ICCD 2026
The 44th IEEE International Conference on Computer Design
Nov. 16-18, 2026 Hong Kong SAR, China

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

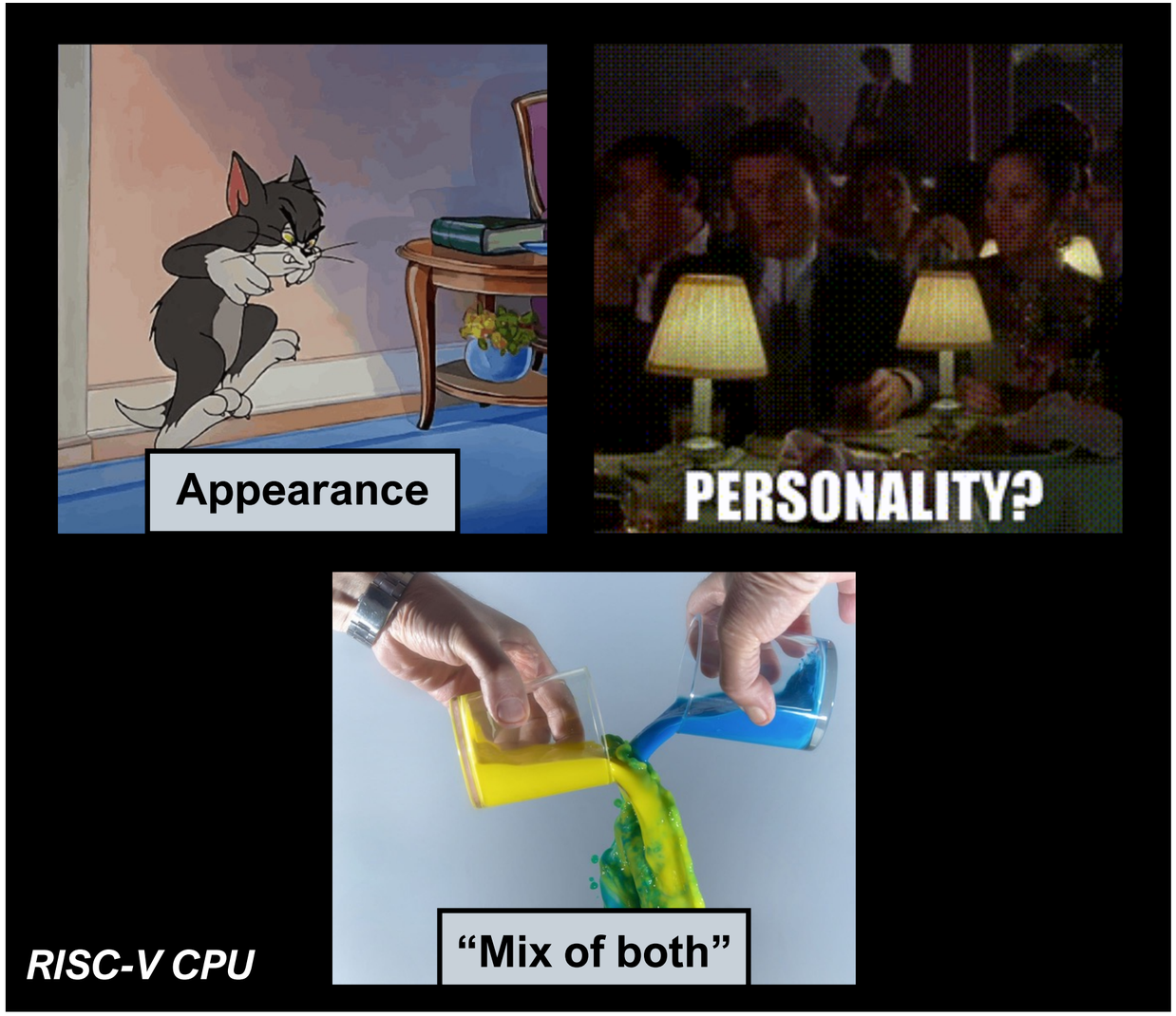
Dawit Chun[1] Minhee Jun[2]

[1]Taejae University, Seoul, Korea
[2]The Catholic University of America, Washington, DC

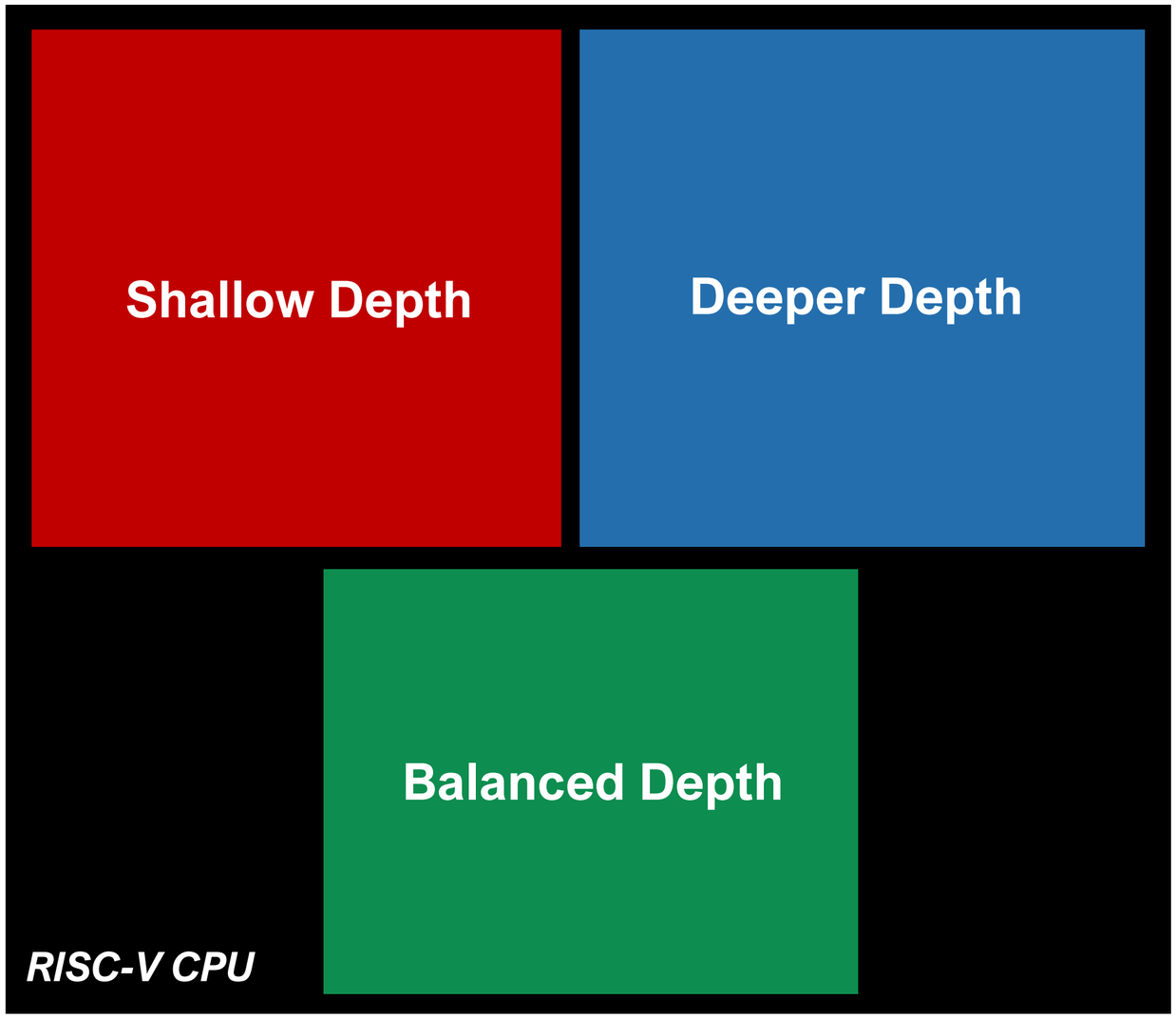


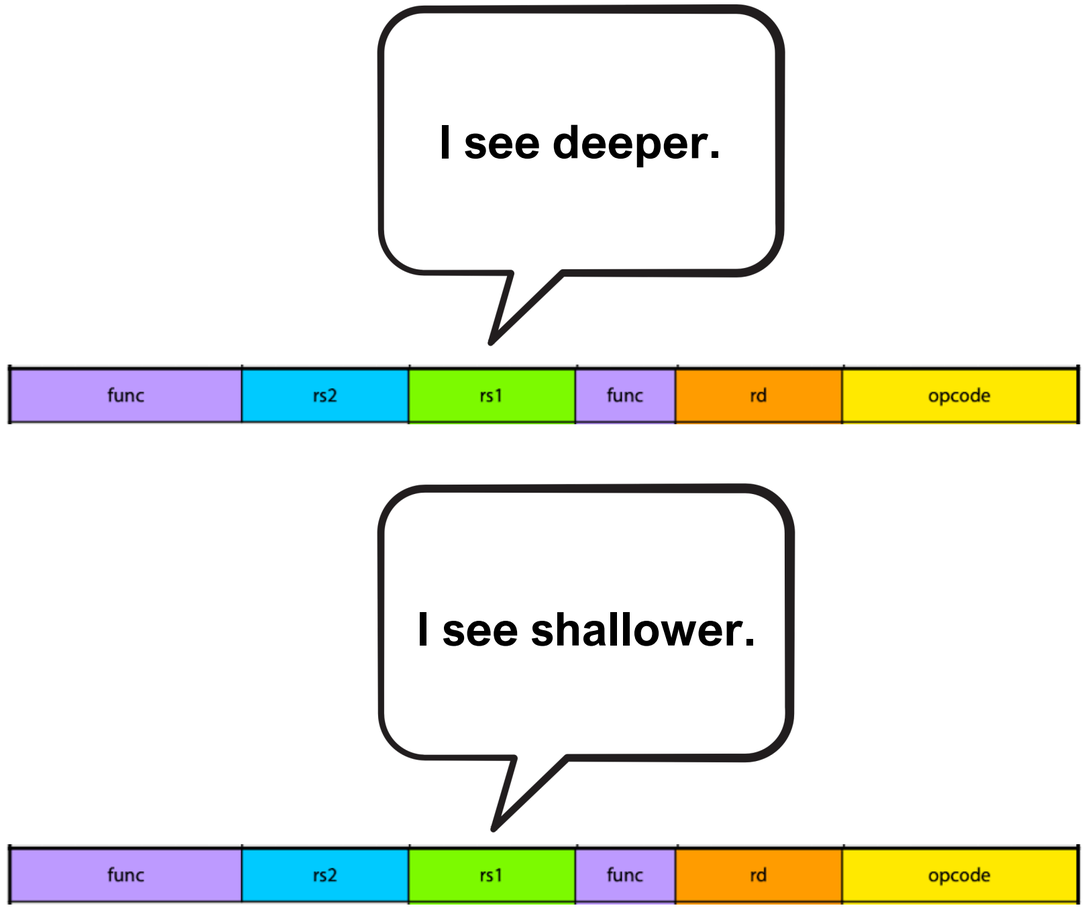
TAEJAE UNIVERSITY
THE CATHOLIC UNIVERSITY of AMERICA

“What do you see in a person?”

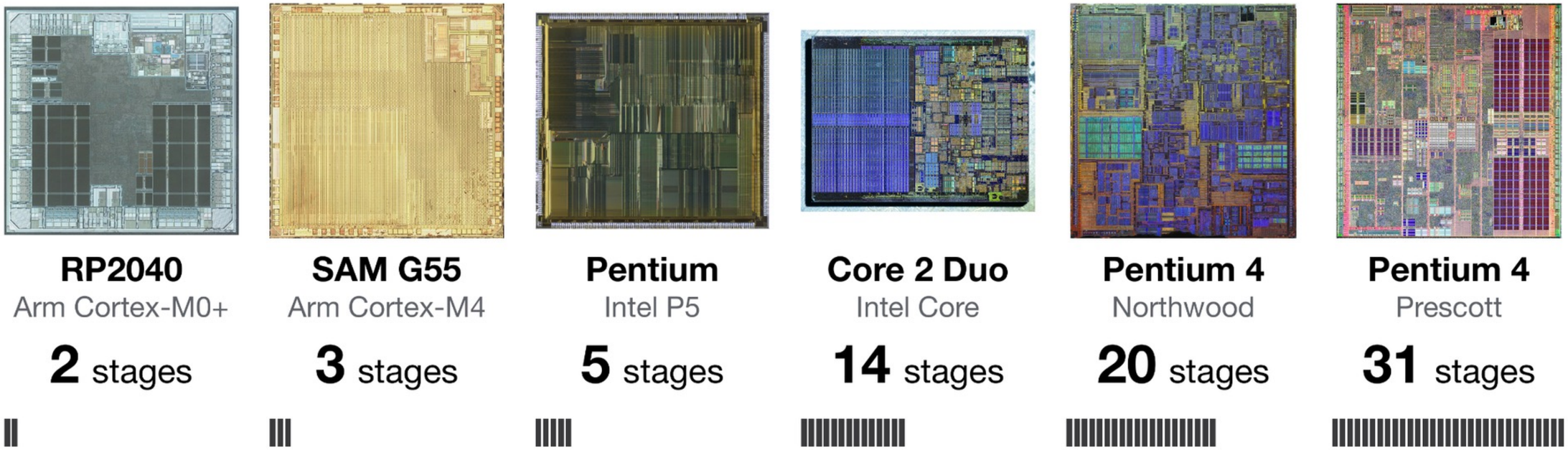


Instructions have depth ‘type’





What is the best architecture? What is the best depth?



