Exploiting Free LUT Entries to Mitigate Soft Errors in SRAM-based FPGAs

Keheng Huang†‡, Yu Hu†, and Xiaowei Li†
†State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
‡Graduate University of Chinese Academy of Sciences
{huangkeheng, huyu, lxw}@ict.ac.cn

Gengxin Hua†‡, Hongjin Liu†‡, Bo Liu†‡
†Beijing Institute of Control Engineering,
‡National Laboratory of Space Intelligent Control

Abstract—As the feature size of FPGA shrinks to nanometers, SRAM-based FPGAs are more vulnerable to soft errors. During logic synthesis, reliability of the design can be improved by introducing logic masking effect. In this work, we observe that there are a lot of not-fully occupied look-up tables (LUTs) after logic synthesis. Hence, we propose a functional equivalent class based soft error mitigation scheme to exploit free LUT entries in the circuit. The proposed technique replaces not-fully-occupied LUTs with corresponding functional equivalent classes, which can improve the reliability while preserve the functionality of the design. Experimental results show that, compared with the baseline ABC mapper, the proposed technique can reduce the soft error rate by 21%, and the critical-path delay increase is only 4.25%.

Index Terms—functional equivalent class, free LUT entries, not fully-occupied LUTs, soft error mitigation, logic synthesis, FPGAs.

I. INTRODUCTION

Field programmable gate arrays (FPGAs) provide an attractive design platform due to their short design cycle and low development cost. With exponential growth in performance and capacity, SRAM-based FPGAs are widely used in many application domains such as telecommunication and embedded applications. On the other hand, due to their vast storage structures, SRAM-based FPGAs are very vulnerable to single event upsets (SEUs) induced by high-energy particles[1], [2]. An SEU in a configuration bit (CB) may alter the implemented design, and then provoke a soft error that manifests as a permanent fault until the affected bit is re-written. With CMOS technology continuously scaling, soft errors increasingly become an important concern in SRAM-based FPGAs.

Soft error mitigation can be conducted at different design stages, such as logic synthesis, technology mapping, clustering, placement and routing. In this paper, we focus on the stage of logic synthesis. Related reliability-oriented logic synthesis work [5], [6], [7], [8] can be classified into two categories: (1) the work utilizes Boolean flexibility of the design and (2) the work exploits free logic resources in the circuit. As functionality of the design can be implemented in different ways, the methods [5], [6], [7] in the first category choose the implementation that has high logic masking effect to improve reliability. On the other hand, the methods in the second category utilize free logic resources, e.g. free LUTs and LUT outputs [8], to introduce logic masking effect, which can improve the reliability of the design.

In this work, we observe that for all benchmark circuits, there are a lot of free LUT entries in not fully-occupied LUTs. Hence, a Functional Equivalent Class (FEC) based soft error mitigation technique by utilizing free LUT entries in the circuit is proposed. For all kinds of not fully-occupied LUTs, their corresponding FEC models which have logic masking effect are established first. Then, for a not fully-occupied LUT, its corresponding FEC model is used to replace the LUT. By utilizing free LUT entries in not fully-occupied LUTs, FEC-based technique will not bring in any additional area overhead.

The rest of this paper is organized as follows. Section 2 introduces the background and motivation. Section 3 describes the procedure of FEC-based technique in detail. Experimental results are shown in Section 4. Section 5 gives the comparison with previous work, and the paper is concluded in Section 6.

II. BACKGROUND AND MOTIVATION

A. Architecture of SRAM-based FPGAs

SRAM-based FPGAs have fixed number of configurable logic blocks (CLBs), switch boxes and wire segments. The CLB is a multi-input, multi-output digital circuits that is composed of many LUTs, multiplexers, and flip-flops (FFs). A k-input LUT has $2^k$ LUT entries. By configuring the internal $2^k$ CBs, a k-input LUT can implement any up to k-input logic functions. Alternatively, by configuring the CBs within switch boxes, wire segments are connected into wires. The architecture of SRAM-based FPGAs is shown in Fig. 1. Prior work [11] reports that, in SRAM-based FPGAs, more than 80% of CBs are dedicated to routing resources, and routing resources take up about 80% of the design failure. Hence in this work, we focus on soft error mitigation of routing resources.

