Speculative GHR Forwarding: Eliminating Stale Branch-Predictor State in Deep FPGA Pipelines

DEVANSH JOSHI, The University of Texas at Austin, USA
SHAFIN ULA, The University of Texas at Austin, USA
SATYA ANANDH, The University of Texas at Austin, USA

Deeper pipelines raise an FPGA soft core's clock frequency, but they make every branch misprediction more costly. As depth grows, the global history register (GHR) of a correlation-based branch predictor takes longer to update. The register grows stale and inflates mispredictions, though the predictor hardware is unchanged. We eliminate this penalty with Speculative GHR Forwarding (SGF). Its key idea is that an in-order pipeline's existing stage registers already hold the per-branch state required for rollback. Recovery therefore needs no reorder buffer, unlike the out-of-order schemes that inspired it. We evaluate SGF on five RISC-V pipelines of four to eight stages, from identical functional-unit hardware on a Xilinx Artix-7 device. The experiment separates this cost into an inherent flush penalty and a correctable stale-GHR penalty. A history-free control predictor and a thirty-two-fold capacity sweep confirm the latter causally. On the seven-stage pipeline, SGF reduces mispredictions by 31 percent at under 1 percent area on single-history predictors, with no frequency penalty; on multi-table predictors like TAGE the area cost is higher. The cycles-per-instruction gain is modest: SGF is a favorable cost-benefit tradeoff, not a large speedup. The same experiment finds the execute-stage split to be the only frequency knee, making a six-stage pipeline throughput-optimal.

CCS Concepts: • **Hardware → Reconfigurable logic and FPGAs**; • **Computer systems organization → Reduced instruction set computing**; *Pipeline computing*.

Additional Key Words and Phrases: RISC-V, pipeline depth, FPGA, Artix-7, branch prediction, gshare, speculative GHR forwarding, microarchitecture, reconfigurable computing

1 Introduction

How deep should an FPGA pipeline be? On ASIC processes the answer is well studied: partition the datapath until gate delay per stage reaches 6–8 FO4 inverter delays, then stop because branch-misprediction penalties and power begin to dominate [6, 8, 23]. On FPGAs, where LUT delay is largely function-independent and routing delay dominates timing [13, 27], that answer does not transfer directly. This paper answers the question with a controlled experiment on Artix-7 and reports three findings: pipeline depth raises clock frequency only through the execute-stage split; beyond it, deeper pipelines add a stale global-history penalty on branch-dense workloads; and a lightweight fix, Speculative GHR Forwarding (SGF), removes that penalty at under 1% area and no frequency loss. Prior RISC-V soft cores differ in ISA support, predictor design, memory architecture, and forwarding strategy, making performance differences across projects unattributable to any single factor. We are not aware of a published study that varies pipeline depth as the single controlled design parameter on a common FPGA target while holding the RTL of all other microarchitectural modules constant. Each stage split does carry structural side effects (an added load-use stall, a fetch bubble, an extra forwarding source), which we make explicit and decompose rather than assume away.

A subtler cost compounds the explicit penalty of deeper pipelines: the predictor's global history register (GHR) becomes stale because the latency between branch resolution and the next prediction spans more cycles. Speculative history updating is itself an established technique. Yeh and Patt's two-level adaptive predictor [29] already updates history speculatively, and ASIC out-of-order processors (Alpha 21264 [11], MIPS R10000 [28]) recover it on misprediction through the reorder

Authors' Contact Information: Devansh Joshi, The University of Texas at Austin, Department of Electrical and Computer Engineering, Austin, Texas, USA, devansh.jos@utexas.edu; Shafin Ula, The University of Texas at Austin, Department of Electrical and Computer Engineering, Austin, Texas, USA, shafin.ula@utexas.edu.

buffer (ROB) or a dedicated branch stack. Gonzalez and Horowitz quantified the staleness cost in ASIC simulations [5]. These recovery mechanisms are tied to the out-of-order substrate, however: the R10000's branch stack checkpoints the register-map alias table and the Alpha's recovery rides the ROB. These are centralized structures that an in-order FPGA soft core lacks and could not add cheaply (the structures alone would dwarf the predictor they protect). To our knowledge, the effect itself has not been measured under controlled conditions on FPGA fabric, and no lightweight in-order realization of speculative history has been demonstrated there.

These findings rest on two contributions. First, we construct a controlled experiment that cleanly separates the CPI cost of deeper pipelining into two mechanisms. Five RV32IM pipeline variants (4–8 stages) share identical RTL for all functional units, synthesized on Xilinx Artix-7 XC7A35T across 50 post-place-and-route runs with seven benchmark workloads:

- **Mechanism A (inherent):** Higher per-misprediction flush penalty from additional stages to drain. Present on all workloads, predictable from pipeline structure alone.
- **Mechanism B (correctable):** Inflated misprediction count from stale GHR state. Present only on branch-dense, data-dependent workloads, two of our seven (+5.7% on CoreMark at 6-stage, +10% on statemate at 7/8-stage), and unpredictable without cycle-accurate simulation.

Two controls establish that Mechanism B is causal rather than correlational: a bimodal predictor (no GHR) shows *zero* depth-dependent inflation on the same workloads, and a 32× PHT capacity sweep shows the inflation is not a table-size aliasing artifact (Section 7.1). A first-order CPI model ($R^2 = 0.977$, $R^2_{CV} = 0.941$) captures the depth decomposition (Section 4.5).

Second, we propose Speculative GHR Forwarding (SGF), an implementation and measurement contribution rather than a new predictor. Speculative history with checkpoint restore is established [11, 29]; SGF does not reinvent it. The novelty is twofold. The first part is the *in-order, ROB-free realization*. An in-order single-issue pipeline admits no nested or out-of-order rollback (Section 5.3), so a single GHR checkpoint forwarded through the *existing* pipeline registers is provably sufficient. It replaces the reorder buffer or branch stack that ASIC schemes require. The second is the *controlled demonstration* that stale GHR state measurably degrades deep pipelines on Artix-7, isolated from the inherent flush penalty so that SGF removes only the correctable Mechanism B. On the 7-stage pipeline SGF cuts mispredictions 31.4% on CoreMark and 22.7% on statemate, pushing deep-pipeline counts *below* the shallow-pipeline baseline, at 49 LUTs and 57 FFs (0.7% area, no frequency loss). The benefit reproduces on a tournament and a downscaled TAGE predictor (Section 5.10); the sub-1% cost holds for single-history predictors and grows on multi-table TAGE.

The value is a favorable cost-benefit tradeoff, not a large speedup: Mechanism B is one of several depth costs, so the 31% misprediction cut is a roughly 3% CPI gain, and SGF does not let a deeper pipeline out-throughput the 6-stage. Its use case is a depth *already forced* by timing, such as a registered-BRAM fetch that mandates the IF1/IF2 split, where the stale-GHR penalty is otherwise unavoidable. At under 1% area and no frequency loss it is a low-risk addition to any gshare-class in-order pipeline with three or more stages of update latency; the gain is largest with tightly coupled memory and dilutes under cache-miss stalls (Sections 6.2, 5.10).

The experiment also reveals a two-tier frequency structure on Artix-7. Shallow pipelines (4/5-stage) cluster at 70–74 MHz; deep pipelines (6/7/8-stage) cluster at 115–118 MHz ($p < 0.001$, Cohen's $d = 18.5$). The execute-stage split at depth 6 is the sole significant frequency knee, which makes the 6-stage the throughput optimum, a directly actionable design rule for any RV32IM-class FPGA soft core independent of SGF.

Table 1. Representative RISC-V FPGA soft cores. None varies pipeline depth as a controlled parameter with invariant functional-unit hardware, and none addresses stale-GHR recovery.

Core	Pipeline	Predictor	Depth-config.?
SERV [12]	bit-serial	none	no
PicoRV32 [26]	multi-cycle	none	no
Ibex [19]	2-stage	bimodal-style	no
VexRiscv [16]	configurable	optional	not indep.
SweRV EH1 [25]	2-wide superscalar	BHT	no
CVA6 [30]	out-of-order	gshare-class	no
This work	4–8 (shared RTL)	gshare+SGF	yes

2 Background and Related Work

2.1 Pipeline Depth and FPGA Timing

RV32IM provides a fixed-width 32-bit encoding with regular operand fields well suited for pipelined implementation [18, 24]. Deeper pipelines reduce per-stage logic delay but increase misprediction penalty, area, and forwarding complexity [7, 8]. Hrishikesh et al. established 6–8 FO4 delays per stage as optimal for ASICs [8]; Sprangle and Carmean showed diminishing returns beyond certain depths [23]. Both assume ASIC timing where gate delay scales linearly with logic depth. A less studied effect is that deeper pipelines also degrade branch prediction accuracy: the increased latency from resolution to the next prediction leaves the GHR stale when branches are closely spaced. Gonzalez and Horowitz identified this in ASIC simulations [5]; we are not aware of a study measuring it on FPGA fabric under controlled conditions. On Xilinx 7-series FPGAs, LUT delay is largely function-independent, and routing delay often exceeds LUT delay [13, 27]. Splitting a combinational path across two pipeline stages therefore yields frequency improvement only when the split targets LUT-dominated logic; routing-dominated paths see diminishing returns [13]. DSP48E1 blocks provide registered multiply-accumulate; Block RAM offers single-cycle synchronous reads.

2.2 Related RISC-V SoP Cores

Prior RISC-V FPGA implementations vary widely in scope (Table 1). They span bit-serial and multi-cycle designs with no pipeline, fixed shallow pipelines, and out-of-order or superscalar cores. None lets pipeline depth vary as a single controlled parameter with the predictor, MDU, and memory hardware held invariant, and none addresses stale-GHR recovery. That combination is what this study requires and supplies.

