University of Southampton

Evaluating Hardware Reliability in the presence of Soft Errors

Bing Xue Mark Zwolinski

26-5-2022

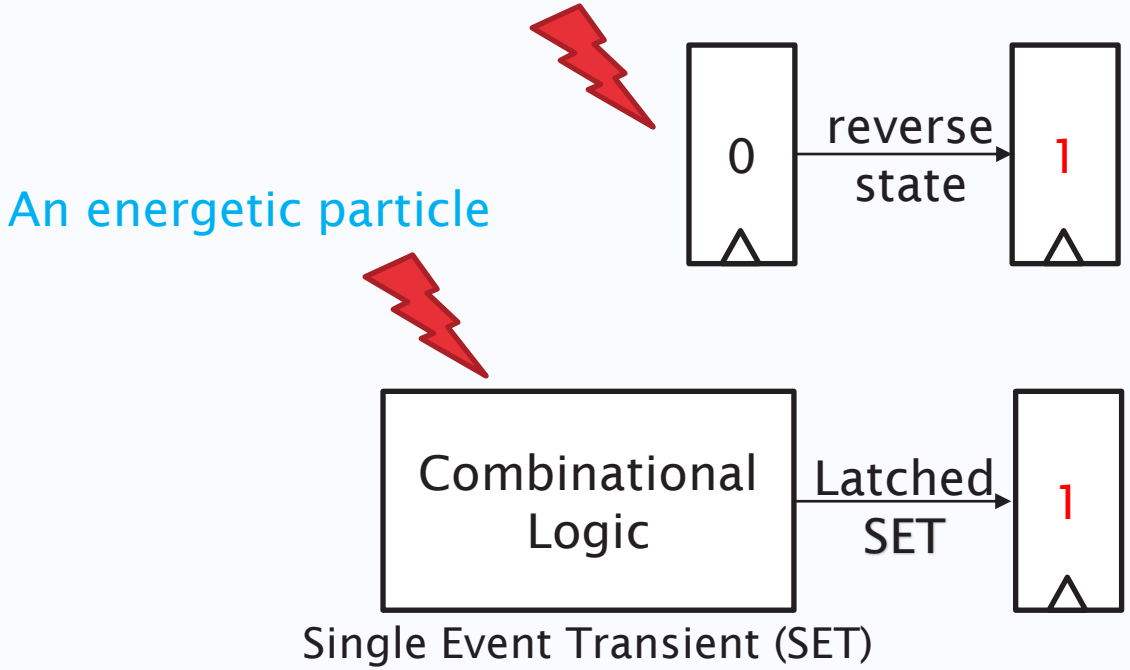
Introduction

- Research Motivation
- Proposed Method with details
- Results with explanation
- Conclusions

Single Event Upset

- Fault model: Single Event Upset (SEU)
- A major concern in terrestrial commercial digital circuits.

An energetic particle(an alpha/neutron particle)