B. Logic masking effect

In gate-level netlist, when a fault occurs in an input of a logic gate, if the fault effect is not propagated to the outputs of the design because the off-path inputs of the logic gate have control values, then the fault effect is logic masked. Taking Fig. 2 (a) as an example, for a 2-input AND gate, the fault-free output is 0 when both of the two inputs are 0s. If a stuck-at 1 (SA1) fault occurs at $I_{n_1}$, since the logic value of off-path input ($I_{n_0}$) is 0, the fault effect of SA1 at $I_{n_1}$ is logic masked and will not

This work is supported in part by National Natural Science Foundation of China (NSFC Program) under Grant Nos. 61076018, 60803031, and 60921002, and in part by National Basic Research Program of China (973 Program) under Grant No. 2011CB302503.

1081-7735/11 $26.00 © 2011 IEEE DOI 10.1109/ATS.2011.25
be propagated to the output of AND gate.

Alternatively, there is logic masking effect in LUT-based FPGAs, but expressed in a different way. As Fig. 2 (b) shows, if both of the two inputs are 0s, the output of the LUT is 0 by looking up the CB stored in \textit{entry}_y0 of the LUT. If a SA1 fault occurs at \textit{In}_1, logic values of input wires change from 00 to 10. Through looking up the CB in \textit{entry}_y2 that corresponds to the input address 10, the output of the LUT remains 0 because of the same CBs stored in both \textit{entry}_y0 and \textit{entry}_y2. Therefore, the fault effect of SA1 at \textit{In}_1 is logic masked, and will not be propagated to the output of the LUT.

To accurately and efficiently describe the logic masking effect in LUT-based FPGAs, we introduce a metric named as Address Hamming Distance. The Address Hamming Distance between two LUT entries, represented as $H[addr(entry_{y0}), addr(entry_{x1})]$, describes the Hamming distance between the addresses of \textit{entry}_{y0} and \textit{entry}_{x1} in an LUT. If a fault occurs at the input of an LUT, then the Address Hamming Distance is 1 between the erroneously selected entry and the correctly selected entry. Furthermore, the position of the different address bits corresponds to the faulty LUT input. If CBs are the same for two LUT entries where their Address Hamming Distance is 1, the fault effect at the corresponding LUT input will be logic masked. In contrast, if CBs are different, the fault effect will be propagated to the output of the LUT. Also taking Fig. 2 (b) as an example, $H[addr(entry_{y0}), addr(entry_{y1})] = H[00,10]=1$ and the CBs stored in \textit{entry}_y0 and \textit{entry}_y1 are the same. Hence for the SA1 fault occurs at the corresponding input \textit{In}_1, the fault effect will be logic masked.

As described above, to improve logic masking effect of the design, we need to store the same CBs in LUT entries that have Address Hamming Distance 1. If there are free LUT entries after logic synthesis, then we can use it to introduce logic masking effect while preserving the circuit functionality, and hence improve the reliability of the design.

\section*{C. Motivation}

At present, most of the FPGAs are based on 6-input or more than 6-input LUTs, but the usage of LUT inputs in real designs is low. For an FPGA based on 6-input LUTs, we perform logic synthesis and technology mapping for the 20 largest MCNC benchmark circuits. As Fig. 3 shows, we observe that only 43.71\% of LUTs that use up all of the 6 inputs, and more than 53\% of LUTs only use 2-5 LUT inputs. If there are $m$ LUT inputs unused, the LUT entry usage is only $1/2^m$. Therefore, there are a lot of free LUT entries in not fully-occupied LUTs of the design, which gives us the chance to perform soft error mitigation while not bring in additional area overhead.

