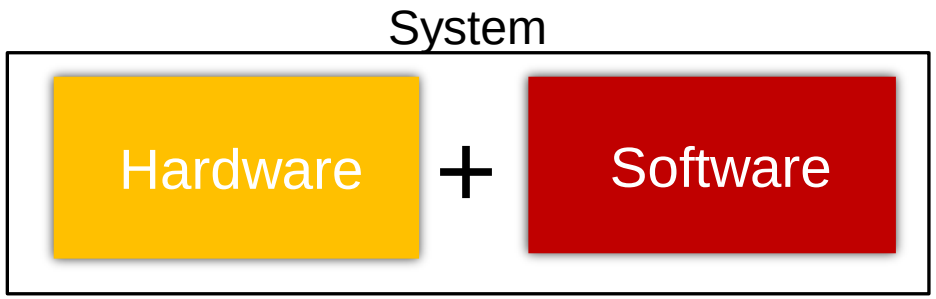
FORMAL PROCESSOR MODELING FOR ANALYZING SAFETY AND SECURITY PROPERTIES ON RISC-V CASE STUDIES

Engineer-researchers: Mathieu Jan, Mihail Asavoae, Belgacem Ben Hedia, Oumaima Matoussi, Farhat Thabet, Hai-Dang Vu
PhD students: Benjamin Binder, Samira Ait Bensaid, Simon Tollec
External collaborators: Damien Couroussé (CEA), Karine Heydemann (LIP6)

MOTIVATIONS AND APPROACH

- **Context**
  - Embedded systems: connected/physically accessible, increased hardware/software complexity
  - From safety-critical to IoT devices

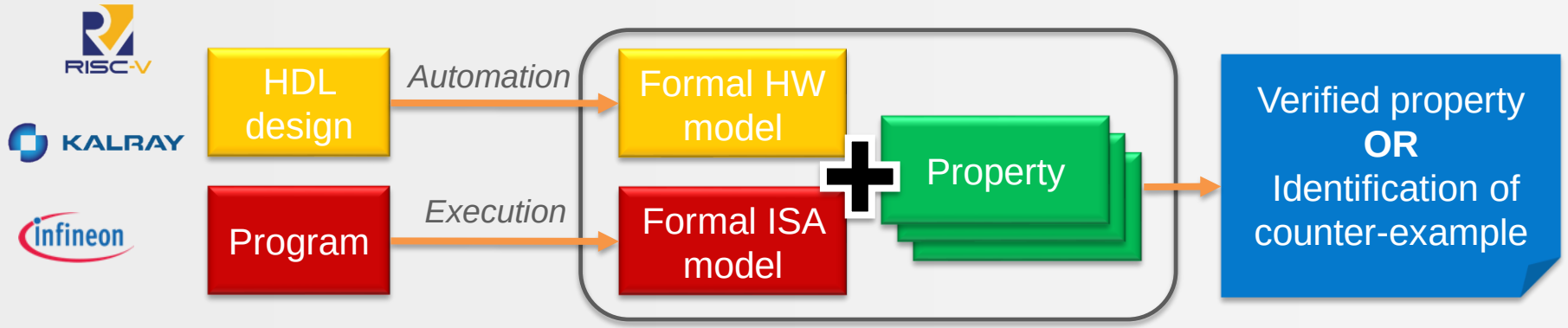
- **Goal: increase the trust in embedded systems using applied formal methods**
  - Software (SW) and hardware (HW) formal verifications are (most often) separated activities
  - Understand system behavior to better design them



- **Approach: combine software and hardware formal verifications**

LEAF: FORMAL ANALYSIS OF HW/SW CPS

- **For what kind of (extra-functional) properties and why?**
  - Safety property: code-specific detection of timing anomalies within pipelines of processors
    - Optimize Worst-Case Execution Time (WCET) analyzers
  - Security property: identification of fault-injection points in a µ-architecture that lead to SW exploits
    - Better understand effect of faults in a µ-architecture and capture that in SW fault model



