Towards Single-Event Upset Detection in Hardware Secure RISC-V Processors

Jeffrey Prinzie, Boris Engelen, Karel Appels, Levi Mariën, Naïn Jonckers

Electronic Circuits and Systems

Advanced Integrated Sensing Lab (ADVISE)

Outline

- Motivation
- Research Methodology
- Fault Injection Simulation Results
- Conclusion

Motivation

Processing systems in radiation environments

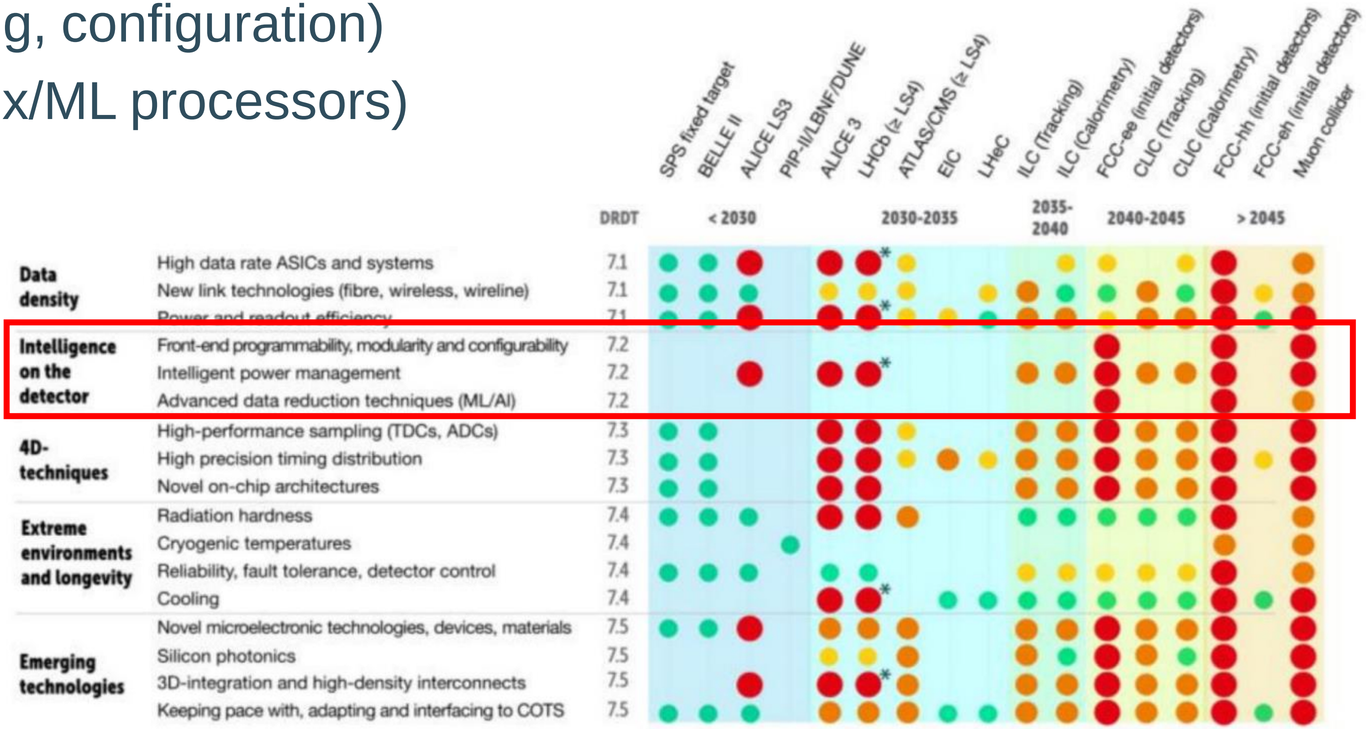
High-Energy Physics

- Housekeeping processors (monitoring, configuration)
- Detector data processing (pixel/vertex/ML processors)

Space applications

- Primary on-board computer
- Secondary computers
  (Data processing, accelerators, …)

