EPFL

Institute of Electrical and Microengineering

Biomedical and neuromorphic microelectronic systems

Test of VLSI Systems 2024, MNIS-MA3, EL-MA1, EL-MA3. EE530-2024-EX07S-v1.0.fm
v1.0 A. Schmid 2023, November 8

TEST OF VLSI SYSTEMS

EXERCISE 7 - SOLUTIONS

1. MARCH TEST FOR DATA RETENTION FAULT (Bushnell and Agrawal 9.14)

Rigorously prove that the IFA-13 test catches all data retention faults.

IFA-13 march test:

{ M0: ⇑(w0); M1: ⇑(r0,w1,r1); M2: ⇑(r1,w0,r0);
M3: ⇓(r0,w1,r1); M4: ⇓(r1,w0,r0); M5: Delay;
M6: ⇑(r0,w1); M7: Delay; M8: ⇑(r1) }

Obviously, IFA-13 has not been designed to test retention faults only. Only a portion of the test is dedicated for this purpose.

Necessary conditions for detecting a retention fault:

Each cell must have a 0(1) written to it, and after a suitable delay (e.g. 100ms), a 0(1) must be read back from the cell. A retention fault has occurred if this value read out has flipped to 1(0).

Fault $<1_T/\downarrow>$

- M6 sensitizes the fault by writing a 1 into all cells;
- M7 produces the necessary delay;
- M8 detects the fault when a 0 is read but a 1 was expected.

Fault $<0_T/\uparrow>$

- M4 sensitizes the fault by writing a 0 into all cells;
- M5 produces the necessary delay;
- M6 detects the fault when a 1 is read but a 0 was expected.

This completes the proof.

2. MARCH TEST FOR LINKED FAULTS AND FAULT MASKING

Figure 2.1 shows one memory bank with linked and unlinked faults. In case (a), two aggressor cells are forcing the state of two victim cells, independently. In case (b), the aggressor cells are forcing the state of the same victim cell, however with different idempotent coupling fault types. We assume that the lower addresses of the memory are located on the left side.

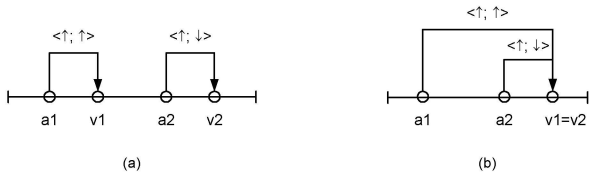

Figure 2.1: Memory bank having (a) unlinked faults, and (b) linked faults.

Show that following march test detects both faults in case (a), but fails in case (b).

{ ⇕(w0); ⇑(r0,w1); ⇓(w0,w1); ⇕(r1) }

Let us first rewrite the march test using labels for consecutive elements:

{ M0: ⇕(w0); M1: ⇑(r0,w1); M2: ⇓(w0,w1); M3: ⇕(r1) }

Case (a)

- M0 fills the memory with 0s;
- M1: w1 to cell a1 will cause cell v1 to flip: <↑ ; ↑> has been sensitized; the fault is detected when the test performs the r0 operation for cell v1 (should be at 0, but has been forced to 1);
- M1 causes the fault <↑ ; ↓> which is not sensitized as the memory v2 is expected to be at 0 at this stage;
- M2 applies an up-transition to all cells. In this case the aggressors are processed after their respective victims. Cell v2 is set to 1 right after M2 has been applied to it. However, in turn, M2 is applied to cell a2 later, which forces cell v2 to 0.
- M3 detects cell v2 at 0, whereas it value in a fault-free circuit should be 1 at this stage.

Case (b)

- M0 fills the memory with 0s;
- immediately after M1 has been applied to cell a1, we have: (format: a1,a2,v1) 101; immediately after M1 has been applied to cell a2, we have: 110; thus, when cell v1 is processed by r0, its value is back to the correct value, and <↑ ; ↑> has been masked by <↑ ; ↓>;
- immediately after M2 has been applied to cell v1, we have: 111; immediately after M2 has been applied to cell a2, we have: 110; immediately after M2 has been applied to cell a1, we have: 111; again, the effect of <↑ ; ↓> has been masked by the effect of <↑ ; ↑>;
- M3 does not detect any irregularities.

Devise a march test sequence that can detect the faults in case (b).

For example:

- { M0: ⇓(w0,w1,w0); M1: ⇳(r0) } detects the <↑ ; ↑> CFid
- { M0: ⇳(w1); M1: ⇑(r1,w0,w1) } the r1 operation of M1 detects the <↑ ; ↓> CFid when processing cell v1
- creating a full march test: { M0: ⇓(w0,w1,w0); M1: ⇳(r0,w1); M2: ⇑(r1,w0,w1) }