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

Simon Tollec[1], Mihail Asavoae[1], Damien Couroussé[2], Karine Heydemann[3], Mathieu Jan[1]

[1] *Université Paris-Saclay, CEA, List* F-91120, Palaiseau, France, firstname.lastname@cea.fr
[2] *Université Grenoble Alpes, CEA, List* F-38000 Grenoble, France, firstname.lastname@cea.fr
[3] *Sorbonne Université, CNRS, LIP6* F-75005, Paris, France, firstname.lastname@lip6.fr

Approach

Motivation

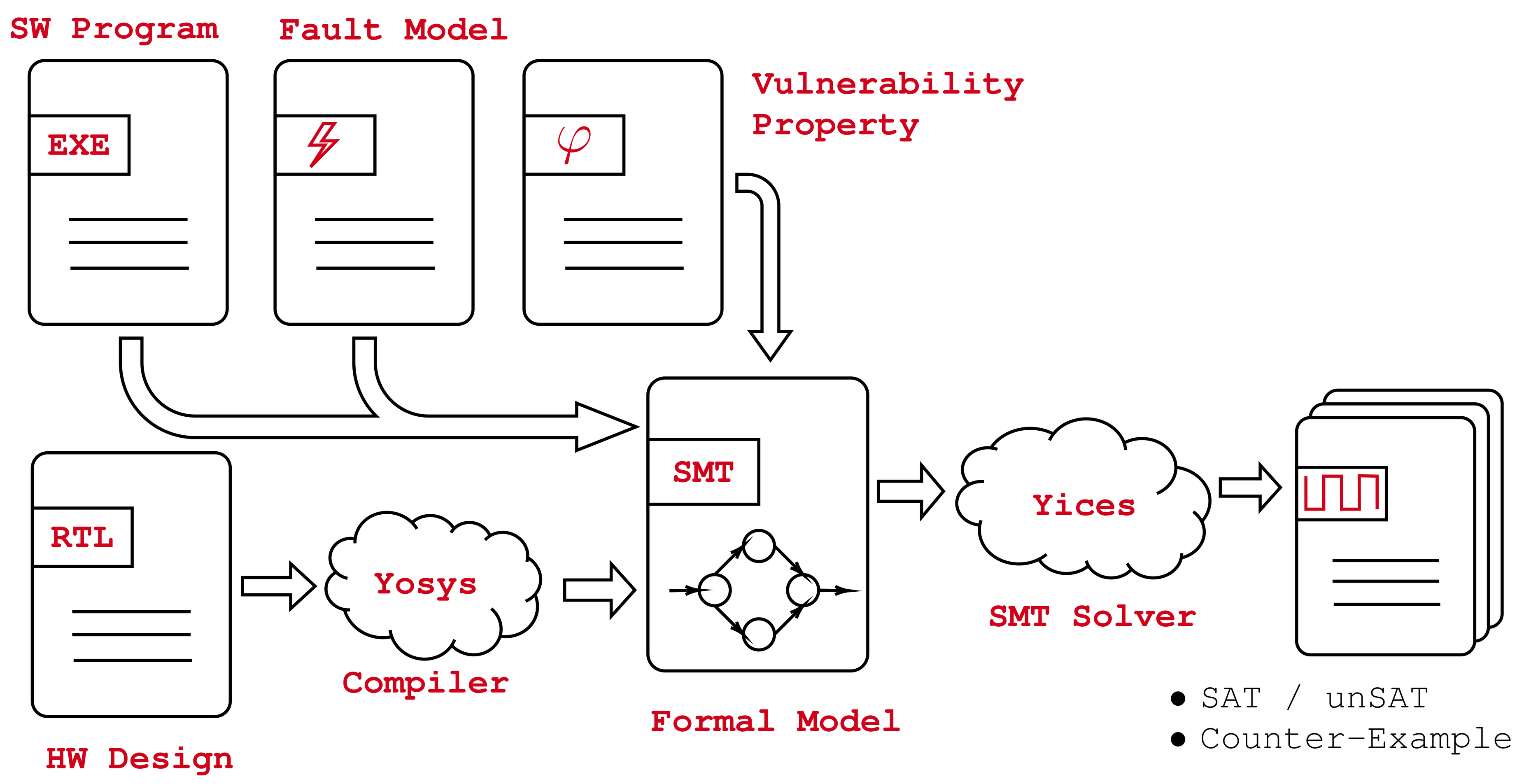
- Fault injection (FI) [1] is a threat to embedded systems
- Security analysis under FI attacks is performed either using:
  - Tests on real platforms [2,3]
  - Simulation [4,5]
  - Formal methods [6]
- Common fault models considered at the ISA level:
  - Instruction replacement
  - Register corruption
- Many fault effects depend on the hardware implementation and are not directly addressable at the ISA level [7], e.g.,
  - Pipeline
  - Speculative execution
- A precise knowledge of the microarchitecture is essential for:
  - A better understanding of the effect of faults
  - A better security evaluation

