronics Engineers, Inc.

ISBN-13: 979-8-3195-0707-5
ISSN: 2576-6996
BMS Part # CFP26ICD-USB

Additional copies may be orderedfrom:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial and cover art production by Javier Gurrola





IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps

2026 IEEE 44th International Conference on Computer Design (ICCD)

ICCD 2026

Table of Contents

Message from the General Chairs ..... **xli**
Message from the Program Chairs ..... **xlii**
Organizing Committee ..... **xliv**
Technical Program Committee ..... **xlv**

Session 1A - Memory Hierarchy, Caching, and Storage Subsystems

ENCORE: Entity-Conditioned Semantic Cache Reuse for Retrieval-Augmented Generation 1
Yu Zhan (School of Computer Science and Technology, Huazhong University of Science & Technology, Wuhan, Hubei, China), and Yuchong Hu (School of Computer Science and Technology, Huazhong University of Science & Technology, Wuhan, Hubei, China)

HACache: Leveraging Read Performance with Cache in a Heterogeneous Array 9
Yaqi Li (School of Computer Science and Technology, East China Normal University, Shanghai, China; and Shanghai Innovation Institute, Shanghai, China), Jialin Liu (School of Computer Science and Technology, East China Normal University, Shanghai, China), Dingcui Yu (School of Computer Science and Technology, East China Normal University, Shanghai, China), Edwin Sha (School of Computer Science and Technology, East China Normal University, Shanghai, China), Qingfeng Zhuge (School of Computer Science and Technology, East China Normal University, Shanghai, China), Jing Wang (School of Computer Science and Technology, East China Normal University, Shanghai, China), and Liang Shi (School of Computer Science and Technology, East China Normal University, Shanghai, China; Shanghai Innovation Institute, Shanghai, China; and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China)

F2FS-D: Toward Billion-Scale File Systems via Hybrid Dynamic File-Metadata Scaling 17
Yaqing Chen (Huazhong University of Science and Technology, Wuhan, China), Xinyi Huang (Huazhong University of Science and Technology, Wuhan, China), and Shenggang Wan (Huazhong University of Science and Technology, Wuhan, China)

CollComp: Co-Design of Collective Communication and In-Network Compression on NICs 25
Junsheng Chang (College of Computer Science and Technology, National University of Defense Technology; and Key Laboratory of Advanced Microprocessor Chips and Systems, National University of Defense Technology, Changsha 410073, China), Chucai He (College of Computer Science and Technology, National University of

Placement-Driven Scheduling for Efficient Multi-Device PIM Execution 368
Junil Kim (Department of Electrical and Computer Engineering, Korea University, Seoul, Korea), Mun Seong Park (Department of Electrical and Computer Engineering, Korea University, Seoul, Korea), Jisung Pack (Department of Electrical and Computer Engineering, Korea University, Seoul, Korea), Seok Young Kim (Department of Electrical and Computer Engineering, Korea University, Seoul, Korea), and Seon Wook Kim (Department of Electrical and Computer Engineering, Korea University, Seoul, Korea)

SD-GEMM: Breaking the Memory Walls of Sparse Matrix Multiplication with PIM Systems 376
Taehyeong Park (Department of Computer Science, Yonsei University, Seoul, South Korea), Yeonoh Jeong (Department of Computer Science, Yonsei University, Seoul, South Korea), and Yongjun Park (Department of Computer Science, Yonsei University, Seoul, South Korea)

Janus: Device-Heterogeneous Self-Speculative Decoding on Mobile NPUs 384
Hao Xu (College of Computer Science and Technology, National University of Defense Technology), Long Peng (College of Computer Science and Technology, National University of Defense Technology), Shezheng Song (College of Computer Science and Technology, National University of Defense Technology), Xinran Hong (KylinSoft Co., Ltd), Xiaodong Liu (College of Computer Science and Technology, National University of Defense Technology), Jun Ma (College of Computer Science and Technology, National University of Defense Technology), Shasha Li (College of Computer Science and Technology, National University of Defense Technology), Jie Yu (College of Computer Science and Technology, National University of Defense Technology), and Xiaoguang Mao (College of Computer Science and Technology, National University of Defense Technology)

Enabling Efficient Multi-Head Latent Attention on Spatial Accelerators 390
Yuxin Yang (School of Computer Science, Peking University, Beijing, China; and Center for Energy-efficient Computing and Applications, Peking University, Beijing, China), Jieran Zhang (School of Computer Science, Peking University, Beijing, China; and Center for Energy-efficient Computing and Applications, Peking University, Beijing, China), Bizhao Shi (School of Computer Science, Peking University, Beijing, China; and Center for Energy-efficient Computing and Applications, Peking University, Beijing, China), and Guojie Luo (School of Computer Science, Peking University, Beijing, China; National Key Laboratory for Multimedia Information Processing, Peking University, Beijing, China; aND Center for Energy-efficient Computing and Applications, Peking University, Beijing, China)

Session 3C - RISC-V, Vector Processing and Heterogeneous Processor Design

PDPM: Perceptron-Driven Pipeline Morphing for Adaptive RISC-V Processors 396
Dawit Chun (Data Science and Artificial Intelligence, Taejae University, Seoul, Republic of Korea), and Minhee Jun (Electrical and Computer Engineering, The Catholic University of America, Washington, DC, USA)

HBL2: Breaking the L2 Coherence Wall for Matrix-Extended AI CPUs 406
Xi Chen (State Key Laboratory of Processors, Institute of Computing Technology, CAS, Beijing, China; and University of Chinese Academy of Sciences (UCAS), Beijing, China), Tianyi Liu (State Key Laboratory of Processors, Institute of Computing Technology, CAS, Beijing, China; and University of Chinese Academy of

PDPM: Perceptron-Driven Pipeline Morphing for Adaptive RISC-V Processors

Dawit Chun
Data Science and Artificial Intelligence
Taejae University
Seoul, Republic of Korea
chundawit@taejae.ac.kr

Minhee Jun
Electrical and Computer Engineering
The Catholic University of America
Washington, DC, USA
junm@cua.edu

***Abstract*—Pipeline depth is typically fixed at design time, but the depth that minimizes execution time varies with the workload, and even between phases of one program. Prior work optimizes depth statically, adapts frequency without the depth–frequency coupling, or reconfigures without feature-driven runtime decisions. PDPM is a morphable RISC-V processor that reconfigures its pipeline at runtime among 3-stage (P3), 5-stage (P5) and 7-stage (P7) depths under depth-specific clock plans. A learned head reads four lightweight features per 200-instruction window and selects one. Its weights are trained offline and frozen at deployment. Rather than classify configurations, the head regresses two cycle ratios. That makes the controller smaller and reusable. A change of clock plan, or of any objective whose per-mode cost is a transformation of cycles, becomes a change of constants rather than a new fit. On 301 benchmarks under nested 5-fold cross-validation, using trace-derived features to isolate predictor capacity, it reaches 94.28% configuration accuracy, standard deviation 2.48, at +0.107% mean overhead against the 12-class classifier's 91.62%, and with six times fewer parameters, 10 against 60 int8 weights; with the single-mode proxy a deployment can measure, 89.40% at +0.255%. On 340 programs of edge inference it matches the per-phase oracle depth on 313 of 315 labeled phases and finishes 0.44% faster than the best fixed depth per program. Implemented in Verilog and validated on a Xilinx Artix-7 XC7A100T, the autonomous closed loop reaches 90.32 MHz in its one-ratio form and 87.1 MHz with the three-depth head, and its committed architectural state is byte-identical to a fixed-depth reference on all 641 programs. The prototype also reconfigures its own clock through the MMCM, at a measured 21.5 µs relock that precludes per-window clock switching.**

***Index Terms*—Morphable pipeline, RISC-V, adaptive microarchitecture, runtime adaptation, perceptron, pipeline depth, FPGA.**

I. Introduction

Pipeline depth in a processor sets a fundamental tradeoff. Deeper pipelines shorten the critical path per stage and allow higher clock frequencies, but they pay a larger penalty whenever branches or hazards disrupt sequential flow. Since this tradeoff has no universal optimum—the right depth depends on workload characteristics that change across programs and across phases of a single program—commercial designs have oscillated between deep, clock-optimized pipelines (Pentium 4 NetBurst, 31 stages) and shallower, IPC-optimized ones (Intel Core, ARM A-class). A branch-heavy parser benefits from a shallow pipeline with a smaller taken-branch penalty; a compute-bound matrix kernel benefits from a deep one running at higher clock [1], [2].