2.3 Branch Predictors and Speculative History

Correlation predictors key a pattern-history table on global branch history. gshare [14] XORs that history with the PC; tournament predictors add a chooser between global and local components; TAGE [21] uses several tagged tables at geometric history lengths; and perceptron predictors [9] are a neural alternative. All of them index on a global history register, so all are exposed to the stale-GHR effect this paper isolates, and we measure three of the four.

Speculative GHR updating with checkpoint rollback is long established and is *not* what we claim as new. Yeh and Patt [29] describe speculative history; out-of-order ASICs recover it through the reorder buffer or a branch stack (Alpha 21264 [11], MIPS R10000 [28]), structures that tie to out-of-order state (the R10000's branch stack checkpoints the register-rename map) and that an in-order soft core lacks. In-order ASIC pipelines have likewise used shift-register history with checkpoint/mask restore, but as dedicated recovery logic. SGF's contribution is the in-order FPGA

realization that instead reuses the *existing* pipeline registers (Section 5), not the rollback concept. In tagged predictors stale GHR corrupts both index and tag (a different error profile than gshare), needing per-table checkpoints [20, 21]; Michaud et al. [15] note that update latency compounds with PHT aliasing.

2.4 FPGA Pipeline Studies

Prior surveys catalog pipelining techniques across RISC families [2], and configurable generators such as Rocket Chip [1] offer parameterizable pipelines with branch-prediction front-ends. Such generators and other in-order soft cores carry lightweight branch-recovery state (PC redirect, BTB/RAS checkpoints), but none, to our knowledge, speculatively checkpoints and forwards the *global history register* of a correlation predictor in an in-order FPGA pipeline, nor isolates pipeline depth as a controlled FPGA variable. The FPGA-specific study of Kuon and Rose [13] characterizes the LUT-routing delay gap that makes FPGA pipelining differ from ASIC pipelining, but does not examine branch-prediction interactions.

3 Experimental Setup

We keep three result types distinct throughout. F_{max}, area, and power are *post-place-and-route* (static timing analysis and Vivado vector-based power estimates). Cycle and misprediction counts are from *RTL simulation*. The CPI decomposition (Section 4.5) is *analytical*, and where it takes a measured misprediction rate as input we say so. Tables and text label which category each number belongs to.

3.1 Common Microarchitecture

All five pipeline variants share the following modules, instantiated from identical RTL source files:

ALU. A combinational 10-operation arithmetic logic unit supporting ADD, SUB, AND, OR, XOR, SLT, SLTU, SLL, SRL, and SRA.

Multiply-Divide Unit (MDU). A 2-cycle pipelined multiplier (DSP48E1-inferred) and a 32-cycle restoring divider, covering MUL/MULH/MULHSU/MULHU and DIV/DIVU/REM/REMU; division-by-zero and overflow follow the RISC-V specification [24].

Branch Predictor. A gshare predictor [14] with a 64-entry pattern history table (PHT) of 2-bit saturating counters [22], a 6-bit global history register (GHR) in the two-level adaptive style [29], a 32-entry direct-mapped branch target buffer (BTB), and a 4-entry return address stack (RAS). The PHT is indexed by XORing the lower PC bits with the GHR. The predictor hardware is identical in all five variants. However, the number of pipeline stages between the fetch stage (where predictions are made) and the execute stage (where predictions are resolved and the PHT is updated) differs across variants. This update latency is the central variable in our analysis and the target of SGF.

Predictor scale justification. The 64-entry PHT and 6-bit GHR are sized for the resource-constrained embedded soft core targeted here (<1% of the Artix-7 fabric); the 1024–4096-entry PHTs of high-performance ASICs would dominate a ~7,500-LUT core. A 6-bit history is short but representative of low-end cores (Ibex's bimodal-style predictor, SweRV's modest branch-history table), and it bounds the SGF checkpoint to 6 bits per in-flight branch. Two controls in Section 7.1 (a 32× PHT sweep and a no-GHR bimodal predictor) confirm the depth effect comes from update latency, not the small size.

CSR Unit. Machine-mode CSRs (`mstatus`, `mtvec`, `mepc`, `mcause`, `mscratch`) and 64-bit performance counters (`mcycle`, `minstret`, branch, and misprediction counts) that each benchmark reads before and after its run.

Memory and register file. A 4096-word (16 KB) `$readmemh`-initialized instruction ROM behind a 64-line direct-mapped instruction cache (combinational hit logic), a 2048-word (8 KB) synchronous

data RAM with byte/halfword/word access, and a 32×32-bit register file (two read ports, one write port, write-through bypass).

3.2 Pipeline Variants

The five variants are constructed by splitting or merging stages of a common datapath. Each split targets a specific combinational path:

4-stage (IF, ID, EX, MEM/WB): Merges memory access and write-back. Load data returns at the end of the combined MEM/WB stage in time to forward to the next EX, eliminating the load-use *stall* but creating the longest critical path.

5-stage (IF, ID, EX, MEM, WB): Classic RISC organization [17], separating MEM and WB at the cost of a 1-cycle load-use stall.

6-stage (IF, ID, EX1, EX2, MEM, WB): Splits the execute phase, separating the forwarding multiplexer (EX1) from the ALU and branch comparator (EX2). This split targets the dominant critical path of the 4/5-stage designs.

7-stage (IF1, IF2, ID, EX1, EX2, MEM, WB): Splits instruction fetch, placing branch prediction in IF2 with a registered PC. Introduces a 1-cycle IF2 bubble on every correctly predicted taken branch (IF1's speculative fetch is discarded). This split targets BRAM read latency. An industrial pipeline might place a zero-delay Next-Line Predictor or BTB in IF1 to remove this bubble; we do not, because embedding specialized IF1 routing would perturb the physical timing paths unevenly across depths and break the structural invariant. The bubble is also orthogonal to SGF: an IF1 next-line predictor changes only *where* the next fetch address is produced, removing the bubble (the $\beta\ B(D)\ f_b$ term of the CPI model, Section 4.5) but leaving the stale-GHR window, and hence SGF's misprediction counts, unchanged.

8-stage (IF1, IF2, ID, EX1, EX2, MEM1, MEM2, WB): Splits memory access, creating a dual load-use stall (loads not available until MEM2). The MEM split leaves the fetch-to-resolution path unchanged, so the predictor's update latency (4 stages), and therefore Mechanism B, is identical to the 7-stage; its only added cost is the dual load-use stall, which is why the 7- and 8-stage baselines share misprediction counts on every workload.

The per-event costs are summarized in Table S5 (supplementary material). The 7/8-stage variants share the same branch penalty because both resolve branches in EX2 with equal numbers of preceding stages. The critical parameter for Mechanism B is the predictor update latency, the number of stages between prediction and resolution: 2, 2, 3, 4, 4 for depths 4–8.

Threats to isolation. Each split introduces structural side effects (load-use stall changes, IF2 bubbles, forwarding-source count) inseparable from the depth increase. The CPI model (Section 4.5) decomposes each into a separate term, so the claim is not that depth alone causes the CPI increase but that each split's Mechanism B component is correctable.

3.3 Synthesis Methodology

To quantify placement-dependent variance in F_{max}, each variant is synthesized and implemented 10 times with different placement and routing directive combinations (Default, Explore, ExtraNetDelay_high/low, ExtraPostPlacementOpt, NoTimingRelaxation, and AggressiveExplore in various pairings). This produces 50 independent implementation runs (5 variants × 10 directive combinations). The target device is Xilinx Artix-7 XC7A35TCPG236-1. All syntheses use Vivado 2025.2 in out-of-context (OOC) mode with a 5 ns clock constraint (200 MHz target) to drive the tools toward maximum optimization effort. The reported F_{max} values are means with standard deviations and 95% confidence intervals computed via Student's t-distribution with 9 degrees of freedom. F_{max} for each run is derived from the worst negative slack (WNS) as $F_{max} = 1000/(5 - \text{WNS})$.

Directive selection. The 10 directive pairings were fixed *before* any run and never re-selected after seeing results, chosen to sample Vivado's optimization heuristics rather than to favor any variant. Run-to-run dispersion (mean, standard deviation, t-based 95% CI) is therefore a sensitivity to *these* directives, not an estimate over all achievable placements. This matters for absolute spreads but not for cross-variant comparisons: every variant (baseline and SGF) uses the *same* directive set, so an SGF-vs-baseline comparison is paired and controlled regardless. Ten runs suffices to separate the 44 MHz inter-tier gap (Cohen's $d = 18.5$, non-overlapping CIs) from within-tier noise, but not the sub-2 MHz intra-tier differences, which we do not interpret.

Simulation environment. All cycle and misprediction counts are from Vivado XSim behavioral (RTL) simulation. The design is a single-clock synchronous pipeline with no asynchronous interfaces, so behavioral simulation is *expected* to reproduce the synthesized cycle count; we present this as an assertion from the deterministic structure, not a hardware-validated result (Section 8). Simulation validates cycle counts and hence CPI, not physical timing, which post-place-and-route static timing analysis establishes separately (the source of all F_{max} values). Determinism is verified by identical instruction counts, branch counts, and memory checksums across all five variants for every workload.

Reproducibility. All benchmarks are compiled with `riscv-none-elf-gcc` 15.2.0 (xPack) using `-march=rv32im_zicsr -mabi=ilp32 -O1 -nostdlib -ffreestanding`. CoreMark uses the official EEMBC source (`github.com/eembc/coremark`) with a bare-metal porting layer; Embench-IoT uses the official source [4]. Linker scripts place code at address 0 (IMEM) and data at 0x10000 (DMEM); the startup routine clears BSS and initializes seed variables before calling `main`. Two of the ten pairings repeat a directive combination under different Vivado random seeds, producing distinct placements; the complete list is in the archived scripts. Complete RTL, synthesis TCL scripts, benchmark hex files, and result logs are archived at `github.com/devanshjoshi08/RISCV-Pipeline-Research`.