Contributions

- Develop a workflow that encompasses the SH/HW to identify harmful faults
- Illustrate the approach on a use case
- Exhibit new fault effects that are difficult to model at the ISA level

Workflow

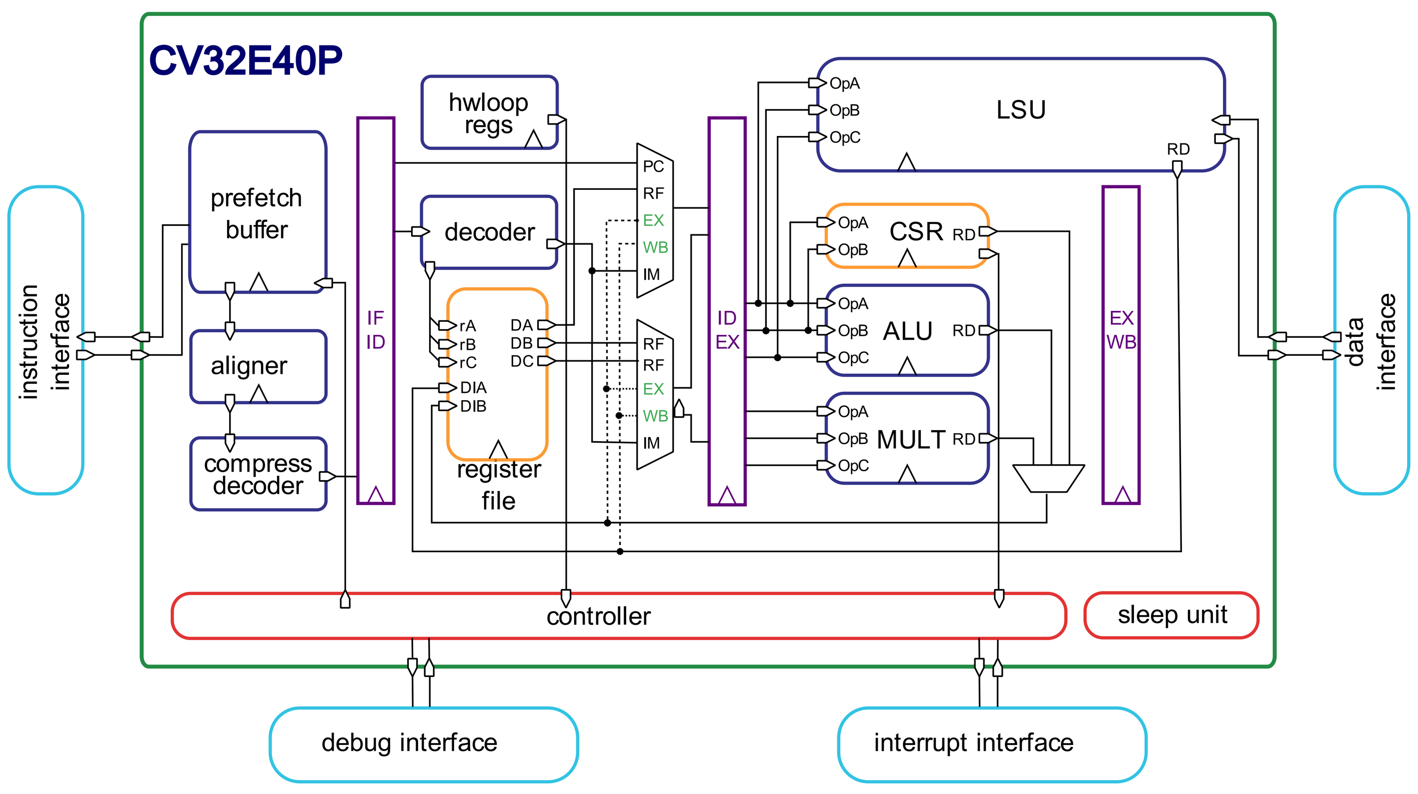
- Yosys produces a formal model (SMT-Lib)
- Software program, FI effects and FI timings are modeled by adding constraints
- Security properties are expressed with assertions
- Yices SMT Solver uses bounded model checking to find vulnerabilities
- Returned counter-examples highlight the propagation of the faults in the microarchitecture