Prior work addresses this through static depth selection guided by analytical models [1], [3], dynamic voltage–frequency scaling that adapts only f [4], and reconfigurable structures that resize caches or issue width. Existing RV32-class designs have not, to our knowledge, combined runtime pipeline-depth reconfiguration, learned feature-based selection, and depth-aware frequency planning in a single implementation. PDPM (Perceptron-Driven Pipeline Morphing) integrates these mechanisms. It is a morphable RV32IM core that reconfigures among 3-, 5- and 7-stage modes at runtime with architectural state preserved, while a learned head reads four lightweight features per window and picks the depth. The target is edge inference at microcontroller scale, evaluated on a 301-program coverage suite in which all three depths are contested and a deployment suite of 340 programs.

Runtime depth selection should not be posed as a generic multi-class problem. Each depth admits a different maximum clock, so the naive decision space is twelve (depth, clock) points—but nine are dominated under the execution-time objective, leaving three real choices. PDPM therefore predicts cycle ratios between depths rather than classifying configurations, which takes the predictor from twelve linear outputs to two regression outputs (60→10 `int8` weights) and gives a score tied to a measurable quantity rather than an arbitrary class.

This paper contributes three things. First, the architecture is a morphable 3/5/7-stage RV32IM pipeline, built as a P7-superset with bypass multiplexers. Its predictor, window accounting and feature extraction included, occupies 478 LUT and 192 FF, 0.75% of the Artix-7 with no DSP. The mode-control FSM adds 344 LUT, 1.30% together. The closed loop runs autonomously on silicon at 90.32 MHz, and committed architectural state is byte-identical to a fixed-depth reference on all 641 programs. Second, a 10-weight regression head that predicts cycle ratios instead of configuration labels. On 301 benchmarks with trace-derived ρ, it reaches 94.28% accuracy at $+0.107\%$ overhead, against the 12-class classifier's 91.62% at $+0.232\%$ with six times the weights. On proxy for deployment, it holds 89.40% at $+0.255\%$. Because the output is a ratio and not a label, a new clock plan is a change of two constants rather than a new fit. Third, on 340 edge-inference programs, the head matches the per-phase oracle depth on 313 of 315 labeled phases and finishes faster than every fixed-depth policy.

II. Background and Related Work

A. Pipeline Depth and Static Optimization

Hrishikesh *et al.* [3] found 6–8 FO4 inverter delays per stage as the historical optimum; Hartstein and Puzak [1] developed an analytical CPI–frequency model that identifies a workload-dependent optimal depth; and Srinivasan *et al.* [5] optimized pipeline depth jointly for power and performance. Rather than treat depth as a design-time parameter, PDPM addresses it as a runtime variable.

B. Reconfigurable and Adaptive Microarchitecture

Lukefahr *et al.* [6] proposed composite cores that switch between a big and a little core sharing register state, but at the cost of more silicon (a complete second datapath). Closer to our mechanism, dynamic pipeline scaling [7] bypasses or merges pipeline registers to shorten the pipeline under light workloads, but switches on coarse power/slack heuristics rather than a learned per-window policy. Adaptive-microarchitecture work also tunes major resources to phase behavior: Albonesi *et al.* [8] resize caches and queues, and Flicker [9] deconfigures horizontal pipeline lanes under a power cap. At the lightweight end, ultra-low-power RISC-V cores such as Ibex [10] fix a 2–3-stage pipeline at design time. PDPM adapts pipeline depth and frequency within a single core via online feature-based learning. It uses bypass multiplexers within a P7-superset to avoid the area cost of a duplicated datapath.

C. DVFS and ML-Driven Microarchitecture

Per-core DVFS guided by hardware counters [4] is widely deployed but optimizes frequency only, leaving the depth–frequency interdependency unexploited. The maximum achievable frequency depends on the pipeline's critical path, which is set by depth. Machine learning has been applied to microarchitectural decisions: reinforcement learning for memory-controller scheduling [11]; coordinated resource management [12]; and, more recently, learning-based design-space exploration including for RISC-V cores [13] (surveyed in [14]). Most of this work targets design-time exploration or resource sizing rather than runtime depth selection. Jiménez and Lin's perceptron branch predictor [15], [16] extended online linear models to branch direction. PDPM extends them further to depth-and-clock selection via cycle-ratio regression.

D. Bandits for Online Configuration

UCB1 [17] is a feature-free online adaptation baseline. Its purpose is not to represent the strongest possible learning controller, but to show the limitation of exploration-only adaptation when workload phases are short. Because it treats each configuration as an independent arm, it must rediscover good configurations after phase changes. PDPM instead maps a new phase directly from its feature vector, which explains its faster convergence and lower phase-transition overhead.

III. Performance Model and Architecture

PDPM targets small in-order RV32IM systems with no dynamic branch predictor, single-cycle local instruction and data memory, and one active application. This is a microcontroller-class design point, where frequent branch recovery and short phases make the depth-dependent flush cost visible. We do not claim the same 3/5/7 tradeoff holds unchanged for out-of-order, predictor-rich, or cache-dominated processors.

A. Performance Model

Execution time follows the Iron Law $T = N \cdot \mathrm{CPI}/f$ for N dynamic instructions [18], and both terms depend on depth d. Decomposing CPI:

$$\mathrm{CPI}(d) = 1.0 + B \cdot P(d) + H \cdot S(d) + C_{\mathrm{mem}} \qquad (1)$$

for taken-branch fraction B, per-flush penalty $P(d)$, hazard frequency H, stall cycles per hazard $S(d)$ and average memory stall C_{mem}. Memory is single-cycle here, so C_{mem} is depth-independent and cancels in the cross-depth ratios the predictor regresses. The limitations quantify the shift once it does not. Measured branch penalties are $P(3)$=1, $P(5)$=2 and $P(7)$=5 cycles. The analytical clock plan from per-mode critical-path analysis is $\alpha(3)$=1.1, $\alpha(5)$=1.3, $\alpha(7)$=1.5, the maximum clock each mode would support if synthesized in isolation. Two clock plans appear in this paper and are never mixed. The first is this analytical one, under which the predictor comparisons are reported, and the measured plan of the superset this paper builds, 87.1/97.1/101.0 MHz, which is flatter because all three depths share one netlist. Defining effective time $E(d)$=$\mathrm{CPI}(d)/\alpha(d)$ and dropping small hazard and memory terms yields:

$$E(\mathrm{P3}) = (1+B)/1.1, \qquad E(\mathrm{P5}) = (1+2B)/1.3,$$
$$E(\mathrm{P7}) = (1+5B)/1.5. \qquad (2)$$

Crossover branch rates are $B^*_{35} \approx 22\%$ and $B^*_{57} \approx 6\%$. These assume ρ=σ=0 and the α vector above. A learned predictor tracks whatever scaling the implementation achieves and finds a 2D boundary in (β, ρ).

B. Superset Pipeline Design

The processor implements a 7-stage RV32IM pipeline as a superset—IF1, IF2, ID, EX1, EX2, MEM, WB—in which four bypass multiplexers (`bypass_if2`, `bypass_id_ex`, `bypass_ex2`, `bypass_ex_mem`) short-circuit selected pipeline registers to realize shallower modes (Figs. 1 and 2). P3 asserts all four, making the IF1/IF2 and EX1/EX2 registers transparent and merging ID+EX and MEM+WB, for a 3-stage pipeline with a 1-cycle branch penalty; P5 asserts only the two register bypasses, yielding the standard 5-stage RISC-V pipeline at a 2-cycle penalty; P7 deactivates all four for the full depth and a 5-cycle penalty. *Why 3, 5 and 7 stages?* They are the three symmetric bypass configurations of one seven-stage datapath: P3 merges ID+EX and MEM+WB, P7 splits IF and EX, the two longest stages. Depths of 4 or 6 would need asymmetric merges and separate hazard logic, and yield no operating point the three do not already provide.