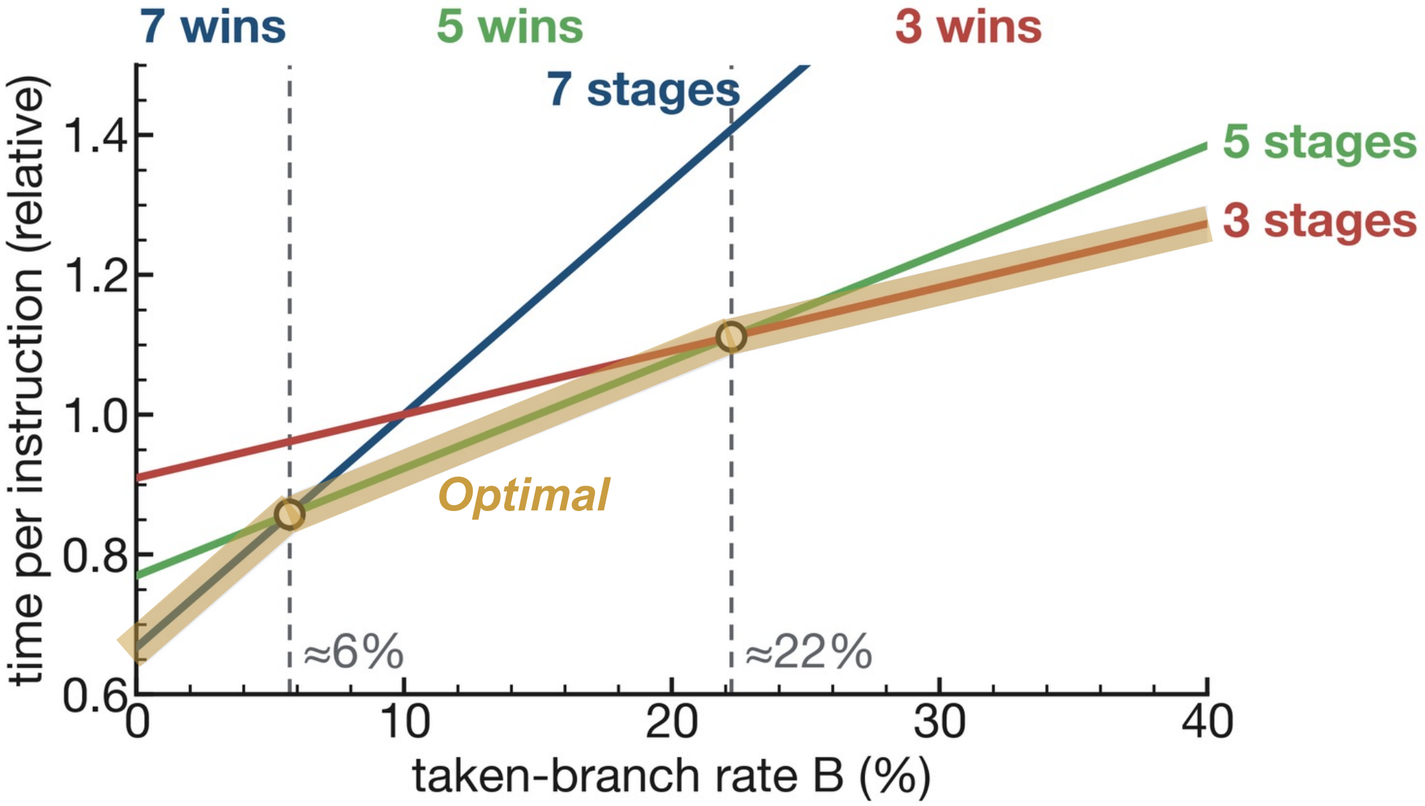
Die photos via Wikimedia Commons: J. McMaster (RP2040, SAM G55; CC BY 4.0), cole8888 (Pentium; CC BY-SA 2.0), M. Boer (Core 2 Duo, Prescott; public domain), Fritzchens Fritz (Northwood; CC0).

- Intel swung from deep, clock-first pipelines (Pentium 4, 31) to shallower ones (Core 2, 14)
- *"**workload-dependent optimal depth**"* (Hrishikesh et al. and Hartstein, 2002)
  - **chip fixes its depth at design time –** for every program it will run

Depth trades clock speed for flush cost

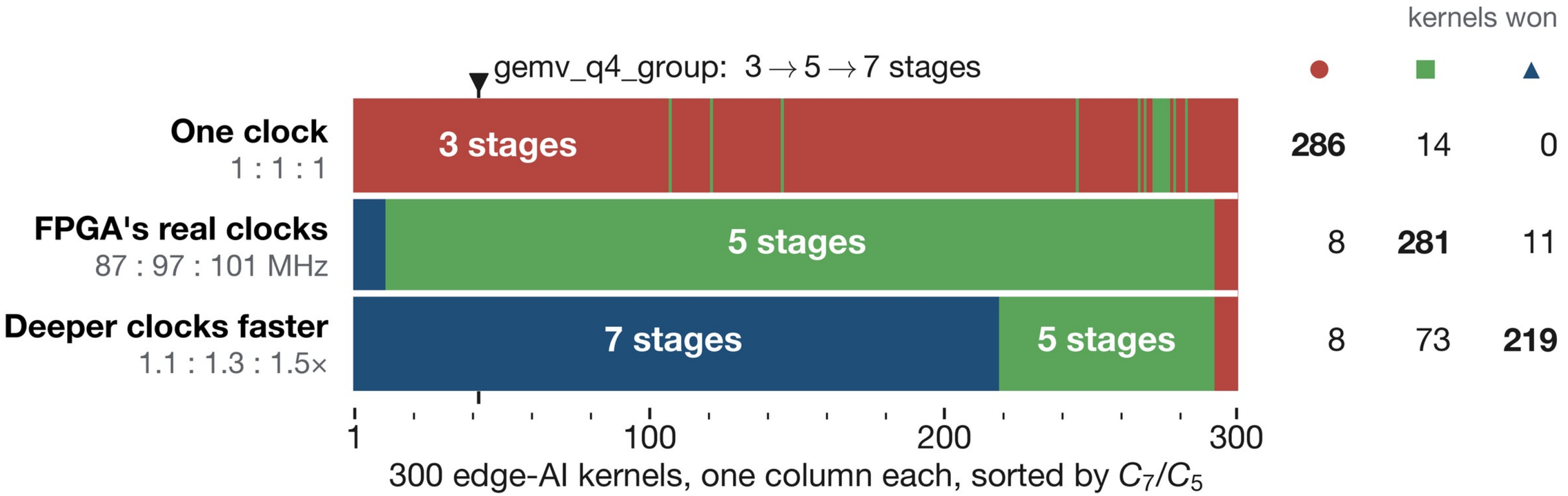
$$T = N \cdot \mathrm{CPI}(d) \,/\, f(d)$$

N: instructions, *CPI(d):* cycles per inst, *f(d)*: clock at depth *d*



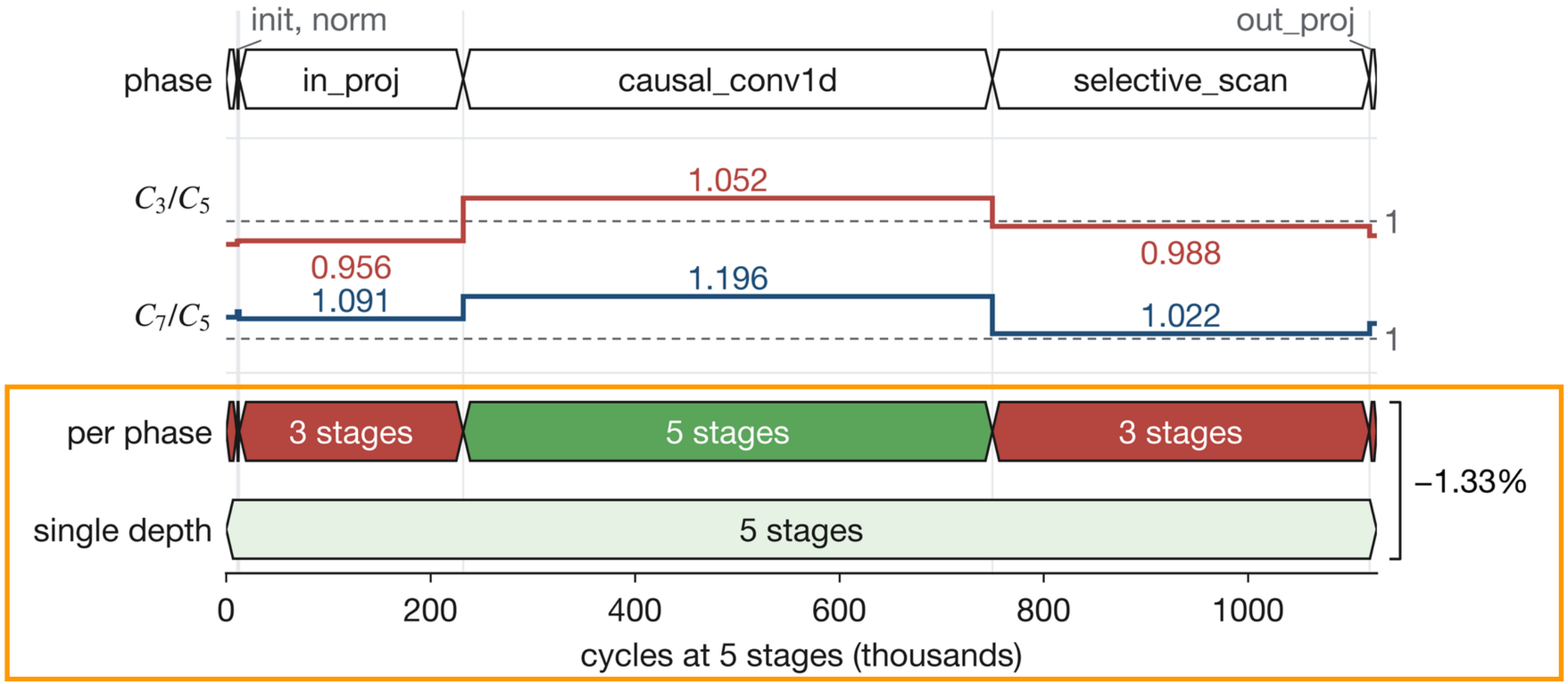
- Deeper: less logic per stage, so a **faster clock (1.1 : 1.3 : 1.5;** *analytical*)
- No branch predictor: each taken branch **flushes** 1, 2, or 5 cycles at 3, 5, 7 stages
- **Why 3, 5, 7:** 4 or 6 need *extra hazard logic,* add no new operating point – the symmetric merges of one 7-stage pipeline

Clock and best depth are correlated



- 300 edge-AI programs, cycles from RTL
- One clock, real clocks, deeper clocks **all have different results** ⟶ **arch + clk control**

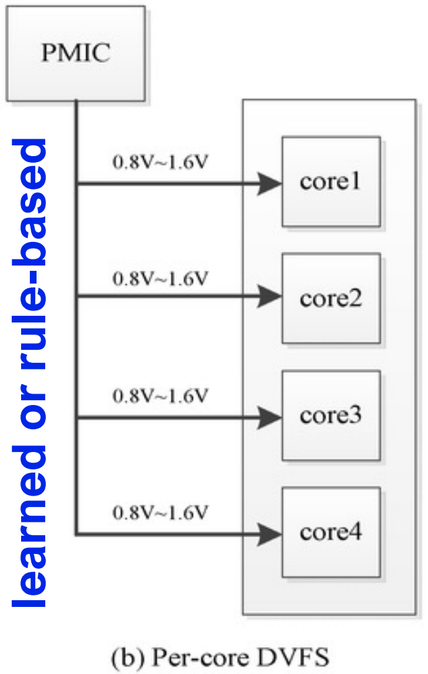
Best depth inside one program



- Mamba-style state-space block with three main phases (3 ⟶ 5 ⟶ 3)
  - Best single depth (5 stages): 1,125,967 cycles; switching per phase: 1,111,014 (−1.33%)
- **Depth chosen at runtime,** inside program

Prior works

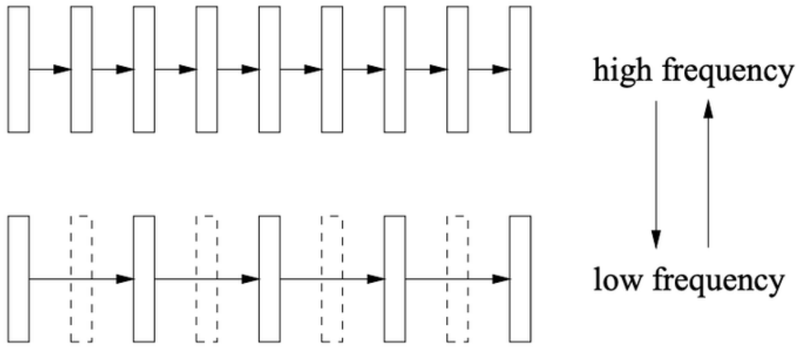
DVFS variants



(b) Per-core DVFS

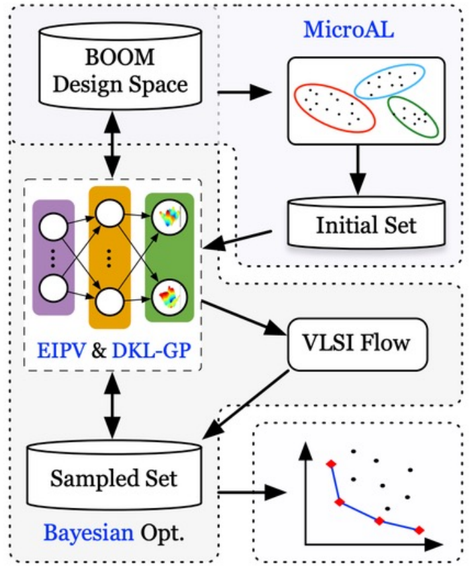
Adapts voltage & clk

Reconfigurable (Datapaths)



Merging stages (power heuristics), or second datapath

Learned Control



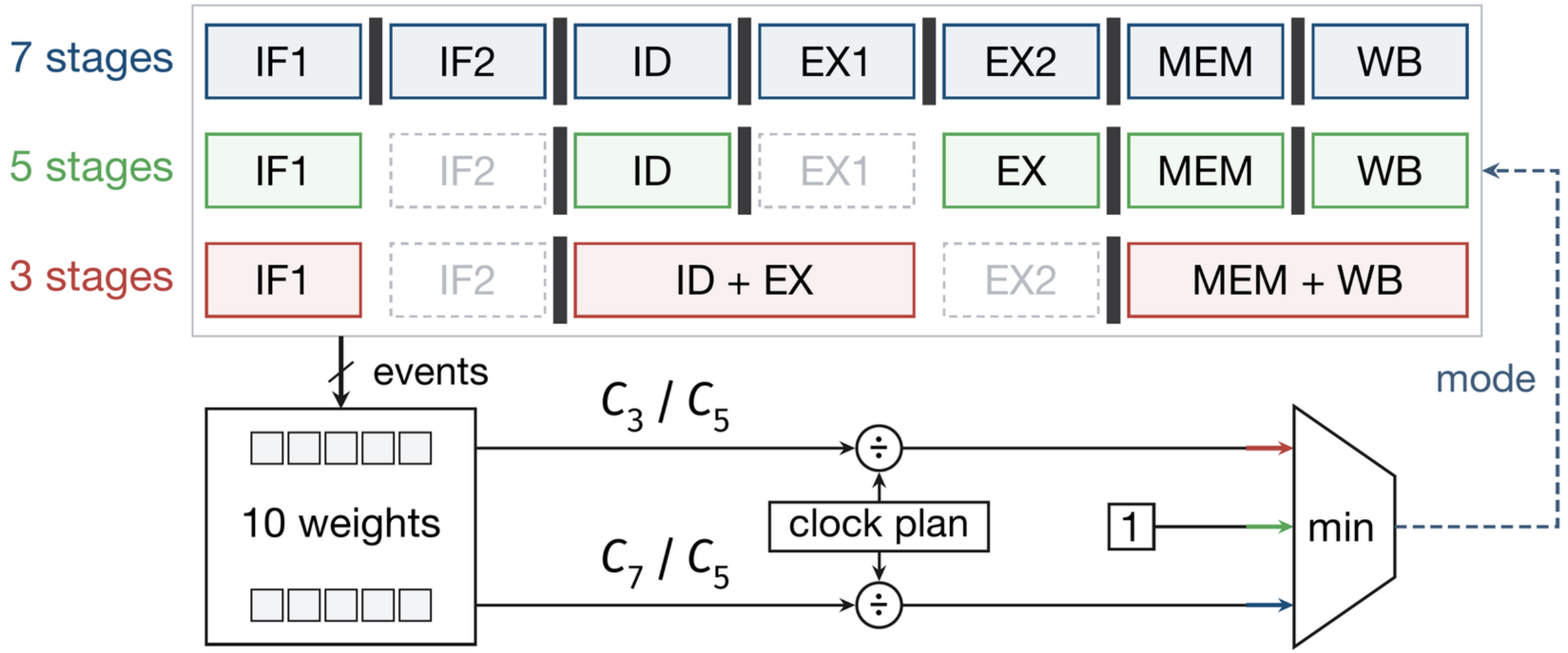
Learns other decisions (branching, memory, etc.)