SEUs can cause data errors, unpredictable behavior or severe crashes



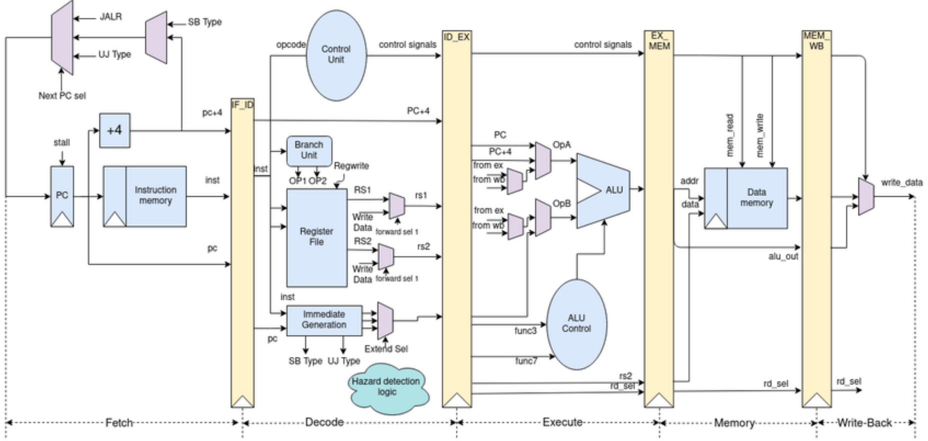
		DRDT	< 2030				2030-2035					2035-2040		2040-2045			> 2045		
			SPS fixed target	BELLE II	ALICE LS3	PIP-II/LBNF/DUNE	ALICE 3	LHCb (≥ LS4)	ATLAS/CMS (≥ LS4)	EIC	LHeC	ILC (Tracking)	ILC (Calorimetry)	FCC-ee (initial detectors)	CLIC (Tracking)	CLIC (Calorimetry)	FCC-hh (initial detectors)	FCC-eh (initial detectors)	Muon collider
Data density	High data rate ASICs and systems	7.1	🟢	🟢	🔴		🔴	🔴*	🟡				🟡	🟡		🟡	🔴		🟠
	New link technologies (fibre, wireless, wireline)	7.1	🟢	🟢	🟢		🟡	🟡	🟡		🟡	🟠	🟢	🟢	🟠	🟢	🔴	🟡	🟠
	Power and readout efficiency	7.1	🟢	🟢	🔴		🔴	🔴*	🟡	🟡	🟢	🟠	🟠	🟡	🟠	🟠	🔴	🟢	🔴
Intelligence on the detector	Front-end programmability, modularity and configurability	7.2												🔴			🔴		🔴
	Intelligent power management	7.2			🔴		🔴	🔴*				🟠	🟠	🔴	🟠	🟠	🔴		🔴
	Advanced data reduction techniques (ML/AI)	7.2												🔴			🔴		🟠
4D-techniques	High-performance sampling (TDCs, ADCs)	7.3	🟢	🟢			🔴	🔴	🟡			🟠	🟠	🔴	🟠	🟠	🔴		🔴
	High precision timing distribution	7.3	🟢	🟢			🔴	🔴	🟡	🟠	🟡	🟠	🟠	🔴	🟠	🟠	🔴	🟡	🔴
	Novel on-chip architectures	7.3	🟢	🟢			🔴	🔴				🟠	🟠	🔴	🟠	🟠	🔴		🔴
Extreme environments and longevity	Radiation hardness	7.4	🟢	🟢	🟢		🔴	🔴	🟠			🟢	🟢	🟢	🟢	🟢	🔴		🟠
	Cryogenic temperatures	7.4				🟢											🟠		🟠
	Reliability, fault tolerance, detector control	7.4	🟢	🟢	🟢		🟢	🟢				🟡	🟡	🟡	🟡	🟡	🔴		🟠
	Cooling	7.4					🔴	🔴*		🟢	🟢	🟢	🟢	🟢	🟢	🟢	🔴	🟢	🔴
Emerging technologies	Novel microelectronic technologies, devices, materials	7.5	🟢	🟢	🔴		🟠	🟠	🟠			🟠	🟠	🔴	🟠	🟠	🔴		🔴
	Silicon photonics	7.5					🟡	🟡	🟠			🟠	🟢	🔴	🟠	🟢	🔴		🔴
	3D-integration and high-density interconnects	7.5			🔴		🔴	🔴*	🟠			🟠	🟠	🔴	🟠	🟠	🔴		🔴
	Keeping pace with, adapting and interfacing to COTS	7.5	🟢	🟢	🟢		🟠	🟠	🟠	🟢	🟢	🟠	🟠	🔴	🟠	🟠	🔴	🟢	🔴

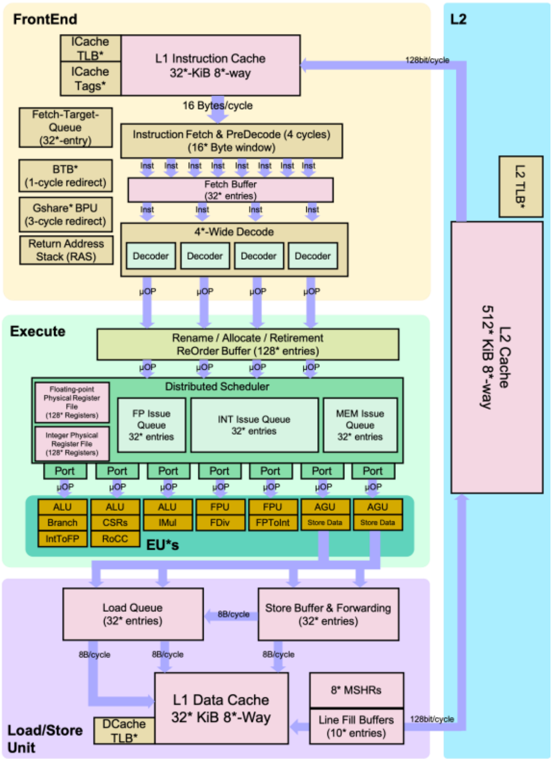
Motivation

RISC-V

- RISC-V Instruction Set Architecture (ISA)
  - Like ARM, x86, MIPS, SPARC, ...
  - Available toolchain (compiler, …)
  - Free to use - Open license
- Many open source cores/SoCs available
  - Availability of source code for fault simulation
  - (Minor) Modifications possible
  - Not limited by vendor and export issues* (i.e. ARM)