\section*{III. \textbf{PROPOSED FEC-BASED TECHNIQUE}}

In this work, we propose a FEC-based technique to mitigate soft errors in SRAM-based FPGAs by utilizing the free LUT entries in the circuit. The proposed flow is shown at the bottom of Fig. 4 labeled as New Flow. Unlike the Baseline Flow, after logic synthesis and technology mapping, the New Flow additionally perform cube-based reliability analysis and FEC-based soft error mitigation to improve the reliability of the design. The New Flow is composed of three steps. At first, for all kinds of not fully-occupied LUTs, the corresponding FEC models are established. Next, the cube-based reliability analysis is performed to evaluate the criticality of each LUT input. Afterwards, for a not fully-occupied LUTs, we select the most critical LUT input as the optimization object, and use the corresponding FEC to replace the LUT, so the reliability improvement is maximized. If there are not any not-fully occupied LUTs existing in the design, the optimized netlist is obtained. Finally, through placement and routing, the circuit is implemented in FPGAs.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig1.png}
\caption{Architecture of SRAM-based FPGAs.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig2.png}
\caption{Example of logic masking effect.}
\end{figure}

\section*{A. Establishing FEC models}

For a not fully-occupied LUT, the FEC is a set of functional equivalent LUTs with logic masking effect. According to the fault effect propagation rules in LUT, if the CBs stored in two LUT entries with Address Hamming Distance 1 are different, then the fault effect at the corresponding LUT input will be propagated to the output of LUT, and may further affect the normal function of the design. Taking Fig. 5 (a) as an example, the LUT implements a 1-input buffer logic. As $H[addr(entry_{y0}), addr(entry_{y1})]=1$, and the configuration bits in \textit{entry}_{y0} and \textit{entry}_{y1} are different, then if a stuck-at 0 (SA0) fault occurs at \textit{In}_0, there are $F_0$ input vectors that can propagate the fault effect to the output. Alternatively, if a SA1 fault occurs at \textit{In}_0, there are $F_1$ input vectors that can propagate to the output. So the total number of fault effect propagation vectors is $F_0 + F_1$. However, if some LUT inputs are unused, we can exploit the free LUT entries to establish FEC with logic masking effect, which can decrease the total number of fault effect propagation vectors, and then improve the reliability of the design.

If there is one LUT input unused, then FEC 1.0 model is established to mask either SA0 fault (corresponds to FEC 1.0...
model) or SA1 fault (corresponds to FEC 1.1 model) at a target LUT input. Firstly, select one of the used LUT inputs as the optimization object, then add a wire from the source of the optimization object to the unused input. In other words, add a redundant wire for the LUT. Please note that, for a not-fully occupied LUT, the synthesis tool will symmetrically configure the unused LUT entries the same functionality as the normally used LUT entries, as Fig. 5 (a) shows. Adding a redundant wire to the unused LUT input will let the input vector look up an entry with the same CB as originally accessed entry. Hence, the functionality of the circuit is preserved with low computational complexity. Afterwards, for a targeted fault, the LUT entries that have Address Hamming Distance 1 with the normally used entries is set the same configuration bits as their counterpart. Consequently, the target fault at the optimization object is logic masked.

As shown in Fig. 5 (b), taking the 2-input LUT as an example, the LUT also implements a 1-input “buffer” logic. Set \( I_{\text{no}} \) as the optimization object, and connect the unused \( I_{\text{n1}} \) to the source of \( I_{\text{no}} \) first. In this case, the normally used LUT entries changes from \( \text{entry}_{y0} \) and \( \text{entry}_{y2} \) to \( \text{entry}_{y0} \) and \( \text{entry}_{y3} \). Then, if there is a SA0 fault occurs at one of the LUT inputs, the input vector will look up LUT entry \( \text{entry}_{y2} \) (corresponds to SA0 at \( I_{\text{no}} \) or \( \text{entry}_{y1} \) (corresponds to SA0 at \( I_{\text{n1}} \)) instead of the normally used \( \text{entry}_{y3} \). For FEC 1.0 model which targets SA0 fault, the CBs stored in \( \text{entry}_{y2} \) and \( \text{entry}_{y1} \) are set the same as \( \text{entry}_{y3} \) to mask the SA0 fault at either \( I_{\text{no}} \) or \( I_{\text{n1}} \). Unwillingly, if a SA1 fault occurs at the newly added redundant wire, there will be additional \( F_1 \) input vectors that can propagate the fault effect to the output, as shown in Fig. 5 (b). Likewise, for FEC 1.1 model which targets SA1 fault, the CBs in LUT entries that have Address Hamming Distance 1 with \( \text{entry}_{y3} \) are set the same as \( \text{entry}_{y1} \). In this case, the SA1 fault effect at any one of the two inputs will be logic masked at the cost of additional \( F_0 \) input vectors that can propagate the SA0 fault at \( I_{\text{n1}} \), as shown in Fig. 5 (c).

