Accurate Modeling of Fault Impact in Arithmetic Circuits
Pierre Guilloux, Arnaud Tisserand

To cite this version:

HAL Id: hal-01404772
https://inria.hal.science/hal-01404772
Submitted on 29 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
1. Objective: Design of Robust Arithmetic Circuits

Study links between fault detection/tolerant solutions and
- types of operations (+, ×, x, x, ÷, √, sin, cos, log, . . .)
- computation algorithms and representations of numbers
- optimizations at architecture and circuit levels
- mathematical error due to faults and rounding modes

Applications: signal/image processing, embedded computing, protections for cryptographic devices, . . .

2. Intensive Logical Fault Simulations: Emulation on FPGA

Cells from ASIC library
Injection controller
x y =
\[ \begin{cases} 
  x \\
  \bar{x} \\
  0 \\
  1 
\end{cases} \]
Logical fault
Injector
Server . . . Level
Netlist
Instrumentation
(add injectors)
Instrumented netlist
FPGA tools
Bitstream
FPGA tools

3. Hardware Platform

Server
LAN (switch)
GbE
Links
Power supply
Gbps links
FPGA 1
FPGA 2
FPGA N

4. Emulation Engine(s) in FPGA

Reference block
Error distribution module
Error analysis module
Injection controller
Main CTRL
Emulation engine

5. Tools Flow

Preparation step
Emulation step
Results analysis

Possible locations for logical fault injection:
- manual on selected internal nets (for debug)
- random on internal nets
- random over circuit area (based on cells area)

6. Example: Multiplier with Residue Code Detection

16×16-bit integer multiplier, moduli \( m \in \{3, 7, 15\} \), with
\[ (a \times b) \mod m = ((a \mod m) \times (b \mod m)) \mod m \]

<table>
<thead>
<tr>
<th>Step</th>
<th>Duration</th>
<th>Area</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impl. ASIC orig. mult.</td>
<td>25 s</td>
<td>1164</td>
<td>–</td>
</tr>
<tr>
<td>Impl. FPGA orig. mult.</td>
<td>43 s</td>
<td>444</td>
<td>9.4 ns</td>
</tr>
<tr>
<td>Impl. FPGA instrum. mult.</td>
<td>48 s</td>
<td>453</td>
<td>10.1 ns</td>
</tr>
<tr>
<td>Impl. FPGA emul. engine</td>
<td>389 s</td>
<td>1714 LUT + 1 BRAM</td>
<td>11.0 ns</td>
</tr>
<tr>
<td>Boot petalinux Zedboard</td>
<td>15 s</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Emulation 1 fault &amp; 1 EE</td>
<td>94.5 s</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Comparison to a CABA simulation in software (Vivado 2014.4):
- 6880 speedup for 1 emulation engine

Error distribution for exhaustive tests (2^{32} inputs) and various single faults (1 color = 1 fault):

7. Future Works

<table>
<thead>
<tr>
<th>Residue code implementation results:</th>
</tr>
</thead>
<tbody>
<tr>
<td>modulo</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

| 100 EEs @ 100 MHz | 100 \cdot 10^6 \times 100 \approx 2^{43} | 100 \cdot 10^4 \times 100 \approx 2^{45} | \approx 2^{49} |

References:
Arithmetic operators library from R. Zimmermann (http://www.ise.ee.ethz.ch/zimmi/arith_lib)
P. Guilloux & A. Tisserand, Compas 2016, French Conf. on Parallelism, Architecture and System