*More important for space applications

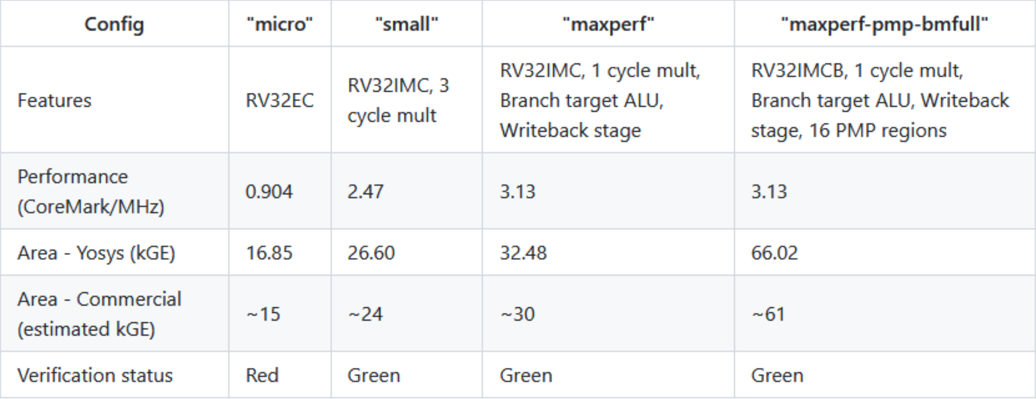



FrontEnd
ICache TLB*
ICache Tags*
L1 Instruction Cache 32*-KiB 8*-way
16 Bytes/cycle
128bit/cycle
Fetch-Target-Queue (32*-entry)
Instruction Fetch & PreDecode (4 cycles) (16* Byte window)
BTB* (1-cycle redirect)
Gshare* BPU (3-cycle redirect)
Return Address Stack (RAS)
Inst
Fetch Buffer (32* entries)
4*-Wide Decode
Decoder
µOP
L2
L2 TLB*
L2 Cache 512* KiB 8*-way
Execute
Rename / Allocate / Retirement ReOrder Buffer (128* entries)
Distributed Scheduler
Floating-point Physical Register File (128* Registers)
Integer Physical Register File (128* Registers)
FP Issue Queue 32* entries
INT Issue Queue 32* entries
MEM Issue Queue 32* entries
Port
ALU
Branch
IntToFP
CSRs
RoCC
IMul
FPU
FDiv
FPToInt
AGU
Store Data
EU*s
Load Queue (32* entries)
Store Buffer & Forwarding (32* entries)
8B/cycle
L1 Data Cache 32* KiB 8*-Way
8* MSHRs
Line Fill Buffers (10* entries)
DCache TLB*
Load/Store Unit

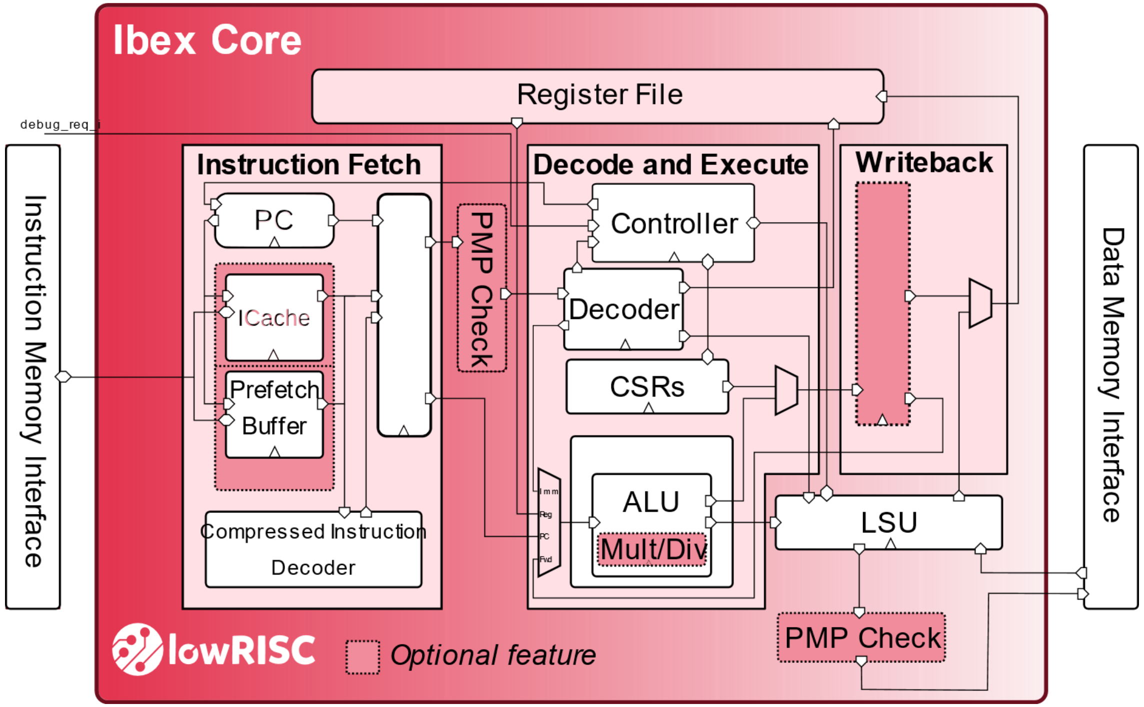
Motivation

Ibex Core overview

- Open source 32-bit RISC-V CPU
- Written in SystemVerilog
- two-stage pipeline (third pipeline stage available)
- Different configurations available



Config	"micro"	"small"	"maxperf"	"maxperf-pmp-bmfull"
Features	RV32EC	RV32IMC, 3 cycle mult	RV32IMC, 1 cycle mult, Branch target ALU, Writeback stage	RV32IMCB, 1 cycle mult, Branch target ALU, Writeback stage, 16 PMP regions
Performance (CoreMark/MHz)	0.904	2.47	3.13	3.13
Area - Yosys (kGE)	16.85	26.60	32.48	66.02
Area - Commercial (estimated kGE)	~15	~24	~30	~61
Verification status	Red	Green	Green	Green