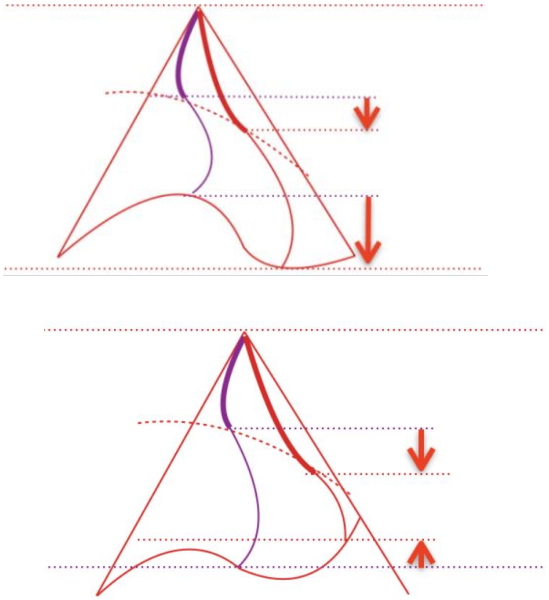
- **Challenges**
  - How to model systems for an efficient verification of extra-functional properties?
  - How to extract extra-functionnal (timing) properties from a system?

WCET AND TIMING ANOMALIES

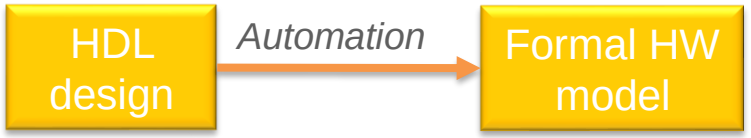
[WCET2018, ERTS2020, FMICS2020, ASP-DAC2020, RTCSA2021, ERTS2022]

- *Real-time systems* are subject to **strong timing constraints**

- *Timing anomalies* are undesired phenomena
  - Amplificiation timing anomalies
    - Prevent from bounding the individual timing contributions of components
    - **Not** possible to perform **compositional static analyses** (pipelines, caches…)
  - Counter-intuitive timing anomalies
    - Prevent from always following local worst cases to determine the global worst case
    - **Jeopardize static analyses** (common assumptions)



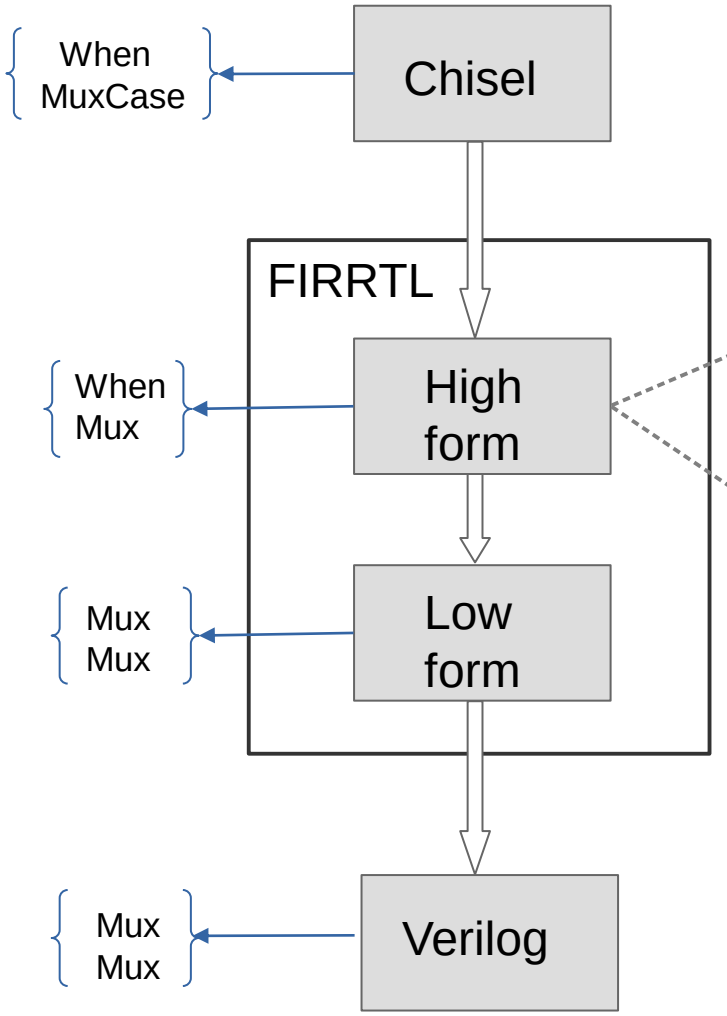
- But how to build the hardware formal models required by WCET analyzers?
  - Few WCET analyzers starts from VHDL/Verilog designs
  - High-level HDL languages (Chisel, SpinalHDL, Clash, etc.)



