Abstract

In this work, targeted experiments were carried out to study the effectiveness of two design-for-diversity techniques in a Triple Modular Redundancy (TMR) system setting with bitwise voting, implemented on a Field Programmable Gate Array (FPGA). The two techniques are Case-Based synthesis and Inverted-Output synthesis. They were used to generate implementation-different yet functionally-equivalent designs of a three-bit multiplier. The diversity of these designs was then extracted as compared to a third implementation (referred to as the base design) generated through topmost high level Verilog synthesis. The diversity metric was used as a determining factor for how diverse each of the designs is relative to the base design, and how diverse the two designs are relative to each other. Both Common-Mode Faults (CMFs) and random faults were considered. However, in both cases, only a single stuck-at fault per module was modeled.

The experimental results indicate that the designs generated using both Case-Based and Inverted-Output synthesis display a high level of diversity from the base design. Augmented with bitwise voting, the implemented TMR system shows a very high level of resilience in the presence of CMFs as well as random faults. The results show a system reliability approaching 99% thereby indicating high levels of data integrity.

I. INTRODUCTION

Redundancy has long been used as a means of increasing the fault-tolerance, and thus the reliability of logical systems. Redundancy has mainly been used in the form of Concurrent Error Detection (CED) techniques such as Dual and Triple Modular Redundancy (DMR and TMR) to detect and correct errors caused by Single Event Upsets (SEUs). However, several recent works, such as [1], [2] and [3], have adapted these principles for use in fault detection, isolation and recovery in adaptive systems. Comprehensive surveys of fault detection and handling techniques can be found in [4] and [5].

While offering real-time error detection in the presence of a single fault, in their raw form, CED systems that replicate the same design remain vulnerable to Common-Mode Faults (CMFs). CMFs in a redundant system are defined in [6] as failures that affect more than one module at the same time, generally due to a common cause which may be due to a design mistake. CMFs in redundant VLSI systems are surveyed in [7]. [8] points out that design faults are reproduced when redundant copies of the same design implementation are made. This compromises the data integrity of the system which is defined in [9] as the ability of the system to either produce correct outputs or generate an error signal when incorrect outputs are produced. In a TMR system, data integrity is maintained as long as no two modules produce identical erroneous outputs (assuming that the comparator is fault-free).

Design diversity has been proposed in the past to protect redundant systems against common-mode failures. Design diversity was defined in [8] as the independent generation of two or more software or hardware elements to satisfy a given requirement. The basic idea is that, with different implementations, common-mode failures will probably cause different error patterns to be produced [6], which will alert the system of the presence of faults allowing it to preserve its data integrity.

Faults can originate from a variety of sources and can have different degrees of impact on the redundant system. One class of faults can be induced due to Single Event Upsets (SEUs) or Single Event Transients (SETs). These tend to have a temporary effect on the system and may cause limited output corruption. Another class of faults has a permanent effect on the system and may be due to manufacturing defects or the degradation of the VLSI circuit. A great discussion on the different types of faults can be found in [4]. In our experiments, stuck-at type faults were modeled at the inputs of the Look-Up Tables (LUTs) of the FPGA. Research in the area of digital testing and diagnosis of combinational and sequential logic circuits has demonstrated the effectiveness of this model [10].

This paper studies the effectiveness of two design-for-diversity techniques, namely Case-Based synthesis and Inverted-Output synthesis, in the preservation of data integrity of a TMR system in the presence of common-mode and non-common-mode stuck-at faults. The TMR system benchmarked implements a three-bit multiplier and is composed of three modules, two of which were generated using the aforementioned techniques. The third module was
implemented through the synthesis of top-level Verilog code, where the synthesizer had full control over the design.

Section II of this paper provides references to some related work in the field of Design Diversity and describes in detail the metric used to quantify diversity among our different designs. Section III describes the two implementation techniques employed to generate the TMR modules. Sections IV and V describe the experiments done and their results respectively. Finally, section VI presents some conclusions drawn from the results as well as ideas for future work.

II. RELATED WORK

Design diversity is not a new concept. N-version programming is an example of diversity in software systems, while hardware diversity is used in the Primary Flight Computer of the Boeing 777. However, the conventional notion of diversity, which relies on “independent” generation of “different” implementations, was qualitative in nature and did not provide a basis to compare the reliabilities of two diverse systems. It wasn’t until 1999 that [6] presented a metric to quantify diversity among several designs. This design diversity metric is defined as follows.

Let \( d_{ij} \) be the diversity with respect to a fault pair \((f_i, f_j)\) where \( f_i \) and \( f_j \) are faults present in the diverse modules M1 and M2 respectively. \( d_{ij} \) is the probability that the designs do not produce identical error patterns, in response to a given input sequence. This prompts the notion of joint detectability, \( k_{ij} \) which is defined as the number of input patterns, in response to each of which, both the implementations M1 and M2 produce the same erroneous output pattern. If we assume that all input patterns are equally likely, then we can write:

\[
 d_{ij} = 1 - \frac{k_{ij}}{2^n}
\]

where \( n \) is the number of input bits.