Architecture & Datapath



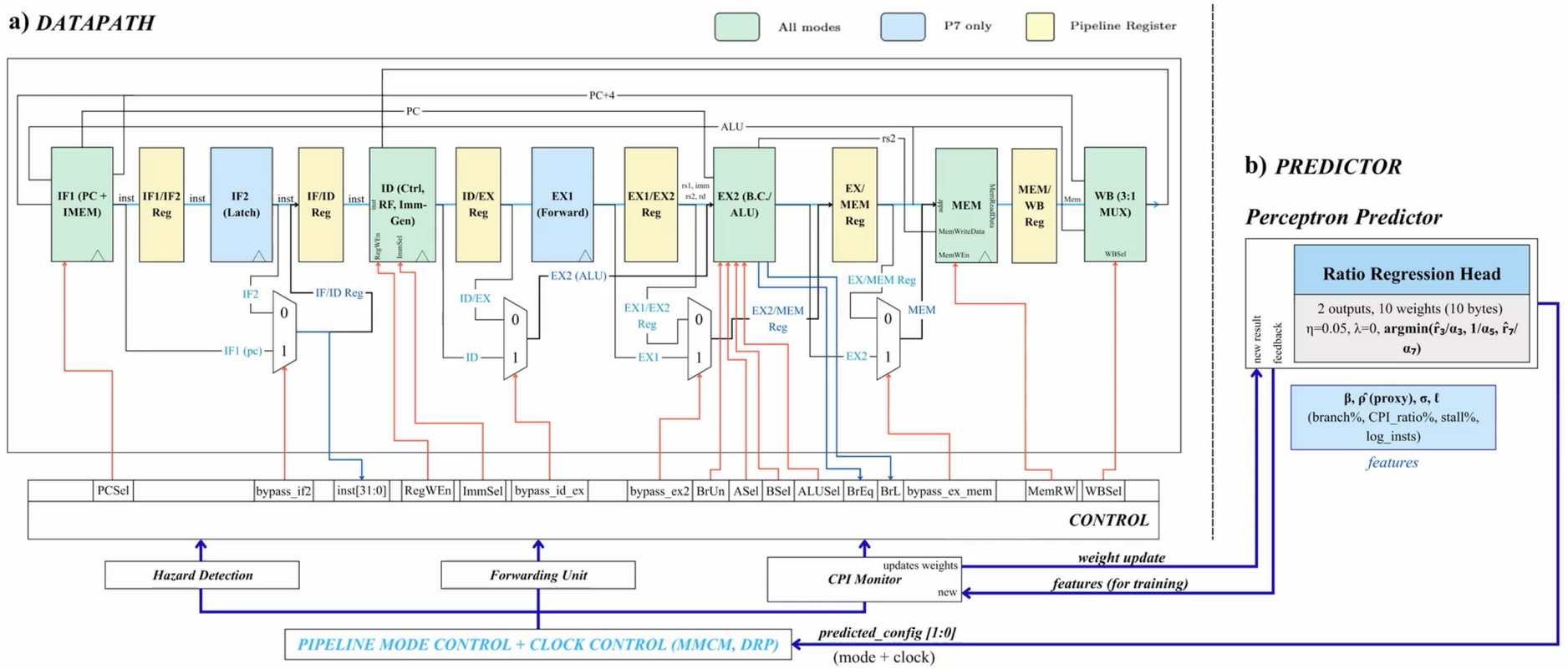
PDPM Explanation

PDPM depth chosen at runtime, inside one datapath, by a learned head



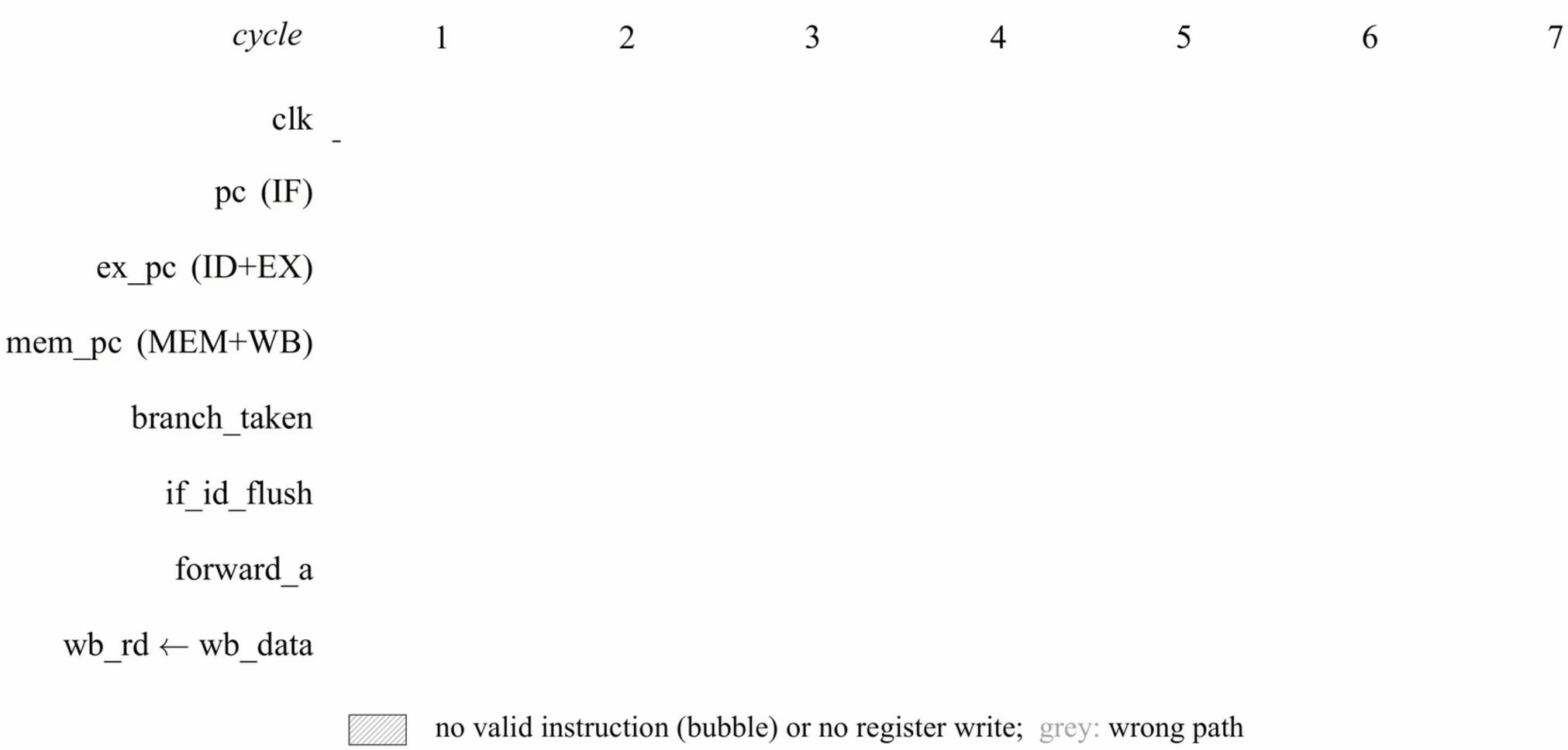
- One 7-stage **superset pipeline** (3, 5 merged)
  - *4 bypass **muxes** beside six pipeline registers*
- Every cycle, hardware counts events: flushes, stalls, cache misses
  - *Learned weights – two **ratios**: cycles at 3 and 7 stages vs. 5*
- Perceptron predicts and switches to the fastest depth real-time

PDPM seven-stage superset (P3)

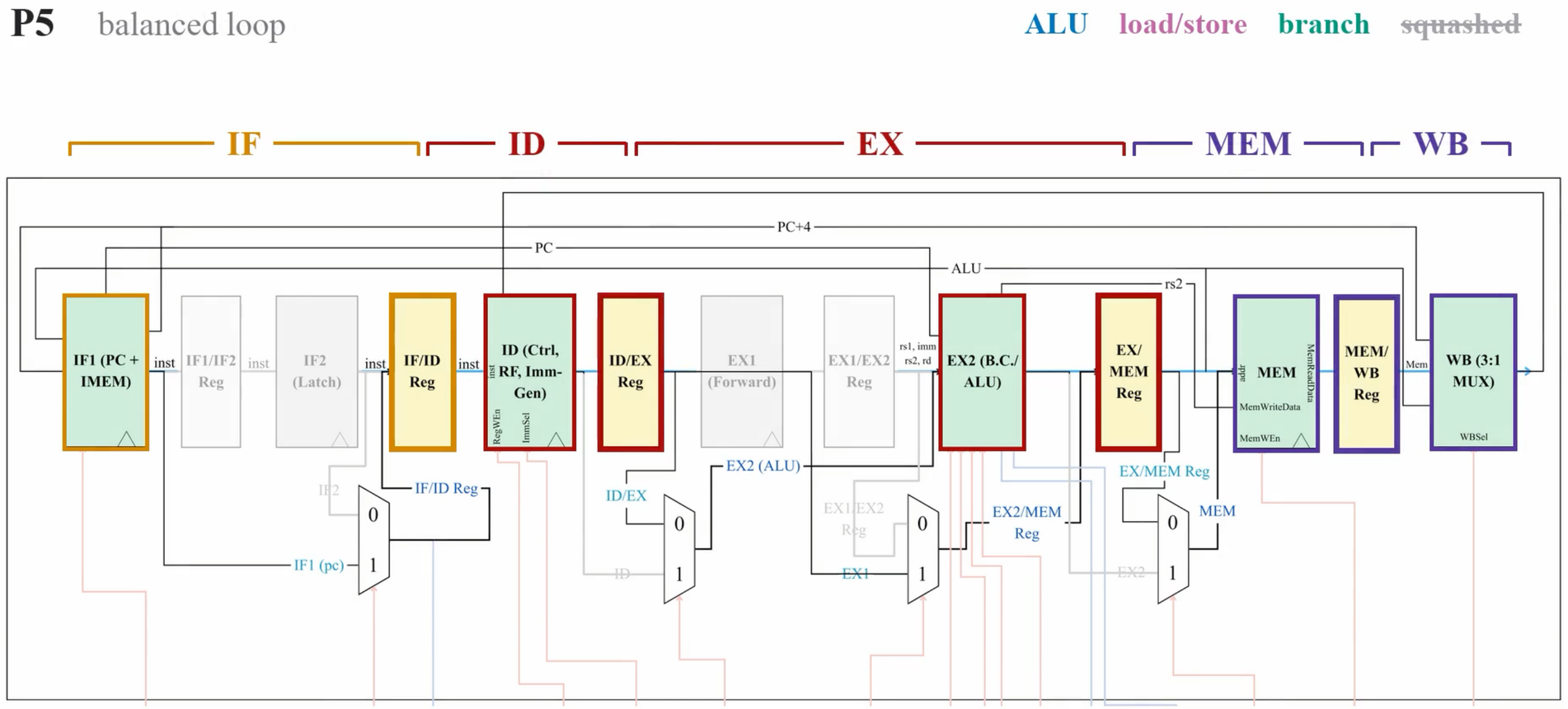


- **Bypass all four** ⟶ 3 stages
- *ID + EX2 goes to EX/MEM*

PDPM seven-stage superset (P3 timing)

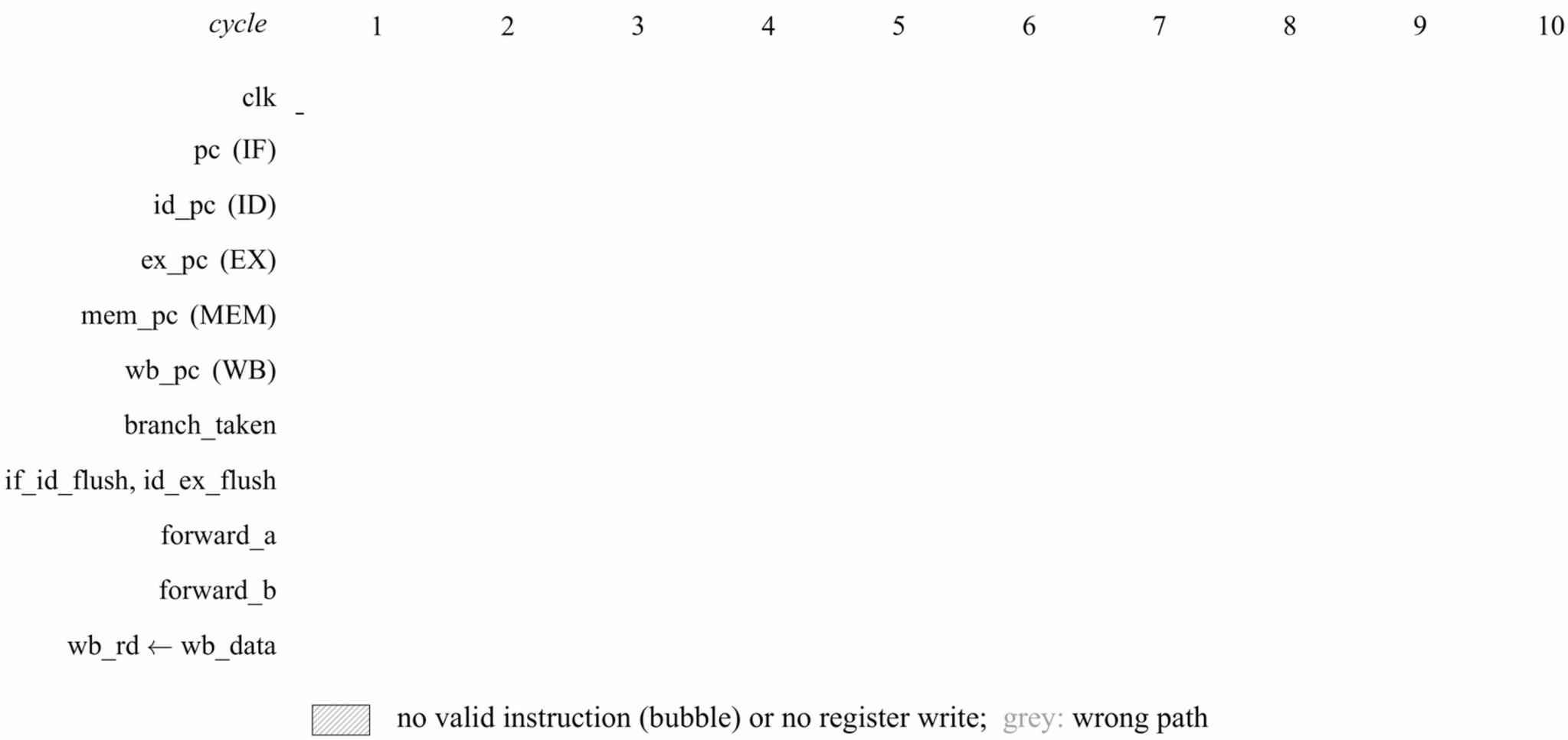


PDPM seven-stage superset (P5)