3.4 Benchmark Suite

Seven workloads from three standard sources measure CPI and branch prediction behavior across all five depths. All are compiled at `-O1`; higher optimization levels may alter branch density and correlation structure (see Section 8). Each benchmark reads the hardware performance counters (`mcycle`, `minstret`, branch count, misprediction count) before and after the workload, computes deltas, and stores results to data memory for testbench readout. Functional correctness is verified by confirming that all five pipeline variants produce identical instruction counts, branch counts, and memory checksums for each workload.

The seven workloads, listed as (instructions, branches, branch frequency), are: **Dhrystone 2.1** [7] (2,926, 400, 13.7%); a mixed **Diagnostic** program exercising arithmetic, load/store, calls, and conditional branches (1,367, 360, 26.3%); official EEMBC **CoreMark** [3] (10 iterations, PERFORMANCE_RUN; 2,893,145, 719,810, 24.9%); and four Embench-IoT [4] kernels, **aha-mont64** (Montgomery multiply, multiply-heavy; 4,453,858, 512,593, 11.5%), **crc32** (4,008,648, 174,591, 4.4%; near-perfectly predictable, 0.19% misprediction rate), **statemate** (FSM with data-dependent transitions; 3,160,225, 376,291, 11.9%), and **edn** (signal processing; 2,972,527, 338,986, 11.4%).

4 Baseline Results

This section establishes the baseline measurements that motivate SGF: the synthesis results reveal the FPGA-specific frequency structure, and the CPI measurements expose Mechanism B as the target for correction.

Table 2. Post-place-and-route synthesis results on Artix-7 XC7A35T (mean ± std from 10 synthesis directive combinations, 50 runs total). 95% confidence intervals computed via Student's t with 9 d.f. Tier gap: two-sample t-test $p < 0.001$, Cohen's $d = 18.5$.

Metric	4-stg	5-stg	6-stg	7-stg	8-stg
F_{max} (MHz)	70.4	74.0	117.3	115.0	117.5
± std (MHz)	2.4	1.5	1.4	2.0	1.8
95% CI (MHz)	[68.7, 72.1]	[72.9, 75.1]	[116.3, 118.3]	[113.5, 116.5]	[116.2, 118.8]
Slice LUTs	7,219	7,429	7,631	7,766	7,841
Slice Regs (FFs)	8,040	8,190	8,501	8,550	8,678
DSP48E1	12	12	12	12	12

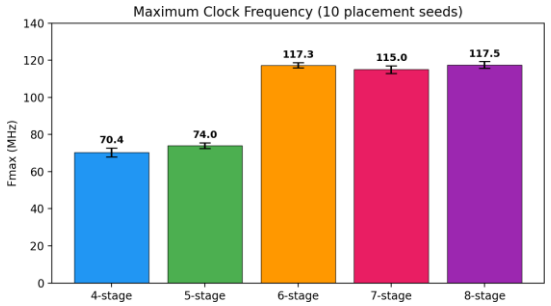


Fig. 1. Maximum clock frequency (F_{max}, post-place-and-route static timing) for each pipeline depth. Error bars show standard deviation across 10 synthesis directive combinations. The 44 MHz gap between the shallow tier (4/5-stage) and the deep tier (6/7/8-stage) has $p < 0.001$ (Cohen's $d = 18.5$).

4.1 Synthesis Results

Table 2 presents the post-place-and-route results for all five variants, with F_{max} means, standard deviations, and 95% confidence intervals.

The results reveal a two-tier frequency structure: the shallow tier (4/5-stage) at 70–74 MHz and the deep tier (6/7/8-stage) at 115–118 MHz (Fig. 1). The 44 MHz gap ($p < 0.001$, Cohen's $d = 18.5$) confirms this is not a placement artifact. The effect size is extreme by convention ($d > 0.8$ is "large"), but expected here: the two tiers' 95% confidence intervals do not overlap, so a large standardized separation is the arithmetic consequence of the small within-tier placement variance.

The jump at depth 6 results from splitting the execute stage: the 4/5-stage critical path traverses the forwarding mux, ALU, and branch comparator in series. The EX1/EX2 split halves this combinational depth, yielding 54–56% frequency improvement. Further splits yield no additional F_{max} because routing delay, not reduced by register insertion [13], dominates the remaining paths. Within the deep tier, differences are <2 MHz (<2%) with limited practical impact.

4.2 Area

The 4-stage uses the fewest resources (7,219 LUTs, 8,040 FFs; Table 2). Each additional pipeline stage adds 52–210 LUTs and 76–150 FFs, for a total increase from 4 to 8 stages of 622 LUTs (8.6%) and 638 FFs (7.9%). All five variants use 12 DSP48E1 blocks for the pipelined multiplier, and no BRAM tiles since both instruction and data memories are implemented in distributed LUT RAM. Figure S1 (supplementary material) shows the LUT and flip-flop growth with depth. The increase is gradual and sub-linear in the added pipeline registers, because the dominant cost is the shared

Table 3. CPI from RTL simulation across all seven workloads and five pipeline depths (f_b = branch frequency). Every variant executes identical instruction, branch, and misprediction counts per workload (determinism check). CoreMark and statemate also show depth-dependent misprediction inflation (Mechanism B; see text and Fig. 2).

Workload (f_b)	4-stg	5-stg	6-stg	7-stg	8-stg
Dhrystone (13.7%)	1.41	1.48	1.68	1.85	1.99
Diagnostic (26.3%)	1.38	1.60	1.66	1.89	2.11
CoreMark (24.9%)	1.54	1.63	1.83	2.02	2.18
aha-mont64 (11.5%)	1.21	1.21	1.25	1.32	1.32
crc32 (4.4%)	1.30	1.30	1.39	1.52	1.56
statemate (11.9%)	1.21	1.30	1.37	1.50	1.69
edn (11.4%)	1.48	1.51	1.66	1.78	1.95

functional-unit logic (ALU, MDU, register file), which is identical across depths; only the inter-stage registers and the slightly larger forwarding/hazard logic grow.

4.3 CPI Measurements

Table 3 reports CPI for all seven workloads across the five depths; it rises monotonically with depth on every workload, and the analytical model of Section 4.5 reproduces these values. Dhrystone and Diagnostic produce *identical* misprediction counts at all depths (203/400 and 69/360 branches respectively): their branches are spaced widely enough that update latency adds none, so only Mechanism A operates. The 5-stage CPI exceeds the 4-stage despite an identical branch penalty because of its 1-cycle load-use stall (absent in the 4-stage merged MEM/WB).

CoreMark, by contrast, exhibits a depth-dependent rise in misprediction count, the signature of Mechanism B. The 6-stage produces 154,077 mispredictions (21.4%) versus 145,735 (20.2%) at 4/5-stage, a 5.7% increase from the longer update latency. The 7- and 8-stage are *identical* at 147,760 (20.5%): the two deepest pipelines share the same 4-stage predictor update latency, so the 8-stage's extra MEM split adds flush-penalty cycles (Mechanism A) but no new staleness (Mechanism B). The official CoreMark/MHz score ranges from 2.23 (4-stage) to 1.58 (8-stage). This inflation is the specific target of SGF. Table S2 (supplementary material) gives the full CoreMark figures across depths.

Among the four Embench-IoT kernels, multiply-heavy aha-mont64 has the lowest CPI (few branches, MDU-bound). crc32 has almost zero mispredictions (344 across all depths, 0.19%) on its highly predictable inner loop. statemate shows the strongest depth-dependent effect (66,603 mispredictions at 4/5-stage vs. 73,263 at 7/8-stage, +10.0%) from data-dependent state transitions. edn has moderate CPI with a low 3.1% misprediction rate.

4.4 Branch Misprediction Analysis: Mechanism A vs. Mechanism B

Deeper pipelines affect CPI through two distinct mechanisms that must be analyzed separately.

Mechanism A: Higher per-misprediction penalty. Each misprediction flushes all stages between fetch and resolution. The 4/5-stage designs flush 2 stages (2 wasted cycles), the 6-stage flushes 3, and the 7/8-stage flush 4. This cost scales linearly with the number of flushed stages and is independent of predictor accuracy. On Dhrystone (203 mispredictions, 2,926 instructions), the per-misprediction CPI contribution increases from $203 \times 2/2926 = 0.14$ at 4-stage to $203 \times 4/2926 = 0.28$ at 7/8-stage.

Mechanism B: Inflated misprediction count from stale GHR state. In deeper pipelines, the latency between branch resolution (where the PHT is updated) and the next prediction (where

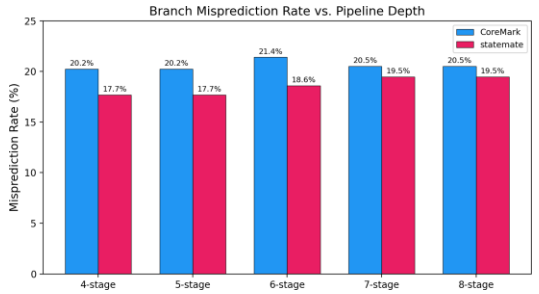


Fig. 2. Branch misprediction rate (%, RTL simulation) on CoreMark and statemate across five depths. The rate climbs with depth, CoreMark at the 6-stage (20.2%→21.4%), statemate at the 7/8-stage (17.7%→19.5%), which is the Mechanism B inflation. The five other workloads (not shown) hold a constant rate at all depths (Mechanism A only).

the PHT is read) spans more cycles. If two branches are closely spaced, the second branch may be predicted using a GHR that has not yet incorporated the first branch's outcome. This produces *additional* mispredictions that would not occur in a shallower pipeline with the same predictor hardware.

Fig. 2 separates these effects. On five of seven workloads (Dhrystone, Diagnostic, aha-mont64, crc32, edn), the misprediction count is identical across all five depths, only Mechanism A operates. On CoreMark, the 6-stage produces 154,077 mispredictions versus 145,735 for the 4/5-stage (5.7% more), adding Mechanism B on top of Mechanism A. On statemate, the 7/8-stage pipelines produce 73,263 versus 66,603 for the 4/5-stage (10.0% more).