To sum up, FEC 1.0 (FEC 1.1) model utilizes a free LUT input to mask the \( F_0 \) \((F_1)\) input vectors that can propagate SA0 (SA1) fault at the optimization object, but at the cost of additional \( F_1 \) \((F_0)\) input vectors that can propagate the SA1 (SA0) fault at the newly added redundant wire. The total number of fault effect propagation vectors that FEC 1.0 model can decrease is

\[
\Delta F_{\text{FEC} 1.0} = F_0 - F_1
\]

\(\Delta F_{\text{FEC} 1.1} = F_1 - F_0 \quad (1)\)

If there are two LUT inputs unused, FEC 2 model is established to mask both SA0 fault and SA1 fault at the targeted LUT input. Also set one of the used LUT input as optimization object, the two unused LUT inputs are simultaneously connected to the source of the optimization object first. In this case, the normally used entries of the LUT are changed, and the functionality of the LUT is preserved by using the functionality duplication from used LUT entries to unused LUT entries. Afterwards, for each normally used LUT entry, find all LUT entries that have Address Hamming Distance 1 with it, and set the CBs of these entries the same as corresponding normally used entry. Consequently, both SA0 fault and SA1 fault at the optimization object will be logic masked.

Taking the 3-input LUT in Fig. 5 (d) as an example, the two unused LUT inputs \((I_{\text{n1}} \text{ and } I_{\text{n2}})\) are connected to the source of the optimization object \((I_{\text{no}})\). In this case, the normally used LUT entries changes from \(\text{entry}_{y0} \) and \(\text{entry}_{y1} \) to \(\text{entry}_{y0} \) and \(\text{entry}_{y7} \). Afterwards, unused entries \(\text{entry}_{y1} \), \(\text{entry}_{y2} \), and \(\text{entry}_{y4} \) are selected as having Address Hamming Distance 1 with \(\text{entry}_{y0} \), and the CBs of these selected entries are set the same as \(\text{entry}_{y0} \). Consequently, the SA1 fault at any one of the three LUT inputs is logic masked. Likewise, unused entries \(\text{entry}_{y3} \), \(\text{entry}_{y5} \), and \(\text{entry}_{y6} \) are selected as having Address Hamming Distance 1 with \(\text{entry}_{y7} \), and the configuration bits of these selected entries are set the same as \(\text{entry}_{y7} \). Consequently, the SA0 fault at any one of the three LUT inputs is also logic masked. Note that, as \(H[\text{addr(entry}_{y0})]H[\text{entry}_{y7}]\)=3, there is no LUT entry, that can concurrently satisfy both \(H[\text{addr(entry}_{y0})]H[\text{entry}_{y7}]\)=1 and \(H[\text{addr(entry}_{y7})]H[\text{entry}_{y7}]\)=1. Hence, for an LUT entry, there is no conflict for the concurrently optimization of SA0 fault and SA1 fault.

In conclusion, FEC 2 model uses two free LUT inputs, and can mask both SA0 fault and SA1 fault at the optimization object. The total number of fault effect propagation vectors that FEC 2 model can decrease is

\[
\Delta F_{\text{FEC2}} = F_0 + F_1 \quad (2)
\]

If there are more than two LUT inputs unused, the corresponding FEC model is the combination of FEC 1.x and FEC 2. For example, if there are three LUT inputs unused, FEC 2 model is used first to mask both SA0 fault and SA1 fault at one of the used LUT input. Afterwards, FEC 1.x model is used to further mask either SA0 fault or SA1 fault at another used LUT input. Likewise, if there are four LUT inputs unused, FEC
2 model is used twice for two used LUT inputs. Consequently, the SA0 fault and SA1 fault at both of the two optimization objects are logic masked.