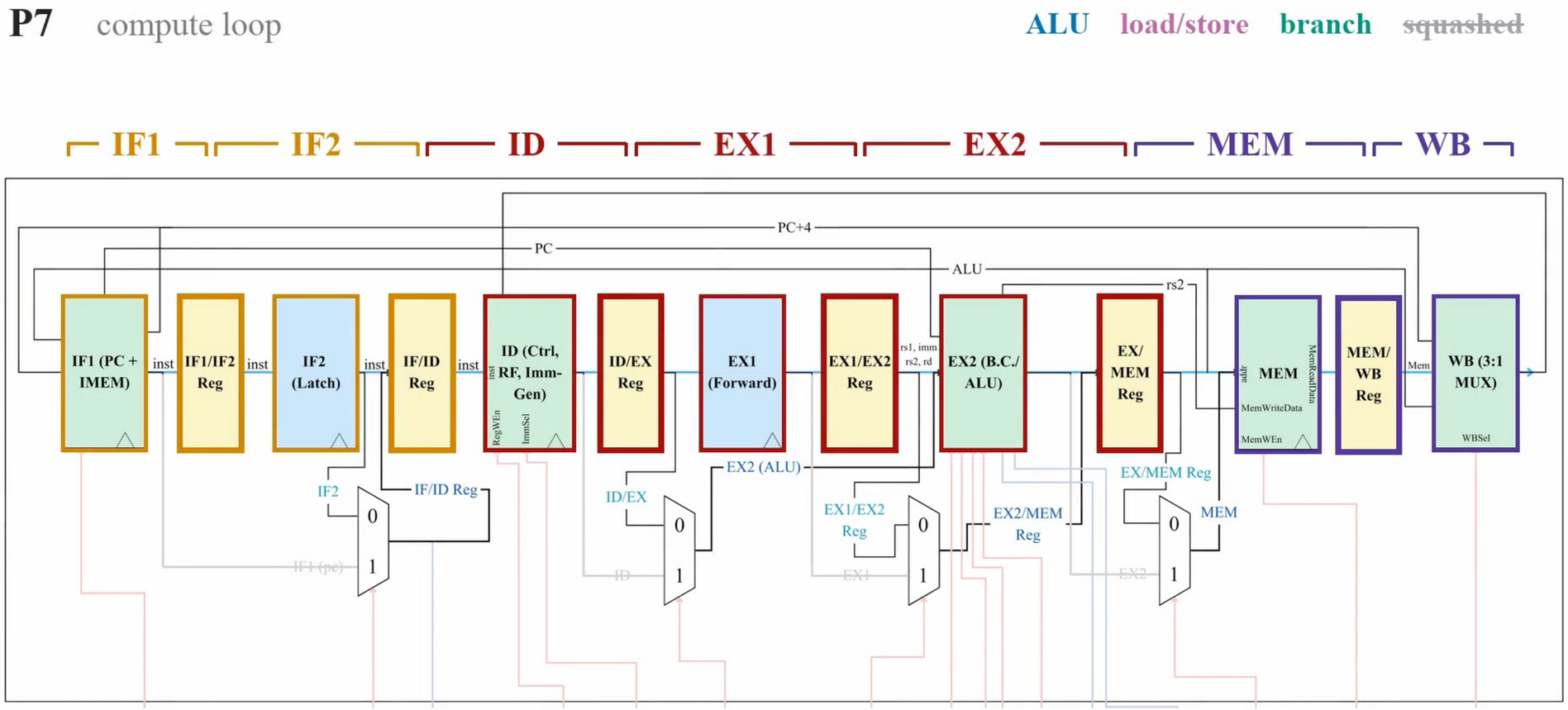


- **Bypass IF2 & EX2** ⟶ classic 5 stage

PDPM seven-stage superset (P5 timing)



PDPM seven-stage superset (P7)



- **Bypass none**

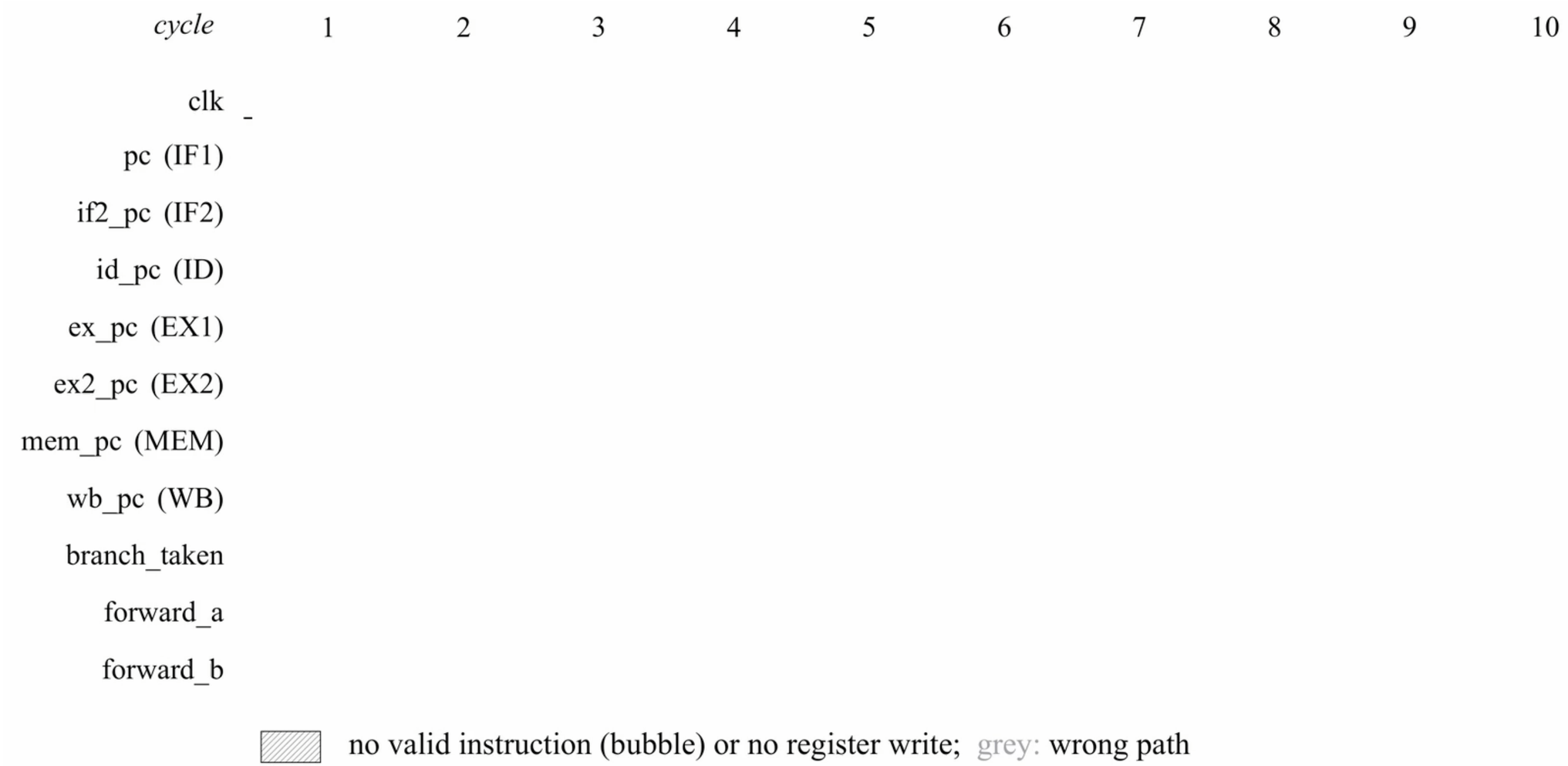
IF1**: compute **next PC

***IF2**: **read** instruction (I-cache)*

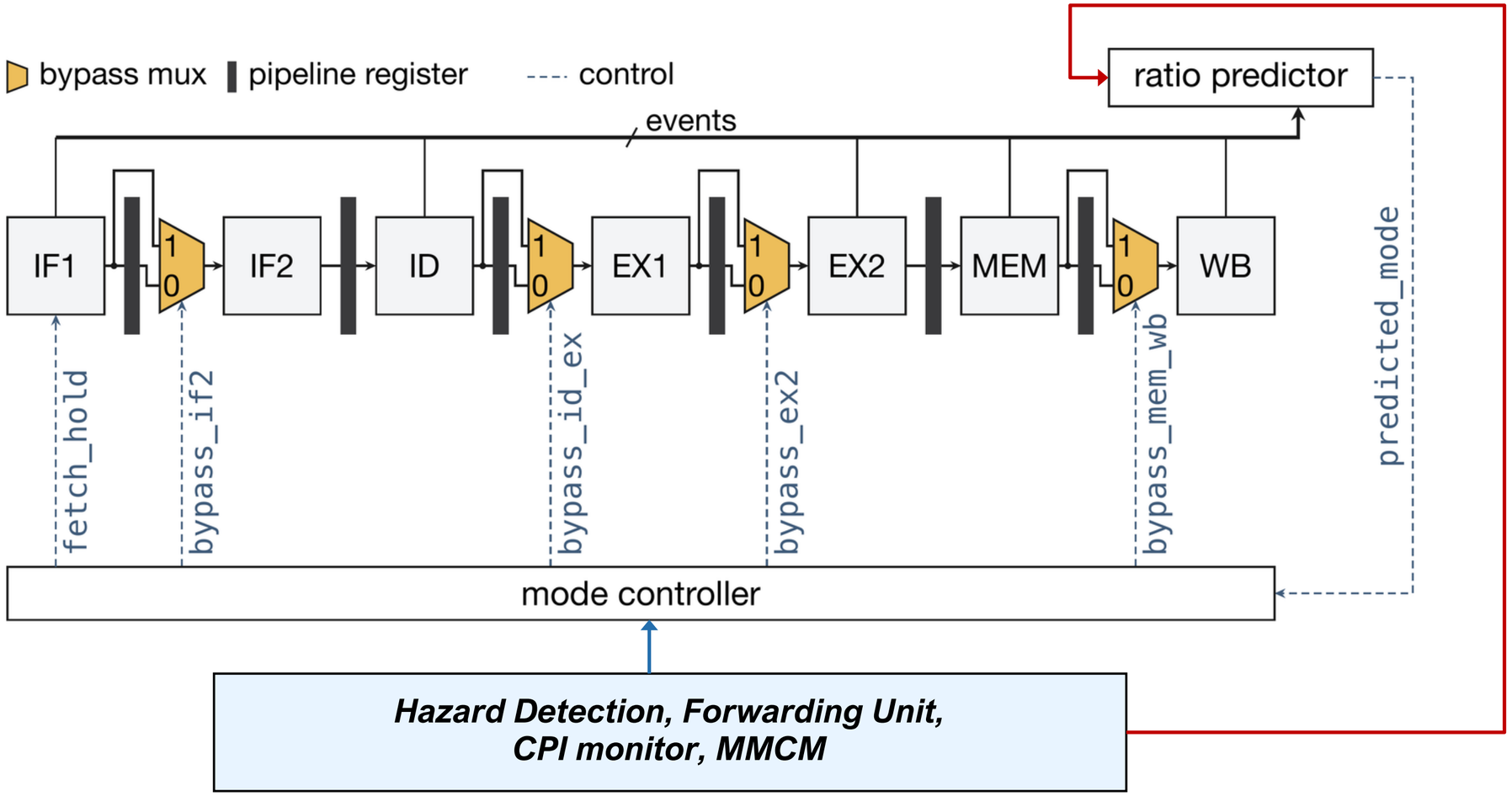
***EX1**: ALU operation **begins**, **operand** selection*

EX2**: ALU **completes**, **branch resolution

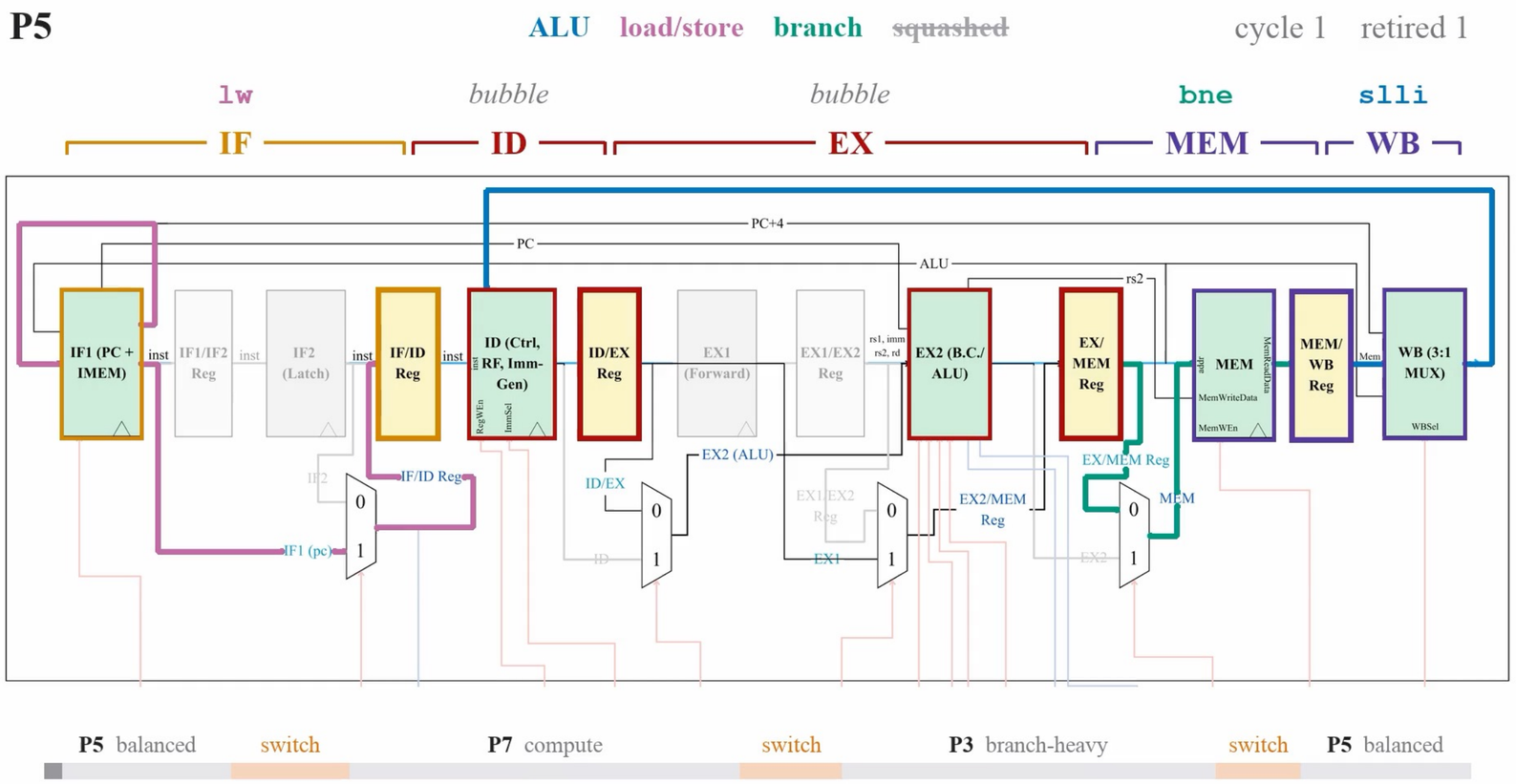
PDPM seven-stage superset (P7 timing)