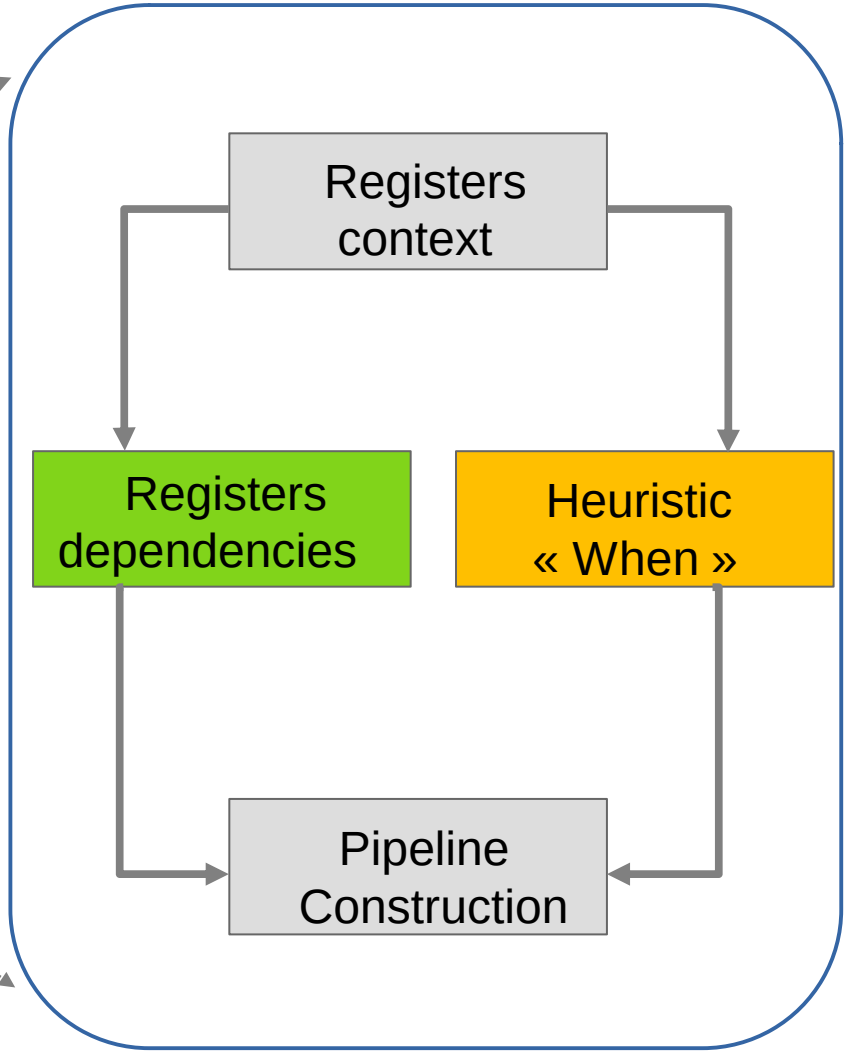
PIPELINE DATAPATH ANALYSIS

[RTAS2022-BP]

