SH7604 cache purge ordering and write-buffer behaviour, measured on a CPS3 board

J.R.Dunkley Board research: XeD

19 October 2026

Abstract. Our CPS3 experiments exposed stale data and a stale instruction fetch after a cache-through store followed by a CCR purge. An ordering rule fits all eight data values: the CCR write waits behind the pending store while hits can still use the old line. Related loops separate write-buffer overlap, load-use and fetch-port effects, and multiply issue from result availability. These observations may help Saturn and 32X emulator, FPGA and homebrew developers frame experiments of their own; they are not measurements of either console.

1 Scope and evidence

This note extracts the CPU-side questions from §§5.4–5.5 and §8.4 of *Cache, Bus, Interrupt Order and Slowdown on a CPS3* [1], using its companion data [2]. There was **one board and one stock SH7604-family part**, running at nominally 25 MHz in a Darksoft-equipped CPS3 under the CPS3 bus settings: BCR1=`0020`, BCR2=`00E8`, WCR=`AA57`. Mechanisms may transfer to related parts. Clock counts transfer only where no external bus cycle is involved, and still depend on the instruction stream, alignment and cache state.

We retain the paper's tags: BOARD, recovered physical observations; MANUAL, Hitachi's documentation; SRC, pinned source inspection; MODEL, the paper's calibrated emulator; INFER, an interpretation or prediction. P2's 16,128-byte capture has CRC-32 `881D2B40`; its raw values and timed samples are in the companion. One FRT tick is eight CPU clocks. No new hardware or emulator execution is reported here.

2 Purge ordering: the eight values and the trap

P2 tests 185 and 186 use cached address `0202AA00` and its cache-through alias `2202AA00`. Each repetition stores 17 through the alias, requests CCR=`11` (CP=1, CE=1), reads the cached address, stores 34 through the alias, then reads the cached address again. Test 186 inserts another CCR=`11` before the second read. Addresses, CCR values and packed results are hexadecimal: `1122` packs reads of 17, then 34.

P2 test (main paper, Table 9)	Rep. 0	Rep. 1	Rep. 2	Rep. 3
185: `data_alias`, one purge	`1111`	`1122`	`2222`	`2222`
186: `data_purge`, two purges	`2222`	`2222`	`2222`	`2222`

These eight values are BOARD; the following explanation is INFER. The sequence has the form of a CCR write ordered behind a pending cache-through store, with hits served meanwhile. Initially the line is invalid: the first miss waits, fills 17, and the second read hits 17. Next repetition, the first read hits that old 17 before the purge takes effect; the second misses and fills 34. Thereafter the first read can hit old 34. Test 186 inherits that line; a later miss waits for the second store and purge. This rule fits all eight values. A never-effective purge or a simple fixed-delay account does not fit the complete pattern; the data do not uniquely identify the internal circuit.

Test 187, `code_alias`, then wrote `MOV #17,R0; RTS; NOP` through the alias, requested a purge and called the cached address. P2 trapped at PC `0202AA00` (BOARD). The old line contained `00000022`; fetching its first halfword as illegal opcode `0000` explains the trap (INFER). **P2 stopped at test 187. Associative purge was never tested; no result or safety claim about it follows.**

3 CPU-side timing observations

The numbers below are throughputs or whole-loop totals, not isolated instruction latencies. P2 took eight samples per CPU test; these are medians of warm samples 1, 3, 5 and 7. The companion identifies the bodies, trip counts, widths and CCR settings [2]. Bus-dependent examples illustrate overlap on this board, not Saturn or 32X wait-state constants.

Conflict cost has the form of a line fill. P2 cycles through d lines for 256 iterations. A 1,024-byte stride selects one set; a 16-byte stride spreads the lines. In four-way mode the cliff is at five lines; in two-way mode it is at three. Across the six thrashing cases,

$$\Delta C = \frac{8(T_{1024} - T_{16})}{256d} = 35.73\text{–}35.94 \quad \text{CPU clocks per load.}$$

That agrees with P1's independently measured 35.97-clock work-RAM line fill (BOARD). Cyclic thrashing has the form expected from LRU replacement (INFER); it does not prove every replacement sequence. The cache-set mechanism may transfer, but the approximately 36-clock miss cost includes CPS3 external bus service.