PDPM simplified architecture



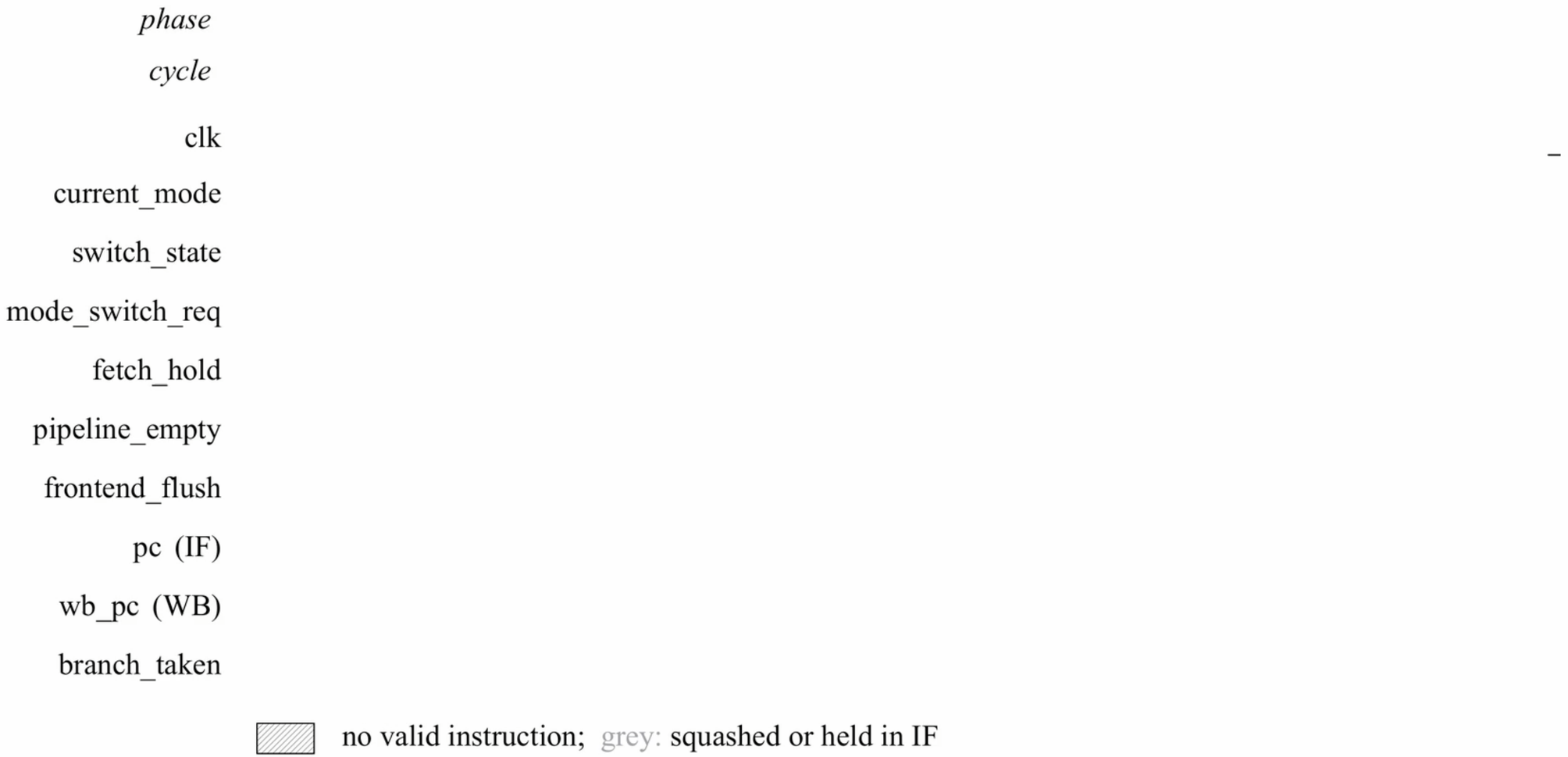
Dynamic Switching Modes



- **Pipeline Switching**: Drains take 3–5 cycles, safely completing all in-flight instructions before switching depths.
- **Clock Switching (if needed)**: MMCM reconfiguration takes 21.5 µs, ensuring safe frequency changes but preventing per-window switching.

Hold fetch ⟶ **Drain** until empty ⟶ **Flip** modes

Dynamic Switching Modes (timing)

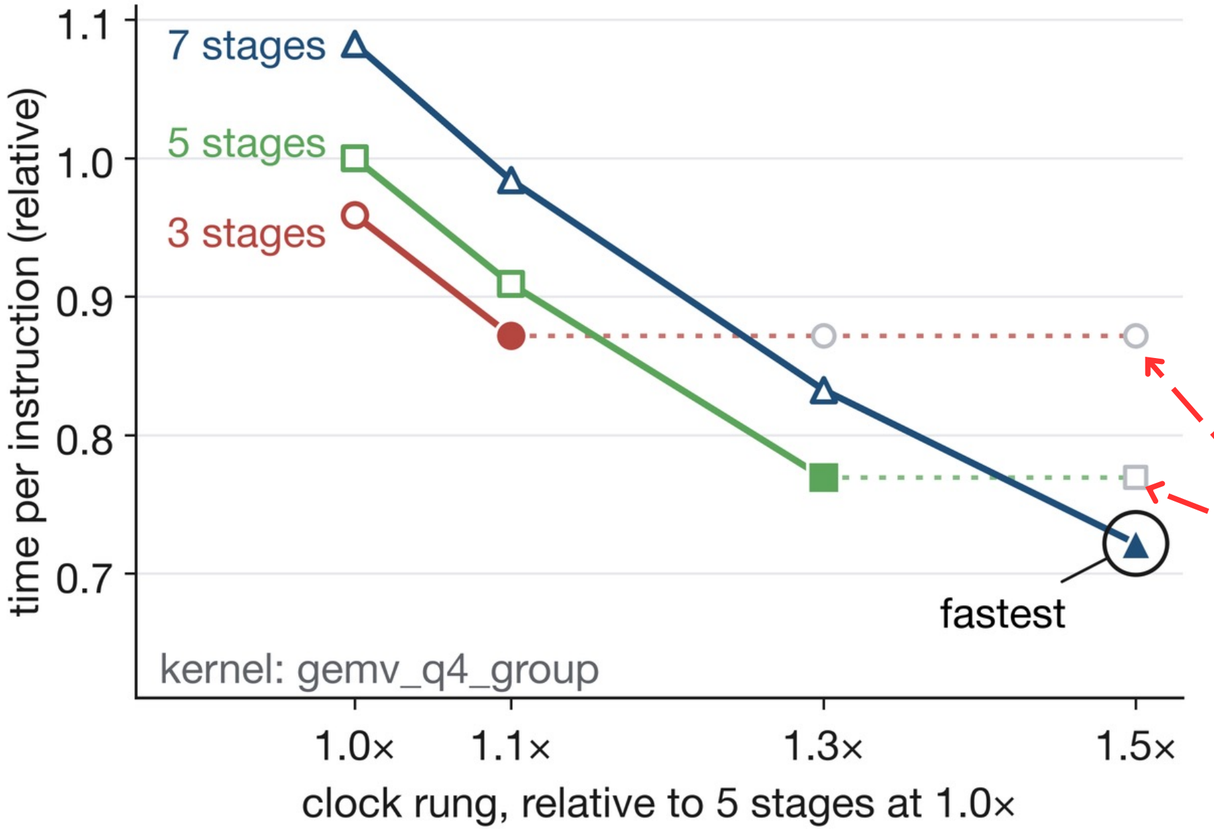


Learned Depth Predictor



How to switch? Classification?

- supervised, **classifying** 3 depths x 4 clock rungs = 12 configs, 60 weights



maximum clock

slower clock

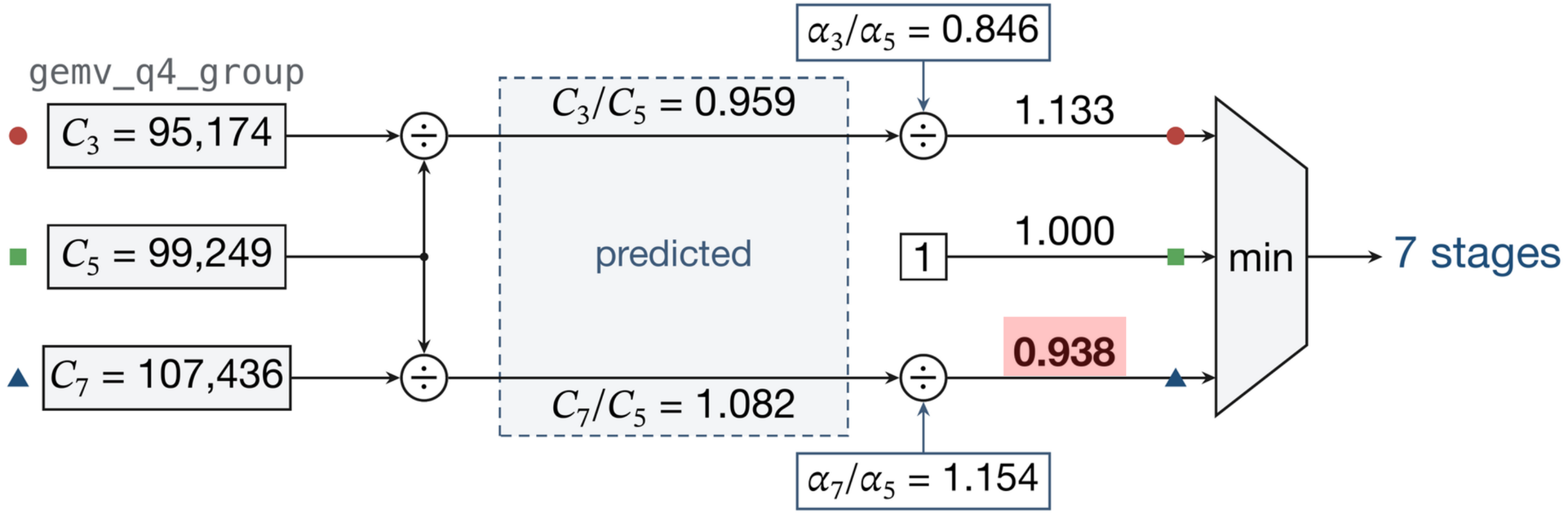
above its cap

WASTE!

Depth's cycles don't change with clk, slower rung **loses** (6)

Above a depth's top clk, **it runs at that clk** (3)

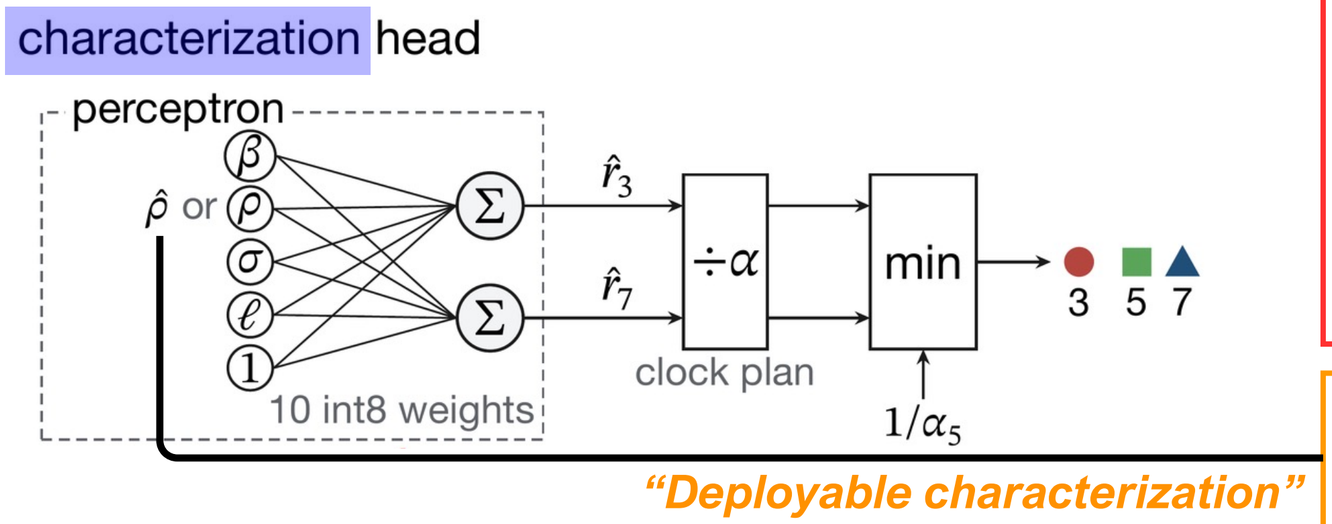
Regression, two numbers decide the depth



A depth's **time** is its cycles over its clk: $T_d = C_d/\alpha_d$

- Divide all three times by T5: program length cancels; only these remain: C_3/C_5, C_7/C_5
- Head predicts **these two ratios** – ***how long P3 or P7 will take more than P5***

Two heads of the predictor



- **evaluate ratio / depth choosing** under **ideal, identical** conditions, RTL sim traces (ratios, labels, etc.)

Is the ratio idea good, and the best option?

