Scale4Edge

Scalable Infrastructure for Edge Computing

Fault Simulation on RISC-V

Endri Kaja, Nicolas Gerlin, Jad Al Halabi, Wolfgang Ecker

SPONSORED BY THE Federal Ministry of Education and Research

infineon

Introduction

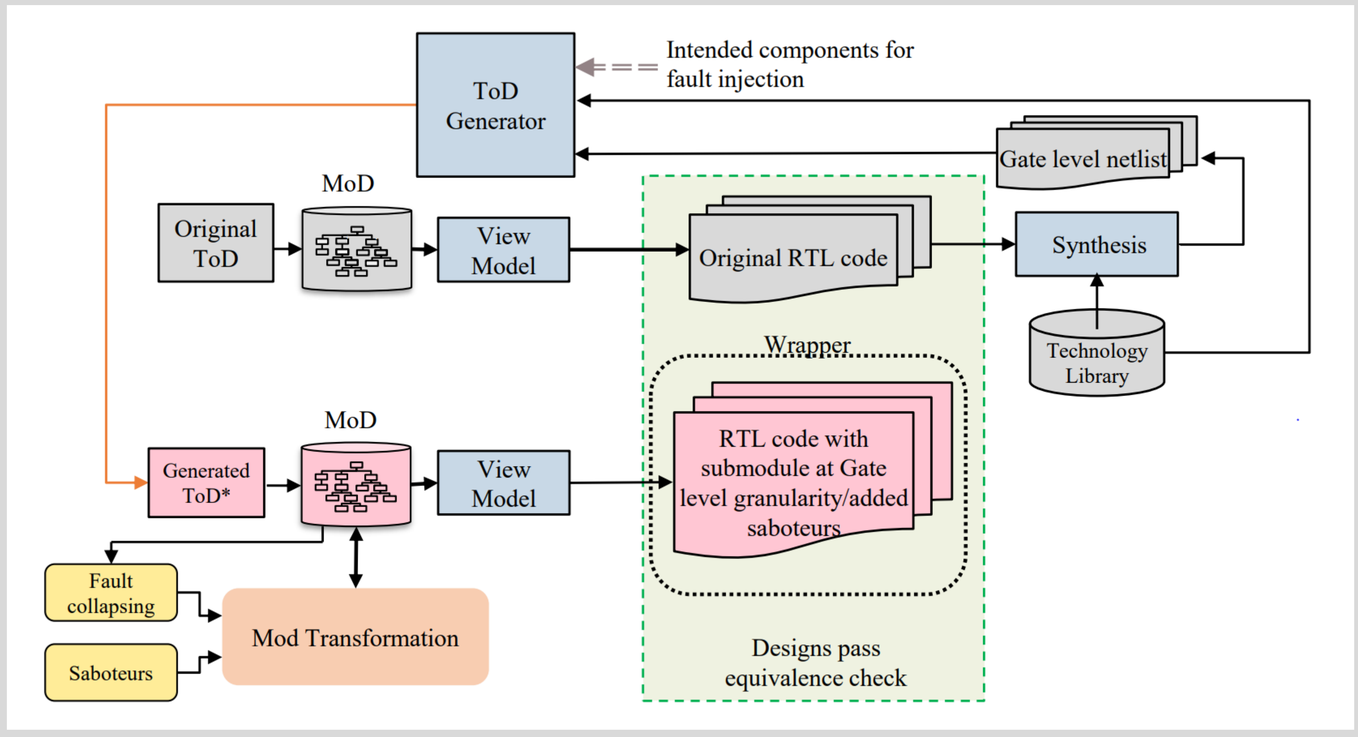
Many System-on-Chips (SoCs) are utilized in safety-critical domain
- 44% of ASICs have integrated safety-critical features[1]

ISO26262 recommends *fault injection* to verify safety-critical systems

A versatile and automated fault injection framework has been developed to generate a wide range of fault injection campaigns
- The framework is vendor-independent
- The design RTL is generated via in-house generator *MetaRTL*
- The design is represented via a mixed granularity, i.e. only the selected parts of the design that are subjected to fault injection are kept at the gate-level granularity and the rest in traditional RTL
- This approach enables fast RTL fault simulation without compromising the accuracy of gate-level fault simulation