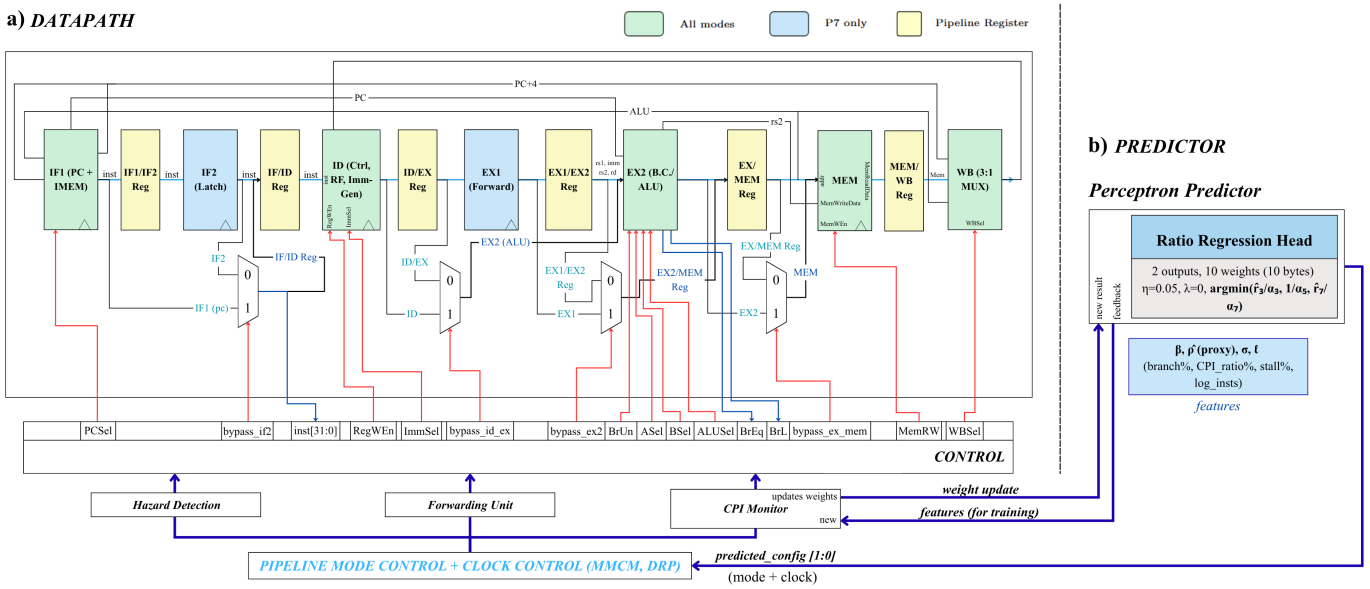


Fig. 1: System architecture. **(a)** Datapath: one seven-stage superset, green present in every mode, blue only in P7, yellow the four bypass multiplexers that realize the shallower modes. **(b)** Control: per-window event counters feed the two-output ratio head, whose $\arg\min$ drives the mode controller, which drains the pipeline and reconfigures the bypass network.

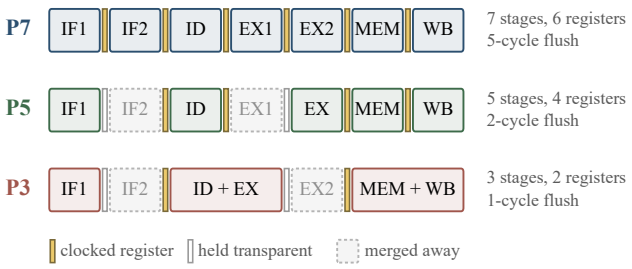


Fig. 2: The three depths as bypass configurations of one seven-stage superset. P7 is the full datapath; P5 makes the IF and EX splits transparent; P3 also merges ID+EX and MEM+WB. Only the four bypass multiplexers change.

The core resolves branches in execute and carries no dynamic branch predictor, taking the flush penalties $P(d)$ on every taken branch. This is a deliberate design point, shared with Ibex [10] and comparable embedded RV32IM cores, chosen to isolate depth-dependent recovery cost in a small in-order core; we do not argue that a production seven-stage core should omit prediction. A predictor would narrow the spread between modes by converting some flushes into correct predictions, but residual mispredictions, data hazards, memory behavior, and clock scaling still set the realized cycle ratios; those ratios are what the head learns, whatever produces them.

The forwarding unit is a priority encoder over four sources: none, MEM/WB, EX/MEM, and an EX2 path active only in P7. The hazard unit inserts a one-cycle bubble on load-use dependencies between EX and ID. Mode switching uses a three-phase FSM. The controller drains the pipeline for d cycles, updates bypass signals atomically with the next clock edge, and resumes. No architectural state is transferred or rebuilt. All three modes share the register file and memory, and only the bypass configuration changes.

Two axes cost differently. A depth change is the drain above, bounded at 5–7 cycles. A clock change reconfigures the MMCM through its dynamic-reconfiguration port while a clock-enable gate holds the core, which takes 21.5 µs measured on the board. We model four rungs, 1.0, 1.1, 1.3 and 1.5× a common base, each depth capped by its critical path—P3 at 1.1×, P5 at 1.3×, P7 at 1.5×. Counting every depth against all four rungs, clamped at each cap, gives the 12-point space.

IV. Learned Depth-Selection Framework

Unless stated otherwise, PDPM denotes the frozen two-output cycle-ratio regression controller defined below. Trace-derived ρ is an analysis ceiling. The single-mode proxy $\hat{\rho}$ of Eq. (3) is the deployable input.

A. Decision Window and Features

The predictor selects a configuration once per decision window of $W{=}200$ instructions to minimize $T_{\text{eff}} = \text{cycles}/f_{\text{clk}}$ over the window. We picked 200 empirically, because per-window feature statistics stabilize within 150–250 instructions. Smaller windows produce noisier features, larger ones miss short program segments.

We use $W{=}200$ throughout the predictor study, where each benchmark contributes one aggregate decision and feature stability rather than reaction time governs. A fixed-clock RTL window-length sweep finds the expected tradeoff: $W{=}100$ halves adaptation latency to 3.2 µs but loses 1.4 pp of cross-validated accuracy. The appropriate window depends on phase duration and implementation target.

Table I defines the four features, normalized to $[0, 1]$ with counters reset at every window boundary. Three need no instrumentation beyond the standard performance-counter unit. The fourth, ρ, is the depth-sensitivity feature. A high branch-flush rate β favors the shallow mode, whose flush costs one cycle instead of five. A high ρ marks code more sensitive to

depth-dependent control-flow penalties, near-zero ρ compute-bound code where P7 wins on clock. Stalls cost more in deeper pipelines as the forwarding path lengthens, so σ pushes the same way. And two benchmarks with identical (β, ρ, σ) can still differ in oracle mode, because longer programs amortize P7's pipeline-fill cost, which is what ℓ carries.

TABLE I: Decision-Window Features

	Definition	Range	Source
β	flushing retires / retires	0–53%	retire, flush counters
ρ	$(C_{P7}-C_{P3})/C_{P3} \times 100$	10–80%	*ceiling:* per-mode traces; *deployment:* $\hat{\rho}$, Eq. (3)
σ	load-use stall cycles / cycles	0–23%	stall, cycle counters
ℓ	$(\log_2 \text{insts} - 4) \cdot 12.5$, clipped	0–99	retire counter

The CPI-depth sensitivity ratio ρ requires special clarification. The ideal CPI ratio equation captures depth-sensitivity directly from cross-mode cycle counts, but this requires simultaneous P3 and P7 measurement of the same window—impossible at deployment time, when only the currently active mode is observable. For deployment, ρ is therefore substituted by a single-mode polynomial proxy of β and σ whose structure matches the Taylor expansion of the analytical relation $\rho \approx 4\beta/(1+\beta) \cdot 100$:

$$\hat{\rho} = a_1\,\beta + a_2\,\beta^2 + a_3\,\beta^3 + a_\sigma\,\sigma + a_0 \tag{3}$$

with coefficients fit on training data and a stall term added since deep pipelines pay more per hazard. Trace-ρ is used here to isolate the predictor's learning capacity from ρ-estimation error, and to set a ceiling any deployment-time estimator can be measured against. It also keeps both heads on the same inputs. Substituting the proxy would collapse ρ into a deterministic function of (β, σ), and the resulting collinearity does not affect a 60-parameter classifier and a 10-parameter regressor symmetrically.