If we assume that all fault pairs are equally probable, and that there are \( m \) fault pairs \((f_i, f_j)\), the design diversity metric, \( D \), for the two implementations is:

\[
 D = \frac{1}{m} \sum_{i,j} d_{ij}
\]

[6] presents a detailed treatment of a time-dependent reliability analysis that shows the relationship between design diversity, system failure rate, and mission time.

The authors of [6] point out in [11] that, for arbitrary designs, the problem of calculating the value of the \( D \)-metric is NP-complete, (i.e., can be of exponential complexity) and they present efficient techniques to estimate the value of the design diversity metric. For datapath designs, they claim to have formulated very fast techniques to calculate the value of the metric by taking advantage of the regularity in the datapath structures. For general combinational logic circuits, they present an adaptive Monte-Carlo simulation technique for estimating accurate bounds on the value of the metric.

In what may be considered one of their most notable contributions to the field of design diversity, the same authors proposed in [9] a systematic technique for the synthesis of diverse implementations of combinational logic circuits in order to maximize the data integrity of diverse duplex systems in the presence of common mode failures. These techniques use the design diversity metric as a cost function during the synthesis process. The first step in their process of generating the “best” alternate design is to generate a new truth table by complementing the outputs of the original truth table of the desired function. The procedure then goes through a number of iterations until it finds the best design whose outputs are the compliment of those of the desired function and adds inverters to it.

III. DESIGN-FOR-DIVERSITY TECHNIQUES

A. Case-Based Synthesis

Case-Based Synthesis is an informal and very simple design technique that involves describing the function to be implemented in the form of a truth table. This truth table is then translated into a HDL case statement which is fed to the synthesizer for logic extraction and schematic generation. This process is straightforward for small combinational circuits. However, the length of the case statement grows exponentially with the number of input bits. This can be overcome by automating the case statement generation process given a functional description of the circuit, or by dividing the complex system into smaller, manageable subcircuits and then tying them together in top module.

The challenge is even greater when dealing with sequential circuits since the output does not depend solely on the inputs. [9] indicates that given the specification of a sequential logic circuit, and an encoding of its internal states, the problem of synthesizing the sequential circuit can be mapped to a combinational logic synthesis problem. This, however, is outside the scope of this experiment.

B. Inverted Output

Once the case statement described above is generated, an inverted-output description of the system is easily produced. This is done by inverting the output associated with each input case. Finally, in order to restore the functionality of the system, an inverter is placed on each of the output bits. This is done simply by adding one non-blocking assignment (per output) after the end of the case statement. Figure 1 illustrates the inverted-output concept for the three-bit multiplier used in our experiment.

Using the implementations of logic functions in their true and complemented forms during duplication was first proposed in [12], and, depending on the synthesizer used and the optimization parameters set,
synthesis of the inverted-output description of a particular function will result in a different implementation from that obtained from the synthesis of a true case-based description.

IV. EXPERIMENTAL SETUP

A. Simulation Tools and Workflow

Our experiments were carried out using Xilinx ISE Design Suite 12.2 equipped with the ISim FPGA simulator. The target configuration was that of a Virtex 4 FPGA, though we did not use the actual hardware in our experiment. The experiments involved three implementations of a three-bit multiplier circuit. All three implementations were generated using the synthesizer provided with the Design Suite. The first implementation, referred to as the base design, was generated using topmost Verilog code whereby the module was described in terms of the relationship of the output to the two inputs using a functional equation. The synthesizer had full control over the implementation process. The only constraint imposed was a physical restriction to a specific region on the FPGA. The second implementation was generated using the case-based description of the function, namely, a 64-line (2^6 input combinations) case statement was written to describe the output value in response to each possible input combination. The third implementation was generated using the inverted-output concept. The same 64-line case statement was used, this time setting the output value in each case to its inverse. Another code statement was added after the case statement to complement one more time the output of the design. A testbench file was created that instantiates each of the designs emulating a TMR configuration (Figure 2) and collects relevant data. The same constraint file as in the base design was imposed on the other two designs. This was possible since the TMR system was not physically implemented on an actual chip. This also simplified the process of CMF injection since the fault location was the same for all three designs.

The testbench recorded the following values relevant to determining the D-metrics among each design pair:

- $k_{ij}$: the joint detectability of fault pairs $(f_i, f_j)$ affecting the base design and the Case-Based design respectively.
- $k_{ik}$: the joint detectability of fault pairs $(f_i, f_k)$ affecting the base design and the Inverted-Output design respectively.
- $k_{jk}$: the joint detectability of fault pairs $(f_j, f_k)$ affecting the Case-Based design and the Inverted-Output design respectively.

The testbench also kept track of values relevant to the reliability of each of the designs and that of the TMR system. Reliability is defined here as the probability that the output obtained from a certain design (or the TMR system) is not erroneous. For example, if the TMR system provides $A$ erroneous outputs out of $2^n$ outputs (where $n$ is the number of input bits) in the presence of a fault sequence $(f_i, f_j, f_k)$, then the reliability $R$ relative to the fault $(f_i, f_j, f_k)$ is defined as:

$$R(f_i, f_j, f_k) = 1 - \frac{A}{2^n}$$

If $B$ fault sequences are recorded, then the reliability can be expressed as:

$$R = 1 - \frac{A}{B, 2^n}$$

Figure 3 shows the workflow of the testbench in the presence of a fault sequence $(f_i, f_j, f_k)$.