The fault injection framework enables automated statistical fault injection (SFI) on RISC-V processors
- Combined with automatic generation of RISC-V designs

[1] Part 7: The 2022 wilson research group functional verification study," https://blogs.sw.siemens.com/verificationhorizons/2022/11/28/ part-7-the-2022-wilson-research-group-functional-verification-study/



Statistical Fault Injection

Commonly utilized technique which aligns with ISO26262 standards

The properties of the entire fault injection population, including all potential fault models occurring at any clock cycle, adhere to a normal distribution[2]

The process of random fault sampling ensures that each individual fault configuration within the initial population, at a specific cycle, has an equal likelihood of being selected for the sample. This is achieved by using a uniform distribution for the random selection

The amount of n faults to inject per campaign can be calculated by[2]:

$$n = \frac{N}{1 + e^2 \times \frac{N-1}{Z^2 \times p \times (1-p)}}$$

- N is initial population size,
- p is the standard error and proven that should be 0.5,
- e is the margin of error, and
- Z is confidence level parameter

Typically N is very large and equation is simplified by assuming N -> ∞

Example: Total amount of fault injections *n* for various parameters

	95% confidence Z =1.96	**95% confidence Z =2.5758**	**95% confidence Z = 3.0902**
e = 5%	n = 384	n = 663	n = 955
e = 1%	n = 9581	n = 16519	n = 23874

[2]: R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in 2009 Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 502–506

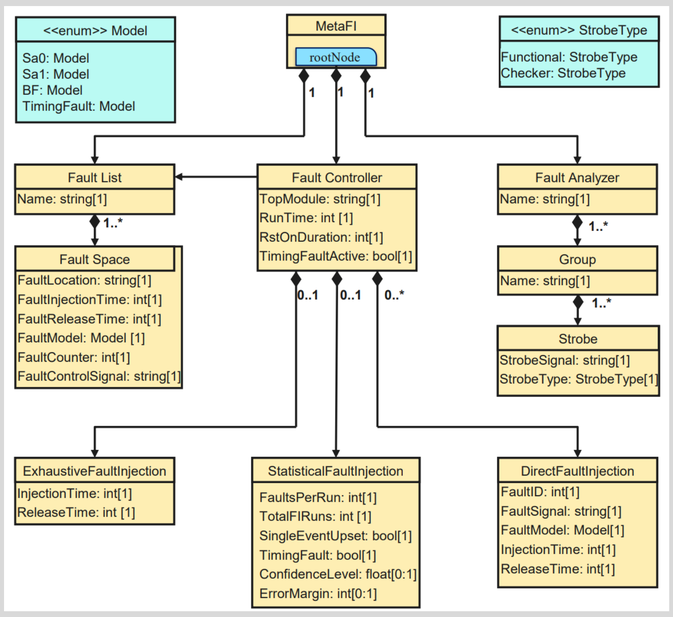
Model-Driven Statistical Approach for Fault Propagation Analysis

The framework reads the design and generates a formal model of the fault list

An intermediate model called the FI-model defines the fault injection features. This model includes all the necessary information for conducting the fault injection campaign, such as fault attributes and design strobes (signals) to be analyzed

Finally, a template engine is employed to translate the model into a final fault handler testbench, which can be implemented in either SystemVerilog or Verilator-based C++

