Cross-Layer Fault-Space Pruning for Hardware-Assisted Fault Injection

Christian Dietrich, Achim Schmider, Oskar Pusz
Guillermo Payá Vayá, Daniel Lohmann

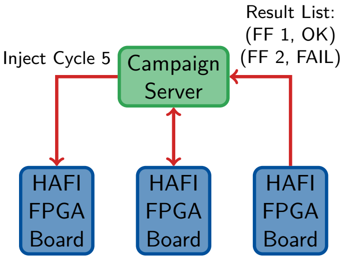
Leibniz Universität Hannover

June 27, 2018

supported by DFG

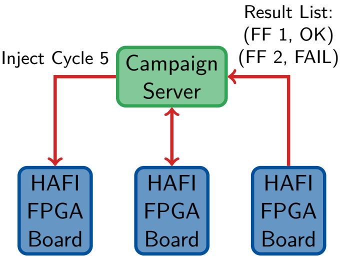
Hardware-Assisted Fault Inject Platforms

- Transient hardware faults are becoming more frequent on sea level
  - Shrinking hardware structure sizes
  - More transistors and more embedded systems
- Safety-critical software must be rated for the resilience
  - Fault injection of one golden run can provide realistic measure
  - Fault space is *Huge*! (cycles × locations)
  - Simulation of faulty behavior is slow (especially for circuits)
- Hardware-Assisted Fault Inject Campaigns
  - Campaign server sends injection commands to FPGA Boards
  - FPGA simulate netlist + fault-injection logic
  - FPGAs can prune the fault list for **benign** faults at run time



Hardware-Assisted Fault Inject Platforms

- Transient hardware faults are becoming more frequent on sea level
  - Shrinking hardware structure sizes
  - More transistors and more embedded systems
- Safety-critical software must be rated for the resilience
  - Fault injection of one golden run can provide realistic measure
  - Fault space is *Huge*! (cycles × locations)
  - Simulation of faulty behavior is slow (especially for circuits)
- Hardware-Assisted Fault Inject Campaigns
  - Campaign server sends injection commands to FPGA Boards
  - FPGA simulate netlist + fault-injection logic
  - FPGAs can prune the fault list for **benign** faults at run time

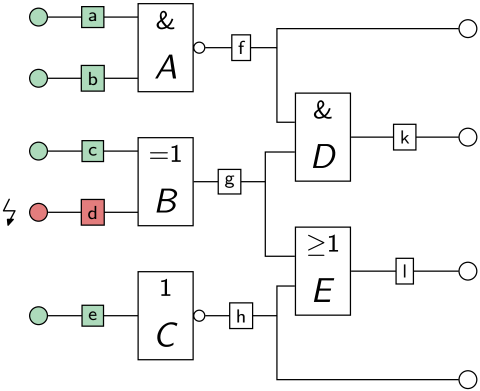


⇒ Prune Fault List depending on the Dynamic State

Approach

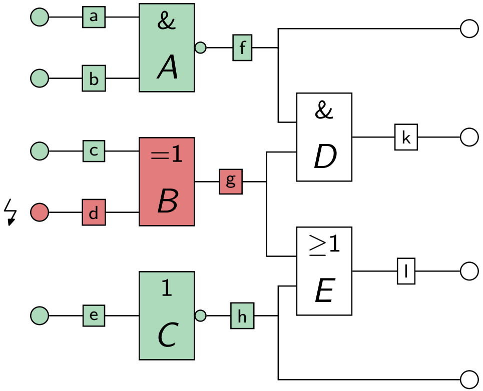
Fault Masking Capabilities of Logic Gates

- Fault model: Single-event upsets in flip flops
  - One flip–flop output becomes untrusted
  - Other flip–flops remain trusted



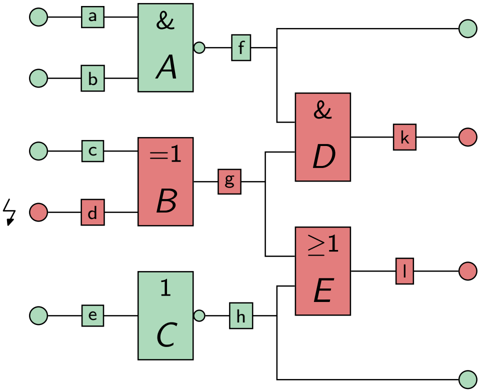
Fault Masking Capabilities of Logic Gates

- Fault model: Single-event upsets in flip flops
  - One flip–flop output becomes **untrusted**
  - Other flip–flops remain **trusted**
- Conservative fault propagation in netlist
  - One untrusted input leads to untrusted gate outputs