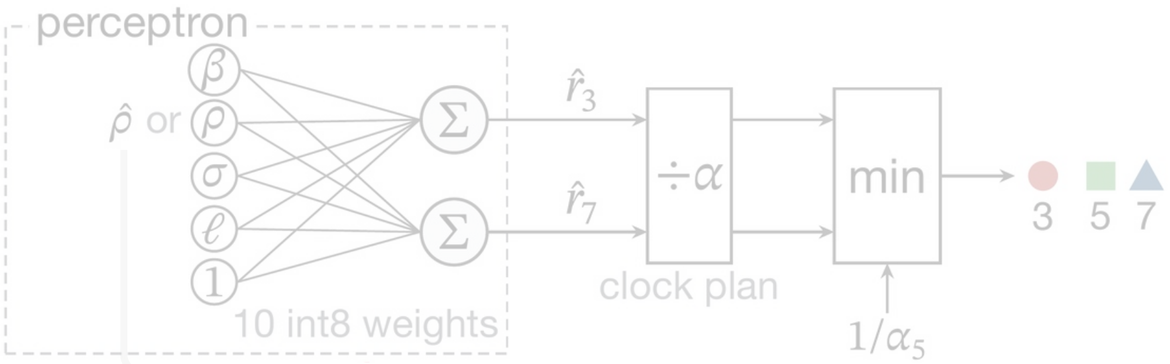
*Does it survive **realistic** inputs a real core could measure?*

*flush rate β, **depth sensitivity ρ**, load-use stalls σ, current length ℓ*

- β, σ and ℓ come from ordinary performance counters.
  ρ needs cycle counts from both P3 and P7

Two heads of the predictor

characterization head



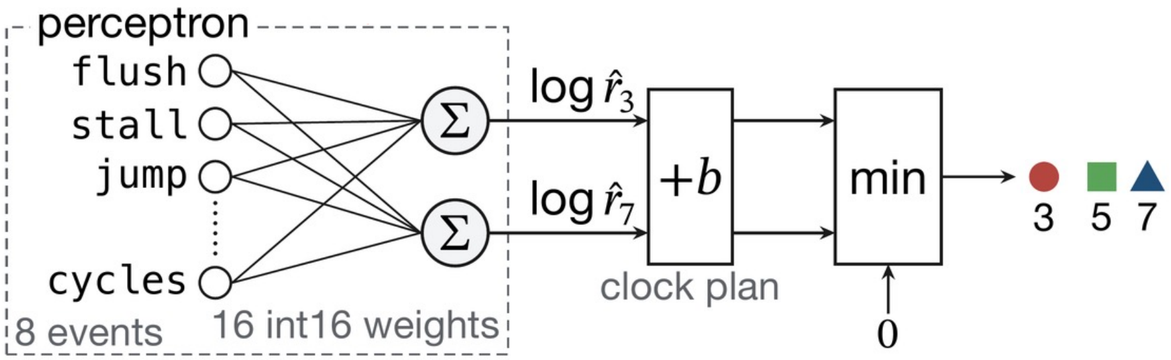
- **evaluate ratio / depth choosing** under **ideal, identical** conditions, RTL sim traces (ratios, labels, etc.)

Is the ratio idea good, and the best option?

"Deployable characterization"

*Does it survive **realistic** inputs a real core could measure?*

hardware head

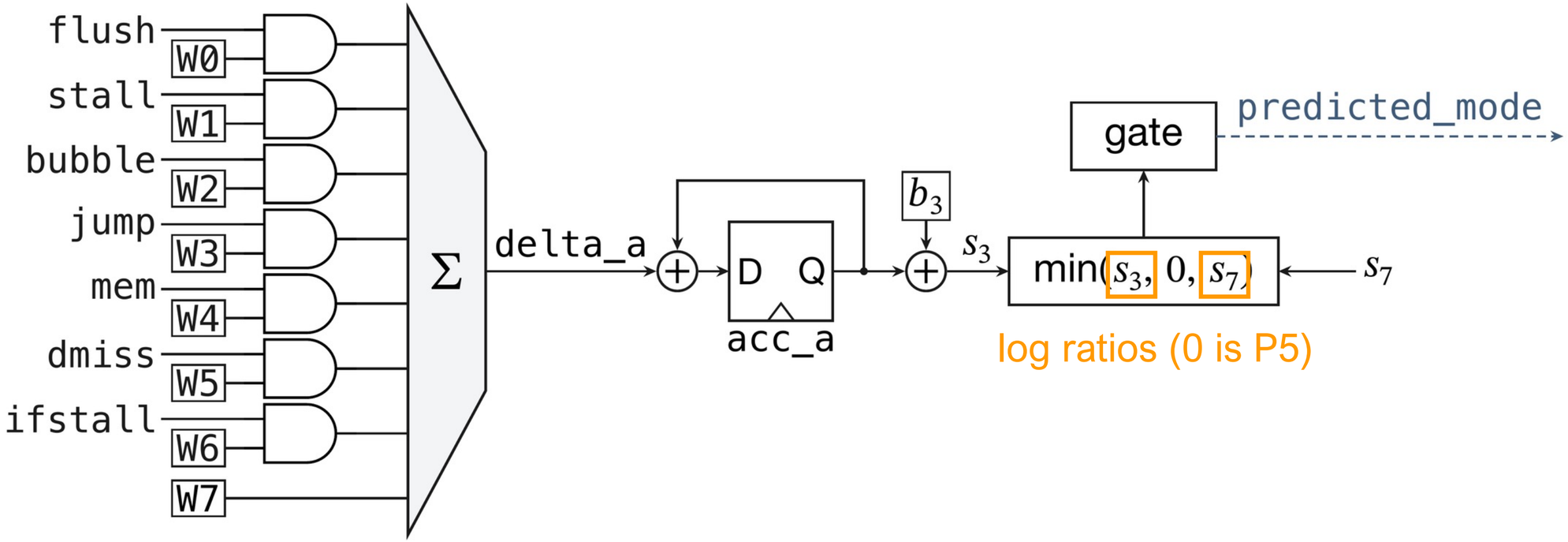


Now validated...

- running – sees only one depth, so it **reads event counts**

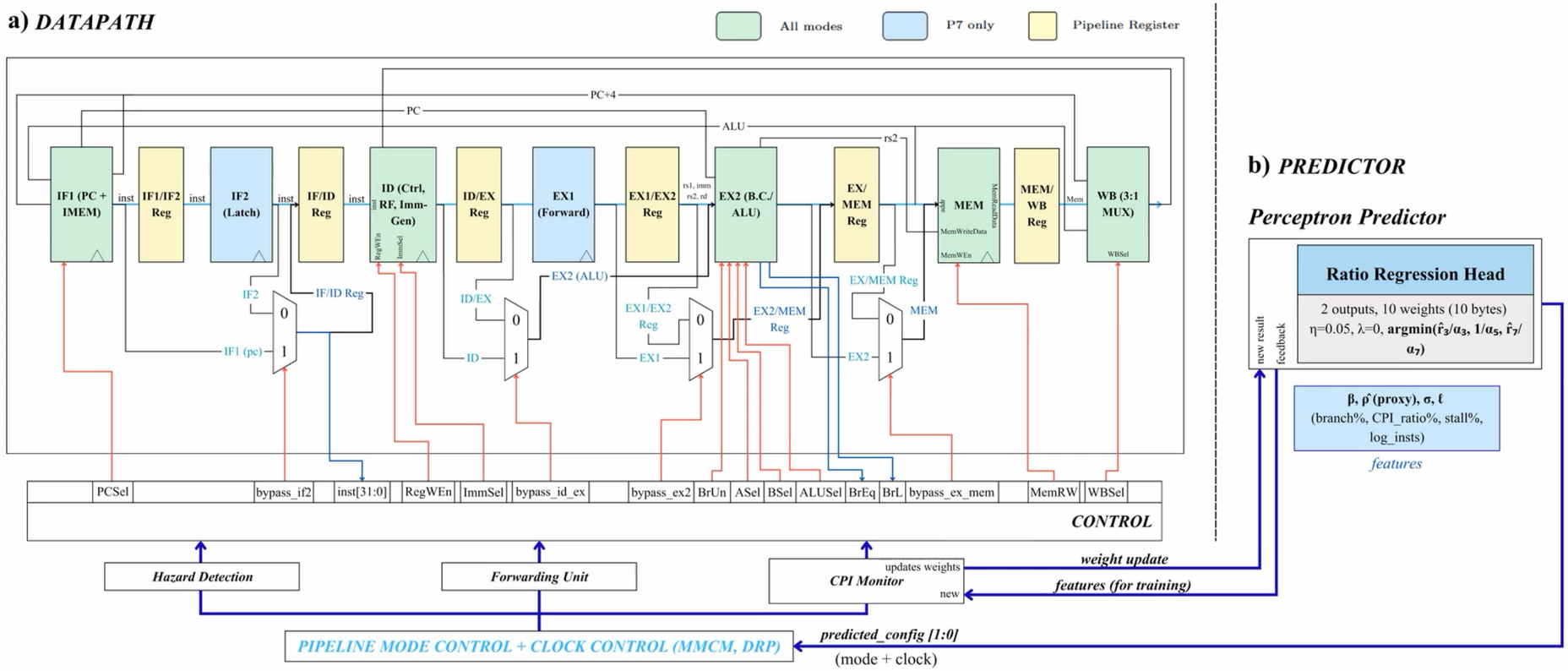
*Does it work on a **real** chip, on real workloads?*

Hardware Head



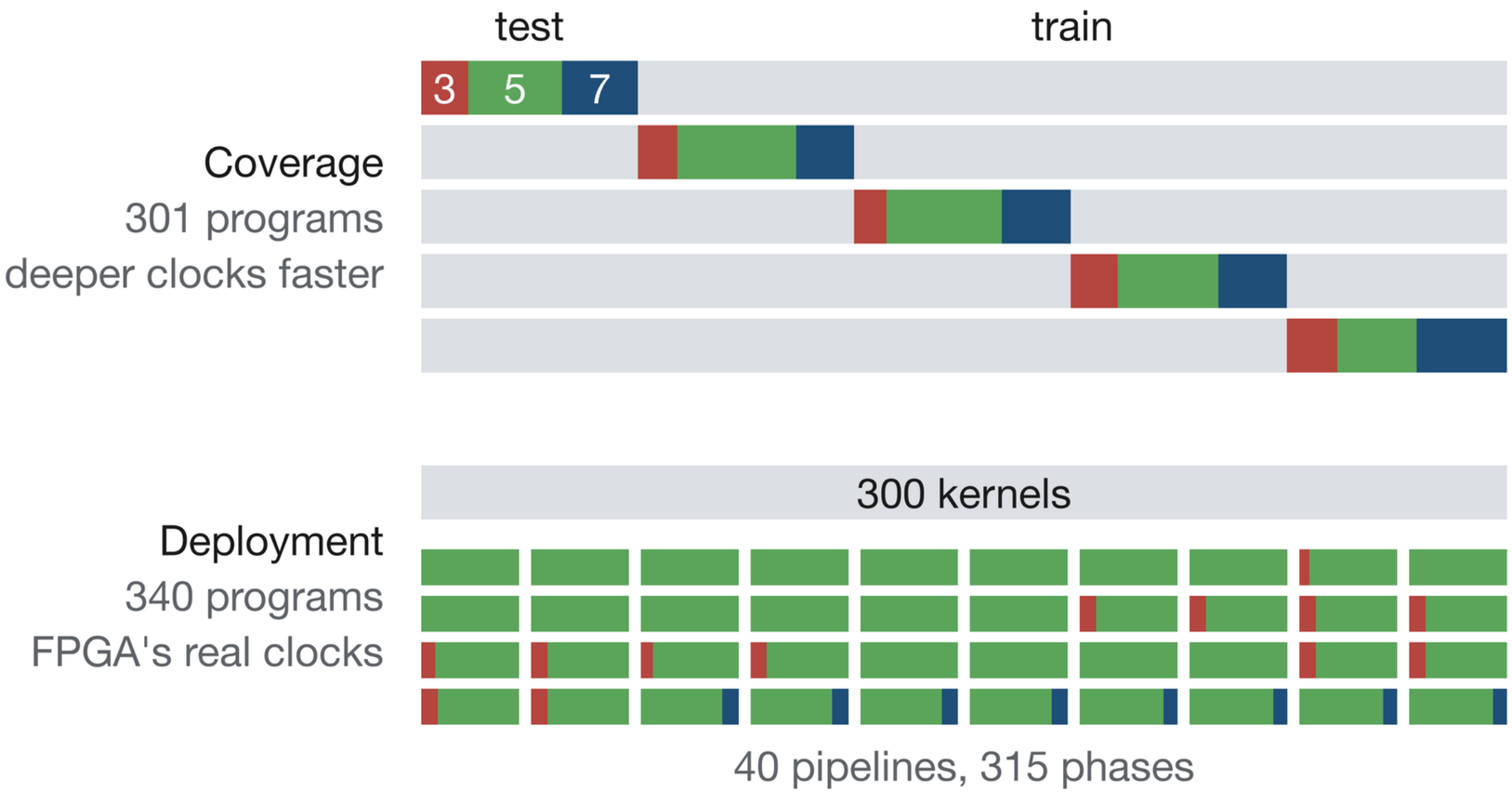
- Each event that fires adds its weight: **no multiplier**
- Smallest wins ⟶ becomes *predicted mode*