B. Cube-based reliability analysis

The first step of performing soft error mitigation is to evaluate the reliability of a design. The evaluation criterion of a design against soft errors is SER [14], which is computed as

\[ SER = \sum_{i=0}^{N_{LUT}} \sum_{j=0}^{N_{element(i)}} N_{ERj} \times EPP_j \]

where \( N_{LUT} \) is the total number of LUTs in the design, \( N_{element(i)} \) describes the number of elements in the LUT, such as LUT inputs, LUT outputs, and LUT entries. For the \( j^{th} \) element, \( N_{ERj} \) describes the probability of a fault occurs at it. On the other hand, error propagation probability of the fault at element \( j \) (\( EPP_j \)) describes the criticality of the element, which is characterized by the number of fault effect propagation vectors \( (N_{prop(j)}) \) scaled to the total number of input vectors \( (N_{inputs}) \). The smaller SER is, the more reliable the design is. During logic synthesis, the proposed FEC-based technique tries to decrease the criticality of each LUT, more specifically, decrease the total number of fault effect propagation vectors \( (N_{prop(j)}) \) of LUT inputs.

In this work, the cube-based reliability analysis [12] is used. A cube defines a relationship between the primary inputs and the primary outputs. A cover is a set of cubes that have the same output value given the function of the design. In each cube, besides traditional 2-value logic, “0” and “1”, don't-care symbol “X” is introduced. Each “X” can be replaced by a “0” or a “1”, which can obtain high compression ratio for input vectors. For each element of the design, such as LUT inputs, and LUT outputs, four types of covers are defined. The two control-covers, namely control-cover 0 and control-cover 1, store the cubes that can control the logic value of a target wire to be 0 or to be 1 during logic simulation, respectively. In this way, control-cover 0 contains cubes that can sensitize a target stuck-at 1 fault, and control-cover 1 contains cubes that can sensitize a target stuck-at 0 fault. Furthermore, there are some cubes in control-covers which can not only sensitize the target fault but also propagate the fault effect. So the cubes in control-cover 0, which can propagate the faulty logic value 1, constitute of a care-cover 1; and the cubes in control-cover 1, which can propagate the faulty logic value 0, constitute of a care-cover 0.

Through forward traverse of the design to compute the control-covers and backward traverse to compute the care-covers of each element, the criticality of each element is the percentage of cubes stored in care-covers to the total number of input vectors.

Note that, the criticality of each element by analyzing by traversing the whole design twice, so the computational complexity is low. Moreover, the cube-based method analyzes the criticality of each element in terms of input vector, so the accuracy is high. The detailed description of cube-based method is given in [12].

C. FEC replacement

After cube-based reliability analysis, the number of fault effect propagation vectors that can propagate the fault at each LUT input is obtained. According to Equation (3), the reliability of the design is high if the total number of fault effect propagation vector is small. For not fully-occupied LUTs in the circuit, corresponding FECs are used to replace them, which can decrease the number of fault effect propagation vectors. Consequently, the reliability of the design is improved.

If there is one LUT input unused, FEC 1.x model is used. According to Equation (1) FEC 1.x can reduce the number of fault effect propagation vector by \( \Delta F_{FEC1.0}=F_{01}-F_{10} \) or \( \Delta F_{FEC1.1}=F_{11}-F_{00} \). To maximize the reliability improvement, we set the used LUT input with \( \max \{F_{01}, F_{10}\} \) as optimization object. Afterwards, if \( F_{01}>F_{10} \), FEC 1.0 model is used to target the SA0 fault. Otherwise, if \( F_{11}>F_{00} \), FEC 1.1 model is used to target the SA1 fault.

If there are two LUT inputs unused, FEC 2 model is employed. As FEC 2 can reduce the number of fault effect propagation vector by \( \Delta F_{FEC2}=F_{00}+F_{11} \), the used LUT input with \( \max \{F_{00}+F_{11}\} \) is set as optimization object. Consequently, the effect of soft error mitigation is maximized.

If there are more than two LUT inputs unused, to maintain the maximum soft error mitigation, the two selection schemes described above are jointly applied. For example, if there are three LUT inputs unused, FEC 2 model is used first for the LUT input with \( \max \{F_{00}+F_{11}\} \), and then FEC 1.1 model is further used for the LUT input with \( \max \{F_{00}+F_{11}\} \).

