LLM INPUT

Context

You have a small 8-bit microprocessor RTL (Verilog). Key modules relevant to this task are:

```
- rtl/add_sub       (performs A ± bus; controlled by inst[5])
- rtl/ALU           (instantiates add_sub and a mux)
- rtl/dff           (8-bit D flip-flop used to register A..F)
- rtl/mux8_2to1   (8-bit 2:1 mux)
- rtl/tribuff_8_2to1 (8-bit tri-state buffer)
- rtl/decode3to8
- rtl/micro.v          (top-level, instantiates registers, decoders, tri-buffers, ALU)
```

Instruction encoding (high level)
- inst[7:6] == 00: IN/MOV/OUT family
- inst[7:6] == 01: ALU family (ADD/SUB)
- inst[5] controls ADD(0)/SUB(1)
- inst[5:3] -> destination for loads (LD decoder)
- inst[2:0] -> source for OE (OE decoder)

Problem description / symptoms

The repo currently exhibits multiple issues (some intentionally injected). When running the provided `tb/micro_tb.v`, the observed failing behavior includes:

1) ADD and SUB appear swapped in some simulations.
2) Register C never updates when LD[3] is asserted.
3) OUT A sometimes produces `zzzzzzzz` instead of A's value.

I also noticed a **subtle timing/race issue** when trying back-to-back operations:
- Sequence:
  - Cycle N: apply an `ADD B` instruction (A ← A + B).
  - Cycle N+1: immediately perform `OUT A` (drive bus with A).
- Expected: `OUT A` in cycle N+1 shows the newly updated A (result of ADD).
- Observed in failing sim: `OUT A` sometimes shows the old A (pre-ADD), or Z — behavior depends on the scheduling of assignments and tri-state drivers.

Environment & testbench logs (failing run)

Below are extracted monitor lines from a failing run (these are the failing traces you must make sense of):

```
Time(ns)  data_in     inst       data_out
---------------------------------------
   0     00000000   00000000    zzzzzzzz
```

```
  10     00001111   00001000    zzzzzzzz   // IN A (A should become 0x0F at next edge)
  20     00000010   00010000    zzzzzzzz   // IN B (B should become 0x02)
  30     00000000   01000010    zzzzzzzz   // ADD B  (A <= A + B)  expect A_new = 0x11 after this edge
  40     00000000   00000001    00001111   // OUT A  (EXPECTED 00010001 but sees 00001111 => old A)
  50     00000000   00000001    zzzzzzzz   // second OUT A (sometimes shows Z)
```

Notes:
- At time 40 the testbench issued OUT A immediately after the ADD; testbench expected the new A to be visible at time 40, but the bus shows old A or Z.
- The repo has earlier documented intentional fixes for ALU, C-load, and tri-state wiring, but there remains a race caused by the DFF style / blocking vs non-blocking semantics and the mux / tri-state wiring ordering in micro.v.

Your task
----
1. Explain the root causes behind the above failing behavior (including why the immediate OUT after an ADD can show old A or Z).
2. Provide precise code changes (diff/patch or full replacement) that fix all three functional issues **and** remove the timing/race so the "ADD then immediate OUT" sequence reliably shows the new A on the next cycle.
3. Explain why your changes fix the problem and show the **expected passing testbench monitor lines** for the same sequence (show before/after logs).
4. Make any recommended small improvements to make the design less fragile (e.g., non-blocking semantics, assertions).

Constraints

- Do not re-architect the whole design — small, targeted changes are preferred.
- Provide the exact Verilog edits (files and code lines).
- Provide expected `tb` console output (passing run) for the same sequence.
- If you cannot run simulations here, clearly state so and provide the expected logs (what a correct simulation would show).

End of prompt.