https://github.com/lowRISC/ibex

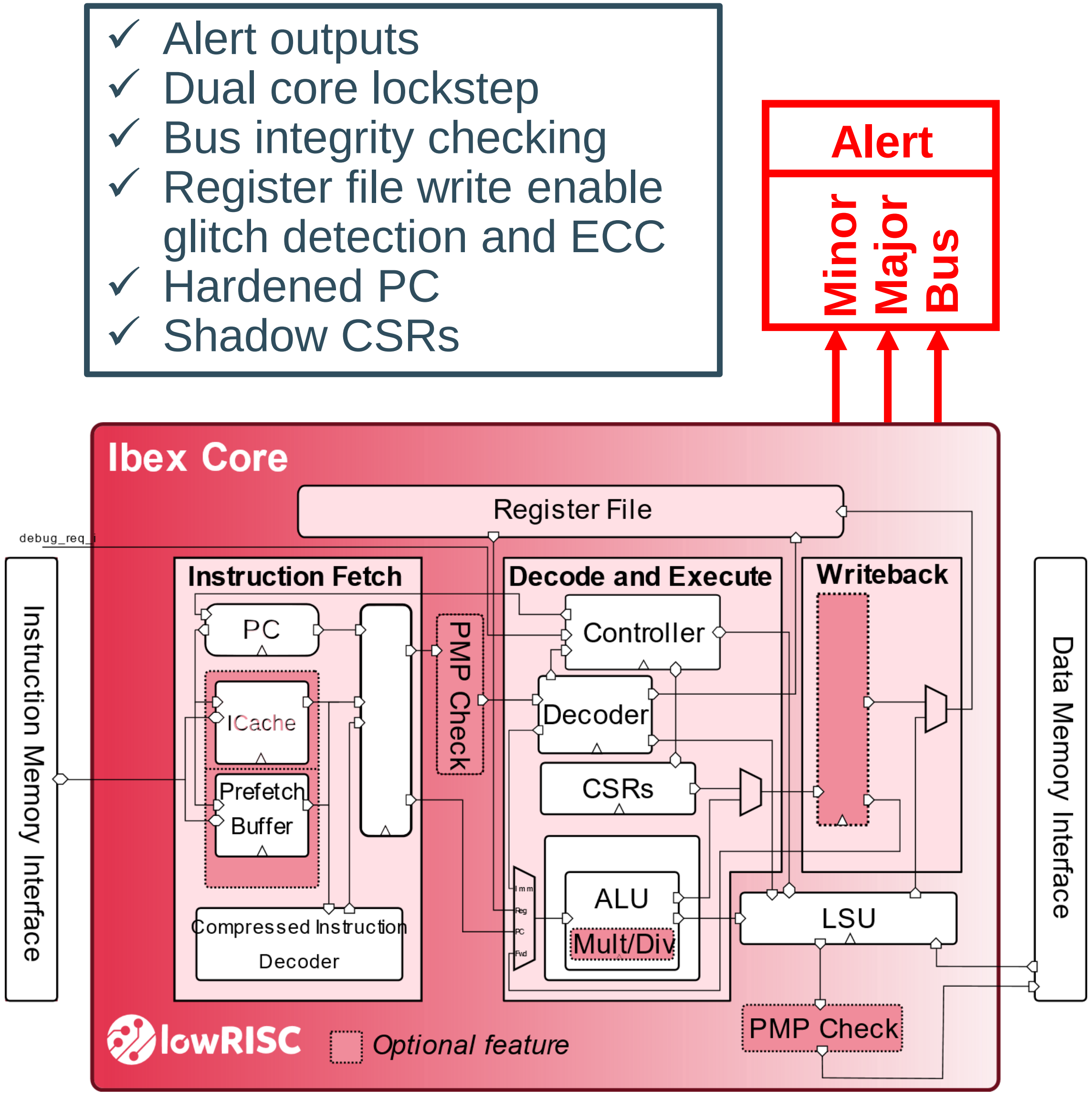
Motivation

Security Features

- Ibex can implement a set of extra features to support **security-critical** applications
- Main strategy: Ibex core can detect external attacks due to corrupted states
- Alerts provided by dedicated signals

Research Question:

Can these built-in security features be used to detect SEUs within the Ibex core?



- ✓ Alert outputs
- ✓ Dual core lockstep
- ✓ Bus integrity checking
- ✓ Register file write enable glitch detection and ECC
- ✓ Hardened PC
- ✓ Shadow CSRs

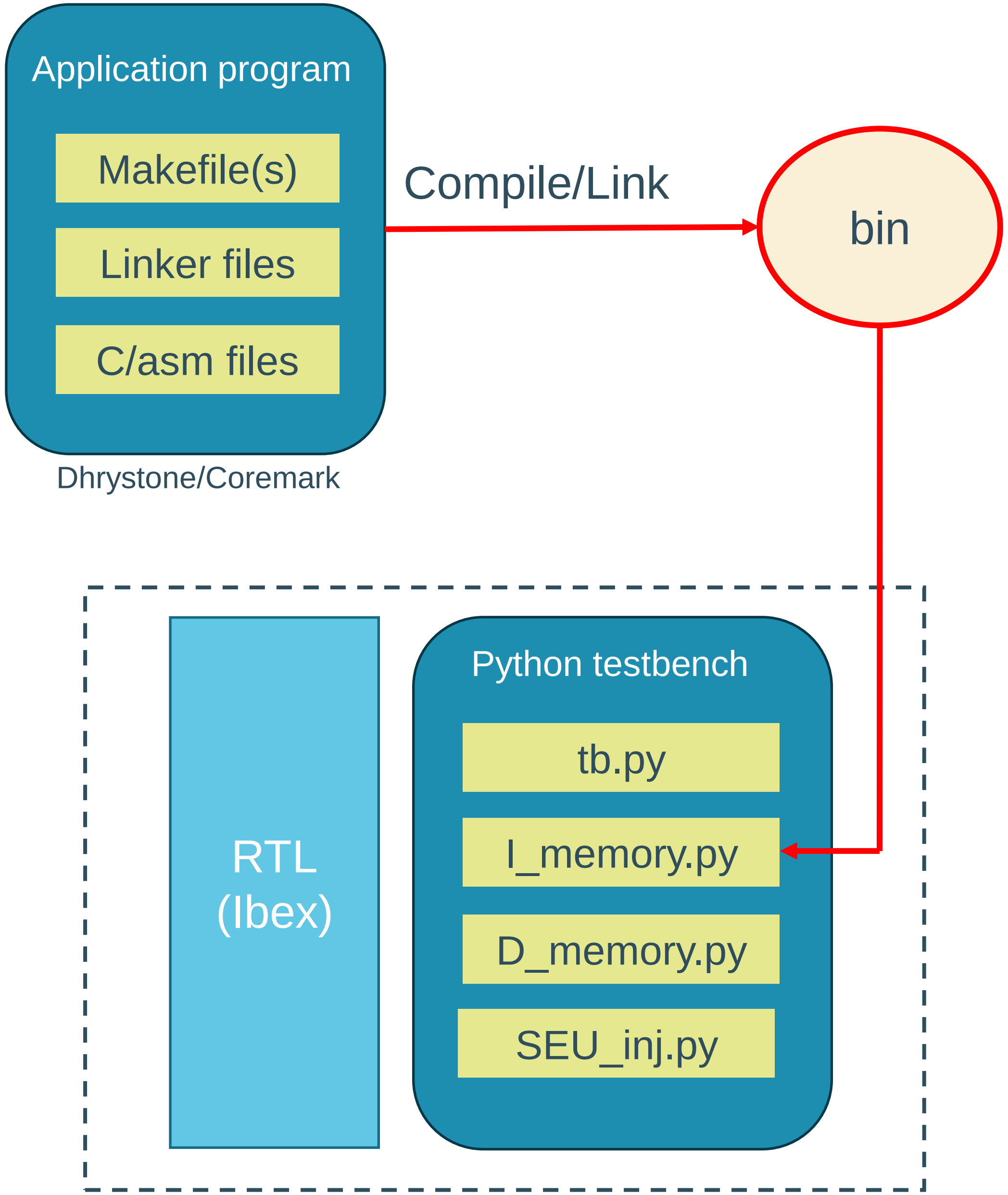
https://ibex-core.readthedocs.io/en/latest/03_reference/security.html