B. Predictor Formulation: Classification Baseline

The straightforward extension of Jiménez–Lin [15] to multi-class is one perceptron per configuration:

$$s_c(\mathbf{x}) = w_{c,0} + w_{c,1}\beta + w_{c,2}\rho + w_{c,3}\sigma + w_{c,4}\ell \tag{4}$$

with $c \in \{0, \ldots, 11\}$, $\mathbf{x} = (\beta, \rho, \sigma, \ell, 1)$ and $\hat{c} = \arg\max_c s_c$: twelve units sharing one input and producing twelve scores in parallel, $12 \times 5 = 60$ int8 weights. Online training uses the multiclass perceptron rule with cost-sensitive class weights $\omega_{c^*} = N/(K \cdot \text{count}(c^*))$ per epoch, correcting the imbalanced oracle distribution. On a misclassification, the model updates two units, $\mathbf{w}_{\hat{c}} \leftarrow \mathbf{w}_{\hat{c}} - \eta\,\omega_{c^*}\mathbf{x}$ and $\mathbf{w}_{c^*} \leftarrow \mathbf{w}_{c^*} + \eta\,\omega_{c^*}\mathbf{x}$, with int8 saturation. The other ten are unchanged.

C. Predictor Formulation: Regression

The classification head spends parameters on configurations that the execution-time objective already dominates. A lower clock at fixed cycle count raises effective time monotonically, so lower-clock variants of a fixed depth are dominated and the twelve outputs collapse to three, P3@1.1×, P5@1.3× and P7@1.5×, leaving the classifier to learn the suppression of the other nine from data.

We instead predict two cycle ratios, normalized so P5 acts as the reference:

$$\hat{r}_3 = \mathbf{w}_3^\top \mathbf{x} \;\approx\; C_{P3}/C_{P5}, \quad \hat{r}_7 = \mathbf{w}_7^\top \mathbf{x} \;\approx\; C_{P7}/C_{P5}. \tag{5}$$

At inference, the configuration selector computes three candidate effective times—$\hat{r}_3/1.1$, $1/1.3$, $\hat{r}_7/1.5$—and picks the minimum, mapped to the matching (depth, max-clock) pair:

$$\hat{c} = \arg\min(\hat{r}_3/1.1,\ 1/1.3,\ \hat{r}_7/1.5). \tag{6}$$

The divisors are the clock plan, written here with the analytical α; retargeting to the measured superset plan of 87.1/97.1/101.0 MHz shifts these two constants and nothing else. They are hard-coded in the inference logic and reduce to bit-shifts and adds at int8, leaving $2 \times 5 = 10$ weights, a 6× reduction.

Training is offline. The two regressors are fitted by SGD on per-mode squared error over the training folds, with learning rate η and ℓ_2 coefficient λ selected by the family comparison, and the weights are frozen thereafter.

D. Confidence Gating, Phase Memory, and Hardware Footprint

Switching carries a drain cost, so every depth change is gated on the prediction margin. The controller commits only if the smallest predicted effective time leads the second-smallest by τ=32 on the int8 scale, about 25% of the maximum representable gap, and clock-only changes bypass the gate. A 16-entry phase-memory table records fingerprint-to-mode tuples, a fingerprint being the four features quantized to 4 bits. After three consistent observations, the entry locks and bypasses the perceptron until the fingerprint changes.

In hardware, the head needs no multiplier. Its inputs are per-window counts of one-bit events, with normalization folded into int16 weights. So, $\mathbf{w}^\top\mathbf{x}$ accumulates as those events fire, with one accumulator per output, an adder tree of gated constants, and the retire counter that closes the window. The accumulated integer is bit-identical to the multiply form, thus the decision is unchanged – remaining a 3-way comparator for the $\arg\min$.

V. Experimental Setup

A. Evaluation Methodology

The multipliers $\alpha(d)$ are the analytical values above, one clock plan of several, not a property of the design. Reported execution times combine them with cycle-accurate per-mode traces from full Verilog simulation of each benchmark in P3, P5 and P7. The plan is an input to the evaluation rather than a property of the predictor, because the head regresses cycle ratios. A plan enters as two added constants and shifts the learned boundary without changing the predictor's structure. Per-mode case analysis on the three-depth closed-loop netlist gives 87.1, 97.1 and 101.0 MHz. That netlist carries the two-ratio head and closes 87.1 MHz overall, against the one-ratio form's 90.32 in Table V. The clock rises with depth as the model assumes, but by 1.000/1.115/1.161 against the analytical 1.000/1.182/1.364. This is because all three depths share one placement and one routing. The shallow mode sets the floor, its

merged IF/ID+EX stage logic-bound where the deeper modes are routing-bound, so a rung safe in every mode is 87.1 MHz.

B. Scope and Assumptions

Measured versus modeled clock scaling — PDPM physically demonstrates autonomous depth morphing and MMCM clock reconfiguration (Section VI-G). Fine-grained clock switching is not used in the runtime experiments because the measured 21.5 µs relock exceeds what a 200-instruction interval can repay. Performance results therefore combine measured per-depth cycle counts with an explicitly stated clock plan, and charge the relock wherever a plan change is credited; fixed-clock deployment results measure the shared-superset FPGA regime as built. The prototype is one superset netlist, so its per-mode frequencies are not those of independently synthesized fixed-depth cores. Both are reported and never mixed.

Every number here should also be read against four conditions. First, training is offline and deployment is inference only. The head is fitted on per-mode traces, then frozen. The datapath carries the update path but no result exercises it, because acquiring the realized ratio r_m^* at runtime needs trial-switching into a mode the policy did not choose. Second, the headline predictor reads trace-driven ρ, which needs simultaneous P3 and P7 cycle counts and is therefore a ceiling; the deployable head reads the single-mode proxy $\hat{\rho}$ and gives up 4.9 pp for it (Fig. 3, Table III). Third, memory is single-cycle unless stated, with 10-, 40- and 240-cycle sensitivity where it matters. Fourth, one application runs at a time, with no co-location, context switching or operating system.

C. Implementation and Tools

The PDPM core is about 4,000 lines of Verilog implementing RV32IM [19]. Cycle counts and event traces come from behavioral simulation in Icarus Verilog; utilization and timing come from Yosys [20] with nextpnr-ice40 [21] on the iCE40 UP5K, and Vivado 2025.1 on the Artix-7 XC7A100T, which carries the benchmarks that exceed iCE40 memory. The testbench runs every benchmark in all three modes under matching reset and instruction-memory conditions, emitting a per-cycle trace that a Python framework replays under each policy to compute features and effective times.

D. Benchmark Suites

The evaluation asks two things of its programs. Comparing predictor designs needs all three depths contested. Where one depth almost always wins, every predictor scores well and the comparison is empty. Measuring the deployed policy needs the workload class the design targets – hence a coverage suite and a deployment suite. The coverage suite is 300 programs plus EEMBC CoreMark [22]: microarchitectural stress cases, generated control- and compute-oriented kernels, compiled C programs, and ML-oriented inner loops.[1] No program is tuned to favor a depth. Each is compiled independently, labeled by execution in all three modes, and kept only if it completes correctly in all three. The oracle distribution covers all three— 60 P3 (19.9%), 141 P5 (46.8%), 100 P7 (33.2%)— no static depth dominates the evaluation set.

The deployment suite is a separate 340 programs of 2026 edge-inference computation, sharing none with the coverage suite. Three hundred are kernels from 44 parameterized families spanning attention, quantized dense algebra, convolution, state-space and recurrent sequence models, decoding, retrieval and robotic action heads. The other 40 chain them into ten model types at four shapes each, with a phase marker at every kernel boundary, giving 315 labeled phases with per-phase cycle counts in all three depths. All are our own C at edge-MCU shape, 16 KB of memory and integer arithmetic only, compiled RV32IM at `-O2`. It is measured at one fixed clock, so $\alpha(d)$ does not enter and a depth change costs a drain, not a relock. A segment is a distinct program image running 1,000 instructions (500 in Full-50), and leave-N-out training excludes every program a stream contains.

E. Evaluation Protocol and Data-Leakage Prevention

Each benchmark contributes one $(\mathbf{x}, c^*)$ pair, with $\mathbf{x}=(\beta, \rho, \sigma, \ell, 1)$ aggregate features and c^* the empirical oracle from Verilog simulation in all three modes. Headline numbers come from nested cross-validation with $9\times$ resampling. An outer 5-fold split (61-benchmark test, 240-benchmark pool) wraps an inner 3-fold split (160 train, 80 validation), repeated across three fold permutations and three weight initializations for 45 outer trials per configuration, from which all confidence intervals are taken. Models are fit on train, hyperparameters chosen on inner validation, accuracy and overhead reported on the test fold.

