Low-cost fault tolerant targeting FPGA devices

Kostas Siozios and Dimitrios Soudris
School of ECE, National Technical University of Athens, Greece

Abstract. Technology scaling in conjunction to the trend towards higher performance introduce mentionable reliability degradation. In this paper, a novel software-supported framework targeting to provide sufficient fault masking against to upsets due to reliability issues, without the excessive mitigation cost of similar approaches, is introduced. Experimental results with a number of industrial oriented DSP kernels prove the effectiveness of our solution, since we achieve considerable delay and power improvements for comparable fault masking.

1 Introduction

More than at any other time, global economics favor programmable chips over costly ASICs and ASSPs, since the costs and risks associated with application-specific devices can only be justified for a short list of ultra-high volume commodity products. However, designs mapped onto FPGAs usually suffer from increased power densities, as compared to alternative ASIC implementations, while this problem becomes far more urgent since there is a trend almost to double power density every three years [11]. Among the consequences imposed by power and thermal stress is the increased number of upsets due to reliability degradation [3].

Up to now, the only known commercial tool targeting to provide fault masking at FPGAs, is Xilinx Triple Modular Redundancy (TMR) [1]. The principle idea of TMR is the usage of hardware redundancy to mask any single failure by voting on the result of three identical copies of the circuit. Even though this approach exhibits mentionable fault masking, it imposes considerable mitigation cost, which rarely could be though as affordable for general-purpose systems (e.g. consumer products).

In order to alleviate the excessive mitigation cost, a number of alternative solutions have been proposed [13] [14] [15]. Even though these approaches introduce lower performance degradation compared to Xilinx TMR, they are applicable solely to mask Single Event Upsets (SEUs) due to cosmic radiation. On the other hand, throughout this paper we propose a framework to provide fault coverage reliability degradation. More specifically, initially by clustering parts of the design based on their sensitivity to failure and then applying selectively redundancy only to suspicious parts, it is possible to provide a spectrum of solutions that trade-off the desired level of fault masking and the consequence mitigation cost.

The rest of the paper is organized as follows: Section II gives the motivation example. The proposed methodology is described in section IV, while experi-
mental results are provided in section V. Finally, conclusions are summarized in section VI.

2 Motivation Example

This section highlights the limitation of Xilinx’s TMR to provide sufficient fault masking against to upsets due to reliability degradation. For demonstration purposes we study the JPEG application [4] mapped onto a Stratix-based FPGA. Note that the selected device does not incorporate any dedicated fault tolerant mechanism. Figs. 1(a) and 1(b) plot (in normalized manner) the thermal maps for JPEG application, when no TMR and uniform TMR (similar to [1]) is assumed, respectively. Thermal analysis was performed with Hotspot-5.0 tool [12] at slice level.

Based on thermal map depicted in Fig. 1(a), the reported temperatures range between 75.2°C and 83.5°C (average temperature 76.4°C). Hence, the reliability degradation varies considerable even between adjacent spatial locations of the device. Note that existing way for protecting designs against to SEUs, depicted in Fig. 1(b), results to additional thermal stress because it introduces redundancy even at regions with almost negligible probability of failure due to aging phenomena. More specifically, the Xilinx TMR leads to average temperature of 79.8°C, while the number of temperature hotspots are also increased.

![Figure 1](image_url)

These conclusions indicate that whenever fault tolerance ignores temperature variation, it is not suitable for protecting designs against to reliability degradation. Hence, the problem we tackle in this paper is summarized as follows: Given the application’s HDL, a general-purpose FPGA and the maximum affordable performance degradation, find an application implementation that corresponds to the maximum possible fault coverage against to reliability degradation with respect to system’s specifications.
3 Proposed Framework

This section introduces the proposed framework for performing application mapping with the maximum affordable (in terms of system specifications) fault coverage against aging phenomena. This methodology, depicted in Fig. 2, is software supported by a number of new and existing CAD tools, which are public available through [6].

![Fig. 2. Proposed methodology for supporting the low-cost fault tolerance.](image)

After application’s synthesis with Quartus Framework, the derived netlist is annotated with TMR (e.g. similar to [1]), resulting to the maximum possible fault coverage. The output from this step protects both logic and routing infrastructure against upsets, but it imposes the maximum mitigation cost. Then, application is P&R with our proposed reliability-aware framework. Upcoming subsections provide additional details about the employed algorithms. Having the application implementation onto the target FPGA, it is possible to compute power consumption per slice, as well as a thermal profile, similar to the one depicted in Fig. 1. Since application is implemented with the maximum possible fault coverage, the derived map of temperature values corresponds to the worst-case scenario.