Research Methodology

Testbench architecture

- CoCoTB testbench
  - Ibex RTL code
  - Python models for SoC
    - Data/Instruction memory
    - Stdio
    - …
  - Random SEU injection

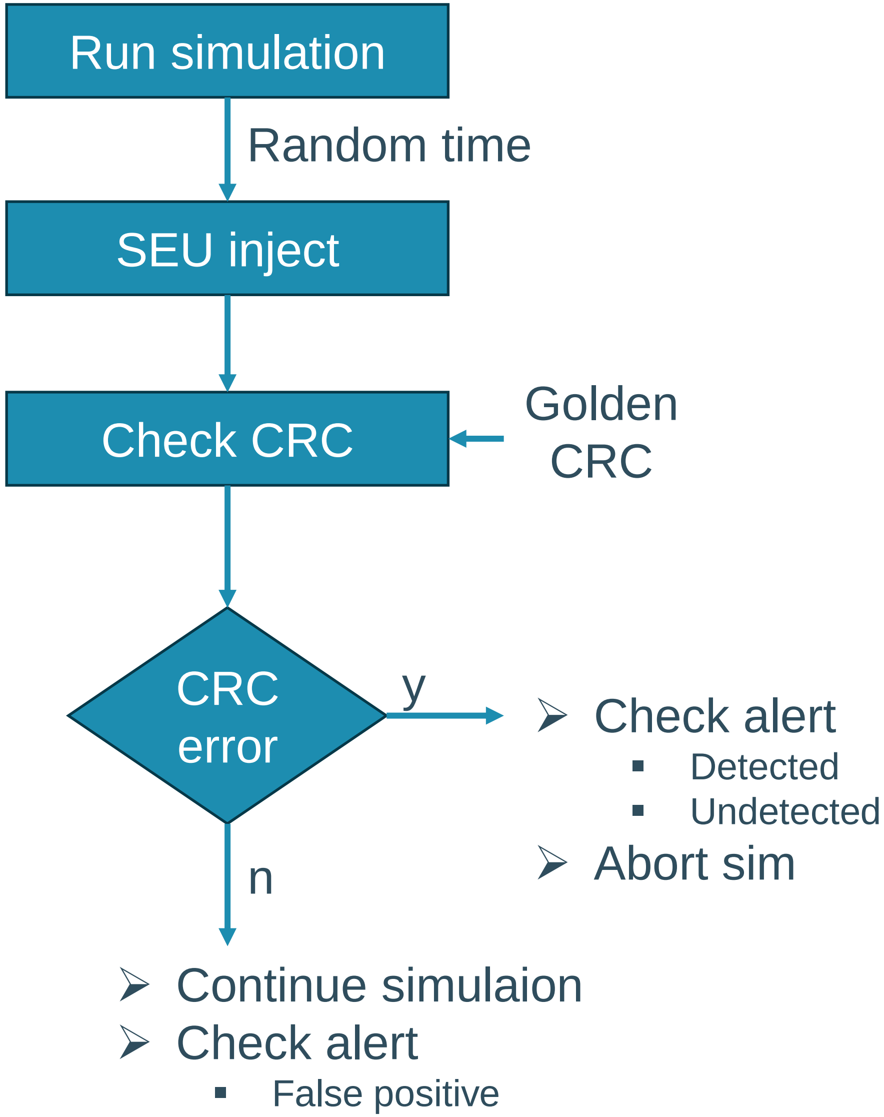
    (Pre-pass with Genus to extract flip-flop list)
- Application code compiled and loaded in I-memory
- Xcelium RTL simulator