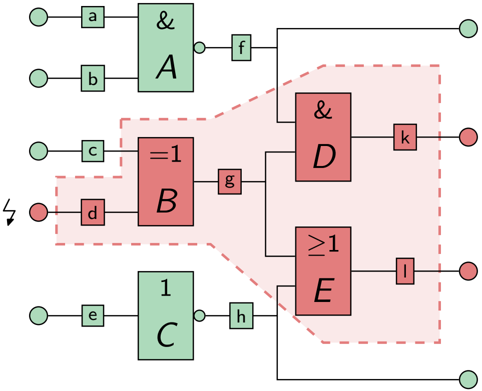
Fault Masking Capabilities of Logic Gates

- Fault model: Single-event upsets in flip flops
  - One flip–flop output becomes **untrusted**
  - Other flip–flops remain **trusted**
- Conservative fault propagation in netlist
  - One untrusted input leads to untrusted gate outputs
  - If fault reaches outputs → fault might lead to error



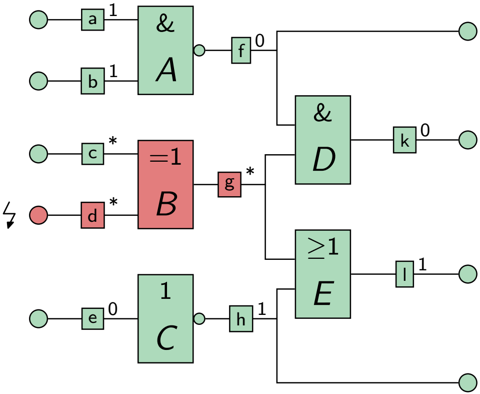
Fault Masking Capabilities of Logic Gates

- Fault model: Single-event upsets in flip flops
  - One flip–flop output becomes **untrusted**
  - Other flip–flops remain **trusted**
- Conservative fault propagation in netlist
  - One untrusted input leads to untrusted gate outputs
  - If fault reaches outputs → fault might lead to error
  - No wire values → we must distrust the fault cone



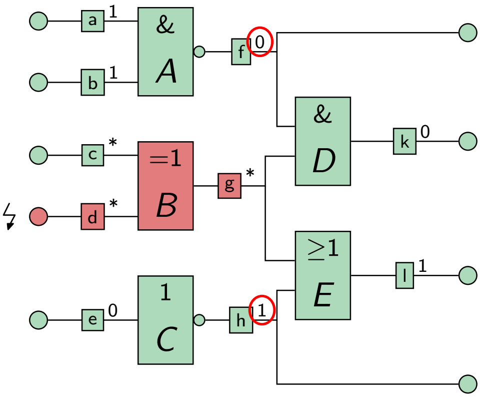
Fault Masking Capabilities of Logic Gates

- Fault model: Single-event upsets in flip flops
  - One flip–flop output becomes **untrusted**
  - Other flip–flops remain **trusted**
- Conservative fault propagation in netlist
  - One untrusted input leads to untrusted gate outputs
  - If fault reaches outputs → fault might lead to error
  - No wire values → we must distrust the fault cone
- Fault-cone–border wires can stop the fault
  - Gates can mask the fault, if some inputs are trusted
  - Constraint on border-wires indicates benign fault



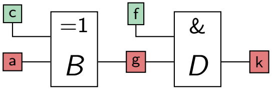
Fault Masking Capabilities of Logic Gates

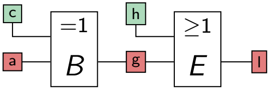
- Fault model: Single-event upsets in flip flops
  - One flip–flop output becomes **untrusted**
  - Other flip–flops remain **trusted**
- Conservative fault propagation in netlist
  - One untrusted input leads to untrusted gate outputs
  - If fault reaches outputs → fault might lead to error
  - No wire values → we must distrust the fault cone
- Fault-cone–border wires can stop the fault
  - Gates can mask the fault, if some inputs are trusted
  - Constraint on border-wires indicates benign fault
- Fault **Ma**sking **Te**rm (MATE)
  - Logic expression of internal netlist wires
  - $\text{MATE}_d = 1$, iff fault_d is known to be benign