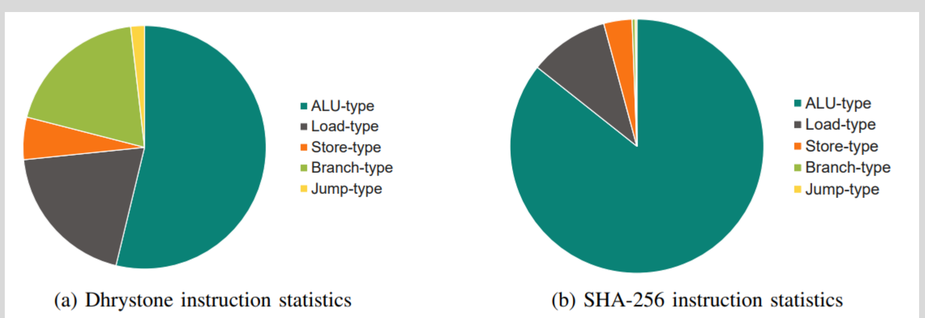
The core element of the automated injection campaign is the MetaFI metamodel:
- Fault Controller
- Fault List
- Fault Analyzer



CPU Workloads for Fault Propagation Analysis

Two benchmarks:
- Dhrystone
- SHA-256

These algorithms pose distinct challenges in terms of fault tolerance and error propagation, making them valuable additions to the suite of benchmarks for fault analysis
- Performance evaluation and cryptographic algorithms

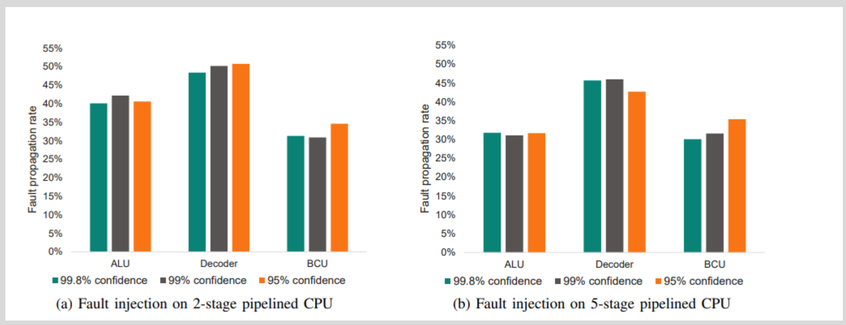


(a) Dhrystone instruction statistics (b) SHA-256 instruction statistics

Application and Evaluation

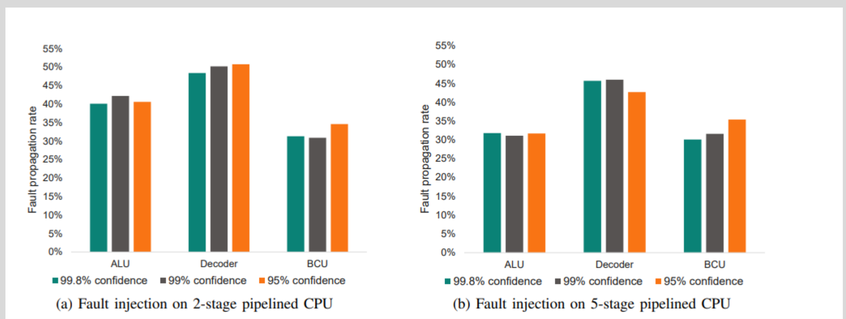
Fault simulation on 2 distinct RISC-V CPUs: 2 stage and 5 stage pipelined RV32-IMC

Dhrystone:



(a) Fault injection on 2-stage pipelined CPU (b) Fault injection on 5-stage pipelined CPU

SHA-256:



(a) Fault injection on 2-stage pipelined CPU (b) Fault injection on 5-stage pipelined CPU

Project Coordination:
Infineon Technologies AG
Wolfgang Ecker
fon +49 89 234-45334
email: wolfgang.ecker@infineon.com

Project Management:
edacentrum GmbH
Andreas Vörg
fon +49 511 936874-62
email voerg@edacentrum.de

Project Duration: 01.05.2020 - 31.12.2025
Homepage: www.edacentrum.de/scale4edge/en/

SPONSORED BY THE Federal Ministry of Education and Research

The Scale4Edge project (project label 16ME0122K-16ME0140+16ME0465+16ME0900+16ME0901) is supported by the German Federal Ministry of Education and Research (BMBF).
The authors are responsible for the content of this publication.