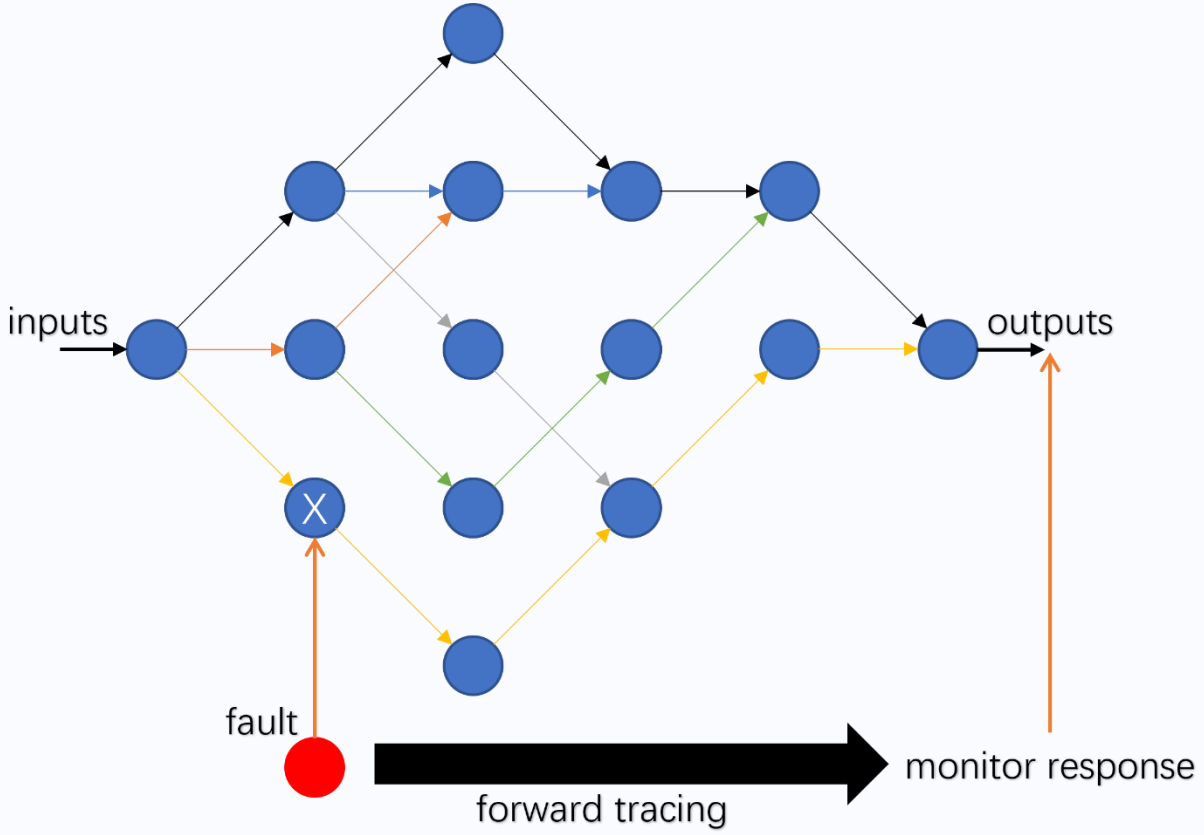


Research Motivation

- Single Event Upset
- Simulation-based Fault Injection

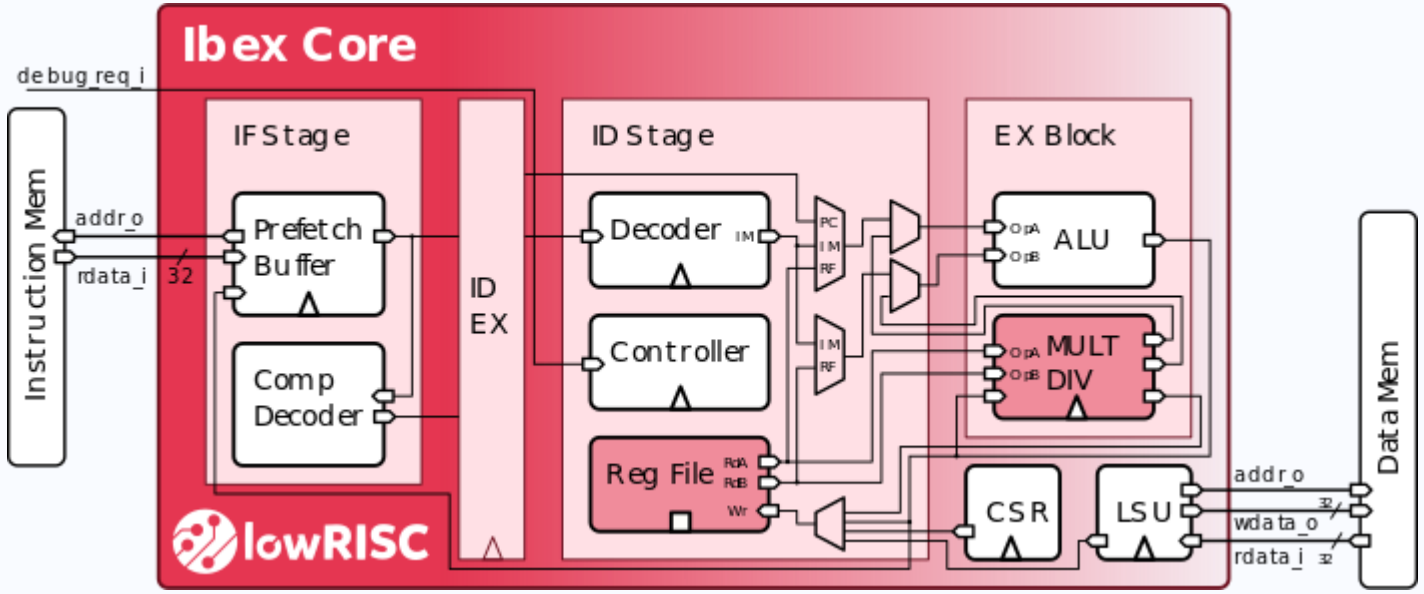
- Problems:
  1. Impossible to test all potential faults
  2. Limited fault coverage
  3. Root cause faults