Mechanism B requires two conditions: (1) branches spaced closely enough that a new prediction occurs before the prior branch's resolution updates the PHT, and (2) outcomes sensitive to global history. A useful proxy metric is the *average inter-branch distance* (IBD), defined as IBD = N / B where N is total instructions and B is total branches. Table S4 (supplementary material) classifies our workloads by IBD and observed Mechanism B behavior. The two workloads exhibiting Mechanism B (CoreMark, statemate) have IBD ≤ 8.4 and data-dependent branch outcomes, while workloads with larger IBD or highly biased outcomes (crc32: 99.8% taken) show no depth-dependent misprediction variation. This suggests a *preliminary* indicator, not a validated design rule: SGF is likely beneficial when IBD < 10 *and* branches are data-dependent rather than loop-structured. Because this rests on only the two Mechanism-B workloads in our suite, we present it as an observation to test rather than a guideline to apply.

SGF eliminates precisely this unpredictable amplification on susceptible workloads. Two controls in Section 7.1 confirm the cause is GHR staleness, not an artifact: a no-GHR bimodal predictor shows *zero* depth-dependent inflation on the same workloads, and a 32× PHT capacity sweep shows the inflation persists at every table size, excluding aliasing.

4.5 Analytical CPI Model

We derive a mechanistic CPI model from the architectural hazard structure of each pipeline variant, following the first-order/mechanistic processor-modeling tradition [10]. Unlike an empirical

regression, each term corresponds to a specific physical mechanism:

$$\mathrm{CPI}(D,w) = 1 + \underbrace{\frac{M_w(D)}{N_w} \cdot P(D)}_{\text{branch flush}} + \underbrace{\alpha \cdot L(D)}_{\text{load-use}} + \underbrace{\beta \cdot B(D) \cdot f_b}_{\text{IF2 bubble}} + \gamma_w \tag{1}$$

where D is pipeline depth, w indexes the workload, $M_w(D)/N_w$ is the misprediction rate (mispredictions per instruction; may vary with D due to Mechanism B), $P(D)$ is the flush penalty in cycles (2, 2, 3, 4, 4 for depths 4–8), $L(D)$ is the load-use stall cycles (0, 1, 1, 1, 1.5), $B(D)$ is the IF2 bubble indicator (0, 0, 0, 1, 1), f_b is branch frequency, and γ_w captures workload-specific base overhead from instruction mix, MDU stalls, and data hazards not attributable to pipeline depth. The 8-stage's $L(D) = 1.5$ is a structural average, not a single per-instruction value. Its MEM1/MEM2 split makes a load-use hazard cost 1 cycle when the dependent use is one slot downstream and 2 cycles when adjacent. The effective stall therefore lies between 1 and 2 over the load-use mix. We use the midpoint, and the 8-stage residual ($\leq$0.04 CPI) confirms it suffices.

The shared α (load-use) and β (IF2 bubble) are fitted across all 35 points and γ_w per workload (9 parameters for 35 observations), giving $\alpha = 0.145$, $\beta = 1.12$. Here α is the per-instruction *frequency* of load-use hazards, not a per-stall cost. The value $\alpha < 1$ reflects that only ~15% of instructions are loads immediately consumed by the next operation, and $\alpha\, L(D)$ multiplies this frequency by the stall cycles each hazard incurs. Three-source forwarding resolves the remaining loads at zero penalty. We note the model's status precisely. Because the branch-flush term takes the *measured* $M_w(D)$ as input, the model *decomposes* observed CPI rather than predicting it ab initio on the two Mechanism-B workloads where M_w varies with D. On the five workloads with depth-invariant M_w it is genuinely predictive from architectural parameters. The LOWO test below isolates the depth-term accuracy from the γ_w offset.

This model achieves $R^2 = 0.977$ across all 35 data points (adjusted $R^2_{\text{adj}} = 0.969$, accounting for the 9 fitted parameters; mean absolute error 0.033 CPI, maximum residual 0.086 CPI), capturing 97.7% of the variance compared to 67% for a simple linear regression.

The model's value is its decomposability: the branch-flush term carries measured (Mechanism A+B) misprediction counts, $\alpha = 0.145$ is the CPI cost per load-use stall cycle, and $\beta = 1.12$ scales the IF2-bubble cost by branch frequency. The per-workload base γ_w ranges from -0.01 (multiply-dominated aha-mont64) to 0.48 (load-heavy edn), capturing instruction-mix effects orthogonal to depth. Because it is orthogonal to depth, γ_w is in principle estimable before implementation from the static instruction profile of the compiled binary (its load, multiply-divide, and branch frequencies) rather than from cycle-accurate simulation. Validating such a static estimator is future work; until then γ_w is fitted. The 4-stage merges MEM/WB, eliminating load-use stalls (it enters the model with $L(D) = 0$) at the cost of a longer critical path. That lengthening is a *frequency* effect (in $F_{\max}$), not a CPI term, and the 4-stage residual stays $\leq$0.04 CPI.

Cross-validation and status. Leave-one-workload-out (LOWO) cross-validation refits on 30 points per fold and predicts the held-out workload from the depth-dependent terms, with γ_w set to the training mean. It gives $R^2_{\text{CV}} = 0.941$ (MAE 0.052 CPI), with α and β stable across folds (0.14 ± 0.01, 1.10 ± 0.05). The largest errors fall on edn and Diagnostic, at the extremes of the instruction-mix distribution. The model is thus *predictive* for its depth-dependent terms (validated out-of-sample) and *descriptive* for the fitted per-workload γ_w; the depth decomposition, not absolute cross-workload CPI prediction, is what we claim. Because α and β barely move across folds, the gap between R^2_{CV} and in-sample R^2 is almost entirely γ_w-substitution error, not depth-term error; an

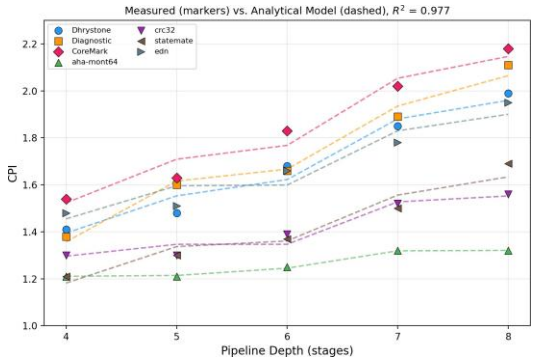


Fig. 3. Measured CPI (points) and the analytical model of Eq. 1 (lines) across the five pipeline depths for all seven workloads. The model captures the depth trend with $R^2 = 0.977$ ($R^2_{CV} = 0.941$ leave-one-workload-out).

extreme instruction mix such as floating-point-heavy code would enlarge only that term. Residuals have mean 0 and standard deviation 0.038 CPI with no depth pattern; the largest (+0.086 on 6-stage CoreMark) reflects the smooth $M_w(D)$ slightly under-capturing the depth-6 misprediction jump. Table S3 (supplementary material) decomposes CoreMark CPI into the four additive terms of Eq. 1 on top of the ideal 1.00: a depth-invariant base ($\gamma_w = 0.45$), a branch-flush term growing with depth (Mechanism A, including the Mechanism B rise at depth 6), a load-use term, and an IF2-bubble term present only at 7/8-stage. Summed, they match measured CPI within the residual ($\leq$0.09) at every depth. Fig. 3 overlays the model on the measured CPI for all seven workloads, and Figure S2 (supplementary material) shows the residuals carry no depth-dependent bias.

4.6 Throughput

Effective throughput (MIPS) is F_{max}/CPI. It combines post-place-and-route F_{max} and simulated CPI, so it is a valid figure of merit for comparing our variants, but not a single end-to-end measurement on running silicon. Computed from 10-directive mean F_{max} and CoreMark CPI, the 6-stage achieves the highest throughput on this fabric (64.1 MIPS on CoreMark, 70.7 MIPS on Diagnostic): its 67% frequency advantage over the 4-stage more than compensates for its higher CPI. The 7-stage reaches 56.9 MIPS (11% below the 6-stage) and the 8-stage 53.9 MIPS, confirming that stages beyond 6 degrade net throughput on this Artix-7 fabric. Fig. 4 plots throughput across depths, showing the 6-stage peak.

5 Speculative GHR Forwarding

Section 4.4 separated two depth penalties: Mechanism A (the inherent flush penalty, irreducible without changing the pipeline) and Mechanism B (the stale-GHR inflation, 5.7–10% on branch-dense workloads). This section presents Speculative GHR Forwarding (SGF), a lightweight mechanism that removes Mechanism B on the workloads that exhibit it.

5.1 Problem: Stale GHR in Deep Pipelines

The baseline gshare updates its GHR only at branch resolution in EX2, creating a 4-stage update latency on the 7/8-stage pipelines. When branches are separated by fewer than 4 instructions, the second branch uses a GHR that does not reflect the first's outcome.

A cycle-level trace makes this concrete. Take two correlated branches B_1, B_2 two instructions apart on the 7-stage, with B_2's outcome depending on B_1's. B_1 is predicted in IF2 at cycle t and

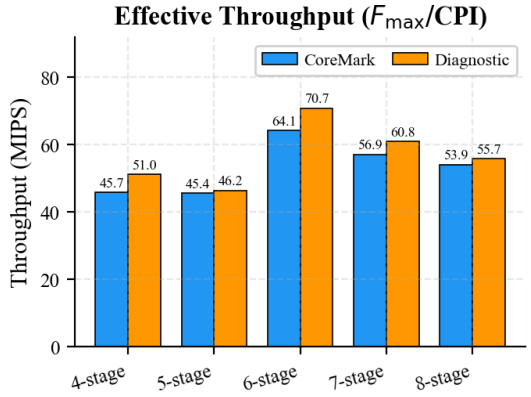


Fig. 4. Effective throughput (MIPS = F_{max}/CPI) across pipeline depths on CoreMark. The 6-stage is the peak: it captures the full execute-split frequency gain at moderate CPI, while deeper pipelines pay higher CPI without further frequency.

resolves in EX2 at t+4, writing its outcome to the GHR there; B_2 enters IF2 at t+2 and is indexed with a GHR predating B_1, selecting the PHT counter for the *wrong* history and mispredicting. On the 4/5-stage the same pair has a 2-stage path, so B_1 resolves at t+2 just as B_2 is indexed and B_2 predicts correctly. The extra misprediction is a control-path timing artifact of the widened resolution-to-prediction window, not a capacity or training deficiency, which is why a larger PHT or more training does not remove it (Section 7.1) but zeroing the update latency does.

