Fuzzy-logic simulation approach to modelling of fault propagation in FPGAs.

Andrzej Cichocki
Outline

• Background and objectives
• The concept of a fuzzy netlist simulation
• Possible fields of application
• FLAVIUS software
• Examples
• Future plans
• Summary
Background

• Challenges for Hi-Rel FPGA design:
  – Which target FPGA is the best for my application?
  – Which FMT (or their combination) is the best to use for my application?
  – What MTTF can I expect?
  – Which part of my design is the most vulnerable?

• Challenges for Hi-Rel FPGA verification:
  – Were all my FMTs implemented correctly?
  – Weren’t they optimized away by the Synthesis Tool? Which synthesizer has better auto-FMTs?
  – Is my FT implementation functionally equivalent to regular one?
Objectives

- To develop a tool able to:
  - compare FMTs (e.g. TMR) for given application, FPGA architecture and radiation environment (D)
  - compare design susceptibility between target FPGAs (D)
  - isolate possible fault-paths that can be critical for mission success (D)
  - estimate probability of major failures (D)
  - prove that FMTs were not optimized away (V)
  - support fault-injection methods in finding most sensitive parts of the design (V)
The concept - overview

• The simulation system uses post synthesis/place and route VHDL/EDIF files and automatically replaces used target architecture primitives to their statistical models

• Statistical model of a primitive uses probabilistic logic domain instead of Boolean (values between 0 and 1)

• Probability of SEE-induced fault at primitive output level is modeled using radiation characteristics acquired from radiation tests data.

• Other factors like time delays (especially important for SETs) and meta-stability issues should also be taken into account
The concept - overview

• Converted VHDL can now be a part of a testbench and be a subject of normal simulation i.e. in ModelSim (the only difference is that it is simulated in probability domain instead of regular logic)

• Dedicated software may be used to do the simulation (performance optimization possible, because it’s structural and not behavioral)
The concept – process flow (1)

- FPGA vendor’s primitive library (functional)
- Probabilistic primitives library
- Radiation tests data (Weibull parameters)
- Standard Post P&R HDL/EDIF/SDF netlist
- Modified HDL netlist
- Radiation Environment Model (LET curves from SPENVIS/OMERE)
- Probabilistic library instance for current environment
- Radiation Environment Model (LET curves from SPENVIS/OMERE)
- Simulation in probability domain ("expected values")
- Statistical model of fault propagation for current architecture and environment
- Testbench (VCD)
The concept – proces flow (2)

IC Model Library (func./timing/rad.)

Standard Post P&R HDL/EDIF/SDF netlist

Testbench (VCD)

Radiation Environment Model (LET curves from SPENVIS/OMERE)

FLAVIUS Framework

User’s code (e.g. assertions)

MTTF estimation, Net Error Histograms, Comparison reports, etc.
Model of a primitive

• Using expected values as signal values (probability of having high logic value)

\[
E[x(t)] = \sum_i p_i x_i
\]

\[
p \in < 0, 1 >
\]

\[
x \in \{0, 1\}
\]

• Each component (gate/FF) introduces additional inversion with probability determined from radiation model (SEE rate calculated for simulation \(dt\))

\[
c(t) = a(t)b(t)(1-r(t)) + [1-a(t)b(t)]r(t)
\]
Model of a primitive

\[
P(A = 0) = 1 - p_A
\]

\[
P(A = 1) = p_A
\]

\[
P(B = 0) = 1 - p_B
\]

\[
P(B = 1) = p_B
\]

\[
p_{\text{OUT}} = P(\text{out} = 1) = p_A + p_B - p_A \times p_B
\]

\[
p_{\text{OUT}}' = p_{\text{OUT}} \times (1 - p_{\text{FLIP}}) + (1-p_{\text{OUT}}) \times p_{\text{FLIP}}
\]

Use modified Karnaugh map – event dependence (ESPRESSO possible?)
or  tree diagram & Law of total probability
Model of a primitive

\[
pQ' = (p_{SEU} - 1) \times (p_{CLR}(p_{PRE} - 1) - p_{CLR}p_{E}p_{PRE}p_{Q} + p_{CLR}p_{PRE}p_{Q}(p_{E} - 1))(p_{CLK}(p_{LCLK} - 1) + 1) - p_{CLK}p_{CLR}p_{D}p_{PRE}(p_{E} - 1)(p_{LCLK} - 1) - p_{SEU}(p_{CLR} + p_{CLR}p_{E}p_{PRE}(p_{Q} - 1) - p_{CLR}p_{PRE}(p_{E} - 1)(p_{Q} - 1)(p_{CLK}(p_{LCLK} - 1) + 1) + p_{CLK}p_{CLR}p_{PRE}(p_{D} - 1)(p_{E} - 1)(p_{LCLK} - 1) - 1)
\]