Note that, the proposed FEC-based technique only uses the free LUT entries in not fully-occupied LUTs, which will not bring in additional area overhead. Besides, during the procedure of establishing FEC models, the proposed method preserves the functionality of the design by utilizing the functionality duplication from used LUT entries to unused LUT entries, so the computational complexity is low. In addition, the FEC-based technique does not alter original interconnection among LUTs, and the newly added wires have the same source and sink as the corresponding optimized object, so the topology of the design is maintained, which has little impact on succeeding placement and routing stages. Moreover, the proposed technique does not rely on any specific architecture of physical device, and is suitable for all LUT-based FPGAs.

After traversing the whole design to replace the not fully-occupied LUTs with corresponding FECs, we evaluate the
TABLE I
COMPARISON OF AREA, SER, AND CRITICAL-PATH DELAY

<table>
<thead>
<tr>
<th>Circuit</th>
<th>PI#</th>
<th>PO#</th>
<th>reg#</th>
<th>Area(# of LUTs)</th>
<th>SER(FIT)</th>
<th>Critical-path delay(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>14</td>
<td>8</td>
<td>0</td>
<td>503</td>
<td>7.78</td>
<td>5.09E-09</td>
</tr>
<tr>
<td>apex4</td>
<td>19</td>
<td>10</td>
<td>0</td>
<td>577</td>
<td>35.78</td>
<td>5.94E-09</td>
</tr>
<tr>
<td>misex3</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td>438</td>
<td>11.38</td>
<td>1.02E-08</td>
</tr>
<tr>
<td>pdc</td>
<td>16</td>
<td>40</td>
<td>0</td>
<td>1359</td>
<td>32.27</td>
<td>1.18E-08</td>
</tr>
<tr>
<td>s298</td>
<td>4</td>
<td>6</td>
<td>14</td>
<td>24</td>
<td>1.53</td>
<td>1.84E-09</td>
</tr>
<tr>
<td>spla</td>
<td>16</td>
<td>46</td>
<td>0</td>
<td>1373</td>
<td>65.58</td>
<td>1.54E-08</td>
</tr>
<tr>
<td>ex1010</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>647</td>
<td>35.91</td>
<td>1.04E-08</td>
</tr>
<tr>
<td>ex5p</td>
<td>8</td>
<td>63</td>
<td>0</td>
<td>354</td>
<td>12.20</td>
<td>5.00E-09</td>
</tr>
<tr>
<td>apex2</td>
<td>38</td>
<td>3</td>
<td>0</td>
<td>643</td>
<td>5.75</td>
<td>5.63E-09</td>
</tr>
<tr>
<td>bigkey</td>
<td>229</td>
<td>197</td>
<td>224</td>
<td>579</td>
<td>63.01</td>
<td>6.95E-09</td>
</tr>
<tr>
<td>clma</td>
<td>62</td>
<td>82</td>
<td>33</td>
<td>2789</td>
<td>9.62</td>
<td>8.81E-09</td>
</tr>
<tr>
<td>des</td>
<td>256</td>
<td>245</td>
<td>0</td>
<td>556</td>
<td>74.93</td>
<td>8.08E-09</td>
</tr>
<tr>
<td>difeq</td>
<td>28</td>
<td>3</td>
<td>305</td>
<td>563</td>
<td>30.65</td>
<td>6.02E-09</td>
</tr>
<tr>
<td>dsip</td>
<td>197</td>
<td>224</td>
<td>0</td>
<td>873</td>
<td>90.41</td>
<td>7.75E-09</td>
</tr>
<tr>
<td>elliptic</td>
<td>3</td>
<td>2</td>
<td>194</td>
<td>315</td>
<td>39.85</td>
<td>5.01E-09</td>
</tr>
<tr>
<td>seq</td>
<td>41</td>
<td>35</td>
<td>0</td>
<td>668</td>
<td>16.73</td>
<td>9.01E-08</td>
</tr>
<tr>
<td>tseng</td>
<td>52</td>
<td>122</td>
<td>382</td>
<td>655</td>
<td>40.42</td>
<td>7.98E-09</td>
</tr>
<tr>
<td>frisc</td>
<td>20</td>
<td>116</td>
<td>866</td>
<td>1762</td>
<td>130.84</td>
<td>1.02E-08</td>
</tr>
<tr>
<td>s38417</td>
<td>29</td>
<td>106</td>
<td>1462</td>
<td>2497</td>
<td>145.81</td>
<td>105.29</td>
</tr>
<tr>
<td>s38584.1</td>
<td>38</td>
<td>304</td>
<td>1260</td>
<td>2318</td>
<td>198.36</td>
<td>9.59E-09</td>
</tr>
</tbody>
</table>

| Geomean | 100.00% | 104.25% |

The raw error rate of an SRAM bit is assumed to be 0.01 (FIT/bit)