Aggregated over the workload, this inflates CoreMark mispredictions by 5.7% at 6-stage and statemate by 10.0% at 7/8-stage relative to the 4/5-stage baseline. These compound with Mechanism A's longer flush penalty to produce a superlinear CPI increase.

5.2 Design: Dual-GHR Architecture

SGF replaces the single GHR with a dual-GHR architecture:

- **Speculative GHR** (`spec_ghr`): Updated at prediction time (IF2) by shifting in the *predicted* branch direction. This keeps the history fresh for subsequent predictions, even before any branch has resolved.
- **Committed GHR** (`committed_ghr`): Updated at resolution time (EX2) with the *actual* branch outcome. This serves as ground truth for rollback.

On a misprediction, `spec_ghr` is restored from `committed_ghr`, discarding the incorrect speculative history. The predictor indexes the PHT from `spec_ghr` at prediction and trains it from the committed history at resolution, so predictions see the freshest history without corrupting the counters.

Why not just update the GHR one stage earlier? Moving the update from EX2 to EX1 cuts latency by only one cycle (4 to 3 on the 7/8-stage), which mitigates but does not eliminate Mechanism B. Worse, the update needs the actual outcome from the EX2 branch comparator. Relocating it to EX1 would drag the comparator along and undo the execute-stage split that gives the 56% frequency gain. SGF instead gives the predictor a zero-latency speculative history without relocating any logic.

For precise PHT indexing, a 6-bit GHR checkpoint (`ghr_checkpoint`, a snapshot of `spec_ghr` taken at prediction) is forwarded through the pipeline registers alongside each branch instruction,

requiring 24 additional flip-flops across the four pipeline stages between IF2 and EX2. On misprediction, the checkpoint stored with the mispredicted branch provides the correct committed GHR state for rollback. The cost scales linearly in the GHR width h: each of the s stages between IF2 and EX2 carries an h-bit checkpoint ($h \times s$ flip-flops), and rollback is a single h-bit 2:1 multiplexer, with no quadratic or fan-out-sensitive term. A BRAM-backed gshare with an 11-bit GHR would need $11 \times 4 = 44$ checkpoint flip-flops and one 11-bit mux on the 7-stage (versus 24 at 6 bits), still under 0.5% of the core; moving the PHT into a BRAM tile removes its LUTs, so SGF's *relative* overhead shrinks as the predictor grows. Only multi-table predictors, needing a separate checkpoint per history length, grow the cost super-linearly (Section 8).

5.3 Integration

SGF changes only the branch-predictor module and the pipeline top-level interconnect; the shared RTL (ALU, MDU, memory, forwarding, hazard detection) and the BTB/RAS/PHT structures are untouched, replacing only the GHR-management logic. The 6-bit checkpoint forwarded with each branch adds 24 flip-flops; with the `spec_ghr`/`committed_ghr` registers, the rollback mux, and synthesis-inserted routing registers, the post-place-and-route total is 57 FFs (Section 5.4).

Correctness invariant. The PHT is always trained on *actual* outcomes, never speculative ones: it is read-indexed by `spec_ghr` but write-indexed by the per-branch `ghr_checkpoint` snapshotted at prediction. A correct prediction needs no action; a misprediction restores `spec_ghr` from `committed_ghr` and updates the PHT counter at the checkpointed index with the actual direction. The training index thus matches the prediction index, and `spec_ghr` reconverges to `committed_ghr` after every recovery; the only transiently-wrong state is `spec_ghr` during a flush, when no predictions are consumed.

Flush-restore timing. Consider a branch fetched into IF2 the cycle after a misprediction flush. The flush, registered at one clock edge, simultaneously invalidates stages IF1–EX1, restores `spec_ghr` from `committed_ghr` (shifted by the actual outcome), and redirects the PC. The next valid instruction therefore enters IF2 *after* the restore, with no intermediate cycle exposing a partially-updated `spec_ghr`; the `!flush` guard further suppresses the speculative shift during the flush cycle.

Absence of nested rollback. Out-of-order cores need a reorder buffer to recover speculative state when branches resolve out of program order [11, 28]; SGF needs none because an in-order single-issue pipeline resolves strictly in program order at EX2. Several branches may be in flight, each carrying its own checkpoint, but only the oldest resolves: a correct prediction leaves younger checkpoints valid, and a misprediction squashes all younger branches and restores `spec_ghr` once. That single restore covers every reachable state, which is the formal reason no ROB is needed. These invariants are encoded as a SystemVerilog assertion suite bound into the predictor and checked across all benchmark simulations (part of the released artifact).

5.4 Synthesis Results

SGF produces no F_{max} degradation under the same 10-directive methodology (Section 3.3): the 7-stage SGF achieves 117.5 ± 2.4 MHz vs. the baseline's 115.0 ± 2.0, and the 8-stage SGF 118.6 ± 1.7 vs. 117.5 ± 1.8. The nominal 2.5 MHz uptick on the 7-stage is within the run-to-run standard deviation ($\sim$2 MHz), and the baseline and SGF 95% confidence intervals overlap. We therefore read it as placement-and-routing noise rather than a real speedup; the defensible claim is that SGF adds no measurable critical-path delay. The area cost is negligible: +49 LUTs / +57 FFs (0.6/0.7%) on the 7-stage and +36 LUTs / +21 FFs (0.5/0.2%) on the 8-stage, with unchanged DSP usage. The speculative-update and checkpoint logic is absorbed by the existing predictor datapath without creating new critical paths. Table 4 reports the full synthesis comparison.

Table 4. SGF synthesis results on Artix-7 (mean ± std, 10 directive combinations). SGF adds +49 LUTs / +57 FFs on the 7-stage and +36 / +21 on the 8-stage (both sub-1%), with no F_{max} degradation and unchanged DSP usage.

Metric	7-stg	7-stg SGF	8-stg	8-stg SGF
F_{max} (MHz)	115.0±2.0	117.5±2.4	117.5±1.8	118.6±1.7
Slice LUTs	7,766	7,815	7,841	7,877
Slice Regs (FFs)	8,550	8,607	8,678	8,699
DSP48E1	12	12	12	12

Table 5. SGF misprediction results on 7- and 8-stage pipelines (Vivado XSim behavioral simulation). These are the shipped SGF numbers (no confidence filter; see Section 5.7). Bold entries indicate workloads where SGF reduces mispredictions below the 4/5-stage baseline.

	7-stg Mispred		8-stg Mispred			7-stg CPI		8-stg CPI	
Workload	**Base**	**SGF**	**Base**	**SGF**	**Δ%**	**Base**	**SGF**	**Base**	**SGF**
CoreMark	147,760	**101,418**	147,760	**102,053**	−31/31	2.02	1.96	2.18	2.13
statemate	73,263	**56,627**	73,263	**56,627**	−23/23	1.50	1.48	1.69	1.68
aha-mont64	139,739	115,644	139,739	115,644	−17/17	1.32	1.30	1.32	1.31
crc32	344	511	344	513	+49/49	1.52	1.52	1.56	1.56
edn	10,452	10,452	10,452	10,452	0/0	1.78	1.78	1.95	1.95
Dhrystone	203	203	203	203	0/0	1.85	1.85	1.99	1.99
Diagnostic	69	69	69	69	0/0	1.89	1.89	2.11	2.11

Δ% column shows 7-stage/8-stage misprediction change (rounded). 8-stage baseline counts equal 7-stage (both have 4-stage update latency). **Hardware cost (7-stage, workload-independent):** +49 LUTs (+0.6%), +57 FFs (+0.7%), 0 DSP/BRAM, F_{max} 117.5 vs. 115.0 MHz (no degradation). Net cycle savings: CoreMark +159,932, statemate +63,222, aha-mont64 +89,044 (others 0).

5.5 Benchmark Results

Table 5 presents the measured misprediction counts and CPI with SGF enabled on the 7- and 8-stage pipelines. All counts are from deterministic behavioral RTL simulation (Vivado XSim), not analytical estimates.

The results show three distinct SGF behaviors:

Strong reduction on branch-dense workloads. CoreMark drops −31.4% (7-stage) and −30.9% (8-stage), statemate −22.7% on both, with CPI following (CoreMark 2.02→1.96, statemate 1.50→1.48). aha-mont64 also drops 17.2% despite *no* depth-dependent inflation (Table S4 classifies it Mechanism-B–free), because the zero-latency speculative GHR is a more accurate history source than the latency-bound committed GHR at *every* depth (Section 5.6).

No effect on sparse-branch workloads. Dhrystone, Diagnostic, and edn show identical misprediction counts with and without SGF. These workloads have branch spacing wide enough that the committed GHR is already current by the time the next branch is predicted.

crc32 edge case. The crc32 benchmark shows an increase from 344 to 511 mispredictions (+48.5% relative): 167 additional mispredictions out of 174,591 total branches (0.096%), with CPI unchanged at 1.52. Section 5.7 dissects this interaction and the confidence-filter countermeasure we measured but deliberately rejected (it would fix crc32 but costs more CoreMark benefit than it saves).