- Can we solve these problems with a backward-tracing approach?



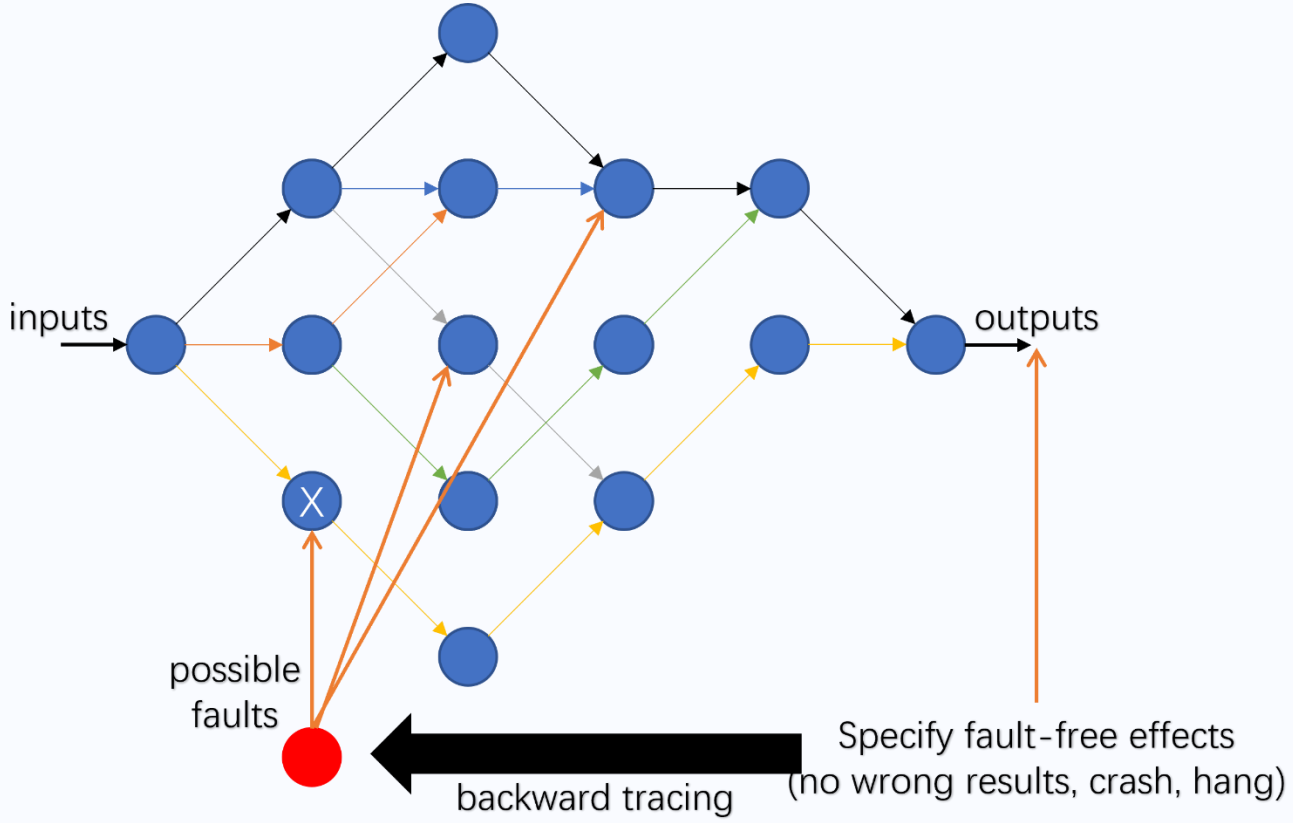
Test Platform

- RISC-V: an open-source Instruction Set Architecture
- Ibex core: an open-source 32-bit RISC-V 2-stage CPU core
  71 registers (2008 bits)



Proposed Method

- Formal verification
- Backward-tracing: from specification to find candidate faults
- Candidate faults