Then, a number of upsets are injected into the application’s configuration file in order to perform error propagation analysis. Based on this analysis it is possible to retrieve the hardware sensitivity in failures (e.g. classify slices based on their high/low temperature values, and hence with/without increased probability of failure). For the sake of completeness, their spatial and temporal distributions follow the studied TDDB reliability degradation model [7]. Note that apart from this aging degradation phenomenon, any other approach can also be studied (e.g. NBTI, EMI, etc). This task is software supported by our tool, named Fault-Inject [6].

Based on the conclusions from this error propagation analysis, our framework eliminates gradually the aggressiveness of redundancy from slices which are not expected to suffer from upsets due to aging phenomena. Since redundancy removal imposes performance enhancement, this procedure is repeated until the retrieved solution meets application’s specifications (e.g. delay, power consumption, etc). The output of this procedure is an application implementation, where only the suspicious for failure hardware resources are protected with redundancy.
Since this imposes that a number of CLBs are not sufficiently utilized, we have developed a new tool, named “Compact Design” that clusters together functionalities from different (non-fully utilized) CLBs, having as goal the minimization of switching capacitance (due to its correlation with the thermal stress).

Finally, there is a refinement step, where we perform partial P&R under reliability constraints. Rather than similar frameworks that optimize either the application’s delay (e.g. [5]), or its power consumption (e.g. [10]), throughout this refinement we are primarily interested to further improve parameters that affect reliability degradation.

The derived application implementation is evaluated against to the system’s specifications, whereas in case these specifications are not meet, there is a feedback loop for additional improvements. Next subsections describe in more detail the new algorithms that support the proposed methodology.

### 3.1 Relax the Aggressiveness of Fault Tolerance

The first step in our proposed reliability-aware framework deals with redundancy removal from application’s functionalities mapped onto resources with reduced probability of failure. For this purpose, application’s functionalities are sorted in descending order based on their probability of failure, mentioned as $PF$. Then, starting from slices with the minimum $PF$ values, we gradually alleviate the aggressiveness of employed fault tolerance. In order to software-support this task, a new open source tool has been developed [6], while the pseudo-code for this tool is depicted at Algorithm 1.

```
Require: P ← utilized slices
Require: M ← thermal map
Require: S ← delay and power specifications
1: Function Relax_TMR (P, M, S)
2:     for (each slice ∈ P) do
3:         G(slice) ← failure_probability(M);
4:     end for
5:     PF ← sort_descending(P, G(P));
6:     while (S NOT satisfied) do
7:         if (P NOT belong to control path) then
8:             for (G(P) ∈ (min{PF}, max{PF})) do
9:                 select a relaxed TMR scenario;
10:                evaluate(new power, new delay);
11:            if (S satisfied) then
12:                break;
13:            end if
14:         end for
15:     end if
16:     end while
17: end Function
```

**Algorithm 1:** Pseudo-code for relaxing the aggressiveness of fault tolerance.
3.2 Compact Design
Since redundancy removal imposes that some of the CLBs are non-fully utilized, our framework aims to derive next a more compact design. For this purpose, application’s functionalities mapped onto non-fully utilized CLBs are candidate to be re-clustered, having as goal to minimize the inter-cluster connections, especially those with increased switching capacitance. For this purpose, a new software tool that is based on a hill climbing algorithm (depicted in Algorithm 2), has been developed. The re-clustering procedure stops either when algorithm reaches the maximum capacity for all the CLBs, or it utilizes all the available (fabricated) input/output pins. We have to notice that the functionality performed with this tool does not correspond to clustering algorithms found in relevant approaches (e.g. [5], [10]), since it focuses on alleviating the consequences posed by reliability degradation.

Require: $U \leftarrow$ LEs utilized by the design
Require: $S \leftarrow$ delay and power specifications

1: Function Compact_Design ($U$, $S$)
2: Clustered_LE $\leftarrow$ NULL;
3: while (Clustered_LE NOT NULL) do
4:  if (Clustered_LE $\in$ non-fully utilized CLB) then
5:       $J \leftarrow$ Max{$Attr(U, \text{MinTemp}(U))$};
6:       evaluate(new power, new delay);
7:       if ($J$ satisfies $S$) then
8:         $U \leftarrow U \cup J$;
9:         Clustered_LE $\leftarrow$ Clustered_LE - $J$;
10:       else
11:         RestoreLastLegalState($J$);
12:       end if
13:   end if
14: end while
15: end Function

Algorithm 2: Pseudo-code for compact design.