Operating regime. Across the seven workloads (Table 5), SGF nets a benefit on three (CoreMark, statemate, aha-mont64), is neutral on three (edn, Diagnostic widely spaced; Dhrystone already

accurate), and is CPI-neutral on crc32 despite a small misprediction blip. No workload shows CPI degradation. The hardware cost is workload-independent (+49 LUTs / +57 FFs, 0.6–0.7%, no F_{max} loss); the largest single benefit is CoreMark's 159,932-cycle saving.

5.6 Key Finding: Deep-Pipeline Accuracy Below Shallow-Pipeline Baseline

SGF does not merely restore deep-pipeline accuracy to the shallow level; it pushes counts *below* the 4/5-stage baseline: CoreMark 101,418 vs. 145,735 (30.4% fewer), statemate 56,627 vs. 66,603 (15.0%), aha-mont64 115,644 vs. 139,739 (17.2%). The reason is the update-latency asymmetry: `spec_ghr` has zero update latency while even the shallowest baseline has two stages, so when predictions are mostly correct (79.8% on CoreMark) it approximates true history better than any latency-bound committed GHR. SGF is therefore beneficial at *any* depth that uses global history, not only where Mechanism B is measurable.

This also explains why the 31.4% reduction far exceeds the +5.7% Mechanism B inflation at the 6-stage: that figure is only the *marginal* latency added by deepening, whereas SGF removes the *entire* update latency, including the 2-stage component shared by every depth. Mechanism B is the depth-dependent slice of a larger latency cost.

CPI impact. The 31% misprediction cut yields a 3% CPI improvement (2.02→1.96 on the 7-stage), since Mechanism B is one of four CPI components (Mechanism A ∼0.20, IF2 bubble ∼0.28, load-use ∼0.15); the rest are structural and outside SGF's scope.

5.7 crc32 Edge Case and the Confidence-Filter Tradeoff

One workload, crc32, shows a misprediction *increase* under SGF: 167 more out of 174,591 branches (0.096%), changing CPI by less than 0.01 (1.52 in both cases). It is a structural edge case, not a counterexample. As a class, SGF can *raise* the raw misprediction count on workloads dominated by saturated-counter, highly-biased branches, where the baseline's update latency was accidentally beneficial. The effect is predictable from the PHT-saturation interaction and stays CPI-neutral across our suite.

Structural mechanics. The crc32 inner loop has one dominant branch (`if (crc & 1)`), taken 99.8% of iterations, so its PHT counter saturates and the committed GHR holds a long run of `1`s. Under the baseline, a rare not-taken misprediction does not reach the GHR until EX2, after the next iteration has already been predicted with clean (stale but accidentally accurate) history, so update latency acts as an accidental filter. SGF instead shifts the wrong `0` into `spec_ghr` immediately, so the next prediction uses a polluted PC XOR GHR index (a different, possibly untrained counter). This adds 1–3 mispredictions until correct predictions or the EX2 rollback restore the pattern.

A confidence filter would close it, at a cost. Suppressing speculative updates when the PHT counter is saturated (one 2-bit comparator, no flip-flops) restores crc32 to its baseline 344 mispredictions. It also forfeits useful freshening on stably-biased hot branches, however: on CoreMark it raises mispredictions from 101,418 to 117,207, cutting SGF's reduction from 31.4% to 20.7%. The trade is lopsided: a 167-misprediction, zero-CPI blip on crc32 against ∼16,000 real CoreMark savings. We therefore ship SGF without the filter (crc32 CPI unchanged at 1.52, the extra flushes negligible against the loop runtime) and retain it only as a build-time option for crc-like, highly-biased workloads.

5.8 SGF at the 6-Stage Design Point

Because the 6-stage is the highest-throughput depth (Section 4.6), we also implemented SGF there, the same dual-GHR mechanism forwarded through the shorter IF/ID/EX1/EX2 path. On CoreMark it reduces mispredictions from 154,077 to 117,207 (−23.9%, CPI 1.83→1.79) and on statemate from 69,933 to 54,329 (−22.3%, CPI 1.37→1.35), with identical instruction and branch counts. The

reduction is smaller than at 7/8-stage (−31.4%) because the 6-stage's 3-stage update latency leaves less staleness to correct, consistent with Mechanism B scaling with depth. The biased-branch edge case (Section 5.7) recurs as small CPI-neutral increases. SGF is thus beneficial at every depth with nonzero update latency.

5.9 Compiler-Optimization Sensitivity (-O2)

A key robustness question is whether aggressive optimization, by unrolling and inlining, widens inter-branch distance enough to dissolve Mechanism B. We rebuilt CoreMark and statemate at `-O2` and re-ran the 7-stage baseline and SGF. `-O2` reshapes the workloads (CoreMark branches −25.8%, IBD 4.0→4.5; statemate IBD 8.4→7.5) but both stay under the IBD ≈ 10 threshold, so Mechanism B persists and SGF still helps: CoreMark 141,212 → 79,705 (−43.6%, CPI 1.86→1.76) and statemate 83,257 → 73,267 (−12.0%, 1.44→1.43). The CoreMark reduction is in fact *larger* than at `-O1` (−31.4%); statemate moves the other way, `-O2` roughly halving its relative benefit (−22.7% to −12.0%). SGF's gain is thus optimization-sensitive in magnitude, but its sign and CPI-neutrality hold under both settings.

5.10 Generalization to Other Correlation Predictors

SGF is developed on gshare, but its mechanism is predictor-agnostic. It applies wherever a prediction reads a history register that the branch updates only after some latency. The fix is always the same: a speculative copy, updated at prediction, forwarded as a per-branch checkpoint, and restored on misprediction. Two properties carry across topologies unchanged. The in-order single-issue pipeline admits no nested rollback (Section 5.3), so one checkpoint per in-flight branch suffices and no ROB is needed. The cost stays linear in the speculative-history width carried per branch: that width across the IF2-to-EX2 span, plus one same-width rollback multiplexer.

To test this beyond gshare, we implemented and measured two further predictor families on the 7-stage pipeline, each in baseline and SGF variants. The first is a tournament predictor: a global gshare PHT and a PC-indexed local PHT, arbitrated by a GHR-indexed chooser. The second is a downscaled TAGE predictor [21]: a bimodal base plus three tagged components with geometric global-history lengths of 4, 8, and 16 bits. Table 6 reports the misprediction change for each. Instruction and branch counts are bit-identical between every baseline and its SGF variant (the predictor is performance-only), so the deltas are exact. Mechanism B reproduces on both predictors, and SGF reduces mispredictions on the history-dense workloads in every case.

The magnitude tracks how history-dependent each predictor is. The tournament reductions are smaller than standalone gshare's because the chooser routes a share of branches to the staleness-immune local component. statemate is the clearest case. Stale history had pinned the tournament's global component at bimodal accuracy (63,274 vs. the 63,272 bimodal control of Section 7.1.2). SGF lifts it to 56,619, within eight mispredictions of standalone gshare+SGF (56,627). TAGE, whose three tagged tables are all history-indexed, has the most stale-history headroom and matches gshare's CoreMark reduction (−31.1%).

The biased-branch edge case of Section 5.7 also generalizes but relocates with predictor structure: on gshare it raises crc32, while on TAGE it instead raises aha-mont64 (+5.0%, CPI-neutral) as biased loops pollute the history-indexed tables, and crc32 itself improves. The saturated-provider confidence filter that fixes the gshare crc32 case does *not* transfer to TAGE: we measured that it worsens aha-mont64 (to +12.2%) and erodes the CoreMark reduction (to −20.1%), because holding the speculative shift across TAGE's three history-indexed tables starves them of fresh history. We therefore report the unfiltered TAGE counts and accept the small, CPI-neutral aha-mont64 increase.

The hardware cost, in contrast, does not generalize uniformly, and the measurements confirm the multi-table caveat of Section 5.2. On the tournament predictor SGF stays cheap: +50 LUTs (0.6%),

Table 6. SGF misprediction change (Δ) across three predictor families on the 7-stage pipeline (Artix-7). Instruction and branch counts are identical between each baseline and its SGF variant, so the deltas are exact. Section 5.10 interprets the trends.

Workload	gshare	Tournament	TAGE
CoreMark	−31.4%	−3.1%	−31.1%
statemate	−22.7%	−10.5%	−11.1%
aha-mont64	−17.2%	−2.8%	+5.0%
crc32	+48.5%	−48.3%	−48.8%

+44 flip-flops (0.5%), and no frequency penalty (112.6 vs. 111.7 MHz across three seeds), since one forwarded global checkpoint corrects both the global PHT and the chooser. On TAGE the checkpoint flip-flops stay modest (+74), but LUTs grow +2,434 (+21%). Each of the three geometric-history tables hashes its index and tag from the speculative history for the read and from the forwarded checkpoint for the write. That per-table dual hashing does not share. This is the super-linear growth anticipated in Section 5.2, now measured. TAGE is also expensive before SGF (about 1.5× the gshare core's LUTs and a frequency tier slower, ~94 MHz), so it is not the predictor a resource-constrained soft core would choose on this fabric. SGF delivers a misprediction reduction at negligible cost on the single-global-history predictors (gshare, tournament) that suit such cores; on a multi-table predictor the benefit persists but the low cost does not.