A store can overlap useful instructions. For eight store groups per iteration, the slope between 64 and 256 iterations removes fixed runner costs. Each group is a cache-through RAM store followed by g independent `ADD`s. The slope $8\Delta T/(8\Delta N)$ gives clocks per group:

Store width	$g = 0$	$g = 2$	$g = 8$
Byte / word	4.22 / 4.23	4.59 / 4.59	10.00 / 10.01
Long	8.46	8.46	10.14

The flat long-store result at $g = 0, 2$ is BOARD. It has the form of a posted write allowing cached instructions to proceed (INFER, consistent with MANUAL §7.11.2). How many instructions fit behind the write depends on its bus duration.

Write then read. For 512 longword pairs, a same-address cache-through RAM store and read take 1,475.5, 1,475.5 and 1,476 ticks at gaps 0, 2 and 8. Different-address RAM reads give 1,475 ticks at each gap (BOARD, tests 91–96). Up to 4,096 added instructions are almost hidden. The near-equality of same- and different-address pairs shows no timing advantage from store-to-load forwarding in these tests; it is not a proof that forwarding can never occur. A read waiting for the external write is a fitting account (INFER). These totals include CPS3 bus cycles.

Cached-load port contention and load-use. P1's cached loads cost 1.49–1.50 effective clocks each after empty-loop subtraction (BOARD). P2 executes 2,048 cached load-and-consume groups, for each of byte, word and long widths. Gaps of 0, 1 and 4 independent `ADD`s yield 898.5–899, 1,026.5–1,027 and 1,922–1,923 ticks. A one-clock load-use interlock together with contention between memory access and instruction fetch fits these nine totals within 0.3% (MODEL). The fitted port rule charges a clock when the following instruction is at a $4n$ address. That rule is a model of the measurements, not a separate observation of an internal port.

Multiply issue and result are different questions. For 2,048 multiplies of constants 3 and 5, P2 measured:

	Result not read			Result read and used		
Instruction / independent gap	0	1	3	0	1	3
`MUL.L`, `DMULS.L`	1091	1155	1667	2307	2435	2435
`MULS.W`	611	899	1411	1411	1411	1923

All entries are ticks (BOARD, tests 151–168). "Read and used" means `STS MACL` followed by an `ADD` consuming that result. Extra instructions can occupy time already spent waiting for a result. The paper's long-multiply model uses four clocks of unit occupancy and a result deadline five or six clocks after issue, depending on fetch alignment (MODEL); those fitted states are not independently measured latencies.

4 A small replication sequence

Portable core, platform-specific harness. Reserve a 16-byte-aligned writable RAM line A, its cache-through alias $U = A$ OR `20000000`, a separate result area and a private stack. On Saturn or 32X, choose an address in that machine's mapped RAM; the CPS3 address above is not portable. Run on one SH-2 with maskable interrupts disabled and the other CPU and DMA kept idle for the experiment. Record the part marking, clock, BSC settings, memory region and code addresses. Keep code, literals and results away from the test line's set where practicable. No display output belongs inside a critical window.

For the baseline, enable the normal four-way cache and establish an invalid test line before test 185 only. Warm the harness without reading A. Then run four repetitions of A followed by four of B, retaining cache contents across repetitions and between A and B. Do not add a purge to the repetition wrapper. P2 used calls, a timer-initialisation helper and a result store outside each critical body; record any wrapper changes when comparing values. The compact body below preserves P2's order, including the CCR-address literal load after the first store. `mov.l =...` denotes an assembler literal-pool load; inspect the emitted instructions and alignment.

```
mov.l =A,r6              ; cached RAM address
mov.l =U,r7              ; same RAM, cache-through
mov #17,r0
mov.l r0,@r7
mov.l =0xFFFFFE92,r2     ; CCR (byte access)
mov #0x11,r0
mov.b r0,@r2
mov.l @r6,r8             ; first value
mov #34,r0
mov.l r0,@r7
; Test B only: insert these two instructions
; mov #0x11,r0
; mov.b r0,@r2
mov.l @r6,r0             ; second value
shll8 r8
or r8,r0                ; packed result in R0
```

Reference output. On the CPS3, A returned `1111,1122,2222,2222`; B immediately afterwards returned `2222,2222,2222,2222` (BOARD). These are the observed target values for a faithful replication, not a guarantee for a port with different bus latency, fetch placement or starting cache state. Record any other pattern rather than adjusting the test to force agreement. P2's timed CPU-loop bodies and test metadata are in `board/p2/probe_build_info.json`; the value-test results are in `lab_values.csv` [2].

