A study on the reliability improvement factor of fault tolerant mechanisms
Jongwhoa Na, Dongwoo Lee

To cite this version:
Jongwhoa Na, Dongwoo Lee. A study on the reliability improvement factor of fault tolerant mechanisms. Safecomp 2013 FastAbstract, Sep 2013, Toulouse, France. pp.NC. hal-00926549

HAL Id: hal-00926549
https://hal.archives-ouvertes.fr/hal-00926549
Submitted on 9 Jan 2014

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.
A study on the reliability improvement factor of fault tolerant mechanisms

Jongwoha Na, Dongwoo Lee
Department of Avionics and Electronics Engineering,
Korea Aerospace University,
Republic of Korea
{jwna, dongwoo1}@kau.ac.kr

Abstract—We present a study on the reliability improvement factor (RIF) to quantify the reliability of the various fault tolerant mechanisms at the system level. First, we find the system level failure rate using co-simulation models and statistical fault injection (StFi). We built co-simulation targets using SystemC simulation models of baseline single-core ARM7, dual-modular and triple-modular redundant ARM7 processors and Mibench embedded benchmark SW. Since the number of experiments in StFi is large, we utilized simulation kernel-modified simulated fault injection tool. Next, we calculated the RIF using the failure probability functions of the co-simulation targets. In this way, we were able to compare the reliability improvement of the fault tolerant mechanism at the system level.

Keywords; statistical fault injection, reliability improvement factor, fault-tolerant processor, fault-tolerant mechanism

I. INTRODUCTION

In the safety-critical embedded systems (SCES) in aircrafts and automobiles, fault tolerant processors (FTP) became a major components. FTPs increase the reliability of the target using various types of redundancies. However, these redundancies also increase the cost of the target considerably. In order to manage the cost increase in SCES, we need a reliability index to quantify these redundancies. We may use MTTF as a reliability index for the target, which has components with sufficient usage history. However, because of the fast developing speed in VLSI/SoC technology, it is difficult to keep the usage history of the components of the modern SCES. This calls for a reliability index without usage history.

In this paper, we explain the application of the reliability improvement factor (RIF) as a reliability index of the effectiveness of the FT mechanism in the FTP. RIF is defined as the ratio of the probability of failure, F(t), of the non-redundant system to that of the redundant system [1,2]. For example, using ARM7 processor as a baseline processor, we may quantify the effectiveness of the TMR mechanism over DMR by finding the $RIF_{TMR}$ of TMR ARM7 over baseline ARM7 and the $RIF_{DMR}$ of DMR ARM7 over the same baseline.

We can calculate the F(t) of RIF by performing the fault injection experiments and finding the failure rate of the FTP and the baseline target. In order to make the failure rate legitimate, we use statistical fault injection (StFi) with confidence level and reasonable targets which can be a real SCES at the final stage or co-simulation target at the early stage of the development life cycle.

We report the case study of RIF for reliability index using StFi and co-simulation target. We built a co-simulation model of a SystemC hardware simulation model of baseline ARM7, DMR ARM7, TMR ARM7 processors and the cross-compiled executable files of the MiBench embedded benchmark suits [3]. For the statistical fault injection experiment, we calculate the required number of fault injections at a 95% confidence level for the given fault models and the SUT [4]. Because of the complexity of the SUT, the required number of fault injections is very large. Thus, the efficiency of the injection tool is important. In this regard, we use a novel simulated fault injection environment that uses a modified simulation kernel instead of saboteur or mutation technique. A detailed explanation of the kernel-modified simulated fault injection is explained elsewhere [5].

II. FAULT TOLERANT PROCESSORS

For hardware, we designed a SystemC simulation model of the ARM7 processor that could execute about 40 instructions from the ARM7 architecture, as shown in Fig. 1.

In order to make the cases of the qualitative comparisons of various FT mechanisms, we designed a SystemC simulation model for the DMR and TMR ARM7 processors. In the case of the DMR ARM7, we duplicated the data path with two ARM processors and added a simple fault recovery controller that could detect faults at the pipeline stages. In the design of the TMR ARM7 in Fig. 2, we implemented the micro-architectural redundancy by triplicating each module and adding a voter. The details of the DMR and TMR architectures can be found in many other studies [4].
III. STATISTICAL FAULT INJECTION EXPERIMENTS

We performed fault injection experiments using the co-simulation models using the three hardware models (baseline single ARM7, TMR ARM7, and DMR ARM7 processors) and the GSM code from a Mibench embedded benchmark suit, and the four fault models (permanent and transient stuck-at-1/0). For each of the experimental setups, we setup statistical fault injection experiments by calculating the required number of injections for the given confidence level. The results of fault injections are summarized in Table 1. In the case of baseline ARM7, the injection result should be one of the not-active, benign, or silent data corruption (SDC) state. In the case of DMR and TMR ARM7, we have two more states: recovered and detected unrecoverable error (DUE).