In order to validate the proposed FEC-based soft error mitigation technique, we use the 20 largest MCNC benchmark circuits, and map them into LUTs by Berkeley ABC mapper [15]. During the stage of placement and routing, VPR (Versatile Place and Route), the state-of-the-art placement and routing tool for FPGAs, is used [16]. The architecture used for all experiments is Virtex-like FPGA architecture. Each CLB contains four 6-input LUTs, and the routing resources contain four types of wire segments, single(8%), double(20%), hex(60%), and long(12%). All the proposed algorithms are implemented in Java and run on a Xeon Linux workstation with 6GB memory.

There are two CAD flows compared in our experiments. For the baseline flow, the benchmark circuits are mapped into LUTs by Berkekey ABC mapper first. Then the mapped designs are implemented in FPGAs through placement and routing by VPR toolset. Finally, the cube-based reliability analysis is performed to evaluate the reliability of the design, as the Baseline Flow shown in Fig. 4. In contrast, for the proposed flow, after logic synthesis and technology mapping, the cube-based reliability analysis and FEC-based soft error mitigation technique are used, which can improve the reliability of the design. Afterwards, placement and routing are performed for the optimized netlist. Finally, the reliability is evaluated again to assess the effectiveness of the proposed FEC-based technique, as the New Flow shown in Fig. 4. Note that, as the reliability of the design is evaluated after the placement and routing stages, the impact of FEC-based technique on succeeding placement and routing stages is also considered. At the step of cube-based reliability analysis, we perform full space input vectors for smaller design with less than 25 combinational inputs (alu4, apex4, misex3, pdc, s298, spla, ex1010, and ex5p). For other designs, we randomly select 100 * 2^25 input vectors. In our experiment, we make the same assumption as [7] that: a soft error that occurs in a routing CB always causes the flip of the logic value of the corresponding routing signal. Note that, this is a pessimistic assumption which estimates an upper bound of SER. Our proposed algorithm can also be applied to more sophisticated fault models such as the work in [17].

The comparison of area, SER, and critical-path delay are shown in Table I. Compared with the baseline result of Berkeley ABC mapper, the proposed FEC-based technique can decrease SER by 21.73% on average. Note that, the effect of soft error mitigation for circuits namely bigkey is not that significant. That is because the high usage of LUT inputs of bigkey limits the optimization space of FEC-based method. In addition, as the FEC-based technique only uses free LUT entries in not fully-occupied LUTs, there is not any additional area overhead. As the FEC-based technique maintains the topology of original netlist, and the newly added wires have the same source and sink as the corresponding optimized object, so there is little effect on placement and routing stages. We can see the critical-path delay increase of FEC-based technique is only 4.25%.

V. COMPARISON WITH PREVIOUS WORK

As described in Section I, the soft error mitigation during logic synthesis can be classified into two categories. For the schemes utilizing Boolean flexibility of the design, the robust resynthesis algorithm (ROSE) [5] establishes robust programmable logic block templates with logic masking effect first, and then performs logic resynthesis based on these robust templates to improve the reliability of the design. Experimental results show that, ROSE can decrease the SER by 25%. Note that, during logic resynthesis, ROSE changes the topology of original design, which has significant impact on placement and routing. Consequently, according to Equation (3), the soft error mitigation obtained by EPP decrease may be offseted by the increase of NER. The work namely IPR [6] selects a cone in the netlist for the high criticality CB, and then chooses the functional equivalent cone with the highest logic masking effect to replace it. Results show that IPR can reduce the SER by 49%. The work namely R2 [7] performs soft error mitigation for interconnecting wires. For interconnecting
wires with high criticality, R2 selects the wires with lower criticality to replace them. Experimental results show that R2 can reduce the SER by 20% for the IPR-optimized netlist. Note that, all the works in this category need Boolean matching to preserve the functionality of the design, which may introduce high computational complexity. The computational complexity of FEC-based technique is $O(N_{LUT})$, which is linear to the scale of the design. The run time of the FEC-based technique for each benchmark is given in Table II. As we can see, the average run time of FEC-based scheme for smaller circuits (less than 25 combinational inputs) and bigger circuits are 5.76ms and 28.83ms respectively. Although there are some small differences in experimental setup, the run time of FEC-based scheme is orders of magnitude shorter than that of ROSE (184.2s [5]), IPR (5.58s [6]), and R2 (90.06s [7]).