Hyperparameter grids span $\eta \in [0.05, 0.7]$, passes up to 200, and $\lambda \in \{0, 10^{-4}, 10^{-3}, 10^{-2}\}$. The selected points are classification η=0.35/200 passes and regression η=0.05/100 passes/λ=0.

Four leakage paths are closed: normalization statistics come from training folds only, benchmarks are parameterized on code pattern with the oracle measured after generation, all policies consume the same per-benchmark Verilog trace, and the family-selection rule was committed before any outer-test result was seen.

F. Oracle and Metrics Definition

Policy overhead is $(T_{\text{policy}} - T_{\text{oracle}})/T_{\text{oracle}} \times 100$. The Per-window Oracle selects the minimum effective time per decision window among the non-dominated configurations, using the same per-mode traces and drain-cost model as the evaluated policy; under the dominated-lower-clocks structure this coincides with the best of all 12. Configuration-correctness is the fraction of decision windows in which a policy selects the same configuration as that oracle,

$$\text{Acc}(\%) = \frac{1}{|\mathcal{W}|} \sum_{t \in \mathcal{W}} \mathbb{1}[\hat{c}_t = c_t^*] \times 100, \tag{7}$$

[1] 10 hand-written archetypes, 90 generated kernels (10 categories × 3 sizes × 3 variants), 33 extended kernels, 8 `gcc -O2` C programs and 159 ML-foundation kernels, plus CoreMark. Counts post-filter.

for decision windows $\mathcal{W}$, with $\hat{c}_t$ selected and c_t^* the oracle configuration at window t. Under the per-benchmark protocol each benchmark contributes one window, so Eq. (7) is the accuracy reported in Section VI-B. In the workload scenarios $|\mathcal{W}|$ counts the windows in the scenario. This oracle changes configuration every window, unlike the per-program oracle, and it is drain-free, so no evaluated policy can fall below it with stream overheads non-negative.

G. Comparison Policies

Five policies are compared. Always P5@1.0× is the fixed 5-stage at base clock, and Always P5@1.3× the strongest static configuration. UCB1 with twelve arms [17] is an online bandit starting cold. The Per-window Oracle above is the reference, and PDPM is the R1b regression head with optional 16-entry phase memory. For workload evaluation, its weights are pre-trained leave-N-out on benchmarks absent from the scenario, then frozen. UCB1 starts from scratch every time, which shows the cost of exploration without prior characterization rather than bounding online learning in general.

VI. Results

A. Regression versus Classification

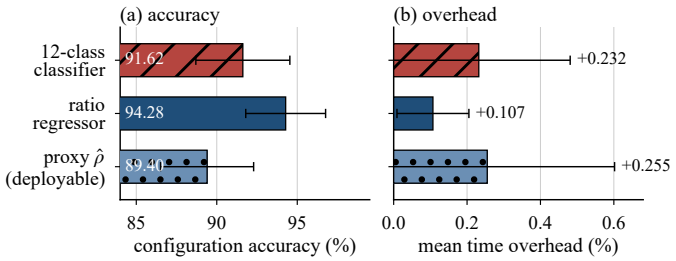


Fig. 3: Classification against regression under identical features, training data and protocol. The regressor leads on both axes with 10 int8 weights against 60. The third bar is the same regressor on the proxy $\hat{\rho}$ a deployment can measure.

Under the identical 45-trial nested-CV protocol on the 301-benchmark suite, and with trace-derived ρ throughout so that the comparison measures predictor capacity rather than feature observability, regression matches or improves on classification on every comparable axis of accuracy, overhead, and 10 int8 weights against 60. These are ceiling figures for both heads. With the deployable proxy the regressor measures 89.40±2.89% at $+0.255\%$, and we did not refit the classifier on the proxy.

Four model families were compared under one protocol, the selection rule committed before any test result (Table II).

TABLE II: Pre-registered family comparison: 45 trials each, trace-derived ρ, λ=10^{-3} for R1 and 0 for R1b.

Variant	Wts.	Inner val. %	Outer test %
A1: 12-class linear	60	89.74±3.18	91.62±2.92
A2: 12-class + quadratic	96	89.50±3.25	90.85±3.10
A3: MLP, 7→4→12	92	71.55±21.17	71.70±25.45
A4: 3-class depth	15	89.59±2.86	90.73±5.12
R1: regression, η=0.1	10	93.05±1.54	92.80±3.15
R1b: regression, η=0.05	**10**	**93.88±1.34**	**94.28±2.48**

A2 and A3 both perform worse than A1 on the outer test, and the regression family wins; the pre-committed rule chooses R1b (η=0.05, λ=0) over R1 on inner-validation accuracy (93.88% vs. 93.05%).

Larger does not help. On 301 labeled points the added capacity of A2 and A3 fits fold noise, and the MLP loses more than 20 pp at five times A1's variance. Simpler does not help either: the analytical β threshold of Fig. 8 misclassifies across all three oracle regions, the boundary being two-dimensional in (β, ρ). That figure also carries the 300 deployment kernels, which no part of the fit saw. The learned boundary places 87.3% of them on the side their oracle implies against 92.7% of the benchmarks it was fitted to. The kernels reach β values the benchmarks never do, and there the one-dimensional model fails. All 37 past β=55% are P7-optimal and a β threshold calls each of them P3, their flushes being cheap relative to the work per instruction.

A4 removes the confound in comparing 60 weights against 10. The twelve-way classifier must represent configurations that are dominated under the execution-time objective, so some of its deficit might be the output space rather than the label representation. Reducing the output space from twelve classes to three does not help, since accuracy drops to 90.73% and overhead from $+0.232\%$ to $+0.178\%$, while the weights fall from 60 to 15. Replacing those three labels with two cycle-ratio targets then raises accuracy to 94.28% at $+0.107\%$ overhead and 10 weights. These ablations indicate the gain is not output-space reduction, rather replacing labels with ratio targets is the key change.

Because A4 and R1b see identical folds, seeds and training sets, the comparison can be made per trial. Paired over 45 outer trials, regression is $+3.11$ pp more accurate (95% bootstrap CI $[+1.78, +4.48]$, Wilcoxon signed-rank p=1.3×10^{-4}) at 0.072 pp lower overhead (CI $[-0.109, -0.037]$, p=1.2×10^{-5}). Its errors also cost less. Across the same trials, the three-class head is wrong 251 times at a mean cost of 1.50% of execution time and a 90th percentile of 3.14%, the regressor 167 times at 1.10% and 2.30%. A ratio target carries the size of each error into the loss. A label target, over three classes or twelve, treats every error alike.

B. Generalization on Held-Out Folds

A separate 5-fold leave-out protocol on the 301-benchmark suite checks for per-fold memorization. Table III shows against every comparison policy and against the deployable input.

TABLE III: 5-Fold Leave-Out Generalization, 301 Benchmarks

Policy	Overhead	Config. correct
Always P5@1.0×	+33.342%	0.00%
Always P5@1.3×	+2.572%	46.84%
UCB1 (12-arm)	+23.565%	9.24%
Per-window oracle	+0.000%	100.00%
PDPM, trace ρ (ceiling)	**+0.107±0.098%**	**94.28±2.48%**
PDPM, proxy $\hat{\rho}$ (deployable)	+0.255±0.347%	89.40±2.89%

Outer-test accuracy matches inner validation with no measurable drop, where memorization would show as one–learned weights depend on the features, not on benchmark identity. Trace ρ is a ceiling no deployment can read. The proxy is what the counters can supply, and it costs 4.9 pp of accuracy

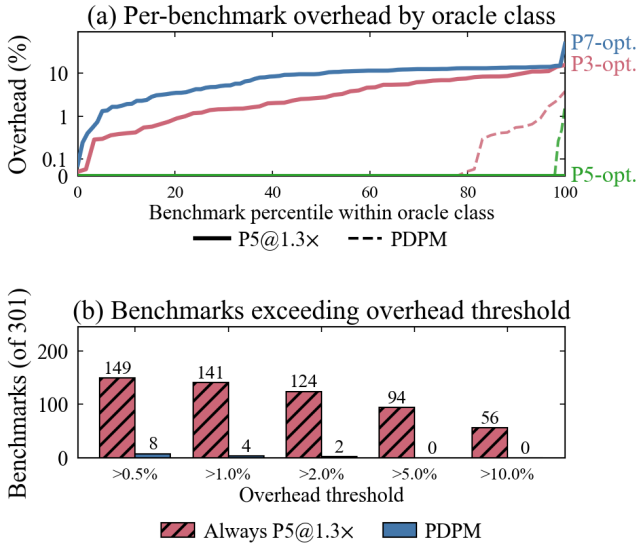


Fig. 4: **(a)** Per-benchmark overhead sorted within each oracle class, solid always-P5@1.3×, dashed PDPM, symlog axis. **(b)** Benchmarks of 301 above each overhead threshold.

and 0.148 pp of overhead. Even so, the deployable head stays an order of magnitude under the strongest static configuration. Nor is the mean hiding a tail (Fig. 4). Across the 45 trials the median per-error overhead is 1.31% and the maximum 6.21%, every error is between adjacent depths, and 8 of 301 benchmarks exceed 0.5% overhead under PDPM against 149 under always-P5@1.3×.