Use Case

Architecture: CV32E40P

- A 32-bit RISC-V processor with a 4-stage, in-order pipeline



Software: VerifyPIN

- VerifyPin program [8] compares two 4-digit codes stored in memory
- Embedded countermeasures against FI:
  - Hardened booleans
  - Loop count verification
  - Inline call
- Attacker's goal: bypassing the authentication mechanism
- Security property: an incorrect password cannot allow authentication

Fault Vulnerability Results

- Investigate single-FI attacks on the logic gates and wires of the design during the VerifyPin execution
- About 6 CPU hours of calculation on an 80-core cluster were needed
- 50 faults injections were identified (2 of them are detailed below on the left)

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

- We identify vulnerabilities already known in the literature [7], e.g., faulting the forwarding mechanism by targeting the `operand_a_fw_mux_sel_o` signal
- We highlight that a fault injection on the prefetch buffer (PFB) (i.e., `status_cnt_n`) leads to effects that are difficult to model at the ISA level:
  1. The fault can force the execution of instructions speculatively fetched in the PFB,
  2. Next instructions are potentially pushed in the pipeline in an incorrect order,
  3. The program jumps to an incorrect address at the next branch instruction.

Module	Wires	Cycle	
		-0g flag	-0s flag
aligner	instr_valid_o	18	
	branch_i	47	
	update_state	47	
controller	deassert_we_o	19	65
	halt_id_o	19	65
	is_decoding_o	19	65
	jump_in_dec	57	68
	operand_a_fw_mux_sel_o	57	65
	pc_set_o		66
decoder	alu_en		65
	alu_op_a_mux_sel_o	57	65
	alu_op_b_mux_sel_o	57	65
	ctrl_transfer_insn	57	65, 68
	ctrl_transfer_insn_in_id_o		65
	regfile_alu_waddr_sel_o	19	
	regfile_alu_we_o	19	
ex_stage	alu_cmp_result	58	66
	mult_multicycle_o	19	65
	regfile_alu_we_fw_o	20	
fifo	empty_o		64
	status_cnt_n	21 - 26	63
id_stage	alu_op_a_mux_sel	57	65
	alu_op_b_mux_sel	57	65
	alu_vec_mode	57	65
	alu_vec_mode_ex_o	58	66
	apu_en_ex_o	20	
	bmask_b_mux	46	61
	branch_in_ex_o		66
	branch_taken_ex	58	66
	halt_id	19	65
	id_valid_o	19	65
	operand_a_fw_mux_sel	57	65
	pc_set_o		66
	reg_d_alu_is_reg_a_id	57	65
	regfile_alu_waddr_mux_sel	19	
	regfile_alu_we_ex_o	20	
	regfile_alu_we_id	19	
if_stage	branch_req		66
	instr_valid	18	64
	instr_valid_id_o	19	65
	is_fetch_failed_o	19	65
mult	mulh_CS		28, 64
	mulh_NS		27, 63
	multicycle_o	19	65
prefetch_ctrl	flush_cnt_q		26, 28-29, 39-40 50-51, 61-64
	next_flush_cnt		25, 27-28, 38-39 49-50, 60-63

Perspectives

- Classify fault injection effects: visible/non-visible at the ISA level
- Improve the scalability of the approach
- Study more complex processors, e.g., CVA6
- Determine the minimal design to model new fault effects at the ISA level

Bibliography

[1] Yuce, B., Schaumont, P. & Witteman, M. Fault Attacks on Secure Embedded Software: Threats, Design and Evaluation. *Journal of Hardware and Systems Security* 111–130 (2018).
[2] Rivière, L. et al. High Precision Fault Injections on the Instruction Cache of ARMv7-M Architectures. *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* 62–67 (2015).
[3] Moro, N., et al. Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller. *Workshop on Fault Diagnosis and Tolerance in Cryptography* 77–88 (2013).
[4] Hoffmann, M., Schellenberg, F. & Paar, C. ARMORY: Fully Automated and Exhaustive Fault Simulation on ARM-M Binaries. *IEEE Transactions on Information Forensics and Security* (2021).
[5] Grycel, J. & Schaumont, P. SimpliFI: Hardware Simulation of Embedded Software Fault Attacks. *Cryptography* (2021).
[6] Pattabiraman, K. et al. SymPLFIED: Symbolic program-level fault injection and error detection framework. *IEEE International Conference on Dependable Systems and Networks* (2008).
[7] Laurent, J. et al. Fault Injection on Hidden Registers in a RISC-V Rocket Processor and Software Countermeasures. *Design, Automation & Test in Europe Conference & Exhibition* (2019).
[8] Dureuil, L. et al. FISSC: A Fault Injection and Simulation Secure Collection. *Computer Safety, Reliability, and Security* vol. 9922 3–11 (Springer International Publishing, 2016).