![Fig. 3. DMR and TMR reliability improvement factors for the transient and permanent fault models](image)

### Table 1: The results of statistical fault injection campaign on co-simulation models and fault models

<table>
<thead>
<tr>
<th>H/W</th>
<th>S/W</th>
<th>Fault type &amp; model</th>
<th>Transient</th>
<th>Permanent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>non-active</td>
<td>stuck-at-1</td>
<td>stuck-at-0</td>
</tr>
<tr>
<td>Base-</td>
<td>Non Active</td>
<td>28,468</td>
<td>71,191</td>
<td>8</td>
</tr>
<tr>
<td>line</td>
<td>Benign</td>
<td>59,035</td>
<td>22,522</td>
<td>12,669</td>
</tr>
<tr>
<td>ARM</td>
<td>Silent Data Corruption</td>
<td>12,497</td>
<td>6,287</td>
<td>87,323</td>
</tr>
<tr>
<td>DMR</td>
<td>Non Active</td>
<td>29,046</td>
<td>71,314</td>
<td>125</td>
</tr>
<tr>
<td>ARM</td>
<td>Benign</td>
<td>33,119</td>
<td>7,928</td>
<td>16,201</td>
</tr>
<tr>
<td>TMR</td>
<td>Recovery</td>
<td>35,217</td>
<td>17,206</td>
<td>44,705</td>
</tr>
<tr>
<td>ARM</td>
<td>Silent Data Corruption</td>
<td>519</td>
<td>769</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>Detected Unrecoverable Error</td>
<td>2,099</td>
<td>2,285</td>
<td>38,894</td>
</tr>
<tr>
<td></td>
<td>non-active</td>
<td>47,554</td>
<td>74,237</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>Benign</td>
<td>2,117</td>
<td>1,279</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Recovery</td>
<td>89,882</td>
<td>34,260</td>
<td>19,138</td>
</tr>
<tr>
<td></td>
<td>Silent Data Corruption</td>
<td>441</td>
<td>224</td>
<td>511</td>
</tr>
<tr>
<td></td>
<td>Detected Unrecoverable Error</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Using Table 1, we calculate the failure rate by dividing the sum of the SDC and DUE failure rates by the total number of fault injections for the three types of processors. Using these failure rates and assuming steady operating condition, we are able to calculate the reliability or failure distribution functions for each case of the baseline ARM, DMR ARM, and TMR ARM co-simulation targets. With the failure function, we can calculate the RIF of the DMR and TMR mechanism as follows:

$$\text{RIF}_{\text{FTM}} = 1 - R_{\text{base,1}}(t)/(1 - R_{\text{TMR}}(t))$$

We presents the RIF$_{\text{TMR}}$ and RIF$_{\text{FTM}}$ over baseline ARM7 processor in Fig. 3. Using the graph, we can compare the effectiveness of the TMR mechanism over DMR mechanism for given time in a quantitative manner. Initially, we can find the effectiveness of the TMR mechanism to be 50–70 times higher than that of the DMR mechanism. Also, we can find that the improvement of RIF$_{\text{TMR}}$ over RIF$_{\text{DMR}}$ decreases over time.

IV. CONCLUSION

We have reported on the reliability improvement factor (RIF) of the DMR and TMR mechanisms. Instead of using the static failure rates from the reliability block diagram, we utilized the dynamic failure rates using the co-simulation targets of SystemC hardware and Mibench benchmark software so that the RIF becomes more practical. The experimental results suggested that the TMR mechanism is initially more resilient than DMR. As such, we may compare the reliability or the cost-effectiveness of the FTM at the system level of various types of redundancy mechanisms.

Using these simulation study as a basis, we are planning to extend the experiments using other benchmark software and other FT mechanisms to investigate the applicability of the RIF as a quantitative reliability index of the fault tolerant mechanism for a group of embedded systems.

REFERENCES