- Exhaustively search
- Reasonable time



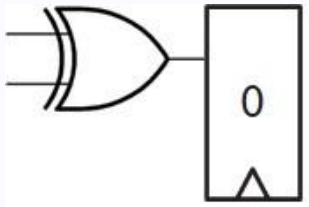
University of Southampton

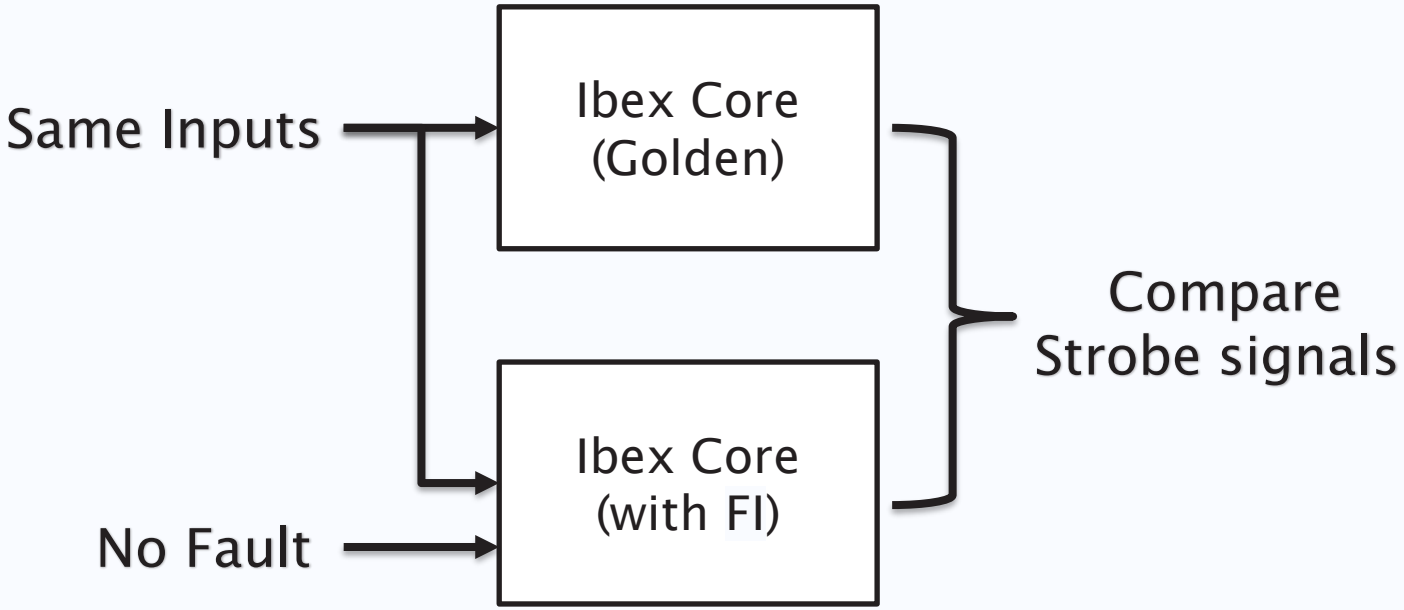
Methods

- 1. Implement a fault injection mechanism
- 2. Develop properties that can search and categorize faults
- 3. Abstract memories and develop constraints
- 4. Run model checking and Cone-Of-Influence (COI) analysis in parallel

