Motivation: Evaluating Soft Error Resilience

Functional Safety (FuSa)
- Fault Tolerance / Fault Detection
- Fault Model: Random
- Example: Cosmic and Package Radiation, EMF
- Static Evaluation (ISO-26262)
- Metrics: Silent Data Corruption (SDC), Detectable Unrecoverable Error (DUE)

Security
- Fault Detection
- Fault Model: Targeted
- Example: Fault Injection Attack, DFA
- Verification against Attack Vectors
- Metrics: e.g., Attack Feasibility

Checkpoint Restore Boot with Masking (CMSK)

Setup:
- Same as CRB. Pre-recorded checkpoint Ω from reference simulation
- Add a dummy Reference CPU core

Warm-up:
- Select Ω as close as possible before fault injection point f

Cooldown:
- After injection during cooldown, load Ω into dummy Reference CPU core
- Perform masking check with a bit-wise comparison of sequential logic: p(x) ⊕ p(y(x)) = 0
  → Additional 15-25% save for uniformly sampled experiments

Experimental Results

Experimental Setup:
- Simple SoC with cv32e40p RISC-V rv32imac core as fault injection target
- Running Embench™ IoT benchmark programs b
- Average simulation time τ and classification of experiments conducted on differently configured RTL. FI. Checkpoint interval of 10,000 clock cycles, 11,000 equal experiments per benchmark and configuration combination.

Fault Classification:
- MSK Fault masked and detected with CMSK feature
- DUE Detectable Unrecoverable Error (exceptions, bus faults, timeouts)
- SDC Silent Data Corruption

pLAT Potentially Latent, i.e., not an SDK, MSK, or DUE

Checkpoints

State-of-the-art and widely used technique to accelerate FI simulations.
→ Around 50% save for uniformly distributed experiments

Publications


Contact: johannes.geier@tum.de
Open Source: https://github.com/tum-ei-eda/vrtlmod

Chair of Electronic Design Automation
Department of Computer Engineering
School of Computation, Information and Technology
Technical University of Munich

Techniques and Tools for Fast Fault Injection Simulations of RISC-V Processors at RTL
Johannes Geier, Daniel Mueller-Gritschneder, and Ulf Schlichtmann

VTLMod: Verilator RTL Fault Injection Modification

Hardware
- Source-Code transformation on VRTL (LLVM/Clang Frontend Tool)
- Automatic insertion of injection expressions in source code
- Small overhead (ca. 10%) compared to plain VRTL
- Integrate as Module in Transaction-level Virtual Platform SoC

Input: Cycle-accurate SystemC/C++ models of Verilog RTL

Output: Fault Injectable VRTL (vRTLmod)

- Add access variables and rewrite source with injection expressions
- Find sequential logic
- Elaboration
- Generates RTL only

vRTLmod Flow [1]

Usage of vRTLmod Outputs in Virtual Platform without masking [1,2]. With masking checks (this work).

[1] Research was partly funded by the Bavarian State Ministry for Economic Affairs, Regional Development and Energy as part of the Bavarian funding program for research and development "Information and Communication Technology" through the project MiVRTL, grant number IUK623/002.

[2] This work has been developed in the project VRTLmod, funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) through the Federal Ministry of Education and Research (BMBF) under contract no. 01IS20096.