Research Methodology

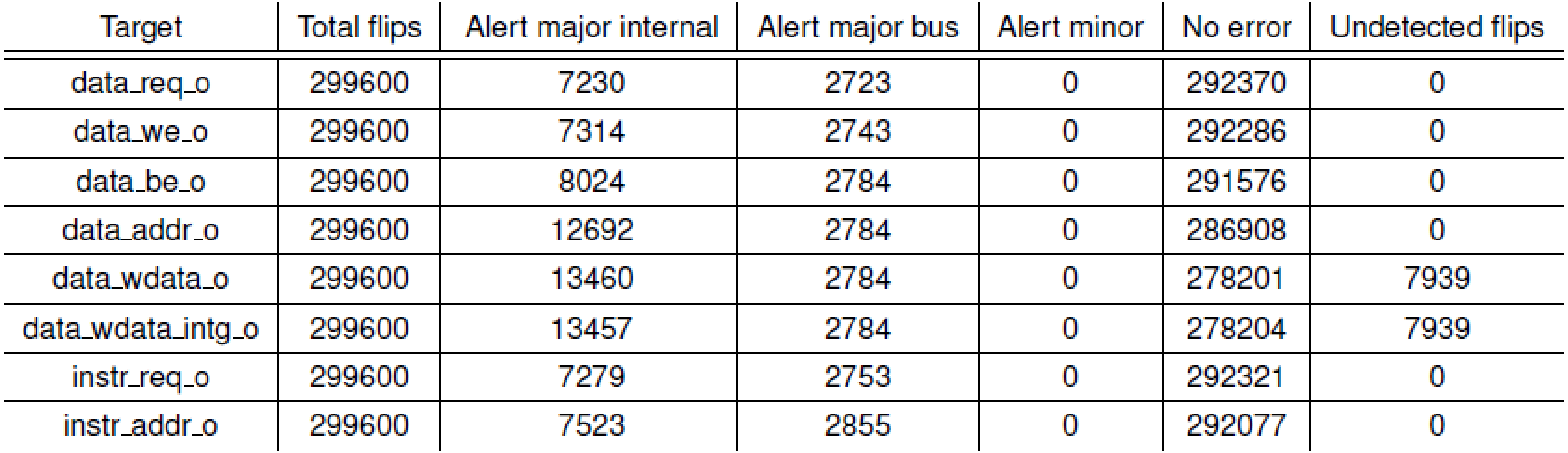
Health checking

- CPU state monitored each clock cycle
- CRC is accumulated on critical internal signals
  →Checksum is signature for correct program flow:
  PC, D-addr, D-data, I-addr, RF, CSR
- Golden simulation is performed initially
- CRC is checked after SEU injection
- 300k SEUs injected



Fault Injection Simulation Results

Results by symptom

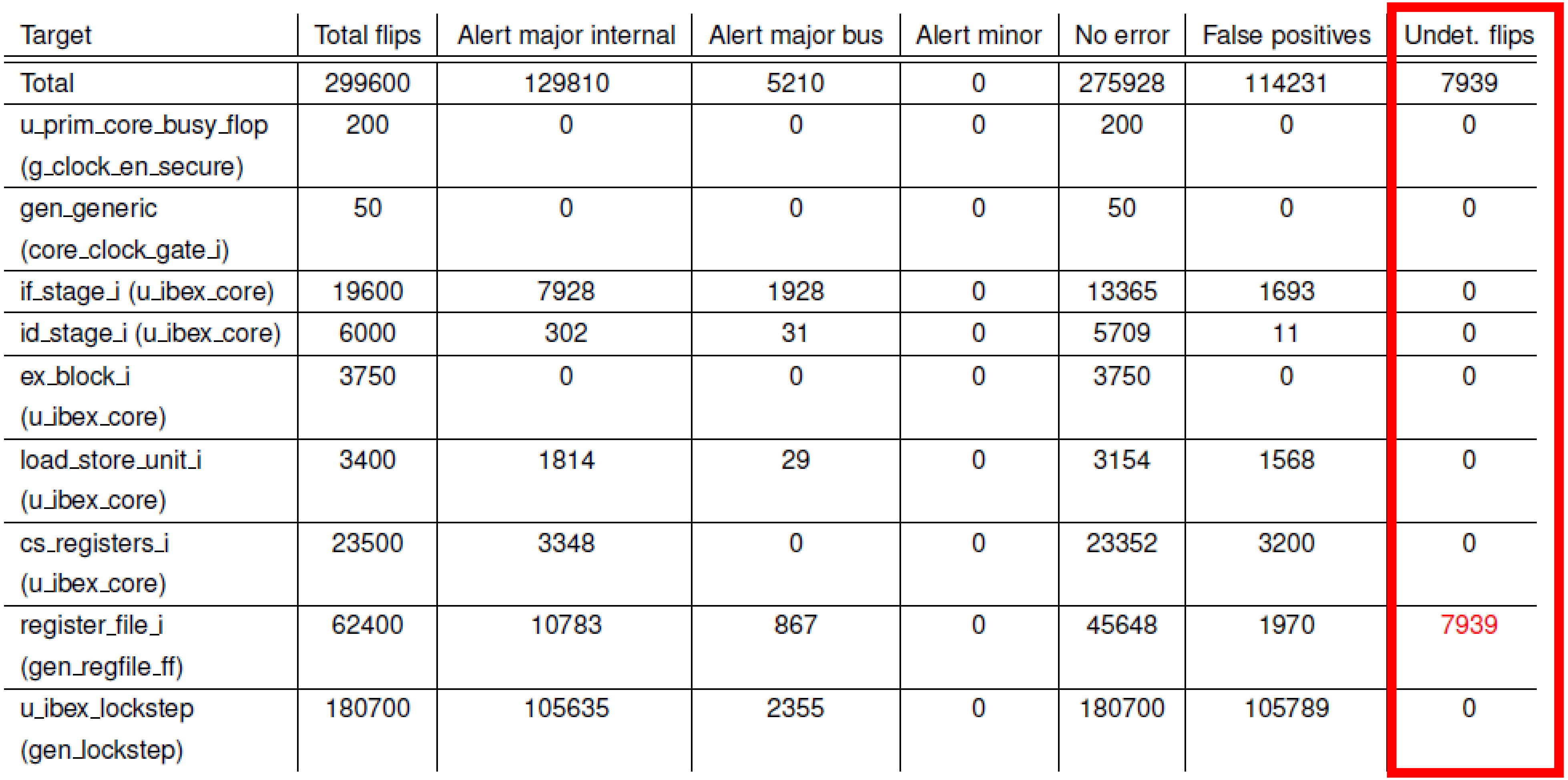


Target	Total flips	Alert major internal	Alert major bus	Alert minor	No error	Undetected flips
data_req_o	299600	7230	2723	0	292370	0
data_we_o	299600	7314	2743	0	292286	0
data_be_o	299600	8024	2784	0	291576	0
data_addr_o	299600	12692	2784	0	286908	0
data_wdata_o	299600	13460	2784	0	278201	7939
data_wdata_intg_o	299600	13457	2784	0	278204	7939
instr_req_o	299600	7279	2753	0	292321	0
instr_addr_o	299600	7523	2855	0	292077	0

