Simulation-based Fault Injection on Ibex Core with UVM Environment

Li Lu*†, Junchao Chen*, Markus Ulbricht*, Milos Krstic*†
*IHP-Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany
†University of Potsdam, Potsdam, Germany

Introduction

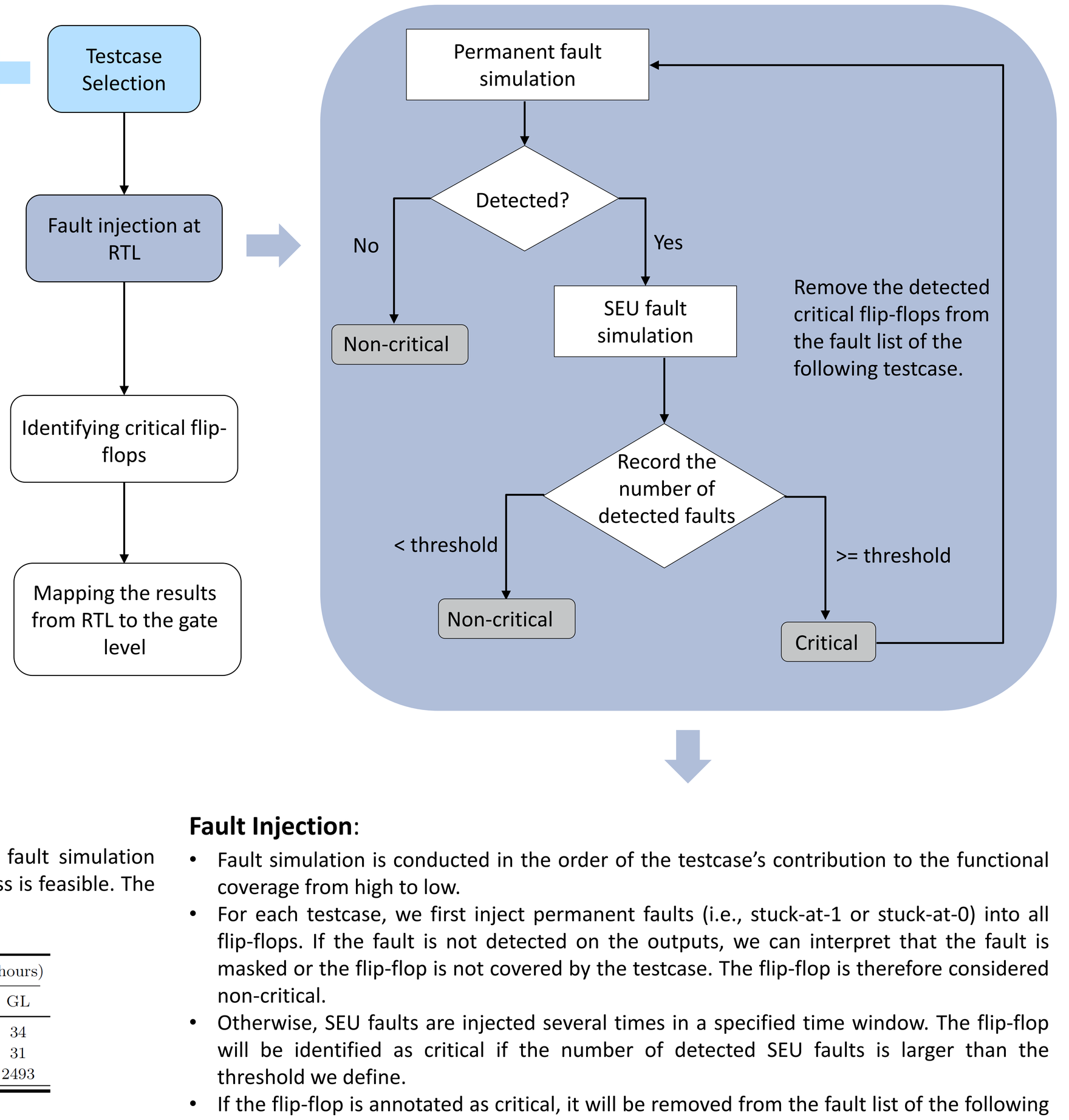
- Simulation-based fault injection enables us to identify the vulnerable parts of a circuit and employ fault mitigation techniques to improve the reliability of circuits before manufacturing.
- Testbenches based on UVM support coverage-driven verification. It can balance verification completeness with minimum verification effort and time.
- This work presents a procedure to conduct simulation-based fault injection at the gate level on the Ibex core with its UVM testbench [1] to identify critical flip-flops which determine the core's correct functioning. The workflow could reduce the time involved in the fault simulation.

Methodologies

- We use the Xcelium fault simulator provided by Cadence for the fault simulation. This simulator enables us to reuse the UVM functional verification testbench to build the fault simulation testbench.

- Integrated Metrics Center (IMC) provided by Cadence is used to analyze the contribution of each test case provided by the UVM testbench to the functional coverage.
- We remove the testcases which don't or less contribute to the coverage and merge some of them. The function coverage of the selected testcases is around 98.2%.

- The fault simulation is implemented at the RTL first to identify the critical flip-flops. These flip-flops are mapped from RTL to the gate level eventually.



RTL vs. GL:

- We investigate the difference between the fault simulation results at the RTL and GL to prove the process is feasible. The results of an example is shown below.

Fault type	Difference	Number of runs		Time (hours)	
		RTL	GL	RTL	GL
Stuck-at-0	7 (≈ 3.3%)	2227	2126	28	34
Stuck-at-1	13 (≈ 6.2%)	2227	2126	26	31
SEU	10 (≈ 4.7%)	18860	19080	1685	2493

Fault Injection:

- Fault simulation is conducted in the order of the testcase's contribution to the functional coverage from high to low.
- For each testcase, we first inject permanent faults (i.e., stuck-at-1 or stuck-at-0) into all flip-flops. If the fault is not detected on the outputs, we can interpret that the fault is masked or the flip-flop is not covered by the testcase. The flip-flop is therefore considered non-critical.
- Otherwise, SEU faults are injected several times in a specified time window. The flip-flop will be identified as critical if the number of detected SEU faults is larger than the threshold we define.
- If the flip-flop is annotated as critical, it will be removed from the fault list of the following testcase.

Test Name	Simulation Time (ns)	Functional Coverage	Number of Flip-flops Injected Permanent Faults	Number of Flip-flops Injected SEU Faults	Number of Detected Critical Flip-flops	Number of Runs	Time (hours)
rand_jump	59,400,814	4598	2227	943	590	23314	5196
pmp_basic	3,611,144	177	1637	954	62	8044	81
debug_branch_jump	21,970,704	10	1575	349	53	6640	338
mem_error	9,818,104	8	1522	442	327	5942	206
debug_ebreakmu	13,274,244	3	1195	179	101	2627	80
invalid_csr	2,261,544	2	1094	14	0	2258	23

Discussion

The procedure could reduce the time involved in the fault simulation. However, there are some limitations to this approach.

- We need to specify the threshold to identify critical flip-flops correctly at the beginning. Otherwise, the simulation of the following testcase might be based on an incorrect fault list.
- The testcases should also be selected based on the relevant applications.
- The simulation time of testcases could be reduced if we could use more efficient constraints. The efficiency could be improved further with these considerations.

The project underlying this report is funded by the German Federal Ministry of Education and Research under grant number 16ME0134 and 16KIS1339K.

[1] ETH Zurich and University of Bologna, 2018, Available: https://ibex-core.readthedocs.io/en/latest/03_reference/verification.html