Methods

- 1. Fault injection (FI) mechanism
- How to prove the mechanism has no impact on the Ibex core?
- Strobe properties





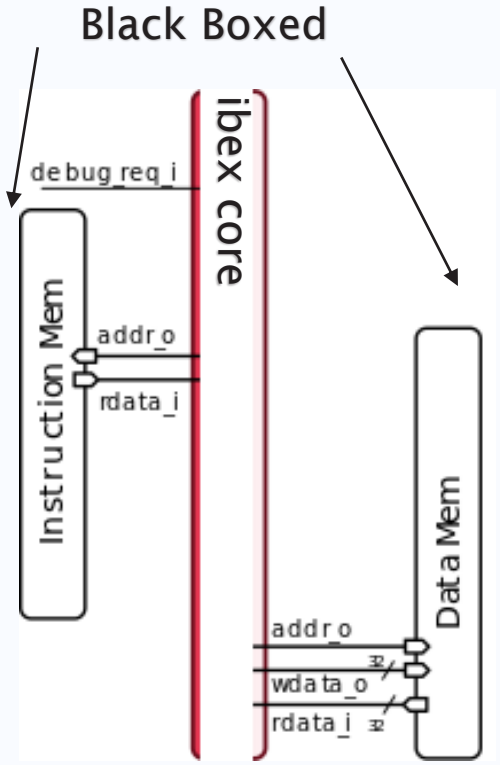
Methods

- 2. Formal properties
- Find faults according to the effects of faults (Silent Data Corruption, crash, hang, nothing)
- SDC: Strobe properties
- Crash: exception code stored in a CSR
- Hang:
  Wait For Instruction
  Dead State
  Live State
- Be careful with liveness!

```
assert_store_access_fault: assert property (
 @(posedge clk_i) disable iff (!rst_ni)
 (crash_priv_mode==2'b11) |->
 (crash_mcause_q!=6'd7) );
```

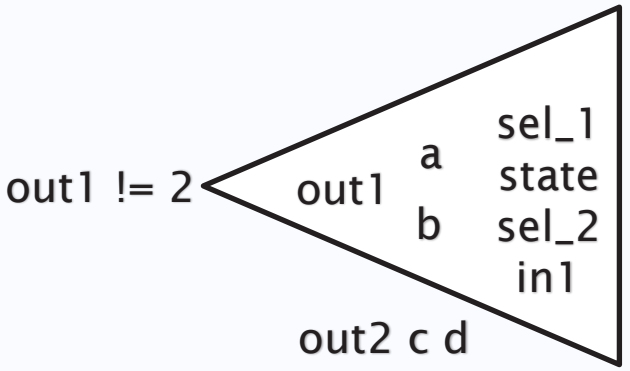
Methods

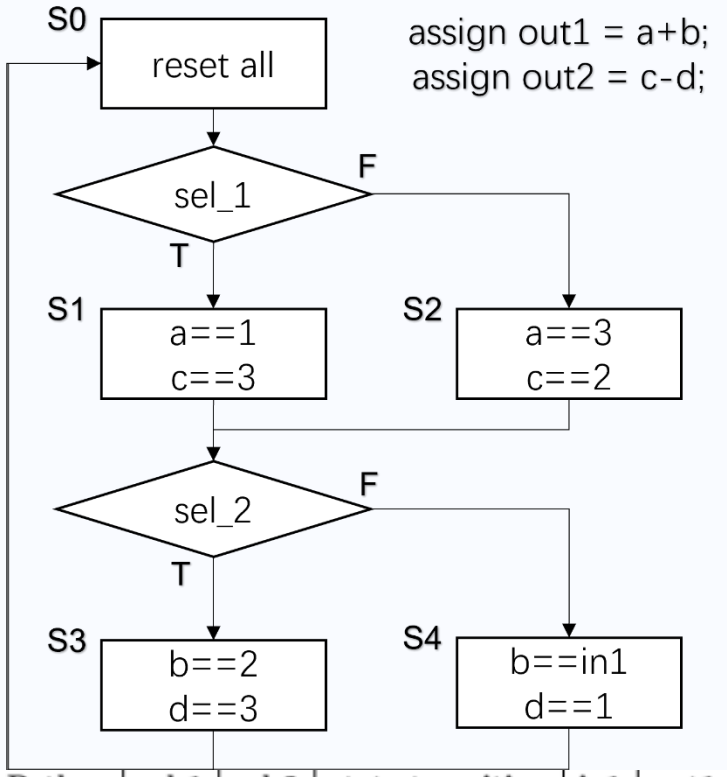
- 3. Abstract memories and develop constraints
- Constraints:
- a. Follow the hand-shake communication protocol
- b. Data from Data Mem are arbitrary
- c. Instructions from Ins Mem are legal RV32IMC instructions, depending on address