For the second category that utilizes free LUTs and LUT outputs in the circuit, the work proposed in [8] duplicates the logic function of not fully-occupied LUTs, and encodes the dual outputs with “AND”, “OR” logic at the next logic stage. The “AND” (“OR”) encoding can mask the SA1 (SA0) fault at one of the preceding duplicated logic, which can reduce SER by 25% in all. Although this method [8] preserves the circuit functionality in architecture aspect, it relies on the feature of dual-output LUT, which is supported by only a small portion of physical devices. Moreover, utilizing the second output proposed in [8] will introduce additional MUX, which brings in additional performance overhead [9]. In contrast, as described in Section III, the proposed FEC-based technique does not rely on any specific feature of physical device, and is suitable for all LUT-based FPGAs. Additionally, the FEC-based technique will not introduce any additional MUX. Consequently, the performance overhead of the FEC-based technique is negligible.

Compared with above mentioned techniques [5], [6], [7], [8], the proposed FEC-based technique can obtain a comparable soft error mitigation effect. Note that, FEC-based scheme is orthogonal to the existing soft error mitigation schemes [5], [6], [7] that utilize Boolean flexibility of the design, and therefore the FEC-based technique can be jointly used with these works to further improve the reliability of the design, while not bring in any additional area and performance overhead. On the other hand, the proposed FEC models will introduce redundant interconnecting wires, so the FEC-based technique is more suitable for the designs with sufficient routing resources.

VI. CONCLUSIONS

In this work, we observe a FPGA-based design usually has a low usage of LUT inputs, and hence contains many free LUT entries. By using these not fully-occupied LUTs, the proposed FEC-based technique introduces logic masking effect to improve the reliability of the design. Compared with the result of traditional ABC mapper, the proposed scheme can very effectively reduce SER. Compared with existing soft error mitigation schemes during logic synthesis, the proposed FEC-based technique is of low computational complexity and of low impact on succeeding placement & routing stages. Moreover, the proposed technique does not rely on any specific architecture of physical device, and is suitable for all LUT-based FPGAs.

REFERENCES


TABLE II

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Run Time (ms)</th>
<th>Circuit</th>
<th>Run Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alu4</td>
<td>5.00</td>
<td>apex2</td>
<td>16.00</td>
</tr>
<tr>
<td>apex4</td>
<td>5.00</td>
<td>bigkey</td>
<td>14.00</td>
</tr>
<tr>
<td>miscx3</td>
<td>2.00</td>
<td>clma</td>
<td>49.00</td>
</tr>
<tr>
<td>pdc</td>
<td>10.00</td>
<td>des</td>
<td>15.00</td>
</tr>
<tr>
<td>s298</td>
<td>10.00</td>
<td>diffeq</td>
<td>14.00</td>
</tr>
<tr>
<td>spla</td>
<td>10.00</td>
<td>disp</td>
<td>16.00</td>
</tr>
<tr>
<td>ex1010</td>
<td>5.00</td>
<td>elliptic</td>
<td>11.00</td>
</tr>
<tr>
<td>ex5p</td>
<td>4.00</td>
<td>seq</td>
<td>18.00</td>
</tr>
<tr>
<td>tseng</td>
<td></td>
<td>frisc</td>
<td>29.00</td>
</tr>
<tr>
<td>s38417</td>
<td></td>
<td>s38584</td>
<td>47.00</td>
</tr>
<tr>
<td></td>
<td>Geomean</td>
<td></td>
<td>5.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>28.83</td>
</tr>
</tbody>
</table>