Comprehensive Project Documentation: Open-Source Radiation Hardening Simulator

Jacob Anderson, David Nichols, Collin Lambert, Parker Allred

July 29, 2024

Contents

1 Project Overview

This project aims to develop an open-source radiation hardening simulator using xschem and NGSpice. The simulator provides a comprehensive library for simulating the effects of radiation on electronic circuits, including modules for fault injection, radiation effect simulation, and results analysis. The project is designed to be user-friendly and accessible to researchers and engineers working in the field of radiation hardening.

2 Library Structure and Core Features

2.1 Fault Injection Module

The Fault Injection Module simulates faults in circuit elements to study radiation effects.

Key Functions:

- `DefineFaultModel(type, parameters)`: Defines the fault model.
- `InjectFault(circuit)`: Injects faults into the circuit.
- `LogFault(details)`: Logs the fault details.

2.2 Radiation Effect Simulation Module

The Radiation Effect Simulation Module simulates the effects of radiation on electronic circuits.

Key Functions:

- `SimulateSET(circuit)`: Simulates Single Event Transients.
- `SimulateSEU(circuit)`: Simulates Single Event Upsets.
- `AnalyzeImpact(data)`: Analyzes the impact of radiation on the circuit.

2.3 Results Analysis Module

The Results Analysis Module analyzes and presents simulation results.

Key Functions:

- `GenerateReport(results)`: Generates a report of the simulation results.
- `PlotResults(data)`: Plots the results for visualization.
- `ComputeSER(data)`: Computes the Soft Error Rate.

3 Installation Instructions

3.1 Prerequisites

Before installing the simulator, ensure you have the following software installed:

- Homebrew (for macOS users)
- Git
- xschem
- NGSpice
- GTK+3

3.2 Step-by-Step Installation

1. Install Homebrew (macOS):

   ```
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.
      com/Homebrew/install/HEAD/install.sh)"
   ```

2. Install Dependencies:

   ```
   brew install gtk+3 cairo pango autoconf automake libtool
      pkg-config at-spi2-core
   ```

3. Clone the xschem-gaw Repository:

   ```
   git clone https://github.com/StefanSchippers/xschem-gaw.
      git
   cd xschem-gaw
   ```

4. Generate Configuration Files and Build:

   ```
   autoreconf --install
   automake --add-missing
   ./configure
   make
   sudo make install
   ```

5. Add the following to your `.bashrc` or `.zshrc` file:

   ```
   export NO_AT_BRIDGE=1
   ```

6. Source the `.bashrc` or `.zshrc` file or open a new terminal window:

   ```
   source ~/.bashrc  # or
   source ~/.zshrc
   ```

7. Run the GTK Analog Wave Viewer:

   ```
   gaw
   ```

4 Usage Examples

4.1 Example Circuit: Memory Cell

Figure 1: Example Memory Cell Circuit

Steps to Simulate:

1. Create the schematic in xschem.
2. Export the netlist as `memory_cell.spice`.
3. Run the simulation with NGSpice:

   ```
   ngspice -b memory_cell.spice -o ngspice_output.txt
   ```

4. Analyze the results using the Results Analysis Module.

4.2 Example Circuit: Operational Amplifier

Figure 2: Example Operational Amplifier Circuit

Steps to Simulate:

1. Create the schematic in xschem.
2. Export the netlist as `opamp.spice`.
3. Run the simulation with NGSpice:

```
ngspice -b opamp.spice -o ngspice_output.txt
```

4. Analyze the results using the Results Analysis Module.

5 Additional Resources

5.1 Documentation and Tutorials

- User Guide: Detailed user guide with step-by-step instructions.
- API Documentation: Comprehensive API documentation for all modules and functions.
- Tutorials: Various tutorials to help users get started with the simulator.

5.2 GitHub Repository

The source code and additional resources for the project can be found on GitHub:

- GitHub Repository: RAD-HARD

6 Conclusion

This document provides comprehensive documentation for the open-source radiation hardening simulator project. By following the instructions and utilizing the provided resources, users can effectively simulate and analyze the effects of radiation on electronic circuits.

Acknowledgment

We would like to thank Dr. Shiuh-hua Wood Chiang for his guidance and support throughout this project. This work was supported by [Funding Source].