Already after MATLAB symbolic optimization!
Problems

• How to estimate pFLIP for a primitive?
  – Demanding HI/Proton tests
  – SET broadening
  – Metastability issues
  – SET susceptibility of routing resources

• How to choose appropriate simulation cycle time (dt)?
  – Simulation performance / accuracy trade-off
  – Primitive’s probabilistic timing models
Fields of application

- **Stimulus is not available, inaccurate primitives’ models**
  - Coarse identification of weak spots (most fault-contributive)
  - Coarse comparison of FMT for particular design (only hardware redundancy)

- **When only stimulus is available (+)**
  - Coverage for time redundancy (oversampling, scrubbing)
  - Comparison of FMTs for particular design and stimulus
  - Identification of the most fault-contributive nets

- **When radiation/environment models and stimulus are available (++)**
  - Estimation of MTTF for given definition of failure
  - Comparison of reliability for different FPGA targets.
FLAVIUS

Fuzzy Logic

Application for
eValuation and
Investigation of

Upset

Spreading
FLAVIUS

• Framework modules written in C++
  – Classes: Simulation, Stimulus, Primitive (virtual)

• Auxiliary software (VB.NET)
  – For conversion of vendor libraries to C++ and then compiled as objects for the linker
  – EDIF parser and converter
  – VCD parser and converter

• Current limitations
  – No support for timings, only SEUs for sequential logic
  – No support for I/O, BRAM and DSP blocks
  – Performance…
FLAVIUS

- Can be used to generate reports
- Full-custom assertions and triggers in C++
- VCD can be dumped and opened by a waveform viewer (e.g. GTKWave)
```cpp
#include <iostream>
#include "simulation.h"

int main (int argc, char** argv) {

    string dummy = "";
    string stim = "D:/SEFUW2018/sim/dut3_f.vcd";
    string netlist = "D:/SEFUW2018/sim/dut3.net";
    clock_t simstart;
    simulation_t *sim;

    sim = new simulation_t (netlist, dummy, dummy, stim, 0, 0);

    sim->vcdCreate("D:/SEFUW2018/sim/dut3_f.vcd");
    sim->vcdAddWave("a");
    sim->vcdAddWave("b");
    sim->vcdAddWave("clk");
    sim->vcdAddWave("d");

    sim->init();

    simstart = clock();
    while (!sim->run()) {
        ...
    }

cout << "\nSimulation time : " << (double)((clock() - simstart) / (CLOCKS_PER_SEC/1000)) << "ms\n";
delete sim;
return 0;
}
```
Validation of the concept

• DUTs
  – Simple designs with FMTs
  – LFSR

• Comparison between fuzzy model and statistics of fault-injection
  – Questa simulation with analog (fuzzy) signals vs. Questa with fault-injection (signal forcing)
  – FLAVIUS runs: fault-injection vs. fuzzy simulation (MT uniform distribution)
Validation of the concept

Convergence

Bit(1) after 100 CC
Examples

- DUT5_A – no TMR
- DUT5_B – TMRed flip-flops

Diagram:

- FPGA
  - A - B > C ?
  - "Very Important Switch"
- ADC
  - Value A
  - Value B
Examples – FMTs comp.

- TMR, SR = 300/bit*day

- No TMR, SR = 0.03/bit*day
Examples – MTTF

• What is the probability that the Switch will be triggered for at least one clock cycle in first 265ns? – example definition of fault.

```c
while (!sim->run(1)) {
    if (tt < 265) {
        pTmp = sim->netlist->getNetVal_f("oswitch");
        pAcc = pAcc + pTmp - pAcc * pTmp;
    }
    tt++;
}
```

• TMR: \( pF = 1.025 \times 10^{-5} \) @ SR = 1/bit*day
• No TMR: \( pF=0.9698 \) @ SR = 1/bit*day
Next steps

• ProASIC3, (RT)AX probabilistic primitives library (other vendors also welcome)

• Concept validation for real designs (w/FMTs)
  – Synplify Premier / Precison Hi-Rel welcome
  – FT-Unshades2 to be used for fault-injection statistics to fuzzy model response comparison
  – Comparison to results from Questa SLEC

• Implementation of time delays and SET support
  – Possible cooperation with Prof. Sterpone’s team

• Integration of radiation response models
  – Generation of database for OMERE and report parser

• Optimizations
  – Design partitioning and parallel simulation (GPU)
Thank you for your attention.

andrzej.cichocki@cbk.waw.pl