Predictor Pipeline



Results

Two benchmark suites



Characterization Head

predictor designs

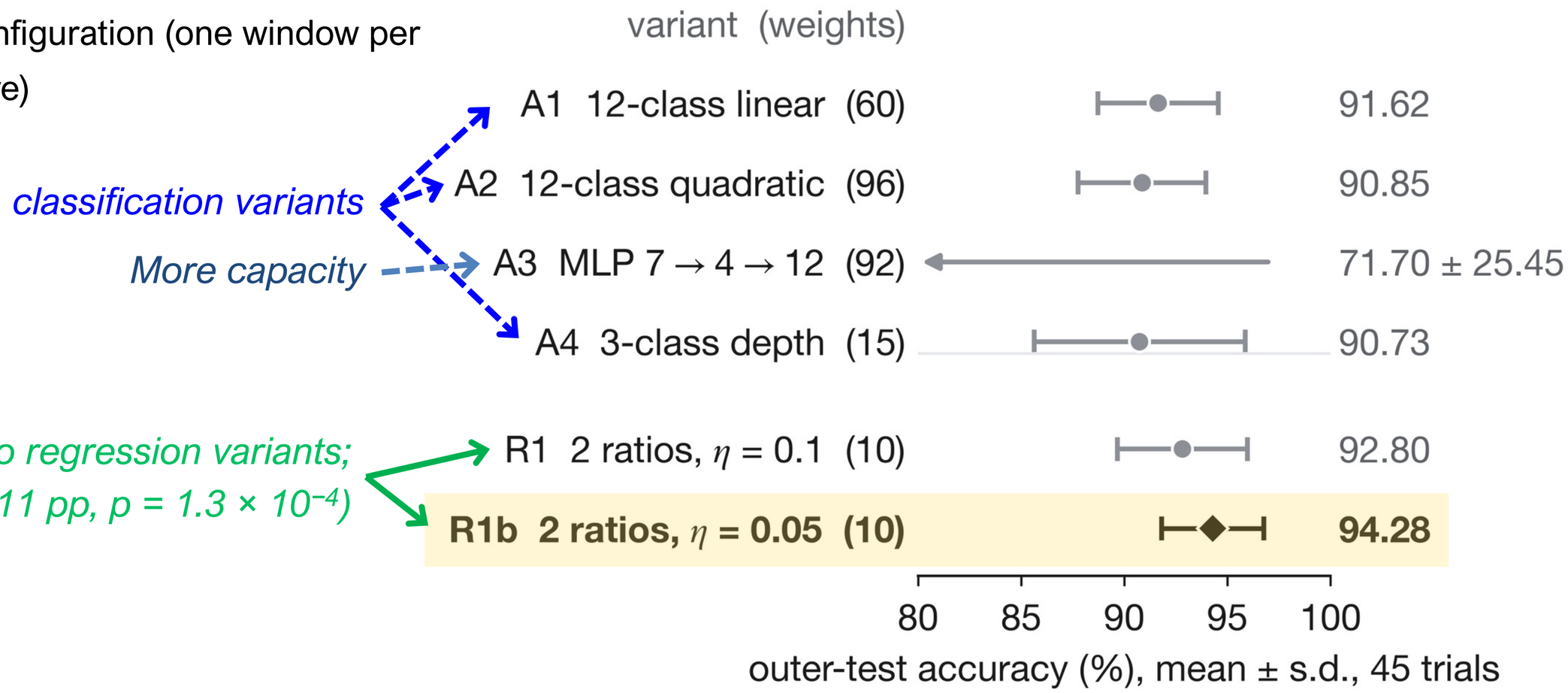
45 trials

Hardware Head

fit once, frozen

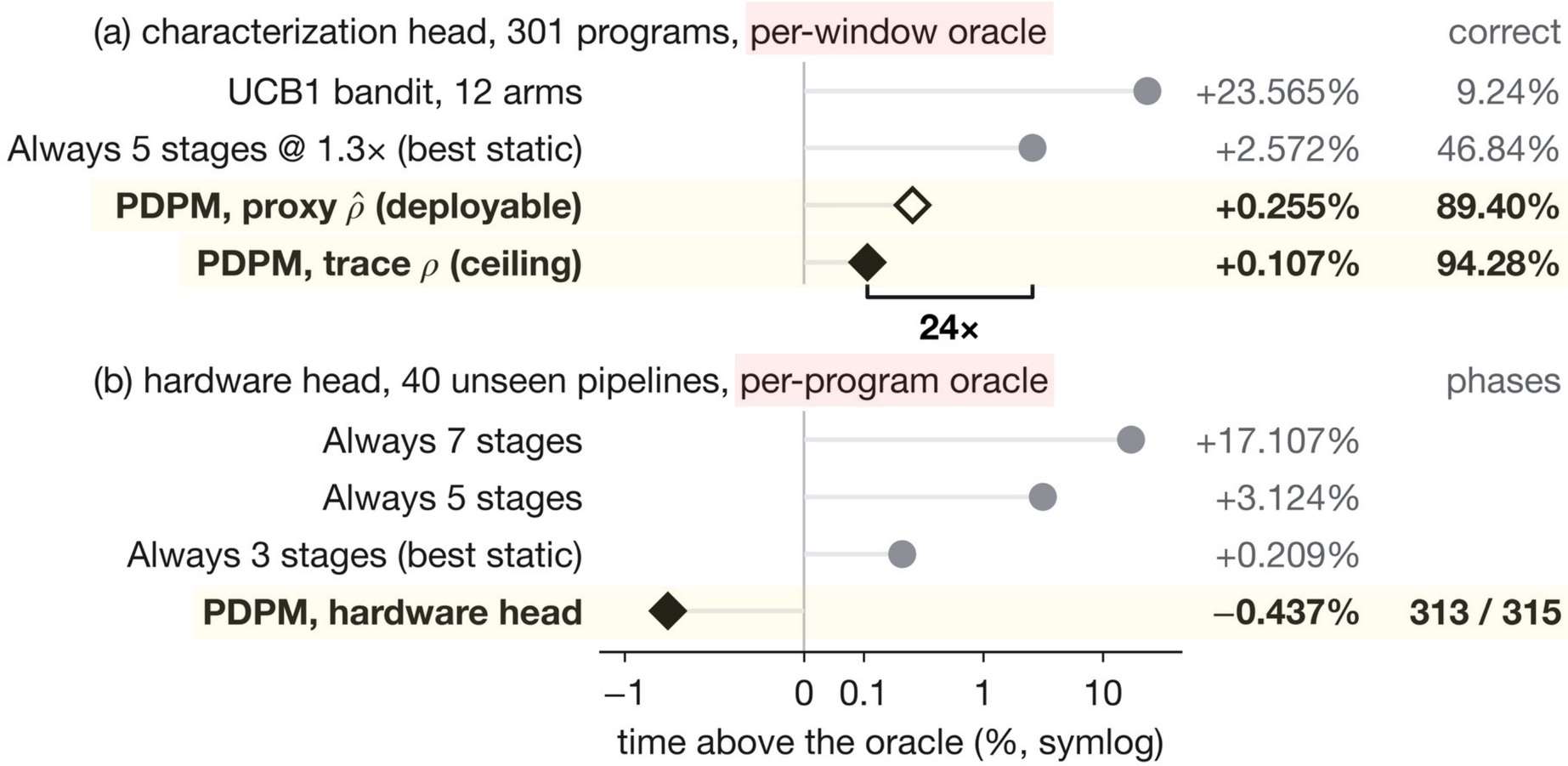
Coverage suite and ratio regression

Accuracy: share of **decision windows** on the oracle's configuration (one window per benchmark here)



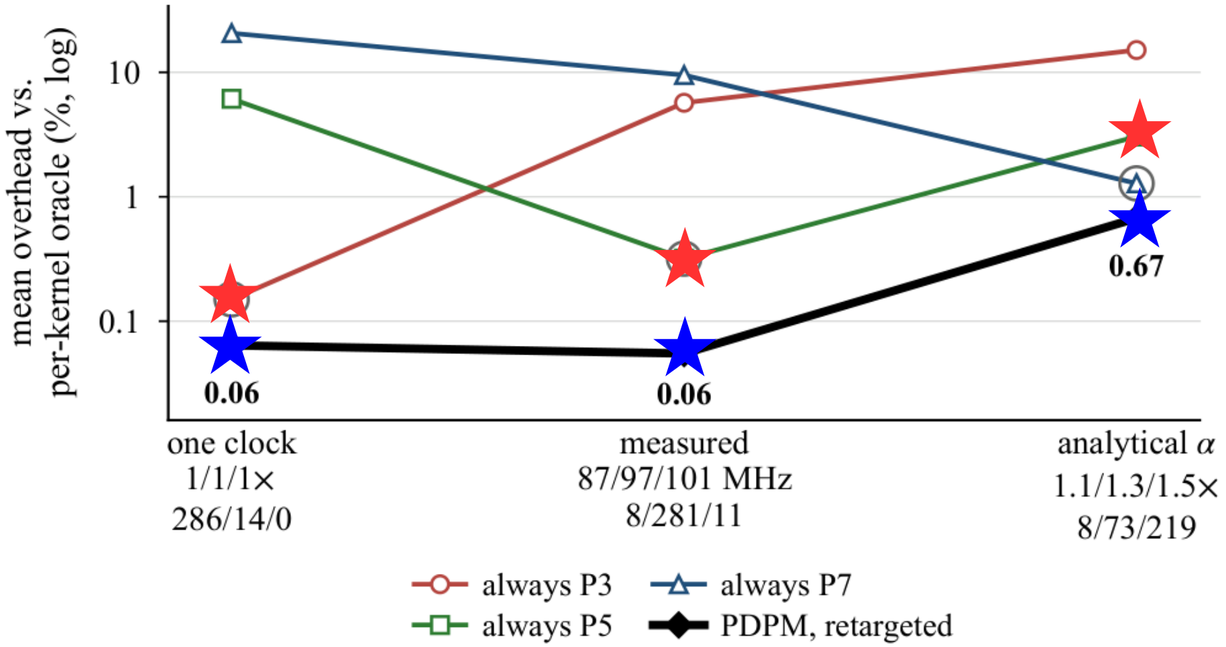
Every head beats static

- Overhead: **time above an oracle,** each form against the fixed depths of its own suite



Head results and clock

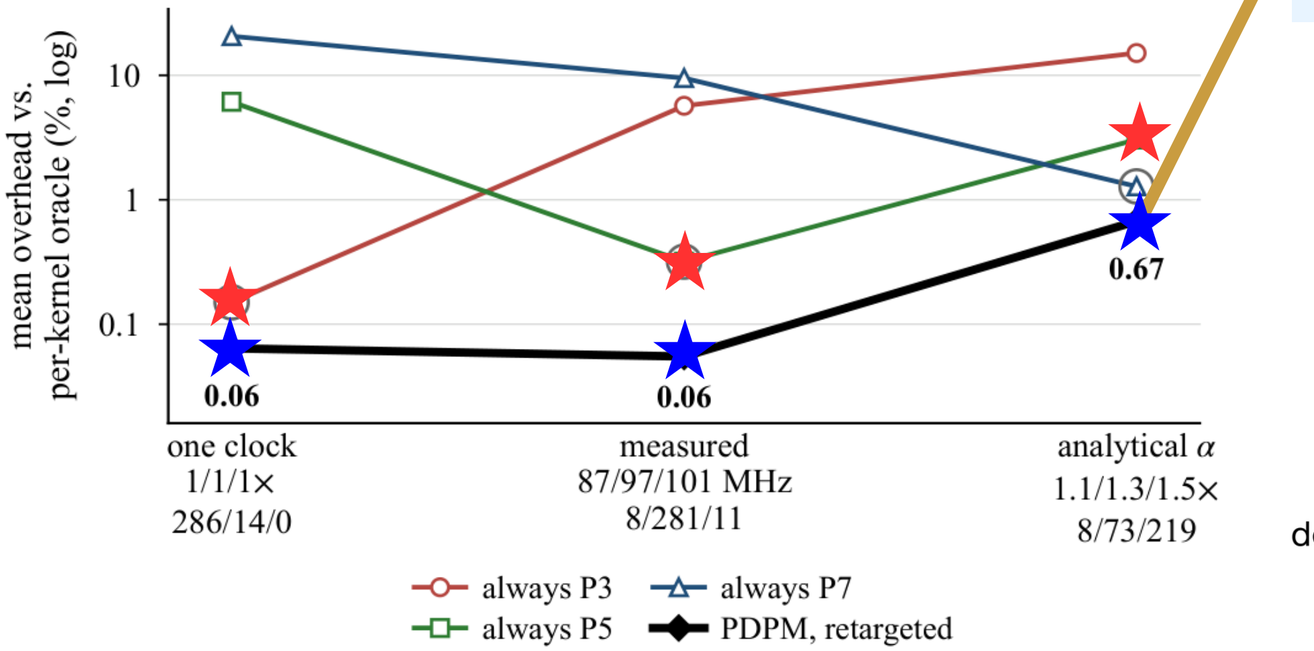
test 300 kernels fitted on + clk change



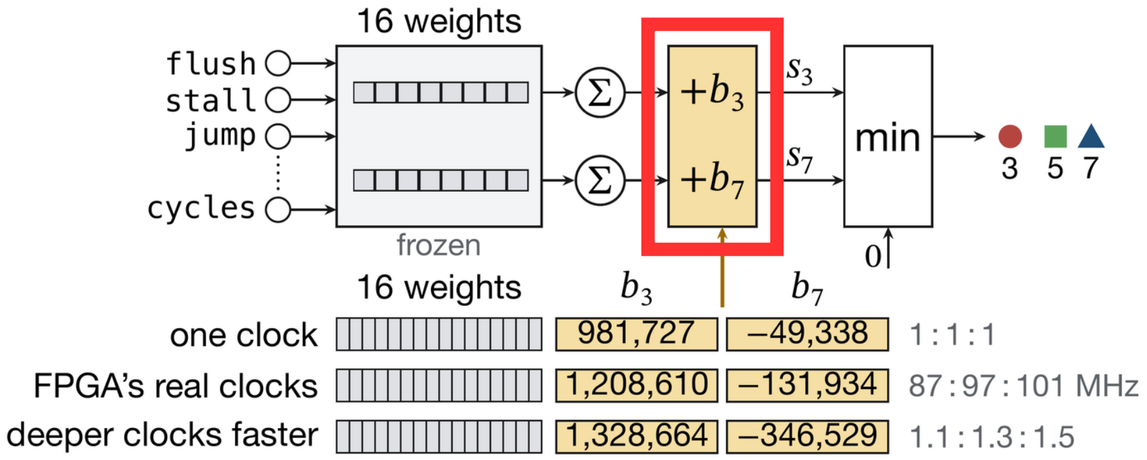
- **each fixed depth is best under one plan** and up to 20.6% worse under another
- hardware head, *weights frozen,* **beats** every fixed depth under every plan
  - new clk plan = *new two biases (no retraining) for hardware head*

Head results and clock

test 300 kernels fitted on + clk change

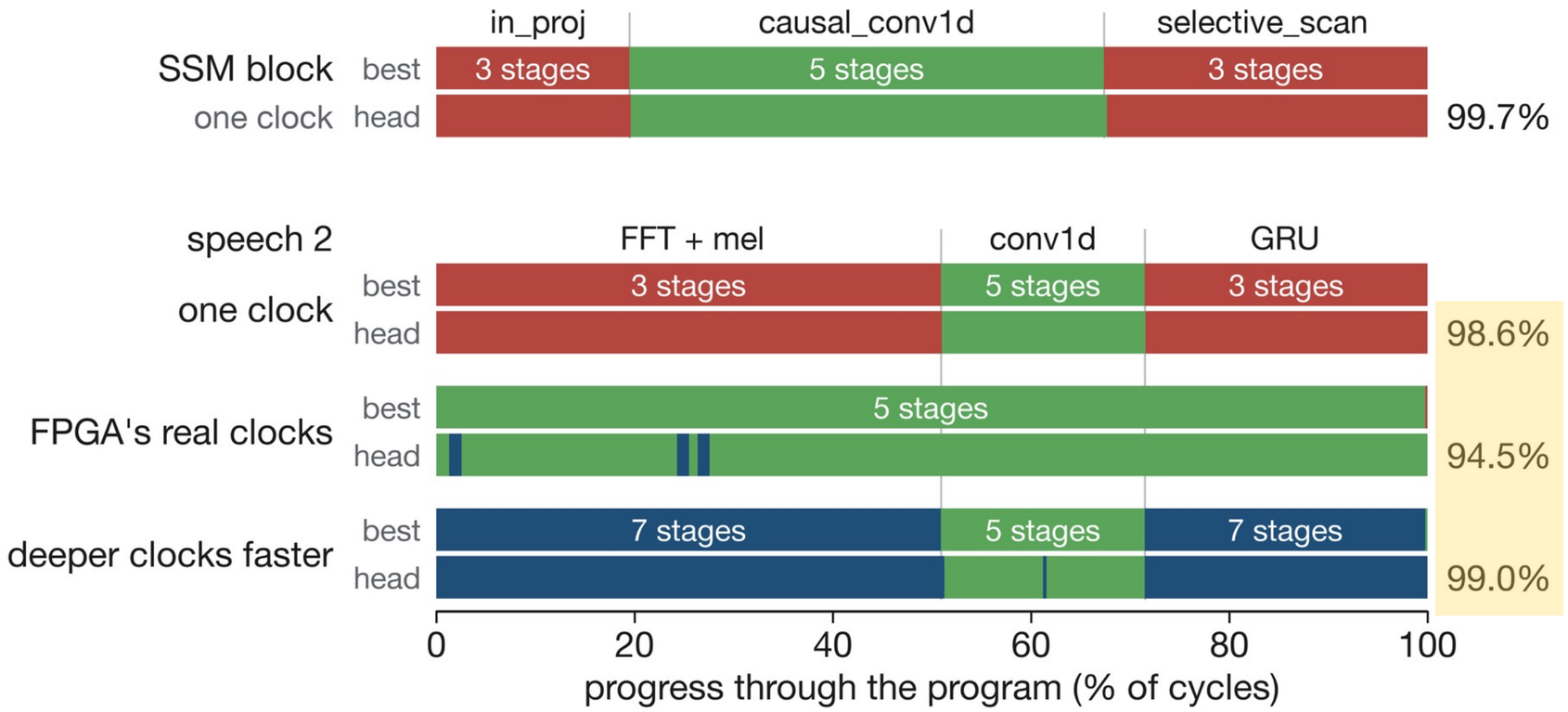


5 and 7 stages **nearly tie** on
122 / 300 due to P7 clocks faster
(< 2% clk cycles)