The selection of candidate LE $J$ to be clustered to CLB $C$ is based on an attraction function ($Attr(J)$), which is defined as follows:

$$Attr(J) = \left( \alpha \times \text{Crit}(J) \right) + \left( 1 - \alpha \right) \times \left( \beta \times \frac{\sum_{i=1}^{\text{networks}} \left( \text{Act}(i) \right)}{\text{Act}_{\text{Avg}}} + (1 - \beta) \times \frac{\sum_{i=1}^{\text{networks}} \left( Q(i) \right)}{\text{networks}} \right)$$  \hspace{1cm} (1)

where $Crit(J)$ denotes how close LE $J$ is to being on the critical path, whereas $Q(i)$ corresponds to the number of networks between the LE $J$ and those LEs already clustered in CLB $C$. $Act(i)$ gives the switching activity of network $i$, whereas $Act_{\text{Avg}}$ is the average switching activity of all the networks in the design. Finally, weights $\alpha$ and $\beta$ tune the importance of either improving application's
delay or reliability, respectively. Throughout this paper we found that optimal results are retrieved with $\alpha = 0.6$ and $\beta = 0.4$.

### 3.3 Reliability-aware Placement and Routing

The last tool in our framework deals with application’s P&R under reliability constraints. The employed algorithms are based on VPR tool [5]; however they have been extensively modified in order to take into account parameters that affect reliability degradation. More specifically, rather than similar approaches aiming either in delay [5], or power reduction [10], our solution aims also to alleviate the thermal stress. The employed cost function during simulated-annealing placement follows:

$$\text{Cost} = (\gamma \times C_{\text{reliability}}) + (1 - \gamma) \times \left[ \delta \times C_{\text{wire}} + (1 - \delta) \times C_{\text{timing}} \right]$$  \hspace{1cm} (2)

where

$$C_{\text{reliability}} = \sum_{i=1}^{\text{Nets}} \left[ z(i) \times (bb_x(i) + bb_y(i)) \right] \times \text{Act}(i)$$  \hspace{1cm} (3)

$$C_{\text{wire}} = \sum_{i=1}^{\text{Nets}} \left[ z(i) \times (bb_x(i) + bb_y(i)) \right]$$  \hspace{1cm} (4)

$$C_{\text{timing}} = \sum_{\forall i,j \in \text{netlist}} \left[ \text{Crit}(i,j)^{\text{const}} \times \text{Delay}(i,j) \right]$$  \hspace{1cm} (5)

Parameters $\gamma$ and $\delta$ define the relative importance of weighting factors (reliability, wire-length and timing). Based on our exhaustive exploration with a representative number of benchmarks [4], we found that optimal balance between thermal stress and mitigation cost is achieved when $\gamma = 0.6$ and $\delta = 0.4$. $\text{Crit}(i,j)$ indicates how close to the critical path the connection is [8], $\text{Delay}(i,j)$ gives a delay estimation from source $i$ to sink $j$, whereas $\text{const}$ is a constant. $\text{Nets}$ parameter corresponds to the total number of application’s networks, while $q(i)$ is used to better estimate wire-length by scaling the bounding boxes for networks with more than 3 terminals [8]. Finally, factors $bb_x(i)$ and $bb_y(i)$ denote the $x$ and $y$ dimensions, respectively, of the bounding box of network $i$.

Similar to placement, our router is a negotiated reliability-congestion-delay algorithm based on PathFinder [9]. Similar to placement task, the cost function at our instantiations of PathFinder router takes also into account reliability-aware parameters, apart from the delay [5] [6] [8] and power [10] optimizations found in relevant approaches.

Next, we describe the employed cost function for evaluating the efficiency of a routing path from source $i$ to sink $j$: 
Cost = \text{Crit}(i, j) \times \text{Delay}(n) + \\
\left(1 - \text{Crit}(i, j)\right) \times \left[\text{Act}(i) \times \text{cap}(n) + \\
\left(1 - \text{Act}(i)\right) \times \text{bc}(n) \times \text{hc}(n) \times \text{pc}(n)\right]
\tag{6}

where \text{Delay}(n) is the Elmore delay of node \( n \) \[9\], while \( \text{bc}(n) \), \( \text{hc}(n) \) and \( \text{pc}(n) \) gives the base cost, the historical cost and the present congestion of node \( n \), respectively.

