Formal Analysis of Fault Injection Effects on RISC-V Microarchitecture Models

Simon Tollec, Mihail Asavoae, Mathieu Jan
Univ. Paris-Saclay, CEA, List
F-91120, Palaiseau, France

Damien Couroussé
Univ. Grenoble Alpes, CEA, List
F-38000 Grenoble, France

Karine Heydemann
Sorbonne Univ., CNRS, LIP6
F-75005, Paris, France

Motivation — Fault injection (FI) [1] is a common threat for embedded systems. Security analysis under FI attacks can be performed either using tests on real platforms [2], [3] or using simulation [4], [5] and formal methods [6], [7] considering an ISA-level fault model such as instruction skipping or register corruption. However, it has been shown that many fault effects are not directly addressable at the ISA level [8] and that a precise knowledge of the microarchitecture is essential for a better evaluation. We propose a solution to model both software (SW) and hardware (HW) parts of a system and then to formally evaluate its robustness to FI attacks. This approach is complementary to the analysis methods used at the software level. It identifies non-visible faults at the ISA level that affect the security of the software and that leverage the specificities of a microarchitecture.

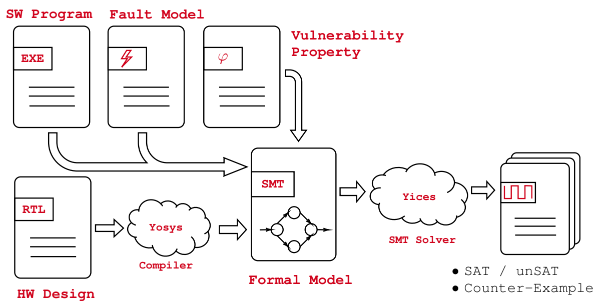

Figure 1: illustration of the proposed workflow.

Workflow to Study the Effects of FI — Fig. 1 illustrates the workflow we propose to analyse the effects of FIs. It requires as input: 1) the hardware description of the evaluated microarchitecture, i.e., both its combinatorial and sequential logics (Verilog 2005 and SystemVerilog 2017 are supported), 2) the executable program, i.e., the instructions to be executed and their associated data, 3) the fault model which describes the localization (both in time and space) and the effects of the attack (e.g., bit flip, bit set) on the hardware, and 4) the security property.

The formal model is a transition system produced by the RTL synthesis tool Yosys and described with the SMT-LIB language. It is constrained by assumptions which describe the execution of the program and the effect of the FIs while the assertions express security properties. Vulnerabilities are found with the Yices SMT Solver using bounded model checking techniques. Returned counter-examples highlight the propagation of the faults in the microarchitecture.

The developed workflow can formally check the execution of a hundred instructions in the presence of faults and is not suited for larger programs. It is also possible to extract a sequence of instructions from a program and check its robustness with degrees of freedom in the initial state of the system.

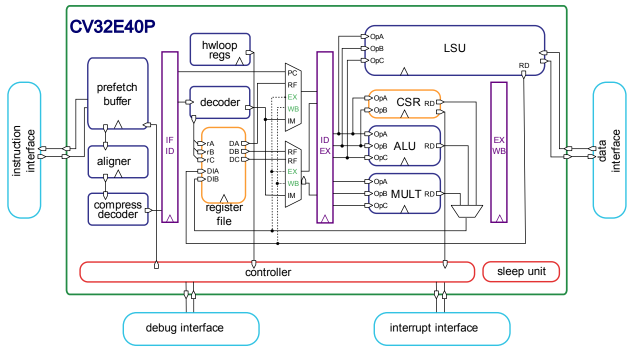

Figure 2: CV32E40P RTL block diagram.

Module	Wire	Timing	Effect
id_controller	operand_a_fw_mux_sel_o	@57	bit-flip
prefetch_buffer	status_cnt_n	@21-26	bit-set
…	…	…	…

TABLE 1: fault injections found by our analysis over VerifyPin.

Results — We carry out a fault vulnerability analysis on the microarchitecture of the RISC-V CV32E40P, a 32-bit processor with a 4-stage, in-order pipeline (Fig. 2).

We study the VerifyPin [9] program which compares two 4-digit codes stored in memory. We investigate single-FI attacks anywhere on the design, at any time during the execution of VerifyPin. About 500 CPU hours are needed to find all faults leading to vulnerabilities on VerifyPin.

Table 1 illustrates some faults that bypass the secure authentication mechanism (about 50 were identified). Our solution confirms some results already known in the literature. For example, it is possible to perform a single bit flip on the forwarding mechanism (i.e., `operand_a_fw_mux_sel_o` at cycle 57) which controls the multiplexer before the execute (EX) stage in Fig. 2. It permits to inject a bad value into the pipeline [8]. In addition, we also identify faults exploiting specific parts of microarchitectures that are not yet known in the literature and cannot be easily represented at the ISA level. The use of the formal model makes it possible to find new and very interesting effects. It contributes to a better understanding of how faults propagate and lead to the vulnerabilities. For instance, a single bit-set FI on the prefetch buffer (PFB) (i.e., `status_cnt_n`) between clock cycles 21 and 26 leads to several effects that are difficult to model:

1) Instructions speculatively fetched in the PFB are executed, whereas they are discarded in the non-faulty behavior,
2) Next instructions are potentially pushed in the pipeline in an incorrect order.
3) At the next branch instruction, the program jumps to an incorrect address.

Further faults identified by the formal analysis on VerifyPin are under investigation.

References

[1] B. Yuce, P. Schaumont, and M. Witteman, "Fault attacks on secure embedded software: Threats, design, and evaluation," *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 111–130, 2018.

[2] L. Riviere, Z. Najm, P. Rauzy, J.-L. Danger, J. Bringer, and L. Sauvage, "High precision fault injections on the instruction cache of armv7-m architectures," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 62–67.

[3] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz, "Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, pp. 77–88.

[4] M. Hoffmann, F. Schellenberg, and C. Paar, "Armory: Fully automated and exhaustive fault simulation on arm-m binaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1058–1073, 2020.

[5] J. Grycel and P. Schaumont, "Simplifi: Hardware simulation of embedded software fault attacks," *Cryptography*, vol. 5, no. 2, p. 15, 2021.

[6] K. Pattabiraman, N. Nakka, Z. Kalbarczyk, and R. Iyer, "SymPLFIED: Symbolic program-level fault injection and error detection framework," in *Intl. Conf on Dependable Systems and Networks (DSN)*, 2008.

[7] J.-B. Bréjon, K. Heydemann, E. Encrenaz, Q. Meunier, and S.-T. Vu, "Fault attack vulnerability assessment of binary code," in *Workshop on Cryptography and Security in Computing Systems*, pp. 13–18.

[8] J. Laurent, V. Beroulle, C. Deleuze, and F. Pebay-Peyroula, "Fault injection on hidden registers in a RISC-V rocket processor and software countermeasures," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 252–255.

[9] L. Dureuil, G. Petiot, M.-L. Potet, T.-H. Le, A. Crohen, and P. de Choudens, "FISSC: A fault injection and simulation secure collection," in *Computer Safety, Reliability, and Security*. Springer International Publishing, pp. 3–11.