Hardware compiler framework



Pipeline analysis

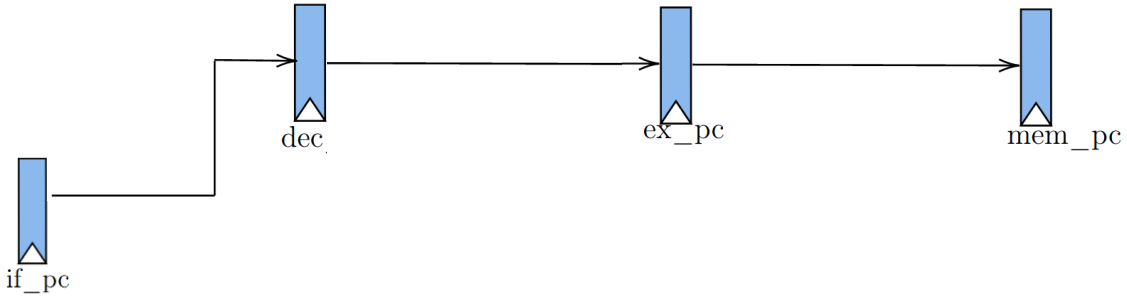


EXAMPLE OVER 5-STAGE RISC-V SODOR PROCESSOR

Reg	Rule	Stage
if_pc	-	1

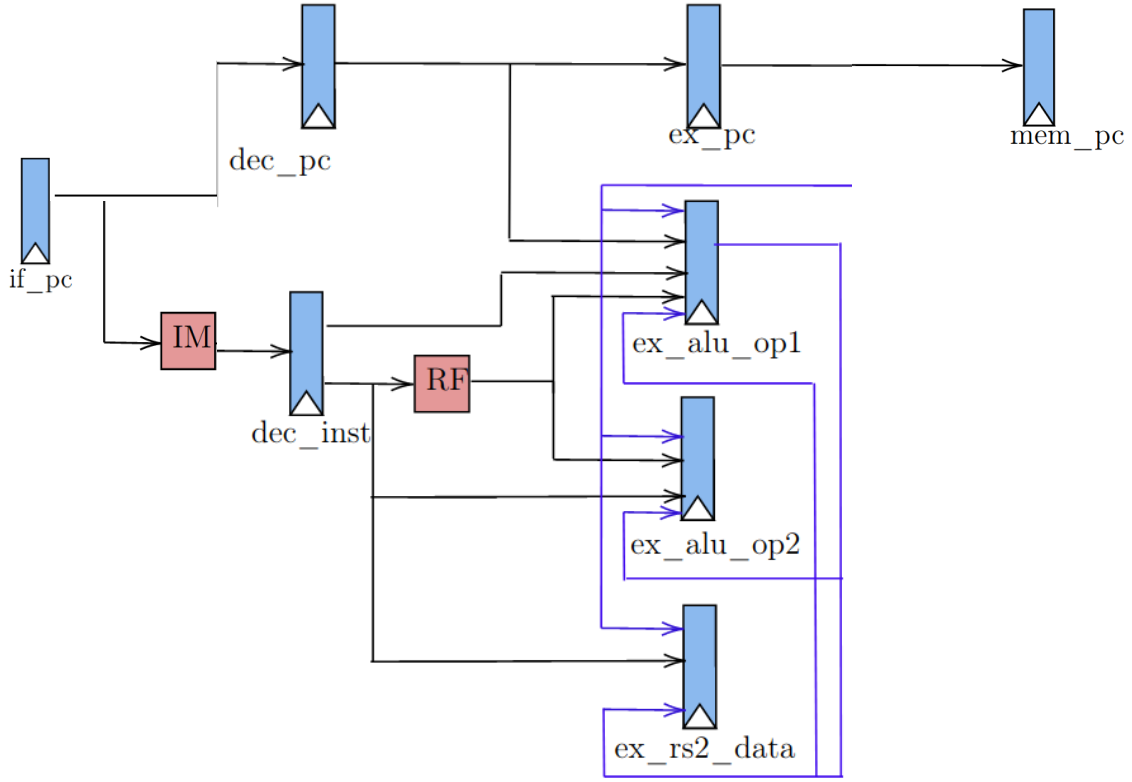
EXAMPLE OVER 5-STAGE RISC-V SODOR PROCESSOR