4 Experimental Results

This section provides a number of experimental results that prove the efficiency of proposed methodology. For evaluation purposes, we employ a number of industrial oriented kernels \[4\], whereas the target device is an Altera Stratix-based FPGA without dedicated fault tolerant mechanism. The number of injected upsets ranges between 3% and 5% of the size of configuration file, while for a given resource, the temporal distribution between consecutive upsets follows TDDB model \[7\]. Note that this claim does not affect the generality of the proposed solution.

Fig. 3 evaluates the efficiency of proposed framework to mask upsets, as we gradually alleviate the aggressiveness of redundancy. Horizontal axis corresponds to the percentage of masked over the injected upsets, whereas the left and right vertical axes give the maximum operation frequency and power consumption, respectively. For demonstration purposes, both axes are plotted in normalized manner over the corresponding maximum values among the alternative solutions. For the scopes of this analysis, JPEG application was mapped with two different scenarios: a timing-aware \[5\] and the proposed reliability-aware frameworks, whereas as reference we employ two border solutions: without fault tolerance (marked as “No TMR”) and with uniform insertion of redundancy (mentioned as “Full TMR”) \[1\].

A number of conclusions might be derived from Fig. 3. Among others, uniform insertion of redundancy, as it is applied with Xilinx TMR \[1\], leads to an excessive mitigation cost, which usually is not affordable for consumer products. More specifically, “Uniform TMR” solution imposes a penalty in delay compared to initial application implementation (i.e. without fault masking) ranging between 16% and 25%, depending on the selected framework. Similarly, the penalty in term of power consumption for this instantiation of TMR ranges from 39% (for our proposed reliability-aware framework) up to 78% (regarding the conventional timing-aware P&R).

On the other hand, our proposed methodology provides a spectrum of solutions that balance the desired fault masking with the consequence mitigation cost. For instance, assuming that a system can afford reduction at maximum operation frequency up to 5%, then a number of alternative implementations are feasible, each of which corresponds to different efficiency in fault masking. More
specifically, in case we employ our proposed reliability-aware framework, then it is feasible to achieve 68% fault coverage (this solution is marked with a green color circle in Fig. 3). Otherwise, this level of fault coverage with the usage of a conventional timing-aware toolflow [5], imposes an additional delay about 8%. Note that apart from performance improvement, our proposed reliability-aware framework achieves also power savings as compared to the existing timing-aware P&R [2]. More specifically, regarding the studied scenario of fault coverage, our solution leads to additional power savings 13%. Furthermore, in case we apply the timing-aware framework and the maximum affordable performance degradation is up to 5%, then our framework reports that only 24% of the injected upsets could be masked (blue color circle in Fig. 3).

Next, we evaluate the delay and power penalties, whenever benchmarks are implemented with three alternative frameworks. More specifically, we provide results about a timing-aware P&R [5], a power-aware framework [10], as well as the proposed reliability-aware solution. For each of these flows, we study three alternative fault tolerant scenarios: (i) without TMR, (ii) with the proposed "Compact TMR", and (iii) uniform insertion of TMR (similar to [1]). Throughout this analysis we assume, without affecting the generality of proposed methodology, that the selected level of fault coverage (horizontal axis in
Fig. 3) is set to 90% of the injected upsets. The results from this analysis are summarized in Tables 1 and 2, respectively.