- **each fixed depth is best under one plan** and up to 20.6% worse under another
- hardware head, *weights frozen,* **beats** every fixed depth under every plan
  - new clk plan = *new two biases (no retraining) for hardware head*

Performance on unseen workloads

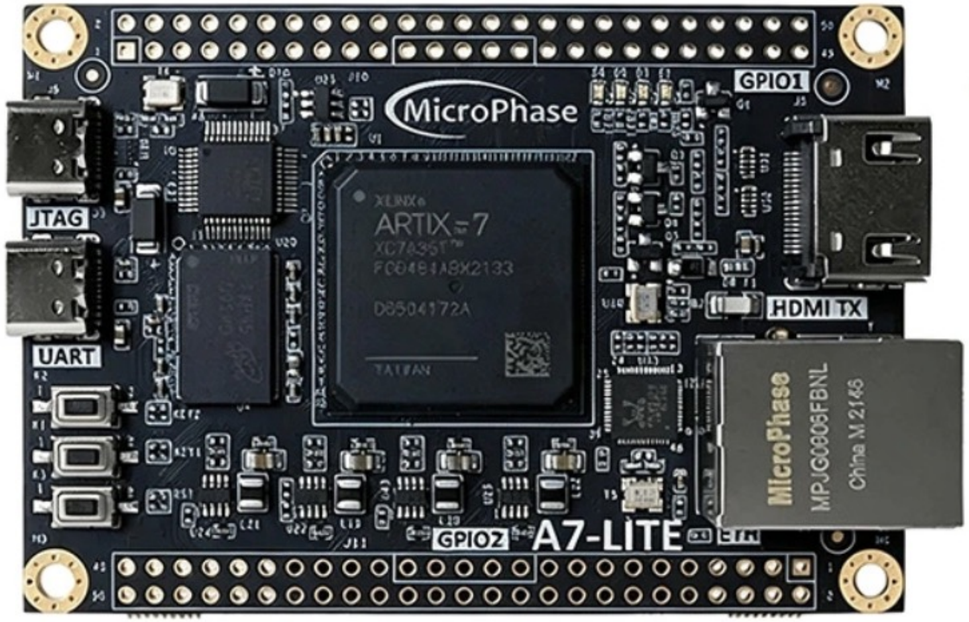


Adaptive to clk changes

40 composite AI real pipelines, 315 phases, never seen in training

- **hardware (real)** head picks best depth for **313 / 315** phases
- *decision (depth-3) tree* for **301 / 315**

FPGA numbers

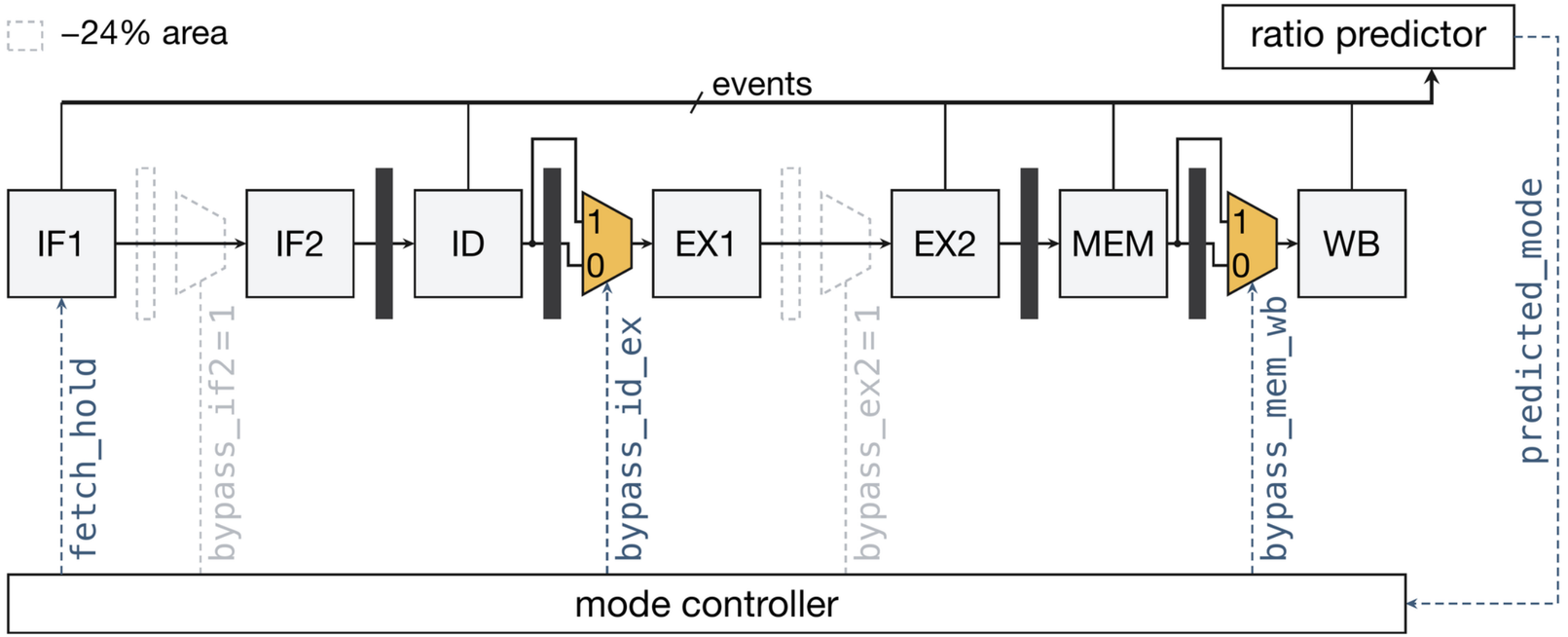


Artix-7 XC7A100T, closed loop	Result
Clock, one-ratio head	90.32 MHz
Clock, three-depth head	87.1 MHz
Core	3,086 LUT · 1,461 FF
Predictor, 2 ratios	478 LUT · 192 FF (0.75%)
Predictor + mode FSM *(switch helper)*	1.30% of the device
Byte-identical runs	**641 / 641 programs**
Self-initiated switches	144–1,315 per suite run

- closed loop: the **hardware head drives** the mode controller, no host
- All 641 programs ***identical***, through up to 1,315 self-initiated switches per suite run

Morphing changes timing, never results!!!

Adaptive hardware optimization



Imagine just working with same clk ⟶ **P7 never wins (0 weight).**

- *Two bypass selects become constants,* ***so synthesis removes*** *the IF1/IF2 and EX1/EX2 registers and their muxes* (-24% area)

PDPM can prune hardware given specific clk plans!

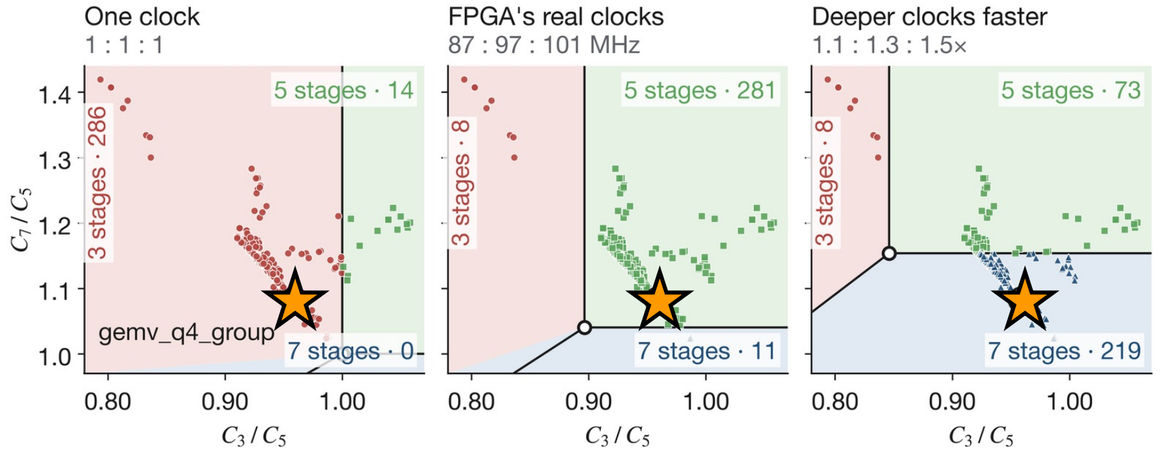
What's Next?

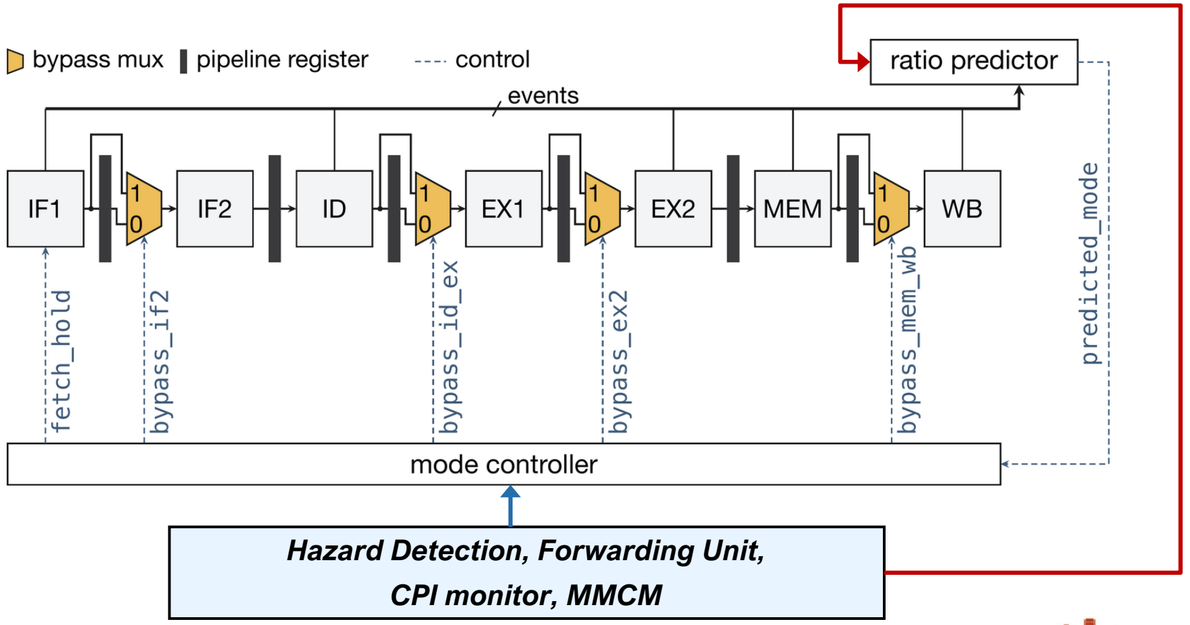
Improvements

Current version	Future
depth sensitivity ρ needs both modes	read it from one mode: close a 4.9 pp gap
single-cycle memory	slower memory: 7 stages win by up to 23.7%
one clock for every depth	a clock that follows the depth: +5.8%
phases shown on composite pipelines	full applications
unprivileged RV32IM	exceptions, interrupts, virtual memory

Takeaways & Contributions

- The **best** depth depends on **program**; **phase** and **clock** also affect
- One **superset** core runs as 3, 5 or 7 stages, **reducing** hardware and allowing convenient depth **switches**
- Predict two cycle **ratios**, not a label: **24× less overhead** than the best fixed depth
- **Frozen, but adaptive perceptron** for clock plan, real AI programs: **~1% hardware**





ICCD 2026

The 44th IEEE International Conference on Computer Design

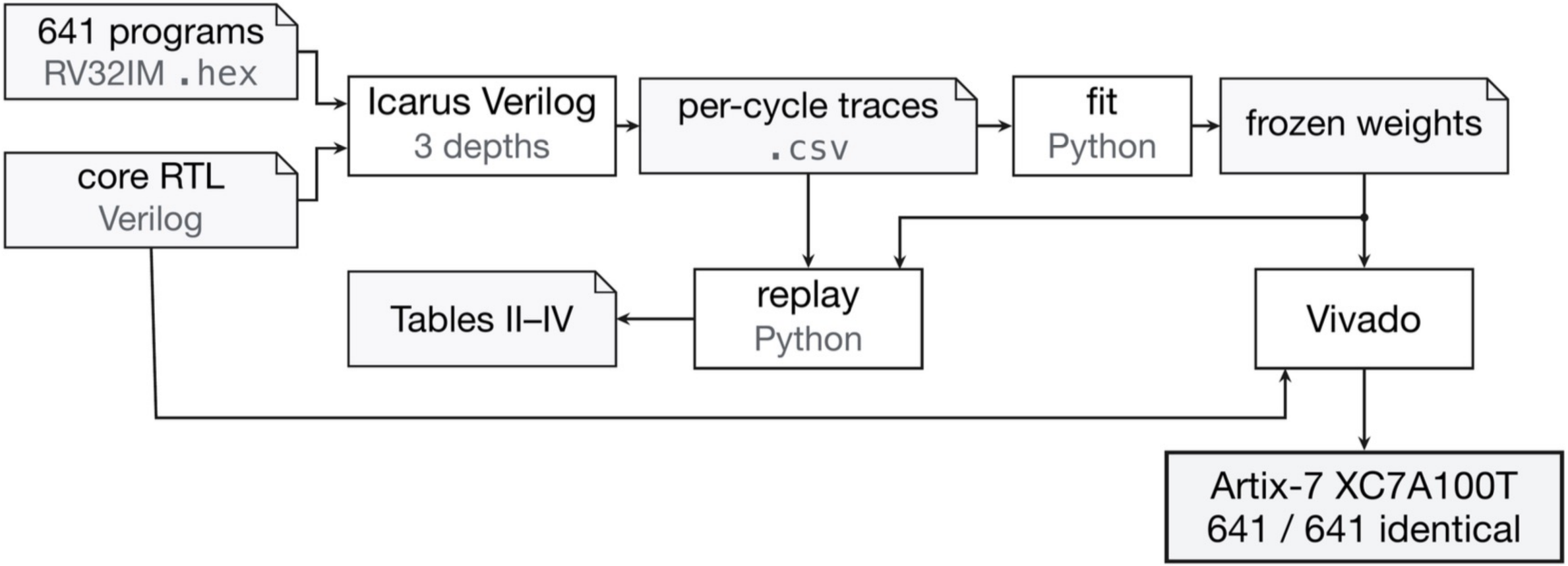
Nov. 16-18, 2026 Hong Kong SAR, China

Thank you!!

github.com/da-dawit/pdpm-riscv



Contact: *chundawit@taejae.ac.kr*


641 programs
RV32IM .hex
core RTL
Verilog
Icarus Verilog
3 depths
per-cycle traces
.csv
fit
Python
frozen weights
Tables II–IV
replay
Python
Vivado
Artix-7 XC7A100T
641 / 641 identical