TB found CRC error but alert was low

Fault Injection Simulation Results

Results by module



Target	Total flips	Alert major internal	Alert major bus	Alert minor	No error	False positives	Undet. flips
Total	299600	129810	5210	0	275928	114231	7939
u_prim_core_busy_flop (g_clock_en_secure)	200	0	0	0	200	0	0
gen_generic (core_clock_gate_i)	50	0	0	0	50	0	0
if_stage_i (u_ibex_core)	19600	7928	1928	0	13365	1693	0
id_stage_i (u_ibex_core)	6000	302	31	0	5709	11	0
ex_block_i (u_ibex_core)	3750	0	0	0	3750	0	0
load_store_unit_i (u_ibex_core)	3400	1814	29	0	3154	1568	0
cs_registers_i (u_ibex_core)	23500	3348	0	0	23352	3200	0
register_file_i (gen_regfile_ff)	62400	10783	867	0	45648	1970	7939
u_ibex_lockstep (gen_lockstep)	180700	105635	2355	0	180700	105789	0

Some errors are not detected!

KU LEUVEN

Fault Injection Simulation Results

Improvement

Observation:

- Some errors from **register file** are undetected
- But … register file is protected with 39/32 Hsiao code = Simple?

Simulations

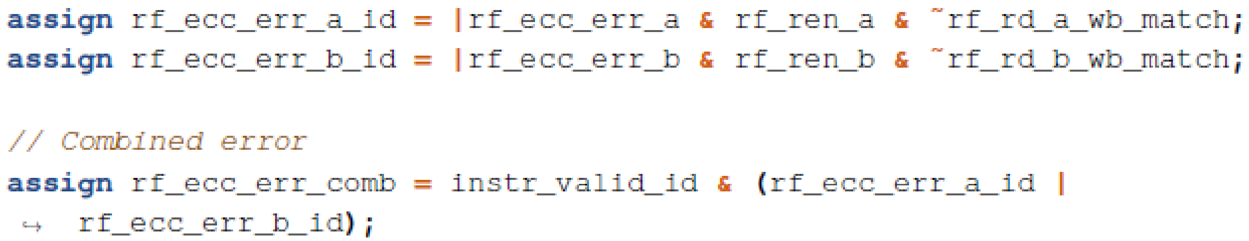
- Run a few cases for undetected errors
- Trace internal alert signals

→ Result: Alerts were raised internally but masked towards the output

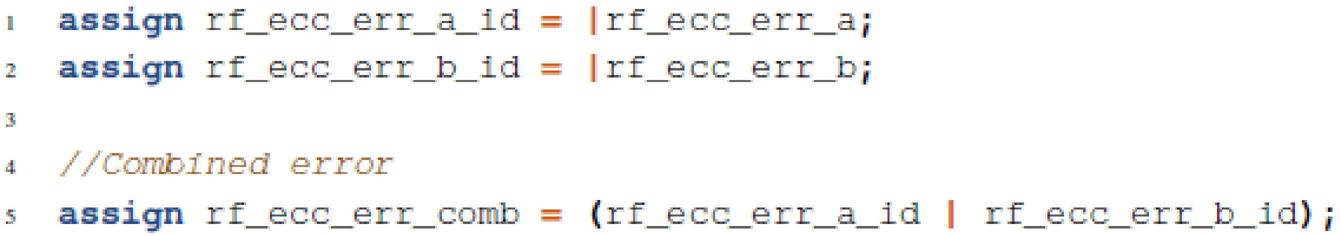
Fault Injection Simulation Results

Improvement

- Modifications to the source code
    - Opening issue on Github
    - Bug?
- Modifications resulted in no undetected bit flips