Reg	Rule	Stage
if_pc	-	1
dec_pc	1	2
ex_pc	1	3
mem_pc	1	4



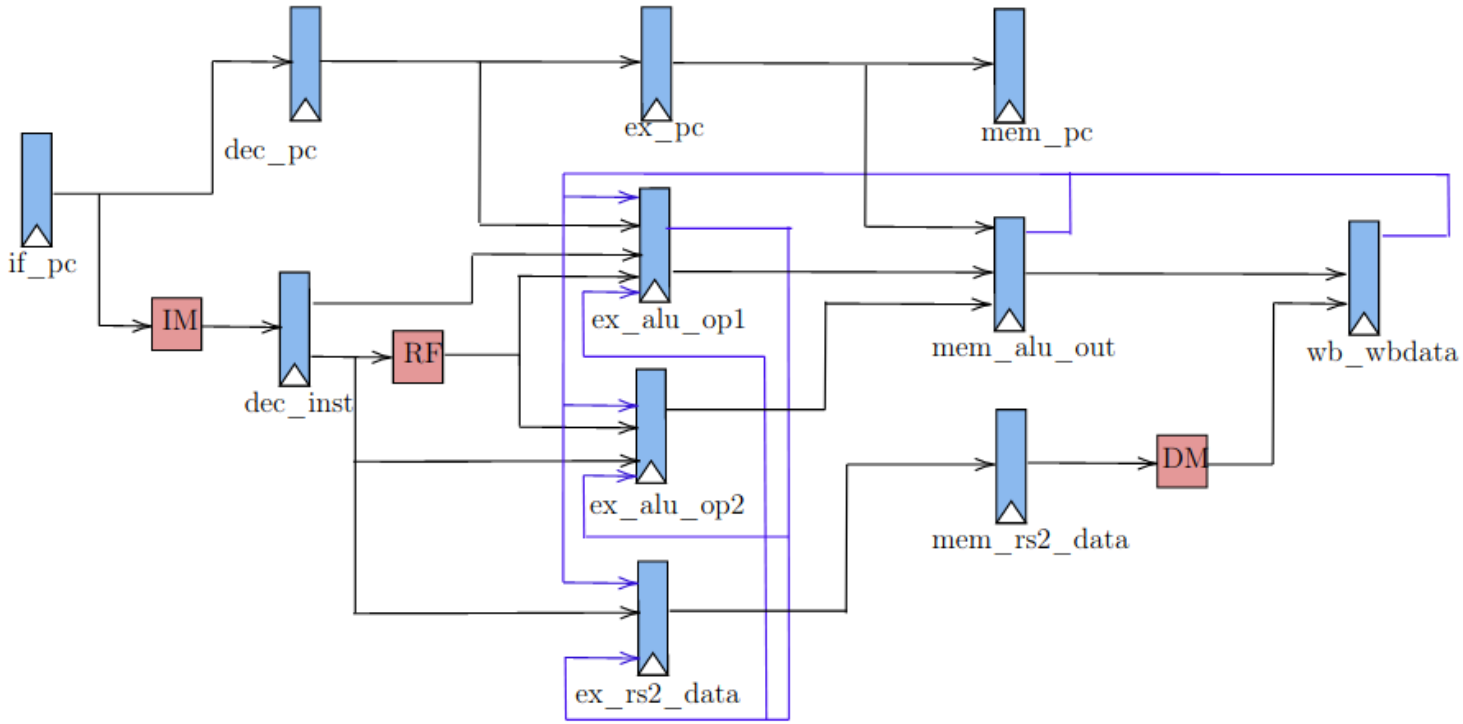
EXAMPLE OVER 5-STAGE RISC-V SODOR PROCESSOR

Reg	Rule	Stage
if_pc	-	1
dec_pc	1	2
ex_pc	1	3
mem_pc	1	4
dec_inst	2	2
ex_alu_op1	2	3
ex_alu_op2	2	3
ex_rs2_data	2	3



EXAMPLE OVER 5-STAGE RISC-V SODOR PROCESSOR

Reg	Rule	Stage
if_pc	-	1
dec_pc	1	2
ex_pc	1	3
mem_pc	1	4
dec_inst	2	2
ex_alu_op1	2	3
ex_alu_op2	2	3
ex_rs2_data	2	3
mem_alu_out	1	4
mem_rs2_data	1	4
wb_wbdata	1	5



EVALUATIONS AND FUTURE WORK

(ON AUTOMATIC CONSTRUCTION OF HW MODELS)

TABLE 3 – Experimental results on RISC-V processor designs.

	#Regs	Case 1	Case 2
RISC-V Mini	15	5	10
Sodor (WFW)	48	36	12
Sodor (FW)	48	34	14
KyogenRV	93	47	36

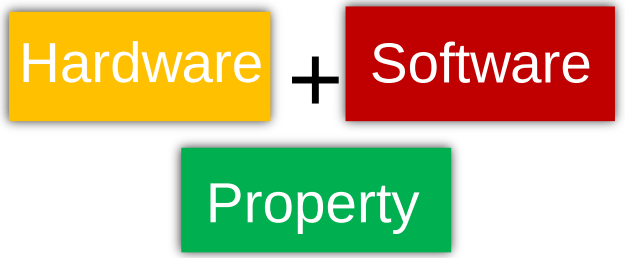
- Extended analysis for multi-modular datapath pipelines (almost done in fact !)
- Experiments on out-of order processors: e.g. Boom processor
- Generate formal abstract models (WCET but not only …)

