An Automated Triple Modular Redundancy EDA Flow for Yosys

Matt Young – Supervised by Assoc. Prof. John Williams

Introduction

Safety-critical sectors require Application Specific Integrated Circuit (ASIC) designs and Field Programmable Gate Array (FPGA) gateware to be fault-tolerant. In particular, high-reliability spaceflight computer systems need to mitigate the effects of Single Event Upsets (SEUs) caused by ionising radiation. One common fault-tolerant design technique is Triple Modular Redundancy (TMR), which mitigates SEUs by triplicating key parts of the design and using voter circuits. Leveraging the open-source Yosys Electronic Design Automation (EDA) tool, in this work, I present **TaMaRa**: a novel fully automated TMR flow, implemented as a Yosys plugin.

Single Event Upsets

SEUs are caused by ionising radiation striking a CMOS transistor on an integrated circuit, and inducing a small charge which can flip bits. This is dangerous, as it can invalidate the results of important calculations, potentially causing loss of life and/or property in safety-critical scenarios.

Prior literature

In the literature, there are two approaches to automated TMR:

- **Design-level approaches** ("thinking in terms of HDL"): Treat the design as HDL *modules*, and introduce TMR by replicating these modules. Operates on HDL source code.
- **Netlist-level approaches** ("thinking in terms of circuits"): Treat the design as a *circuit* or *netlist*, which is internally represented as a graph. TMR is introduced using graph theory algorithms to *cut* the graph in a particular way and insert voters.

Design-level approaches are usually more intelligible and extensible, as they operate on HDL source code directly. However, it's difficult to account for EDA synthesis optimisations that can remove the redundancy. Whilst being less intelligible, netlist-level approaches can support many HDLs, and operate safely after optimisation.

Triple Modular Redundancy

Triple Modular Redundancy (TMR) mitigates SEUs by triplicating key parts of the design and using voter circuits to select a non-corrupted result if an SEU occurs (see Figure 1).

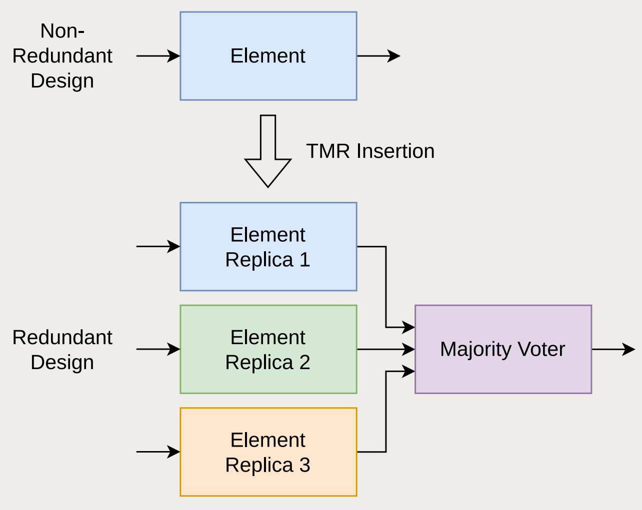

Figure 1: Diagram demonstrating TMR being inserted into an abstract design

Results: Circuits

Figure 3 shows a netlist schematic for a simple 2-bit multiplexer, and Figure 4 shows it after the application of TaMaRa TMR.

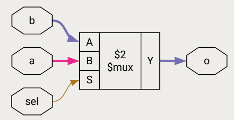

Figure 3: 2-bit multiplexer

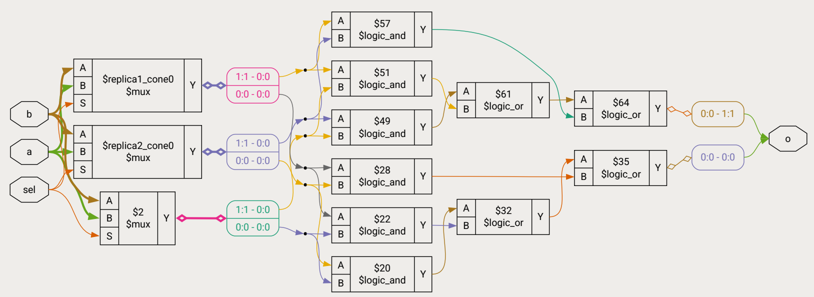

Figure 4: 2-bit multiplexer with TaMaRa TMR

TaMaRa Methodology

The **TaMaRa** algorithm (Figure 2), introduced in this work, automates the insertion of TMR at the post-synthesis netlist level.

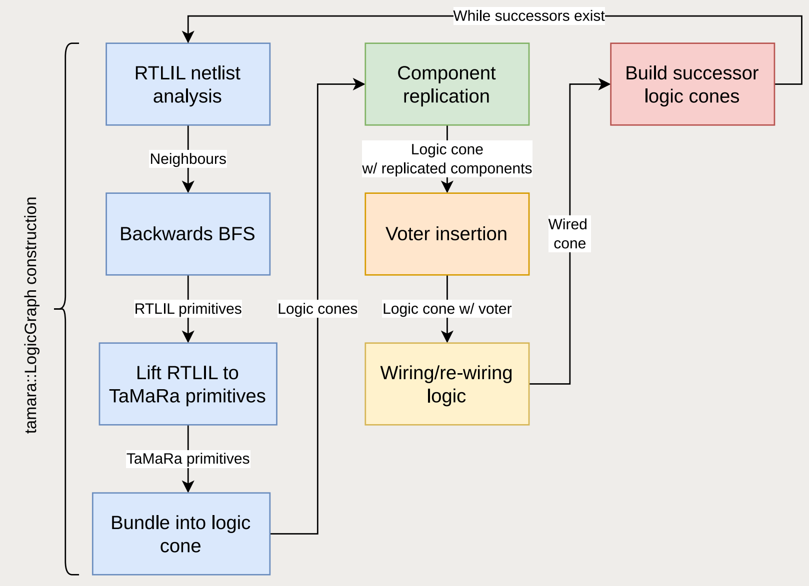

Figure 2: Description of the TaMaRa algorithm

Results: Reliability

TaMaRa demonstrates the capability of mitigating simulated SEU faults in a large-scale formally verified fault-injection campaign. When the voter is itself protected from faults (Figure 5), the algorithm performs well; but in more realistic unprotected scenarios, faults can still occur (Figure 6).

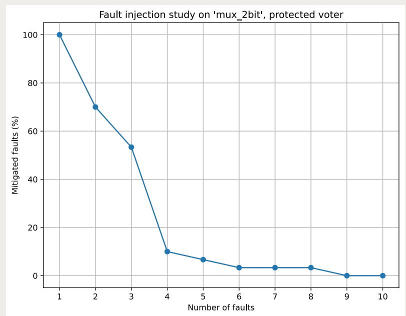

Figure 5: Protected voter

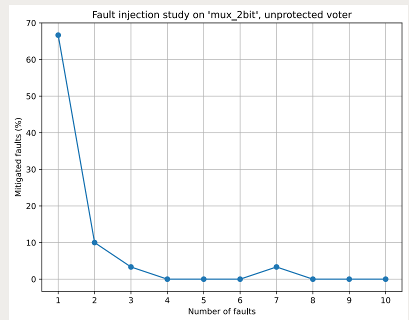

Figure 6: Unprotected voter

School of Electrical Engineering and Computer Science

Acknowledgements

N. Engelhardt, George Rennie, Emil J. Tywoniak, Krystine Sherwin, Catherine "whitequark", Jannis Harder, John Williams, Peter Sutton, and my friends and family.

The University of Queensland
Australia