### Table 1. Evaluation in term of maximum operation frequency (MHz).

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Timing-aware P&amp;R</th>
<th>Power-aware P&amp;R</th>
<th>Reliability-aware P&amp;R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Compact Uniform</td>
<td>No Compact Uniform</td>
<td>No Compact Uniform</td>
</tr>
<tr>
<td></td>
<td>TMR</td>
<td>TMR</td>
<td>TMR</td>
</tr>
<tr>
<td>jpeg</td>
<td>73.36</td>
<td>61.37</td>
<td>60.68</td>
</tr>
<tr>
<td>flp_risc8</td>
<td>49.06</td>
<td>43.17</td>
<td>35.15</td>
</tr>
<tr>
<td>aes_core</td>
<td>108.67</td>
<td>98.89</td>
<td>89.81</td>
</tr>
<tr>
<td>des_des3perf</td>
<td>90.24</td>
<td>81.21</td>
<td>69.73</td>
</tr>
<tr>
<td>huffman_video</td>
<td>228.28</td>
<td>192.83</td>
<td>110.38</td>
</tr>
<tr>
<td>oc_mem_ctrl</td>
<td>86.92</td>
<td>73.89</td>
<td>76.80</td>
</tr>
<tr>
<td>oc_pips</td>
<td>24.3</td>
<td>21.63</td>
<td>20.08</td>
</tr>
<tr>
<td>oc_mips</td>
<td>91.8</td>
<td>74.36</td>
<td>80.24</td>
</tr>
<tr>
<td>oc_correlator</td>
<td>51.02</td>
<td>39.90</td>
<td>43.18</td>
</tr>
<tr>
<td>fn_receiver</td>
<td>42.5</td>
<td>32.06</td>
<td>35.19</td>
</tr>
<tr>
<td>Average:</td>
<td>78.02</td>
<td>65.63</td>
<td>62.52</td>
</tr>
<tr>
<td>Overhead:</td>
<td>- 18.87% 24.78% - 18.13% 32.00% - 16.77% 32.92%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. Evaluation in term of power consumption (mWatt).

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Timing-aware P&amp;R</th>
<th>Power-aware P&amp;R</th>
<th>Reliability-aware P&amp;R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Compact Uniform</td>
<td>No Compact Uniform</td>
<td>No Compact Uniform</td>
</tr>
<tr>
<td></td>
<td>TMR</td>
<td>TMR</td>
<td>TMR</td>
</tr>
<tr>
<td>jpeg</td>
<td>190.00</td>
<td>283.10</td>
<td>507.30</td>
</tr>
<tr>
<td>flp_risc8</td>
<td>98.00</td>
<td>141.12</td>
<td>263.62</td>
</tr>
<tr>
<td>aes_core</td>
<td>170.00</td>
<td>181.61</td>
<td>335.28</td>
</tr>
<tr>
<td>des_des3perf</td>
<td>47.00</td>
<td>73.08</td>
<td>116.67</td>
</tr>
<tr>
<td>huffman_video</td>
<td>218.00</td>
<td>368.42</td>
<td>647.46</td>
</tr>
<tr>
<td>oc_mem_ctrl</td>
<td>193.00</td>
<td>306.87</td>
<td>482.50</td>
</tr>
<tr>
<td>oc_pips</td>
<td>56.00</td>
<td>74.48</td>
<td>166.88</td>
</tr>
<tr>
<td>oc_mips</td>
<td>229.00</td>
<td>384.72</td>
<td>638.91</td>
</tr>
<tr>
<td>oc_correlator</td>
<td>119.00</td>
<td>176.12</td>
<td>354.62</td>
</tr>
<tr>
<td>fn_receiver</td>
<td>95.00</td>
<td>137.75</td>
<td>264.10</td>
</tr>
<tr>
<td>Average:</td>
<td>177.20</td>
<td>278.73</td>
<td>482.73</td>
</tr>
<tr>
<td>Overhead:</td>
<td>- 57% 172% - 49% 161% - 56% 166%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total penalty:</td>
<td>- 57% 172% -14.02% 28.43% 124% -6.78% 45.37% 148%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Based on the results summarized in Table 1, uniform insertion of redundancy leads to the maximum performance degradation among the studied solutions. More specifically, this approach imposes an average delay overhead, as compared to application implementation without fault tolerance (“No TMR”), ranging from 25% up to 33%, depending on the selected implementation flow. On the other hand, the proposed “Compact TMR” flow introduce significant lower average delay overhead, which ranges between 17% (reliability-aware) and 19% (timing-aware).

Apart from the performance improvement, the selectively insertion of TMR results also to considerable power savings. Based on Table 2, the power overheads imposed by the uniform insertion of TMR range between 161% and 172%,
as compared to initial application implementation (without TMR). On the other hand, the power overhead whenever only critical for failure resources are triplicated (proposed solution), ranges from 49% up to 57%, respectively. Moreover, we have to notice that power savings reported in this table are complementary to those already depicted about maximum operation frequency.

5 Conclusion

A novel framework for supporting efficient application mapping under reliability constraints, was introduced. Rather than similar approaches that protect the entire design, our solution provides a trade-off between the desired fault coverage and the consequence delay and power overheads. Experimental results proven the efficiency of introduced framework, since it outperforms similar solutions for comparable fault masking.

References