A SIMILAR APPROACH FOR SECURITY …

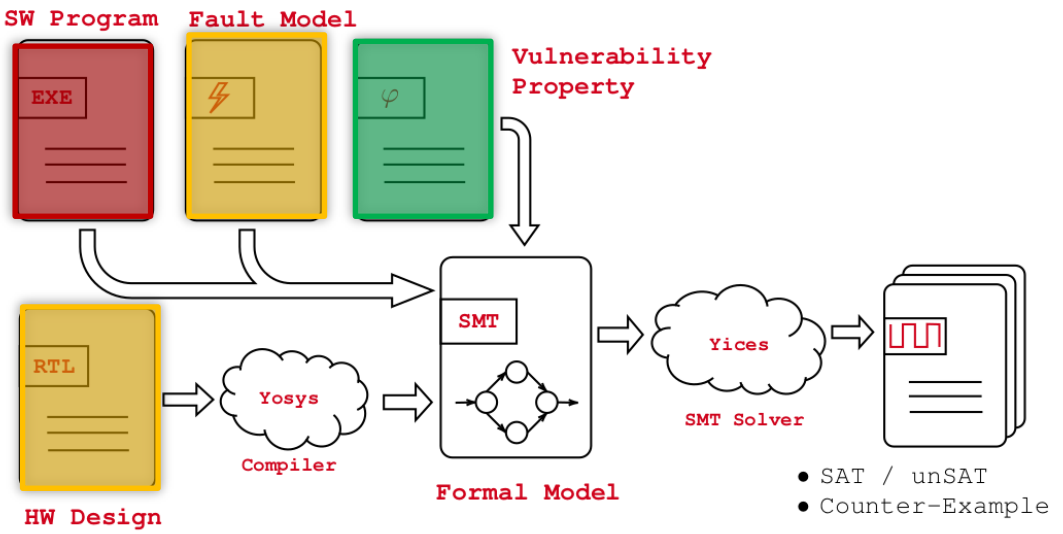
- New fault-injection attack paths shown at the RTL-level

[Johan Laurent, Vincent Beroulle, Christophe Deleuze, Florian Pebay-Peyroula: Fault Injection on Hidden Registers in a RISC-V Rocket Processor and Software Countermeasures. DATE 2019]

  - Identified by manual code review
  - Analysis of effects by relying on simulations

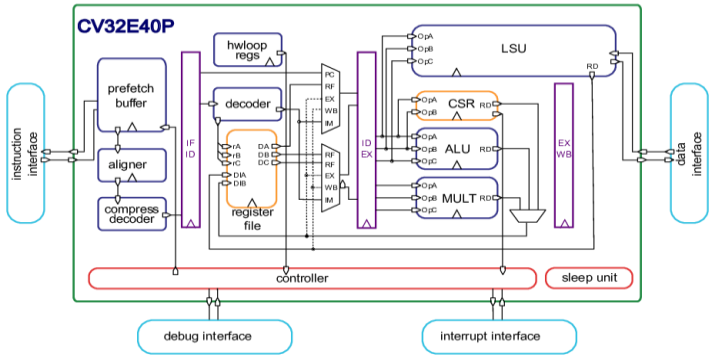


- Extend **LEAF** approach to identify and understand the impact of fault injections on processor microarchitectures
  - SMT-based formal hardware model (both sequential and combinatorial logics): generated by Yosys
  - Software model: sequence of instructions and data
  - Fault model: spatial, temporal and effect dimensions
  - Security property: encode an expected behavior



PRELIMINARY RESULTS

- Use-case: verifyPIN over RISC-V CV32E40P



- Identifies non visible faults at the ISA level based on microarchicture specificities
  - Forwarding mechanism (not new)
  - Prefetch buffer (new)
    - Instructions speculatively fetched in the PFB are executed, whereas they are discarded in the non-faulty behavior
    - Next instructions are potentially pushed in the pipeline in an incorrect order
    - At the next branch instruction, the program jumps to an incorrect address

Module	Wires	Cycle	
		-Og flag	-Os 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
	wfi_active		65
decoder	alu_en		65
	alu_en_o		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	19	
...			
mult	mulh_CS		28, 64
	mulh_NS		27, 63
	multicycle_o	19	65
prefetch_controller	flush_cnt_q		26, 28-29, 39-40, 50-51, 61-64
	next_flush_cnt		25, 27-28, 38-39, 49-50, 60-63

CONCLUSION AND FUTURE WORK

- LEAF approach: combine software and hardware formal for the co-verification of extra-functional properties
  - Come to see our posters on the subjects!
  - « Pipeline Datapath Models from RISC-V based cores » by Samira Ait Bensaid
  - « Formal Analysis of Fault Injection Effects on RISC-V Microarchitecture Models » by Simon Tollec

- Future work
  - Further investigate abstract modeling flavours to improve formal verifications
  - Towards mitigation strategies: SW and/or HW