Methods

- 4. Model checking & COI analysis
- Model checking: find faults which violate specifications
- COI analysis: find faults which are structurally safe





Path	sel_1	sel_2	state transition	in1	out1
1	0	0	S0-S2-S4	0	0-3-3
2	0	0	S0-S2-S4	1	0-3-4
3	0	0	S0-S2-S4	2	0-3-5
4	0	0	S0-S2-S4	3	0-3-6
5 to 8	0	1	S0-S2-S3	x	0-3-5
9	1	0	S0-S1-S4	0	0-1-0
10	1	0	S0-S1-S4	1	0-1-2
11	1	0	S0-S1-S4	2	0-1-3
12	1	0	S0-S1-S4	3	0-1-4
13 to 16	1	1	S0-S1-S3	x	0-1-3

counter-example

University of Southampton

Results

Name	Proven	Bounded	Failure	Name	Proven	Bounded	Failure
Instruction_is_done	81	689	1238	memory_read_data	81	710	1217
Instruction	81	675	1252	memory_read_mask	81	739	1188
rs1_address	81	691	1236	memory_write_data	81	704	1223
rs2_address	81	693	1234	memory_write_mask	81	736	1191
rd_address	81	690	1237	Insn_access_fault	2002	0	6
rs1_read_data	81	692	1235	Illegal_insn	1908	5	95
rs2_read_data	81	693	1234	breakpoint	1963	2	43
rd_write_data	81	689	1238	load_access_fault	2006	0	2
current_pc	81	690	1237	store_access_fault	2006	0	2
next_pc	81	689	1238	Ecall_Mmode	2004	0	4
memory_address	81	693	1234	Hang_WFI	1995	10	3

SDC

crash

Results

- **Proven**: all faults in a bit cannot cause a corresponding error.
- **Bounded**: a fault in a bit is less likely to cause a corresponding error.
  If there exists a failure, it is beyond the (time/trace) limit.

SDC
crash

Name	Proven	Bounded	Failure
Instruction	81	675	1252
Insn_access_fault	2002	0	6
Hang_WFI	1995	10	3

- In formal verification, bounded proof is an acceptable type of results. It is hard to fully prove some formal properties due to various reasons (state explosion, resource limit).
- **Failure**: a fault in a bit can cause a corresponding error.
- Sum of **Proven**, **Bounded**, and **Failure** in each error type is 2008 – 2008 bits in the core.
- Now we know vulnerability of each bit & possible fault effects in each bit.

University of Southampton

Findings

- Some registers are more vulnerable than others:
  faults in some registers can cause multiple types of errors
  faults in some registers cannot cause errors
- We can use formal verification to find and classify faults according to fault effects.
- Some bits are vulnerable to certain errors:
  faults in some bits cannot cause crash or hang_WFI but can cause SDC
- Even in the same register, fault effects in different bits may be different.

University of Southampton

Conclusions

- Our method combines formal verification and fault injection, exhaustively searchs the whole state space and the fault list, and performs backward tracing of SEUs.
- Our method can successfully categorize SEU effects in hardware.
- Next Steps:
  - Hang (Dead State & Live State) – avoid liveness in formal verification
  - Protection – we have shown some bits are vulnerable to certain errors; we can use different technologies (with different costs and different efficiencies) to protect different vulnerable bits.
  - Evaluation – we can use the proposed method to evaluate different protection technologies.

University of Southampton

Questions?