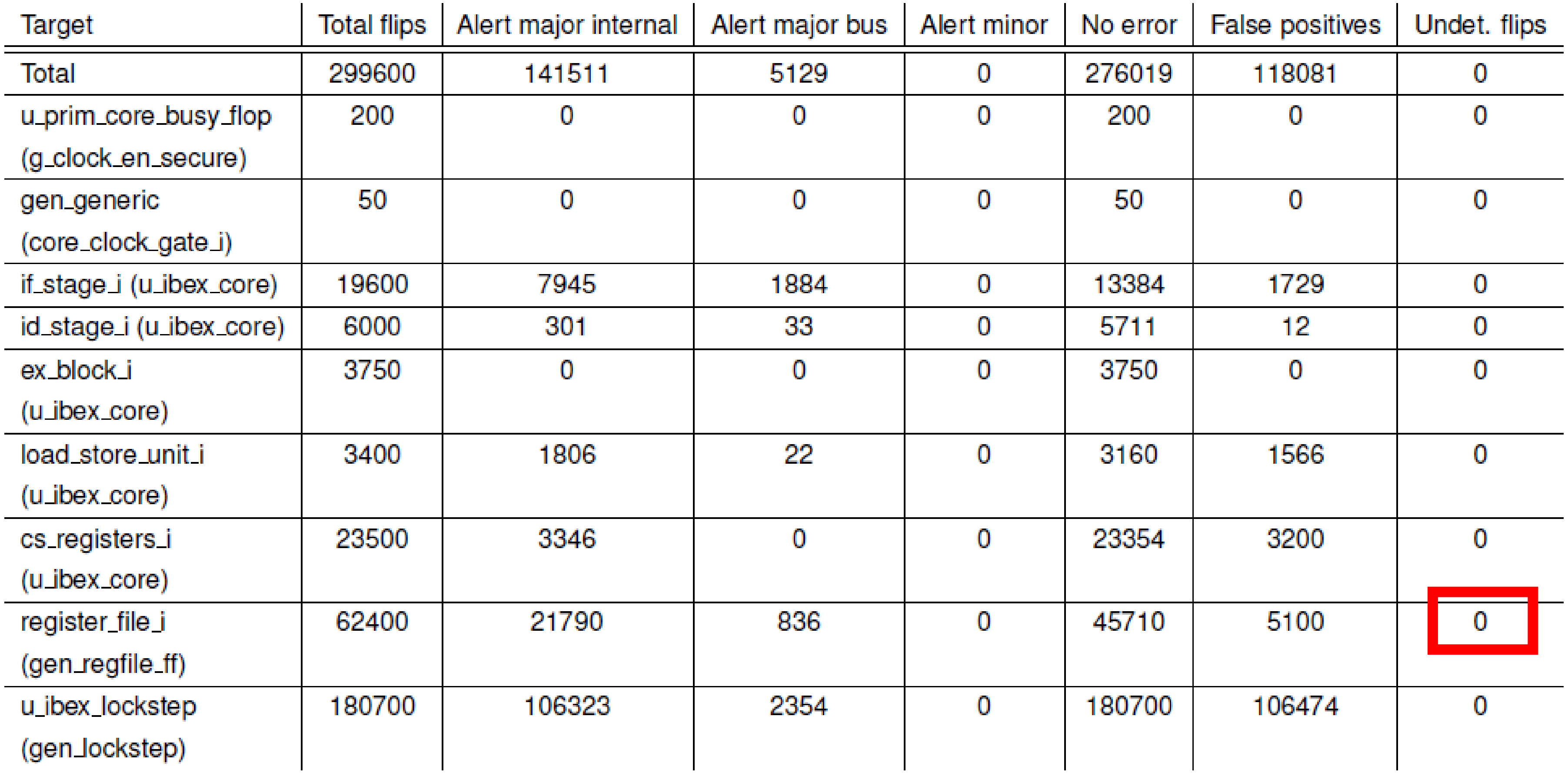


```
assign rf_ecc_err_a_id = |rf_ecc_err_a & rf_ren_a & ~rf_rd_a_wb_match;
assign rf_ecc_err_b_id = |rf_ecc_err_b & rf_ren_b & ~rf_rd_b_wb_match;

// Combined error
assign rf_ecc_err_comb = instr_valid_id & (rf_ecc_err_a_id |
  ↪  rf_ecc_err_b_id);
```



```
assign rf_ecc_err_a_id = |rf_ecc_err_a;
assign rf_ecc_err_b_id = |rf_ecc_err_b;

//Combined error
assign rf_ecc_err_comb = (rf_ecc_err_a_id | rf_ecc_err_b_id);
```

Fault Injection Simulation Results



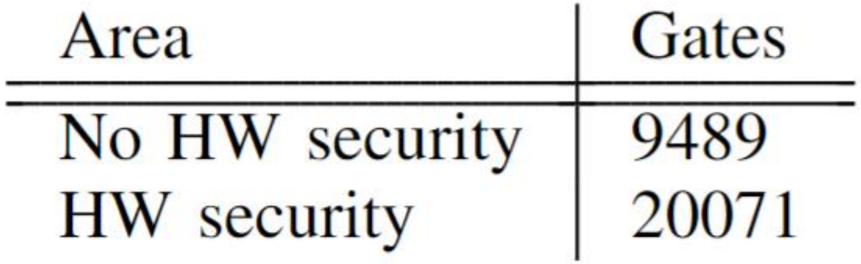
Target	Total flips	Alert major internal	Alert major bus	Alert minor	No error	False positives	Undet. flips
Total	299600	141511	5129	0	276019	118081	0
u_prim_core_busy_flop (g_clock_en_secure)	200	0	0	0	200	0	0
gen_generic (core_clock_gate_i)	50	0	0	0	50	0	0
if_stage_i (u_ibex_core)	19600	7945	1884	0	13384	1729	0
id_stage_i (u_ibex_core)	6000	301	33	0	5711	12	0
ex_block_i (u_ibex_core)	3750	0	0	0	3750	0	0
load_store_unit_i (u_ibex_core)	3400	1806	22	0	3160	1566	0
cs_registers_i (u_ibex_core)	23500	3346	0	0	23354	3200	0
register_file_i (gen_regfile_ff)	62400	21790	836	0	45710	5100	0
u_ibex_lockstep (gen_lockstep)	180700	106323	2354	0	180700	106474	0

Fault Injection Simulation Results

Area comparison

Synthesis performed in 180nm

- ~2x area overhead
- Mostly due to lockstep datapath
- Overhead includes comparison logic



Area	Gates
No HW security	9489
HW security	20071

Conclusion

Research Question:

Can these built-in security features be used to detect SEUs within the Ibex core?

→**Yes, but a slight modification to the core was necessary (bug?)**

Only error detection is present, how should we correct for errors?

- TMR directly corrects errors but large overhead
- Software/architecture correction required (checkpoint, rollback)

→Alert signals can be connected to CPU interrupt controller

No scrubbing in register file

Fault accumulate until Hsiao code cannot correct anymore

→ Registers must be refreshed in software regularly (compiler add-on required)

Conclusion

- RISC-V provides opportunity for HEP and Space applications
- Hardware secure RISC-V cores can provide a solution to ride along a much larger community → We can focus on SoC design
- Ibex RISC-V core was evaluated – CoCoTB simulaton environment
- Most errors were detectable
- Small RTL correction was necessary to provide 100% coverage

Thank you

Jeffrey Prinzie, Boris Engelen, Karel Appels, Levi Mariën, Naïn Jonckers