6 Discussion

6.1 Pipeline Partitioning Boundaries on Artix-7

The data identify one partitioning boundary that matters on this fabric: splitting the monolithic execute stage into EX1 (forwarding) and EX2 (computation). This split produces a 56% frequency improvement (74 to 117 MHz). No other split, IF into IF1/IF2, MEM into MEM1/MEM2, produces a statistically significant frequency gain (within-tier differences are <2%). The practical design question on Artix-7 is therefore binary: split the execute stage (gain 56% frequency at ~0.12 CPI per added stage) or leave frequency capped regardless of depth. This contrasts with ASICs, where F_{max} rises roughly monotonically with depth because gate delay scales with logic levels [8]. On FPGAs, function-independent LUT delay and undiminished routing delay [13] mean only breaks of LUT-dominated chains (the execute stage) help. The boundary is thus specific to Artix-7's 6-input LUT fabric and Vivado 2025.2; a different LUT depth, routing topology, or process node could move it. The CPI model's per-stage costs, however, are hazard-driven and fabric-independent, so the model and SGF's misprediction reduction transfer to other FPGA families; only the frequency tier boundaries shift. The physical cause is concrete. In the 4/5-stage designs the execute stage chains the forwarding mux, the ALU (32-bit CARRY4 adder ~3.2 ns plus barrel shifter), and the branch comparator on one path (~8.5 ns: ~4.1 ns logic, ~4.4 ns routing, i.e. ~74 MHz). The EX1/EX2 split registers mid-path, halving the per-stage logic delay and dropping the path to ~3.5 ns (~117 MHz). Subsequent IF/MEM splits yield no gain because those paths are routing-dominated (~1.5 ns LUT vs. ~4.0 ns routing), and register insertion does not reduce routing delay.

6.2 Cache Miss Sensitivity Analysis

All benchmarks execute from single-cycle memory (near-100% hit). In production, cache misses add a depth-independent term $CPI_{total} = CPI_{core} + r_{miss}\, p_{miss}$ (miss rate × penalty) that dilutes branch-prediction gains. SGF's *absolute* cycle saving on 7-stage CoreMark (159,932 flush cycles) is miss-rate-independent, so its *relative* benefit shrinks as memory stalls grow: 3.0% at zero misses,

to 2.7% (1%/20-cycle), 2.0% (2%/50-cycle), and 0.9% (5%/100-cycle). SGF is thus most valuable in embedded systems with tightly coupled memories, the primary FPGA-soft-core use case, and least where cache-miss penalties dominate. Table S1 (supplementary material) works the 7-stage CoreMark case through four miss scenarios.

6.3 Scope and Design Guidance

We separate what generalizes from what does not. The mechanism is an *architectural fact*: Mechanism B arises for any global-history predictor with nonzero update latency, shown causally by the bimodal and PHT controls, so the decomposition and the ROB-free construction transfer. The *magnitudes* are workload- and predictor-specific, and the IBD threshold is a *heuristic* grounded in only two positive workloads (Section 4.4). SGF is a cost-benefit improvement, not a throughput gain, and the optimal-depth conclusion holds only under the stated `-O1`, benchmark, and single-cycle-memory boundaries (Sections 6.2, 8).

For RV32IM-class soft cores on Artix-7: the 6-stage is the highest-throughput depth (64.1 MIPS CoreMark), capturing the full execute-split frequency benefit at moderate CPI; the 5-stage is dominated (4-stage frequency ceiling plus load-use stalls), useful only when area is binding; and the 7/8-stage match the 6-stage frequency but pay higher CPI, justified only when the IF split resolves a real timing bottleneck such as synchronous BRAM. SGF does *not* let a deeper pipeline out-throughput the 6-stage (7-stage+SGF: 58.7 vs. 64.1 MIPS), but is worth adding at any forced depth with ≥ 3 stages of update latency, including the 6-stage (-23.9% CoreMark), for 0.5–0.7% area against a 17–31% misprediction cut.

7 Causal Isolation and Supporting Analyses

7.1 Causal Isolation of Mechanism B

The baseline measurements in Section 4.4 established a correlation between pipeline depth and misprediction count on branch-dense workloads. Two alternative explanations must be excluded before attributing this to GHR staleness: (1) the inflation could be an aliasing artifact of the 64-entry PHT, resolvable by increasing table capacity; or (2) some other pipeline-structural effect (e.g., forwarding interactions) could cause additional mispredictions at deeper depths independent of the predictor. We design two controlled experiments to rule out each alternative.

/.1.1 PHT Capacity Sweep on CoreMark and statemate. We swept the PHT across six sizes from 32 to 1024 entries on all five pipeline depths for the two workloads that exhibit Mechanism B, CoreMark and statemate (60 simulation runs). Table 7 presents four representative sizes (the 32× extremes plus the 64-entry default and 256-entry midpoint); the trends hold at the omitted 128- and 512-entry points.

statemate (controlled result). We treat statemate as the load-bearing result of this experiment. Its branch outcomes are data-dependent, yet its branch pattern is sparse enough (IBD = 8.4) that aliasing does not dominate, so the depth effect is observed against a stable baseline. Across a 32× capacity range (PHT=32 to 1024), the misprediction inflation at 7/8-stage depth is +10.0% at every PHT size from 32 to 256. Absolute counts vary by fewer than 8 mispredictions out of 73,000. At PHT=1024 the larger table resolves some aliasing and lowers the absolute baseline. Yet the depth-dependent inflation does not merely persist; it grows to +11.8%. No PHT capacity from 32 to 1024 entries closes the depth gap. This is the expected signature of an effect that is not caused by table capacity: a 32× sweep cannot remove what aliasing did not create.

CoreMark (high-aliasing corroboration). CoreMark has the densest branch pattern of any workload (IBD = 4.0) and therefore the highest aliasing pressure. Its absolute baseline counts swing non-monotonically with table size (145K–172K; the 64-entry table happens to alias favorably, the

Table 7. Misprediction counts (RTL simulation) across PHT sizes and pipeline depths on CoreMark and statemate (gshare predictor). Δ shows inflation relative to the 4/5-stage baseline at each PHT size.

	PHT=32		PHT=64		PHT=256		PHT=1024	
Depth	**CM**	**SM**	**CM**	**SM**	**CM**	**SM**	**CM**	**SM**
4-stg	150,502	66,599	145,735	66,603	172,161	66,607	164,997	56,623
5-stg	150,502	66,599	145,735	66,603	172,161	66,607	164,997	56,623
6-stg	177,857	69,929	154,077	69,933	179,277	69,935	166,945	63,282
7-stg	152,376	73,259	147,760	73,263	174,436	73,266	166,934	63,283
8-stg	152,376	73,259	147,760	73,263	174,436	73,266	166,934	63,283
Δ 6-stg	+18.2%	+5.0%	+5.7%	+5.0%	+4.1%	+5.0%	+1.2%	+11.8%
Δ 7/8-stg	+1.2%	+10.0%	+1.4%	+10.0%	+1.3%	+10.0%	+1.2%	+11.8%

Table 8. Bimodal predictor (no GHR) misprediction counts on CoreMark and statemate. The counts are identical across all five depths: zero Mechanism B inflation.

Depth	CoreMark	statemate
4-stg	85,493	63,272
5-stg	85,493	63,272
6-stg	85,493	63,272
7-stg	85,493	63,272
8-stg	85,493	63,272

256-entry table unfavorably). This aliasing noise partially masks the depth effect, so we deliberately do *not* rest the causal claim on CoreMark's inflation magnitude. Two facts nonetheless survive the aliasing: the 6-stage exceeds the 4/5-stage baseline at *every* one of the six table sizes, and the 7/8-stage shows a consistent +0.7–1.9% inflation at every size. The contrast between the two workloads is itself diagnostic. The table-size-dependent baseline swing isolates the *aliasing* axis, while the table-size-invariant depth gap isolates the *staleness* axis; the two are orthogonal. A larger table reduces absolute mispredictions but does not correct the stale-GHR inflation that SGF targets, and only the latter is the subject of this paper.

/.1.2 Bimodal Predictor Control Experiment. To confirm that Mechanism B requires a GHR, we replaced the gshare predictor with a bimodal predictor that indexes the PHT using only the PC (no global history register). We ran CoreMark and statemate on all five pipeline depths. The bimodal predictor produces *exactly* the same misprediction count at every depth on both workloads: 85,493 (CoreMark) and 63,272 (statemate), not a single additional misprediction at any depth (Table 8). On the same workloads and pipelines, gshare shows +5.7% inflation on CoreMark at 6-stage and +10.0% on statemate at 7/8-stage (Table 7). The only architectural difference is the presence of a GHR in the prediction index.

Causal conclusion. Together these two controlled experiments isolate the cause:

(1) The bimodal control shows that Mechanism B *requires* a GHR: without one, no depth-dependent misprediction variation occurs on any workload.

(2) The PHT capacity sweep shows that Mechanism B is *not* an aliasing artifact: the inflation persists at every table size from 32 to 1024 entries.

(3) These two controls rule out the alternative explanations, leaving stale GHR update latency as the cause; SGF removes that latency and is the matching fix.

Is SGF better than abandoning global history? For area-constrained designs the bimodal counts answer whether a correlation predictor with SGF beats a GHR-free bimodal of the same size, and the answer is workload-dependent. On statemate global history pays off: gshare+SGF (56,627) < bimodal (63,272) < stale-GHR gshare (73,263), so SGF makes the predictor worth its GHR. On CoreMark the 64-entry gshare is aliasing-bound and bimodal (85,493) beats gshare even with SGF (101,418), since SGF removes staleness but not table-capacity aliasing. The guidance is thus not universal: SGF is worth it when global history helps; where a small table leaves gshare aliasing-bound, a bimodal predictor can be the better choice.

7.2 BRAM Instruction Memory Effect

To test whether the 7-stage IF split becomes beneficial with synchronous memory, we synthesized BRAM variants of all five pipelines using a registered-output instruction memory that infers Block RAM. With BRAM the deep tier rises to 121.5 MHz (6-stage), 120.9 MHz (7-stage), and 121.2 MHz (8-stage). The 7/8-stage improve from their LUT-memory 115.0/117.5 MHz to nearly match the 6-stage, while the shallow tier holds (4-stage 69.3, 5-stage 73.7 MHz). The 4-stage dips slightly because the registered BRAM output lengthens its already-critical execute path. The IF1/IF2 boundary naturally absorbs the BRAM's 1-cycle read latency, making the 7-stage frequency-competitive; but its deeper-pipeline CPI overhead remains, so it still needs SGF and moderate-branch-density workloads to match the 6-stage's throughput. This also answers the memory-mapping concern: the main study uses distributed LUT RAM to hold memory organization *invariant* across depths, and these BRAM variants are the measured counterfactual with true synchronous-read primitives. Mapping to BRAM shifts frequency but not the CPI ordering, so the 6-stage stays the throughput optimum and a real-BRAM mapping strengthens rather than invalidates it. Figure S3 (supplementary material) compares the two memory mappings across depths.