C. Multi-Program Stress Test

Unlike having depth changes inside one composite program, this experiment concatenates separate program images and tests selection from the first window of each. We evaluate six unseen multi-program streams under 200-instruction windows, including alternation and delayed-recall cases, and two built the same way from the deployment kernels. PDPM stays within +0.1% of the per-window oracle on all coverage streams and within +0.54% on deployment streams (Fig. 5). The strongest static policy is 1.5–5.2 pp worse and cold-start UCB1 21–26 pp worse. Persistence beats static depth but incurs 20.32% overhead on Repeating, where PDPM is neutral. The gain is feature-based first-window selection, not temporal locality.

D. Depth Selection Across Clock Plans

The two suites are measured under different clock plans, and separating those axes explains the evaluation (Fig. 6). The 300 deployment kernels split 286/14/0 across P3/P5/P7 at one clock, 8/281/11 under the measured superset, and 8/73/219 under the analytical plan. The 301 coverage benchmarks were similar, 300/0/1 at one clock against 60/141/100 under the analytical plan. Depth is a property of the workload and the clock plan together, not of the workload alone.

At one clock, both suites collapse onto the shallow mode. P7 needs a 15% clock advantage over five stages before it takes 77 of the 300 kernels. The measured superset supplies 4.1%, enough for 11, because past five stages the critical path is the memory interface rather than datapath depth. Under the analytical plan, which pays the deeper pipeline for its shorter stages, the kernels lean deeper. The 219 preferring P7 there are led by attention, GEMV/GEMM, MoE routing, RoPE/YaRN and the Mamba scan, winning by up to 11.3% over P5, and 41 of the 44 families place at least one kernel on it, the largest contributing 10.

The wrong depth is expensive in the tail too. Always-P7 costs up to 78.9% on one kernel at one clock, always-P3 up to 31.6% under α. The same holds inside a program. No pipeline prefers all three depths under any single plan—at one clock the 315 phases split 301/14/0—because the plan decides which two are contested, yet all 40 chained benchmarks use all three modes across the plans (Fig. 7).

Workload and clock plan decide the depth jointly, which is the case for choosing at runtime and the reason the head regresses cycle ratios, which are invariant to the plan. The results above use one clock. Letting it follow the mode is worth a further 5.8% on the deployment kernels, 5.1% once the measured 21.5 µs relock is charged and 11.0% at a 240-cycle miss latency; on the coverage suite it is neutral, its 4 µs programs spending on relock what they gain.

For a single clock, though, the exception to the shallow mode is one family. Causal dilated 1-D convolution runs 3–5% fewer cycles in five stages, with two further phases under 2%. Always-P7 costs 19.1% against a per-program oracle here and always-P5 5.2% (Table IV).

E. Phase Changes Inside Unseen Programs

The depth also changes within a program. Across the 40 pipelines, the best depth differs at 20 of the 275 phase boundaries, and holding the best depth per phase is worth 0.45% of execution time over the best single depth per program (Fig. 7). A head fitted only on the isolated kernels follows those transitions inside pipelines it has never seen, settling within roughly 200 cycles of a marker. It matches the per-phase oracle depth on 313 of 315 labeled phases and finishes 0.437% faster than the best fixed depth for each program, a reference no policy switching mid-program is bounded by. The three references here are distinct: per-program, per-phase and per-window. Against the per-phase reference, which does bound this policy, the head is 0.011% short. A depth-3 tree fitted identically does not transfer, landing 2.9% behind.

TABLE IV: Fixed-Clock RTL Results on the 2026 Edge-Inference Programs

Policy	300 kernels Ovh. %	40 pipelines Ovh. %	Phases at oracle
Always P7	+19.076	+17.107	—
Always P5	+5.177	+3.124	—
Always P3 (best static)	+0.203	+0.209	—
Depth-3 tree, other suite	+0.204	+0.211	301/315
PDPM, fit on the kernels	**+0.048**	**−0.437**	**313/315**

Overhead against a *per-program* oracle fixing the best single depth per program; a negative entry is faster than that oracle, which only a policy switching inside a program can be. The pipelines are unseen by both heads. The per-phase oracle is itself −0.448% against this per-program reference, so PDPM's −0.437% is 0.011 pp short of it.

What a head must represent follows from its regime. Fitting one head across both costs 0.65 pp on the pipelines (93.3±2.0%

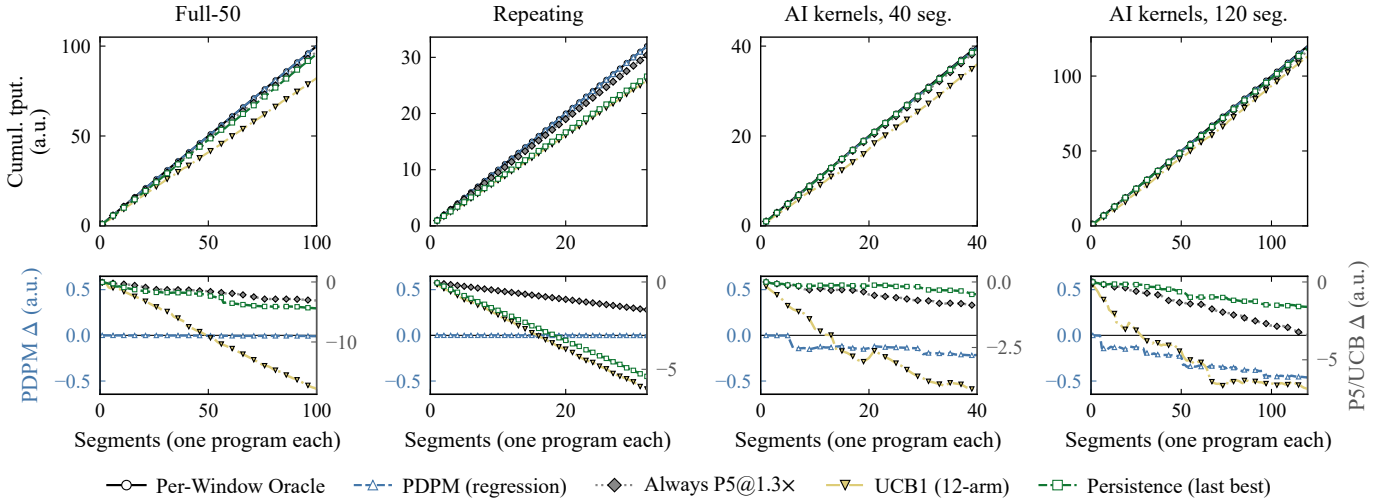


Fig. 5: Cumulative throughput of sequential, non-co-located program streams. *Top:* PDPM tracks the per-window oracle from the first window. *Bottom:* deviation from it, PDPM on the left axis and the baselines on the right, two orders wider.