$\neg f \wedge h \Rightarrow$ *fault*(d) is benign

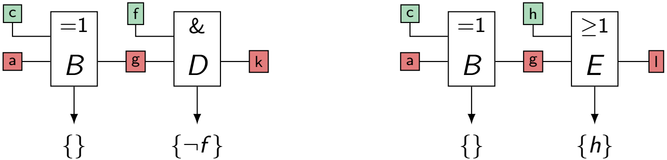
Finding and Employing Fault MATEs





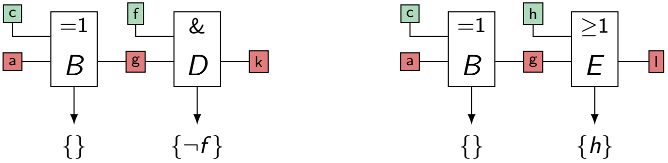
- For every input wire: Fault must be masked on any path input ⟶ output

Finding and Employing Fault MATEs



- For every input wire: Fault must be masked on any path input → output
  - Every gate has a set of masking terms that stop propagation here
  - Combine one masking term from every path into a candidate MATE
  - Collect MATE-candidate sets overall input wires
  - Use VCD trace of the circuit to find and rate effective MATEs

Finding and Employing Fault MATEs



- For every input wire: Fault must be masked on any path input ⟶ output
  - Every gate has a set of masking terms that stop propagation here
  - Combine one masking term from every path into a candidate MATE
  - Collect MATE-candidate sets overall input wires
  - Use VCD trace of the circuit to find and rate effective MATEs
- Integrate TOP-N MATEs into FPGA fault-injection platform
  - MATEs are connected to the netlist-internal wires
  - If MATE triggers, the corresponding fault(s) can be remove from the fault list
  - MATE prune the fault list depending on the dynamic state in every cycle

Results

Experimental Setup

- Test Benchmarks
  - ASIC synthesis using Synopsys Design Compiler 2017.09-SP1
  - 15nm FinFET-based Open Cell Library
  - Sythesized netlist for 2 processor designs: AVR, MSP430 (neo430)

- Search for heuristically for candidate MATEs
  - Use sets of flip-flop outputs as start points
  - Two sets: All flip-flops (FF) and flip-flops outside of register file (FF w/o RF)
  - One MATE can prune several detect several benign flip-flops

- Select and rate MATEs with wire trace of running program
  - Fibonacci and convolution
  - Select MATEs that triggered most (Top-N)
  - Calculate fault-list reduction
  - Cross validation between selection and rating in the paper

Fault List Reduction

- AVR: 8-bit RISC microcontroller, implementing a two-stage pipeline design
  - 383 flip–flops, without register-file: 135 FFs
  - Average Fault-Cone Size: 656 gates
  - 164 seconds for MATE exploration

- neo430: 16-bit multi-cycle MSP430-compatible microcontroller
  - 743 flip–flops, without register-file 519 FFs
  - Average Fault-Cone Size: 287 gates
  - 126 seconds for MATE exploration

Fault List Reduction

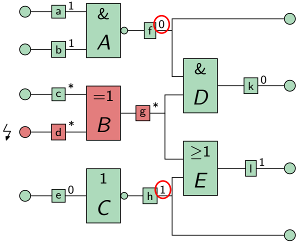
- AVR: 8-bit RISC microcontroller, implementing a two-stage pipeline design
- neo430: 16-bit multi-cycle MSP430-compatible microcontroller

- Results for 8,500 cycles of a convolution:

	AVR		neo430	
	FF	FF w/o RF	FF	FF w/o RF
#Eff. MATEs	390	247	441	437
Avg. #inputs	5.8± 1.8	4.9± 1.2	3.4± 1.9	3.4± 1.9
Masked Faults	7.90 %	16.48 %	14.32 %	20.45 %
Top 10	2.58 %	7.05 %	4.97 %	7.11 %
Top 50	5.90 %	15.86 %	13.11 %	18.77 %
Top 100	7.79 %	16.43 %	14.01 %	20.02 %
Top 200	7.89 %	16.48 %	14.32 %	20.44 %

Conclusion

- Hardware-assistance increases feasibility of fault injection
- Fault-masking terms detect **surely benign** faults
- Easy to integrate with FPGA-based injection platform
- Reducion of fault list by 8-14 percent (up to 16-20 percent w/o RF)



Backup: Analysis Time

	AVR		MSP430	
	FF	FF w/o RF.	FF	FF w/o RF
Faulty Wires	383	135	743	519
Avg. Cone [#gates]	656	840	287	151
Med. Cone [#gates]	547	581	236	27
Run Time [s]	164	34	126	90
#Unmaskable	81	57	96	70
#MATE candid.	$3 \cdot 10^7$	$7 \cdot 10^6$	$4 \cdot 10^7$	$2 \cdot 10^7$
#MATE	24,536	3,226	19,180	17,649