7.3 Power and Efficiency

Power is estimated with Vivado's `report_power`, annotated with workload-specific switching activity (a SAIF file from a behavioral CoreMark run of each variant). These are vector-based estimates at "Medium" confidence, not physical measurements (±20–30% versus silicon), so we rely on the *relative ordering across depths*, not absolute watts. Total on-chip power *falls* with depth, from 0.235 W (4-stage) to 0.217 W (8-stage): deeper pipelines add registers, but their lower IPC reduces average switching activity, so on the same workload they draw less dynamic power, a trend only a workload-derived SAIF captures. By energy ($E = P_{\text{dyn}} \cdot \text{Cycles}/F_{\max}$) and EDP, the 6-stage is best (7.0 mJ, 318 mJ · ms, 286 MIPS/W) versus 10.6 mJ/675/195 for the 4-stage and 8.0 mJ/432/248 for the 8-stage, making it the most energy-efficient depth as well as the highest-throughput (a relative comparison, not a measured criterion). SGF's +49 LUT/+57 FF add <1% to dynamic power. Figures S4 and S5 (supplementary material) show the power breakdown and the efficiency.

8 Limitations

The boundaries of this study constrain every conclusion drawn above and must be stated precisely.

Fabric and tool dependence. All synthesis, timing, and power data are from Artix-7 XC7A35TCPG236-1 with Vivado 2025.2. The two-tier $F_{\max}$ structure is a property of the 6-input LUT fabric and Vivado's placer/router; a different LUT architecture, routing topology, process node, or tool could shift the execute-split knee or eliminate it, and the absolute $F_{\max}$/area/power figures do not transfer without re-synthesis. The 10-directive methodology mitigates single-run bias within Vivado but not cross-tool variation.

Single microarchitecture and memory scope. The core is a research vehicle lacking the C extension, out-of-order execution, multi-level caches, TLB/MMU, and I/O; a different forwarding

topology, superscalar issue, or deeper memory hierarchy could change the optimal depth and SGF impact. All workloads fit in 16 KB at near-100% hit rate, so the analytical cache-miss bound (Section 6.2) needs re-evaluation against a real hierarchy.

CPI model. The model (Eq. 1) predicts the depth-dependent terms from architectural parameters, but its per-workload base γ_w is fitted, not first-principles, and is validated only on the seven workloads used for fitting.

Predictor scope. SGF is measured on three families (gshare, tournament, downscaled TAGE; Section 5.10); the stale-GHR effect is not gshare-specific. On TAGE [21] the benefit persists but the area cost grows to +21% LUTs. A measured evaluation on perceptron [9] and local-history [29] predictors is future work, and the crc32 confidence filter ships off by default.

Benchmark and compiler scope. The seven workloads (branch frequency 4.4–26.3%, 1.4K–4.5M instructions) exclude floating-point, OS-level, and multi-threaded code, and the IBD indicator (Table S4, supplementary material) rests on only two positive workloads. Headline results use `-O1`; an `-O2` re-measurement (Section 5.9) shows Mechanism B and SGF's benefit persist (CoreMark's reduction grows), so the effect is not an `-O1` artifact. `-O3`, floating-point, OS-level, and multi-threaded workloads remain untested.

9 Conclusion

This paper makes two contributions to pipelined FPGA soft-core design. First, a controlled depth experiment (five RV32IM variants from shared RTL on Artix-7) separates depth-dependent CPI into an inherent flush penalty (Mechanism A) and a correctable stale-GHR penalty (Mechanism B, up to 10%), isolated causally by a no-GHR bimodal control (zero inflation) and a 32× PHT sweep (inflation at every size), and captured by a first-order CPI model ($R^2 = 0.977$, $R^2_{\mathrm{CV}} = 0.941$). The same experiment reveals a two-tier $F_{\max}$ structure ($p < 0.001$, $d = 18.5$) whose sole knee is the execute-stage split, making the 6-stage the highest-throughput depth (fabric- and tool-conditional, not universal).

Second, Speculative GHR Forwarding eliminates Mechanism B with a ROB-free dual-GHR architecture that updates speculatively at prediction and rolls back from pipeline-register checkpoints. It cuts 7-stage CoreMark mispredictions by 31.4% and statemate by 22.7% at 0.5–0.7% area and no frequency penalty, pushing deep-pipeline accuracy below the shallow baseline with no CPI degradation on any workload. SGF is a low-risk addition to any gshare-class in-order pipeline with three or more stages of update latency.

What transfers without re-measurement is the Mechanism-A/B decomposition and the ROB-free construction, not the specific percentages; measured tournament and TAGE predictors confirm the benefit reproduces, at sub-1% cost on single-history predictors and a larger cost on multi-table TAGE. Future work extends this to `-O3` and floating-point/OS workloads, to perceptron and local-history predictors, and to other FPGA fabrics with physical-hardware validation. The complete RTL, testbenches, synthesis scripts, and benchmark programs are available at https://github.com/devanshjoshi08/RISCV-Pipeline-Research.

References

[1] Krste Asanović, Rimas Avižienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, et al. 2016. *The Rocket Chip Generator*. Technical Report UCB/EECS-2016-17. EECS Department, University of California, Berkeley.

[2] M. Biglari, S. M. Fakhraie, and M. T. Manzuri. 2019. A Survey on Pipelining Techniques in RISC Processors. In *IEEE 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 1–6.

[3] EEMBC. 2009. CoreMark: An EEMBC Benchmark. https://www.eembc.org/coremark/. Accessed May 2026.

[4] Embench Task Group. 2019. Embench: A Modern Embedded Benchmark Suite. https://github.com/embench/embench-iot. Accessed May 2026.

[5] Roberto Gonzalez and Mark Horowitz. 1999. A power-aware methodology for designing processor pipelines. *IEEE Journal of Solid-State Circuits* 34, 3 (1999), 345–354.
[6] A. Hartstein and Thomas R. Puzak. 2002. The optimum pipeline depth for a microprocessor. In *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 7–13.
[7] John L. Hennessy and David A. Patterson. 2017. *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann, Cambridge, MA.
[8] M. S. Hrishikesh, Norman P. Jouppi, Keith I. Farkas, Doug Burger, Stephen W. Keckler, and Premkishore Shivakumar. 2002. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 14–24.
[9] Daniel A. Jiménez and Calvin Lin. 2001. Dynamic branch prediction with perceptrons. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 197–206.
[10] Tejas S. Karkhanis and James E. Smith. 2004. A first-order superscalar processor model. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 338–349.
[11] Richard E. Kessler. 1999. The Alpha 21264 microprocessor. *IEEE Micro* 19, 2 (1999), 24–36.
[12] Olof Kindgren. 2019. SERV: The SErial RISC-V CPU. https://github.com/olofk/serv. Accessed May 2026.
[13] Ian Kuon and Jonathan Rose. 2007. Measuring the gap between FPGAs and ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26, 2 (2007), 203–215.
[14] Scott McFarling. 1993. *Combining branch predictors*. Technical Report WRL Technical Note TN-36. Digital Western Research Laboratory.
[15] Pierre Michaud, André Seznec, and Richard Uhlig. 1999. Trading conflict and capacity aliasing in conditional branch predictors. In *Proceedings of the 26th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 292–303.
[16] Charles Papon. 2018. VexRiscv: A RISC-V implementation written in SpinalHDL. https://github.com/SpinalHDL/VexRiscv. Accessed May 2026.
[17] David A. Patterson. 1985. Reduced instruction set computers. *Commun. ACM* 28, 1 (1985), 8–21.
[18] David A. Patterson and John L. Hennessy. 2020. *Computer Organization and Design: The Hardware/Software Interface, RISC-V Edition* (2nd ed.). Morgan Kaufmann, Cambridge, MA.
[19] Pasquale Davide Schiavone, Davide Rossi, Michael Gautschi, Antonio Pullini, Francesco Conti, and Luca Benini. 2017. Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications. In *27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 1–8.
[20] André Seznec. 2006. *Revisiting the perceptron predictor*. Technical Report PI-1878. IRISA.
[21] André Seznec. 2011. A new case for the TAGE branch predictor. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. ACM, 117–127.
[22] James E. Smith. 1981. A study of branch prediction strategies. In *Proceedings of the 8th Annual Symposium on Computer Architecture (ISCA)*. 135–148.
[23] Eric Sprangle and Doug Carmean. 2002. Increasing processor performance by implementing deeper pipelines. In *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 25–34.
[24] Andrew Waterman and Krste Asanović. 2019. *The RISC-V Reader: An Open Architecture Atlas*. Strawberry Canyon LLC, Berkeley, CA.
[25] Western Digital Corporation. 2019. SweRV EH1 RISC-V Core. https://github.com/chipsalliance/Cores-SweRV. Accessed May 2026.
[26] Clifford Wolf. 2017. PicoRV32: A Size-Optimized RISC-V CPU. https://github.com/YosysHQ/picorv32. Accessed May 2026.
[27] Xilinx Inc. 2018. *7 Series FPGAs Configurable Logic Block User Guide (UG474)*. Xilinx Inc., San Jose, CA. v1.8.
[28] Kenneth C. Yeager. 1996. The MIPS R10000 superscalar microprocessor. *IEEE Micro* 16, 2 (1996), 28–40.
[29] Tse-Yu Yeh and Yale N. Patt. 1991. Two-level adaptive training branch prediction. In *Proceedings of the 24th Annual International Symposium on Microarchitecture (MICRO)*. ACM/IEEE, 51–61.
[30] Florian Zaruba and Luca Benini. 2019. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27, 11 (2019), 2629–2640.