An Open-Source Framework for Xilinx FPGA Reliability Evaluation

Aitzan Sari, Vasileios Vlagkoulis, Mihalis Psarakis

Dept. of Informatics, University of Piraeus, Greece
Motivation

• FPGAs gain acceptance in critical applications (e.g. space avionics)
  • SRAM FPGAs are vulnerable to Single Event Effects (SEEs)

• Need for FPGA design automation tools for the analysis of SEE vulnerability and design of mitigation approaches
  • Commercial and academic tools are available

• Our goal: provide an open-source framework to support FPGA reliability assessment and improvement tasks
Existing academic frameworks

• Reliability analysis and Implementation tools
  • VERI-Place (by Politecnico di Torino): for the placement of hardened circuits
    • an executable version is available upon request
  • SEVAX (by University of Piraeus): analytical approach for the SEU sensitivity analysis
    • open-source, available in github

• Fault injection platforms
  • FTUNSHADES (by University of Sevilla and ESA): hardware-accelerated fault injection platform
    • available as a cloud-service, free of charge for research projects
  • JTAG configuration manager (by Brigham Young University): supports the JTAG protocol and FPGA configuration function and runs on an embedded board
    • not publicly available
An open-source framework

To provide various FPGA configuration functions at low cost:

- No extra equipment
- No DUT modifications
Framework architecture

User application

TCL I/F handler

FPGA configuration
JTAG functions (TCL)

Local TCP/IP server

Vivado

1. On-chip logic
2. Target FPGA
3. User Logic

High-level functions

JTAG configuration engine
On-chip logic

- Provides communication with the FPGA through the JTAG port
  - Slower but less sensitive to radiation effects

- Access to the **configuration memory** and registers
  - For reconfiguration (fault injection), readback and scrubbing

- Access to the **user logic**
  - For monitoring and debugging

- Three different alternatives
1 - Basic setup

JTAG controller provides access to the configuration memory and user logic (through BSCAN primitive)

- Enables:
  - FPGA (re)configuration and configuration memory readback
  - Fault injection to the configuration frame(s)
  - Monitoring of the user logic
2 – Frame ECC-based scrubbing setup

- **FRAME ECC** provides access to the embedded ECC logic
- **FIFO** retains the erroneous frames detected by the FRAME ECC
  - FIFO is read through the BSCAN primitive
- **HeartBeat** logic provides watchdog functionality for the FRAME ECC
- Enables the building of a configuration memory scrubber
  - FRAME ECC scans the configuration memory and stores the erroneous frames
  - FIFO is read periodically
    - in case of error, configuration frames are repaired by partial reconfiguration
  - Heartbeat is read periodically
    - in case of alarm, FPGA is fully reconfigured
3 – Hardened (TMR) version

- FIFO and HeartBeat modules are triplicated and voted
- Multiple primitives simplifies the routing and facilitates the implementation of the Xilinx Isolation Design Flow

- This setup is for use in a radiation environment (e.g. radiation experiments)
  - Tolerates all the programmable resources of the on-chip logic against SEUs
JTAG configuration engine (CE)

- Provides the **low-level JTAG operations** (as TCL functions) for accessing the FPGA configuration memory
  - basic boundary scan commands
    - e.g. `scan_ir_hw_jtag` and `scan_dr_hw_jtag`
  - complex **Vivado TCL commands**
    - e.g. `Configure`, `Readback`, `ReadbackCapture`, `ReadbackVerify`, `RegisterWrite`, `RegisterRead`, `FrameWrite`, `FrameRead`

- **accepts commands** from the user application and
- **executes** the associated low-level TCL functions
High-level functions

• Consists of the user application and the interface functionality with the JTAG CE (TCL I/F handler)

• Target application communicates with the JTAG CE
  • Using the APIs exposed by the TCL I/F handler

• Proposed framework has been designed using Qt and PySide2
  • Qt is a cross-platform application and UI development framework
  • PySide2 is a Python binding for Qt
TCP client-server approach

- JTAG CE runs in a separate thread as TCP server
- TCL I/F handler starts a Vivado instance in batch mode and a TCP client
- TCP client is used as Inter-Process Communication (IPC) between the user application thread and the JTAG CE

- TCP client-server scheme was used to reduce the execution time overhead introduced by the Vivado instance
  - Creates a single Vivado instance instead of instantiating Vivado for every TCL script execution
Multiple client/server schemes

- TCP client-server solution enables the development of:
  - **multi-client schemes**: multiple applications (clients) run in the same platform targeting the same FPGA device
    - example: a fault injection tool and a memory scrubbing process run in parallel for the same target FPGA
  - **multiple-server schemes**: multiple servers communicate with a single client and target different FPGA devices
    - example: concurrent fault injection in multiple FPGA boards
Use cases

• Implemented for the needs of a radiation testing experiment
  • for the characterization of Xilinx Zynq-7000 devices under heavy-ion irradiation

• Radiation testing logger
  • records the bit upsets of the FPGA memories during irradiation
  • JTAG commands: readback, readback-capture and readback-verify

• Configuration memory scrubber
  • checks the embedded ECC, monitors the heartbeat, reads and writes (repairs) erroneous frames
  • User application runs a 2D error correction algorithm
  • JTAG commands: readback, readback-capture, FPGA configuration frame(s) read & write, configuration register(s) read & write
An open-source framework that provides access to the FPGA configuration memory and circuit logic via the JTAG protocol

Supports the development of FPGA reliability-aware methodologies

Open-source Project

Source code (TCL scripts, GUI, Python code for simple use cases and VHDL code for on-chip logic) are available to the github project FREtZ (FPGA Reliability Evaluation through JTAG)

Project is licensed under the GNU GPLv3
Thanks for your attention