Two discriminating controls. First, repeat with a same-address cache-through read after each store that precedes a requested purge. Hitachi describes such a read as a way to wait for a device write to finish (MANUAL, §7.11.2); removal of the race is a hypothesis to test, not a board result. It does not make an unpurged cached copy coherent: a simple immediate-purge cache may give `1111` for A and `1122` for B (INFER). Second, compare the manual's cache-disable/purge/re-enable procedure (§8.4.6). Neither control was established by these P2 data. Associative purge is outside this note's measured scope.

Optional fetch extension. After recording the data tests, install an exception handler that records PC and SR. Through U, write `E011000B` as a longword and `0009` as a word at $U + 4$; request CCR=`11`, then execute `JSR @r6; NOP`. The CPS3 stopped at the first call with PC=A. This extension deliberately probes the stale-fetch trap; retain the failure record separately from successful return values.

Timing extension. Use the companion's `wr_rd_same_g0/g2/g8` bodies (four pairs per iteration, 128 iterations) and warm sampling to compare with 1,475.5/1,475.5/1,476 ticks on CPS3. For CPU-resident tests, use `load_use_*` and `mul_*`: eight groups per iteration, 256 iterations, and the gaps in §3. Preserve loop branches, delay slots and halfword alignment. Compare slopes or matched controls when changing the runner; raw totals include its overhead. The question on another machine is whether the same overlap and dependency patterns occur, not whether its external RAM has the CPS3's timings.

5 What the implementation checks establish

The following is SRC, re-read at pinned commits on 19 October 2026. It is not a test result for any emulator or FPGA. "Immediate" here means that invalidation occurs inside the CCR-write handler, without a separately scheduled purge-completion event; it does not describe the host's speed or settle every surrounding guest-pipeline effect.

Mednafen. At mirror commit `f0ee9d595db6`, `SH7095::SetCCR` invalidates all tags and clears LRU in the call that handles CP. In the Mednafen-derived Beetle Saturn port at `faef2779a756`, the C function is `SH7095_SetCCR`, with the same synchronous invalidation loop [4, 5]. The symbol names distinguish the two sources; the port is not presented as the current upstream Mednafen release.

Ymir. At `8cdf0a530ec1`, `Cache::WriteCCR` writes CCR, calls `Purge()` when CP is set for a normal, non-debug-poke write, then clears CP [6]. The same pin's `saturn.cpp`, lines 735–736, says the alternative timings below it pass `misctest` but run too slowly in practice because cache, write-buffer and BSC emulation are inaccurate. This is the project's own description of a modelling gap, not a measured diagnosis of its output on P2.

jgenesis. At `b4f943bb5f6f`, `write_control` calls `purge_all` immediately when bit 4 is set; that function clears valid bits and LRU [7]. These three implementations therefore offer a useful comparison with the board's delayed-effect interpretation. Their outputs on the sequence above remain **untested here**. A cache with synchronous stores, immediate purges and no intervening eviction may yield A=`1111` and B=`1122` on every repetition (INFER). A cacheless model may yield `1122` for both. The main paper's Table 9 row labelled "any emulator" is consequently not carried forward as a general claim.

srg320's SH7604 RTL. The re-check does *not* establish that `CACHE.sv` queues CCR writes behind a pending store. In the Saturn MiSTer copy at `9aa944985a96`, lines 419–423 latch CCR on `CCR_SEL && CBUS_WR && CBUS_REQ` at `CE_R`, without an `IBREQ` or pending-store guard. Its hit path (373–377) can assert `CACHE_READ` without waiting for `IBREQ`; misses wait for the bus. The JTCPS3 copy pinned by the main paper, `bfced64dad83`, also latches CCR without that guard (644–648), and its `CBUS_BUSY` expression additionally includes `IBUS_WRITE && !IO_AREA` (660–661) [8]. Thus the source variants must be distinguished. The presence of a posted-store path and a hit path is not by itself proof of queued CCR completion. A whole-core trace may resolve interactions with CPU request/acknowledge timing; no claim that either variant reproduces the eight board values is made.