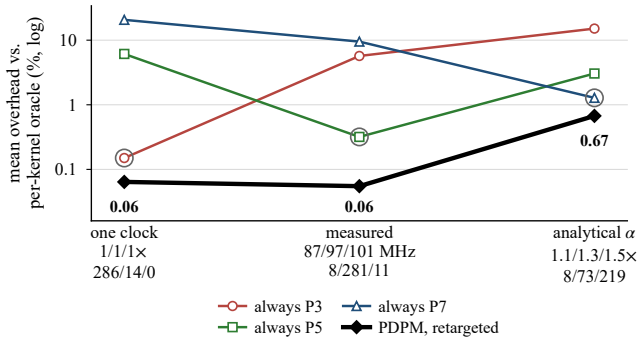


Fig. 6: No fixed depth survives all three clock plans; the head does. The same 300 kernels and measured cycle counts throughout—only each depth's clock changes, named under each column with the oracle's P3/P5/P7 split. Every static depth is best under one plan (circled) and up to 20.6 % worse under another. PDPM has weights frozen, two bias integers retargeted per plan. Unretargeted, it keeps picking P3, at +5.90 and +14.98 %.

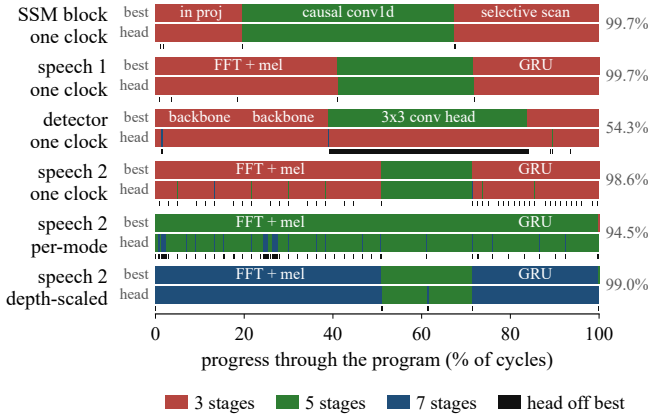


Fig. 7: PDPM tracks kernel-delimited depth changes inside unseen composite programs. Upper bar: the per-phase oracle; lower bar: the depth served; thin marks: mismatches, with agreement at right. Rows 1–3 are three programs at one clock; rows 4–6 one program under all three clock plans, where the depth-scaled plan puts 79% of its cycles in seven stages.

against 97.3±2.6% cross-validated). At a fixed clock, the seven-stage mode is never selected, so a head that drops it—one ratio in the same module, the second output not instantiated—lets synthesis prune the P7 datapath for 24% of the design, and is the better policy there. Under a clock plan the depth is worth keeping, which is why the paper evaluates the three-way head.

F. Learned Decision Boundaries

Negative w_β and w_ρ on $\hat{r}_3$ make P3 cheaper than P5 as either rises, and the dominant w_ρ=+78 on $\hat{r}_7$, nearly three times $|w_\beta|$, drives P7 worse than P5 at high CPI ratio.

The boundaries are 2D lines in the (β, ρ) plane, not 1D β-thresholds. Analytical thresholds (β^*_{35}=22%, $\beta^*_{57}\approx$6%) are the slice of these lines at ρ=σ=0, but real workloads lie between ρ=10% and ρ=80%, where the 1D thresholds are off by tens of pp. Fig. 8 plots all coverage and deployment benchmarks in (β, ρ). The learned 2D lines (dashed) separate the three oracle classes, while the analytical β-only thresholds (solid, vertical) miss most of the boundary structure. This explains the 24× overhead gap between the static P5@1.3× baseline (+2.57%) and PDPM (+0.107%).

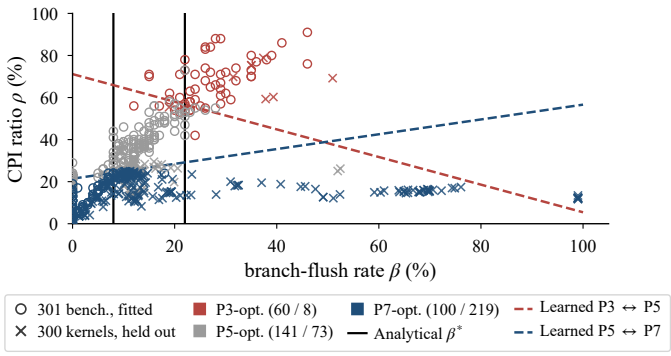


Fig. 8: The learned rule uses branch rate and cross-depth cycle sensitivity; a branch-only threshold cannot separate the oracle-depth regions. Color is oracle depth under the analytical plan, shape the suite: circles the 301 fitted benchmarks, crosses the 300 held-out kernels. Solid verticals are the analytical β thresholds, dashed the learned two-dimensional ones, which place 92.7% and 87.3% correctly.

At deployment, ρ is replaced by the single-mode proxy $\hat{\rho}$ of Eq. (3), whose coefficients come from ordinary least squares of trace-measured ρ on $(\beta, \beta^2, \beta^3, \sigma)$ over the 240-benchmark

training folds, the cubic structure chosen a priori from the Taylor expansion of $4\beta/(1+\beta)$ rather than by feature search. Ordered as in Eq. (3) they are $(+3.04, -0.046, +4.4\times10^{-4}, +1.02, +4.37)$. Under the same 5-fold protocol with R1b retrained on $\hat{\rho}$, accuracy is $89.4 \pm 2.89\%$ at $+0.255 \pm 0.347\%$ overhead, a 4.9 pp gap with trace ρ. Out of fold, the proxy tracks ρ at r=0.959 with 4.36 pp mean absolute error; refitting the head on β and σ alone, with no $\hat{\rho}$ term, costs a further 1.70 pp (paired over the 45 trials, t=4.33).

G. Autonomous Operation on the FPGA

If above is a policy result from traces, then the hardware question is whether the loop closes on a real device without changing what programs compute. On the Artix-7 board, the predictor drives the mode controller directly, no host in the loop. Features come from the core's own counters, a decision is taken at each window boundary, and the controller drains and updates the bypass network. The board holds the one-ratio head at 90.32 MHz. The three-depth two-ratio head certifies 87.1 MHz, missing by 57 ps the 87.5 MHz rung it was constrained against, and has run on the board at that rung as well.

A separate bitstream makes the clock runtime-reconfigurable. The mode controller drives the MMCM through its dynamic-reconfiguration port, gating the core with a `BUFGCE` for the whole reprogram-and-relock so it never sees an unlocked edge, and ordering the two changes so a mode is never clocked above its certified rung—slowing down before a deepening switch, speeding up after. Running autonomously, the core reconfigured its own clock alongside its depth with the switch counter reaching 470. An integrated logic analyser sampling 27 of those switches across three sessions puts relock at 21.08–21.44 µs and the core-off interval at 21.30–21.66 µs, against a datasheet bound of 100 µs. Joint depth-and-clock actuation is therefore measured on this prototype. What the 21.5 µs forecloses is doing it often, which is why the evaluation treats the clock plan as a parameter and charges the relock whenever a change is credited.

The deepest mode did not run on the board until the policy could select it, due to morphable design rather than to this prototype. The deployed counter tree never chooses P7 at a single clock, so constant propagation under the frozen policy made mode 2 unreachable; the tools removed the upper mode-register bit together with the seven-stage datapath. A depth the policy cannot select is not a depth the configured fabric contains. When a clock plan makes P7 reachable, reinstating the two-output head retains the seven-stage datapath and the autonomous controller selects it on the board.

Correctness is checked more strictly than a checksum. For every program, we record the ordered sequence of architectural commits—each register write and each store—from the autonomous run and from a fixed-depth reference run of the same binary. They are identical on all 301 benchmarks, 300 deployment kernels, and 40 model programs at a 10-cycle as well as 40-cycle memory latency, across 144 to 1,315 self-initiated depth switches. Morphing depth mid-program is architecturally invisible, changing when results are produced.

H. Resource Utilization and Timing

TABLE V: FPGA Resource Utilization Across Two Targets

	iCE40 UP5K 4 KB mem, bringup		**Artix-7 XC7A100T** 16 KB mem, full suite	
Resource	Used	Util. %	Used	Util. %
Logic cells / LUTs	4,506	85%	3,086	4.87%
of which logic	3,506	66%	3,018	4.76%
as memory	—	—	68	0.36%
DFF / FF	1,302	25%	1,461	1.15%
F7 / F8 muxes	—	—	42 / 16	0.13%
Block RAM	4	13%	5.5	4.07%
DSP / multiplier	0	0%	4	1.67%
Predictor, 2-output reg.	—	—	478 / 192	**0.75%**
one-ratio form	—	—	181 / 92	**0.29%**
Predictor, 12-way class.	—	—	331 / 58	0.52%
$f_{\max}$ post-route	52.96 MHz		**90.32 MHz**	
WNS	n/a (icetime)		+0.013 ns	

Predictor rows are LUT / FF post-route; Util. % is of device LUTs.

The CPU was brought up on a Lattice iCE40 UP5K (Yosys + nextpnr, $f_{\max}$=52.96 MHz, 4,506 logic cells) and moved to a Xilinx Artix-7 XC7A100T once 16 KB memories became necessary, the RTL identical across both. The first 16 KB port closed at 70.19 MHz with 3,378 LUTs and 573 FFs under Vivado 2025.1 and runs the full 301-benchmark suite including CoreMark. Registering the instruction-memory read on the falling edge and adding the directives described later gives the closed-loop build of Table V at 90.32 MHz, 3,086 LUTs and 1,461 FFs, the flip-flop count rising because the memory read and predictor pipeline are now registered. Post-route, the two-output regression head measures 478 LUT / 192 FF with no DSP, its one-ratio specialization 181 LUT / 92 FF, and a classification head over the same counters 331 LUT / 58 FF: 0.75%, 0.29% and 0.52% of the device. The 6× weight reduction does not appear as module area, because in either head the dominant term is extracting window features from the event counters rather than the dot product.

One design decision sets the achievable clock. P3's merged IF/ID+EX stage needs a combinational instruction-memory read, which a synchronous block RAM cannot supply. Registering the array read on the falling edge resolves this without adding an architectural fetch cycle. The read becomes a half-period path that block-RAM clock-to-out meets comfortably, and re-simulation confirms 301 of 301 traces byte-identical. The closed loop then closes 90.32 MHz at WNS +0.013 ns by two independent implementation routes.[2]

Past five stages, the limit is the memory interface rather than datapath depth. The seven-stage mode reaches its binding path in six logic levels against the five-stage mode's sixteen, but has half a clock period in which to do it. The worst setup paths all end in the data-cache tag-and-valid lookup, none traversing a bypass multiplexer or the mode-select logic. Morphability does cost clock, and the cost is the shared bypass muxing. A standalone five-stage build of the same core certifies at

[2]A placement directive (+0.013 ns) and a sequential-area pass (+0.008 ns). The three-depth two-ratio loop certifies 87.1 MHz in the same flow, 57 ps short of the 87.5 MHz rung it was constrained against.

136.6 MHz, while removing P3 from the morphable netlist closes no higher.

VII. Discussion

The formulation retargets to energy as it does to a clock plan. Per-mode dynamic power measures 0.448, 0.365 and 0.413 mW/MHz, so energy per instruction scales as $\mathrm{CPI}(d) \cdot P_{\mathrm{dyn}}(d)$ and those coefficients fold into the α_d divisors of Eq. (6) with the weights untouched. The closed loop goes from $+0.454\%$ to -0.265% of a per-program energy reference.

In a morphable design, the policy and the datapath are not separable. Constant propagation removes any mode the frozen controller cannot reach, so a head that never selects P7 deletes the seven-stage pipeline as surely as editing it out of the RTL. Used deliberately, a deployment needing two depths need not be designed twice. Freezing the third output lets the tools reclaim its datapath, the 24% area reduction, obtained by changing ten weights rather than the pipeline. Depth becomes a parameter of the policy.

Furthermore, a clock plan enters as two added constants, one bias per predicted ratio, because of what the head outputs. The characterization head of Eq. (6) regresses the cycle ratio. The head synthesized for the board regresses its logarithm, which makes a clock plan additive and gives the dead band its units.

On the twelve Embench IoT 1.0 [23] programs that fit our runtime, accuracy holds at 93.3±3.1% with +0.12% overhead; `nsichneu`, `cubic`, `minver`, `nbody`, `st`, `wikisort` and `qrduino` are excluded on footprint, soft-float or allocation limits, not algorithmic ones.

VIII. Limitations

The α-scaled results use analytical clock factors (Section V-B). Built separately the shallow-to-mid ratio is within 1% of it. Inside the superset the ordering is monotonic but compressed. The feature ρ is obtained from per-mode traces offline, and the deployable substitute is the proxy of Eq. (3), at a cost of 4.9 pp. Probing the other depth instead is implemented in the RTL and does not pay at these margins. Enabling it moves the gate-level closed loop from $+0.501\%$ to $+0.923\%$. Closing that gap at low overhead remains the open problem. Also, intra-program premise rests on the composite pipelines alone, and memory is single-cycle. Slower memory separates the depths further, not less. The best single depth falls from $+0.48\%$ to $+1.31\%$ of a per-program oracle, and the deepest mode wins outright on 28 of the 301 programs at a 10-cycle miss latency and 56 of 298 at 240, by up to 23.7%. Finally, the core addresses the unprivileged RV32IM datapath only, not precise exceptions, interrupts, virtual memory or speculative execution.

IX. Conclusion

PDPM demonstrates that adaptive pipeline-depth selection can be implemented with a small learning-driven controller on a morphable RV32IM datapath, and that the problem is better formulated as cycle-ratio regression than as twelve-way configuration classification. The two-output regressor uses 10 weights and reaches 94.28% accuracy at $+0.107\%$ mean overhead on trace-derived ρ, against the twelve-class classifier's 60 weights, 91.62% and $+0.232\%$, and holds 89.40% at $+0.255\%$ on the deployable proxy. Implemented in Verilog and validated on a Lattice iCE40 UP5K and a Xilinx Artix-7 XC7A100T, the deployed head occupies 478 LUT, under 1% of the device; the closed loop runs autonomously at 90.32 MHz, byte-identical to a fixed-depth reference on all 641 programs. These results indicate that lightweight feature-driven prediction can connect static pipeline design with dynamically varying workload behavior in embedded RISC-V systems.

References

[1] A. Hartstein and T. R. Puzak, "The optimum pipeline depth for a microprocessor," *ISCA*, 2002.

[2] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed., Morgan Kaufmann, 2019.

[3] M. S. Hrishikesh *et al.*, "The optimal logic depth per pipeline stage is 6 to 8 FO4 delays," *ISCA*, 2002.

[4] W. Kim *et al.*, "System level analysis of fast, per-core DVFS using on-chip switching regulators," *HPCA*, 2008.

[5] V. Srinivasan *et al.*, "Optimizing pipelines for power and performance," *MICRO*, 2002.

[6] A. Lukefahr *et al.*, "Composite cores: Pushing heterogeneity into a core," *MICRO*, 2012.

[7] J. Koppanalil *et al.*, "A case for dynamic pipeline scaling," *CASES*, 2002.

[8] D. H. Albonesi *et al.*, "Dynamically tuning processor resources with adaptive processing," *IEEE Computer*, vol. 36, no. 12, 2003.

[9] P. Petrica *et al.*, "Flicker: A dynamically adaptive architecture for power limited multicore systems," *ISCA*, 2013.

[10] P. D. Schiavone *et al.*, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for IoT applications," *PATMOS*, 2017.

[11] E. İpek *et al.*, "Self-optimizing memory controllers: A reinforcement learning approach," *ISCA*, 2008.

[12] R. Bitirgen *et al.*, "Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach," *MICRO*, 2008.

[13] C. Bai *et al.*, "BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework," *ICCAD*, 2021.

[14] N. Wu and Y. Xie, "A survey of machine learning for computer architecture and systems," *ACM Comput. Surv.*, vol. 55, no. 3, 2022.

[15] D. A. Jiménez and C. Lin, "Dynamic branch prediction with perceptrons," *HPCA*, 2001.

[16] D. A. Jiménez, "Fast path-based neural branch prediction," *MICRO*, 2003.

[17] P. Auer *et al.*, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, 2002.

[18] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design*, RISC-V ed., Morgan Kaufmann, 2017.

[19] A. Waterman and K. Asanović, eds., "The RISC-V instruction set manual, vol. I: unprivileged ISA, v20191213," RISC-V Foundation, 2019.

[20] C. Wolf and J. Glaser, "Yosys—A free Verilog synthesis suite," *Austrochip*, 2013.

[21] D. Shah *et al.*, "Yosys+nextpnr: An open source framework from Verilog to bitstream for commercial FPGAs," *FCCM*, 2019.

[22] S. Gal-On and M. Levy, "Exploring CoreMark," EEMBC, 2012.

[23] J. Bennett *et al.*, "Embench: an evolving benchmark suite for embedded IoT computers," *RISC-V Workshop*, 2019.