6 Prior measurements and possible relevance

A web search for SH7604 purge ordering, Saturn cache/write-buffer timing and 32X memory benchmarks found relevant prior hardware-oriented work. jgenesis issue #493 links the `32xspd` benchmark and a prior 32X performance discussion. Its July 2025 reports compare hardware bandwidth with emulator results; the 9 January 2026 follow-up attributes a cached-read discrepancy to SH-2 pipeline MA/IF conflicts [9]. This is directly relevant prior evidence for contention, but it does not replicate P2's purge sequence. The accessible 32XDK notes collect further bus-timing and cache observations [10]. We found no independently documented hardware run of this exact eight-value CCR/store sequence in the sources checked. That is a bounded search result, not a priority claim.

Mednafen's `PROBLEMATIC-GAMES` list separates games relying on instruction-fetch-driven data-cache coherency, direct execution from cache, and data-cache incoherency [4]. It shows why preserving cache contents and access type may matter, but it does not connect those games to this purge race. **The impact of these CPS3 observations on commercial Saturn or 32X games is unknown.** For homebrew and reimplementation work, the small value sequence may expose an ordering distinction that a throughput benchmark misses; the timing loops may distinguish an overlap mechanism from a uniform per-instruction cost.

References and replication record

[1] J.R.Dunkley; board research by XeD. *Cache, Bus, Interrupt Order and Slowdown on a CPS3: Darksoft probes, board-cost model and slowdown census for Street Fighter III: 3rd Strike*. 16 October 2026. §§5.4–5.5, §8.4, Table 9. doi:10.5281/zenodo.23435893.

[2] J.R.Dunkley and XeD. Companion data and code for [1], `data-and-code.zip`, in the same Zenodo record. Archive root: `where_the_frame_goes_board_companion`. Relevant paths: `board/p1_second_return/`, `board/p2/lab_cpu.csv`, `lab_values.csv`, `initial_observations.json`, `probe_build_info.json`; `numbers/`; and `scripts/verify.py`. P2 capture SHA-256: `821dcbf85b981a6cb1116b0c649d97c87875a6026f00a6622c8292cee21dc983`.

[3] Hitachi. *SH7604 Hardware Manual*, ADE-602-085C, rev. 4.0, 19 September 2001. §7.11.2, pp. 210–211; §8.4.6, p. 224. Archived manual PDF. The cache-initialisation discussion distinguishes purge execution from the extra time needed to write CCR and notes instruction reads during that write (MANUAL).

[4] Mednafen contributors. Source mirror, commit `f0ee9d595db68ad5247ba5ac6a8367fdced9c3fc`. `src/ss/sh7095.inc`, `SetCCR`; `src/ss/notes/PROBLEMATIC-GAMES`, lines 1–47.

[5] Beetle Saturn contributors, Mednafen-derived core. Commit `faef2779a7568c1734addc4fe63cc76e9fa2566b`. `mednafen/ss/sh7095.inc`, `SH7095_SetCCR`.

[6] Ymir contributors. Commit `8cdf0a530ec1927f44bf3cd02c6ab51a586c4407`. `sh2_cache.hpp`, `WriteCCR`; `saturn.cpp`, lines 735–756.

[7] J. Groth and contributors. jgenesis, commit `b4f943bb5f6f8e3857a37dd9915d27bdf88ed5dd`. `cpu/sh2-emu/src/cache.rs`, `write_control` and `purge_all`.

[8] S. Dvodnenko (srg320) and contributors. SH7604 `CACHE.sv`, two copies checked: Saturn MiSTer, `9aa944985a96433983ffba27bdb520009fd9a8e8`, `rtl/SH/SH7604/`; J. Tejada's jtcores, `bfced64dad833daa3864e601e9c0e90614c56df8`, `modules/jtframe/hdl/cpu/sh7604/`.

[9] Genos3, J. Groth and contributors. jgenesis issue #493, "[32X] memory benchmark", opened 19 July 2025. See especially J. Groth's 9 January 2026 comment on MA/IF conflicts. Accessed 19 October 2026.

[10] viciious and contributors. "SH2 on the 32X", 32XDK wiki, memory-system and cache notes. Accessed 19 October 2026. This is contextual prior work, not a replication of P2.

Replication. Recording the CPU marking, RAM choice, bus settings, instruction addresses and all eight values, followed by the drained-store control and the guarded fetch extension, may help separate internal ordering from the timing window created by CPS3 memory. A different value pattern would be informative too. **The replication we would most like is this test sequence run on a real Saturn.**