![Figure 2: Generic TMR configuration with bit-wise voting](image)
B. Experiment Descriptions

Experiment 1: TMR with diverse designs and a single CMF per module.

In this experiment, the TMR system consists of the three designs described earlier. CMFs were injected in eight LUTs belonging to 4 physical slices utilized by all three designs. A total of 64 fault combinations were injected corresponding to all possible single fault states. This experiment gives the best indication of the D-metric as it addresses the main pitfall of CED using duplicated designs.

Experiment 2: TMR with diverse designs and a single random fault per module.

This experiment uses the same setup as Experiment 1, but injects random stuck-at faults at random locations in each module. In conjunction with Experiment 3, this will give indication of the benefit (if any) of using design diversity over replicated design in a TMR system. 12 fault combinations were simulated. Figure 4 illustrates the concepts of CMFs and random faults.

Experiment 3: TMR with duplicated design and a single random fault per module.

In this experiment, the TMR system is made up of 3 identical modules (base design). Random stuck-at faults are injected at random locations in each module. The expectation is that the modules will produce different outputs since the fault locations are not similar, and therefore the system might not behave much worse than one implementing diverse modules. Eight error patterns were simulated.

Figure 4: (a) In CMF modeling, the faults affect the same physical location in all three modules. (b) In Random fault modeling, one fault is randomly injected in each module independent of the other modules.
Experiment 4: TMR with diverse designs (Inverted-Output, Template-Based, and NAND-Based) and a single CMF per module.

[13] and [14] have carried out a similar study to this one using other design-for-diversity techniques. [13] experimented with the Template-Based design technique where a VLSI circuit is broken into lower-level functional units, which are combined to perform the desired function. Several implementations of each of these sub-circuits are generated and then mixed and matched to generate several implementations of the desired circuit. [14] constrained the synthesis process to use only NAND gates in its implementation of the circuit. In this experiment a TMR system was built using an Inverted-Output module, a Template-Based module and a NAND-based module. The point here is to study the diversity among each pair of modules generated using these techniques. A single CMF was injected per module, and 12 CMFs were simulated.

Experiment 5: TMR with diverse designs (Inverted-Output, Template-Based, and NAND-Based) and a single random fault per module.

This experiment uses the same setup as in Experiment 4, but injects random stuck-at faults at random locations in each module. Combined with the results of Experiments 2 and 3, this should confirm (or refute) the presence of any advantage for using diversity in the presence of random faults. Eight error patterns were simulated.

V. EXPERIMENTAL RESULTS

Several combinations of properties have been used to generate a broad spectrum of data to allow for a clear characterization of the performance of each design technique under different error conditions. For each experiment, the aggregate fault impact shows the total number of faulty outputs articulated by each of the used modules and by the TMR system. This shows the fault-masking ability of the TMR system under different setups. The D-metric is recorded for each pair of designs tested under CMFs. Finally, the reliability of each module and of the TMR system is recorded to indicate the level of data integrity exhibited by the system.

Experiment 1: We see from Figure 5 that the Case-based and the Inverted-Output designs displayed a varying degree of tolerance towards faults compared to the Base design depending on the LUT affected. This is due to the fact that the LUTs affected had different levels of prominence in the datapath of each design. However, we see that the TMR system as a whole was consistently less affected by the faults. This is due to two factors. First, even though the number of faults was high in one module or the other, it did not always coincide with a high number of faults in the other modules. Second, even when all three modules generated faulty outputs, in many instances the bit-wise voting module generated the correct output for the TMR system. Of course, had the one design been replicated to form the TMR system, the bit-wise voting becomes transparent in the presence of CMFs, and the number of faulty outputs produced by the TMR system will be equal to the number of faulty outputs produced by the duplicated module. This highlights the importance of using design diversity in CMF-prone environments.

Figure 6 shows that, even though each of Case-based and Inverted-Output show very high diversity from the Base design, they are less diverse among each other, which is to be expected since the two designs were generated using related techniques.

Figure 7 shows that the reliability of the TMR system over the range of the faults simulated is considerably higher than that of any individual design.

It is noteworthy that the two modules generated using Case-based and Inverted-Output designs were generally not affected by the nature of the stuck-at fault (0 or 1).
Figure 6: The D-metric among each pair of the designs in the presence of CMFs.

Experiment 2: Figures 8 and 9 show the fault impact, and the calculated diversity among each design pair in the presence of random faults respectively. We see that the TMR system still exhibits a high level of fault-masking ability, while we see that the diversity between the Base design and the Inverted-Output design has decreased from the value found in the previous experiment. This would be acceptable if not for the fact that diversity has actually increased between the Case-Based design and the Inverted-Output design. The latter result is counter intuitive, and puts in question the validity of the D-metric when calculated using random faults only. This is reaffirmed in Experiment 3 where the same module was duplicated, yet the D-metric (not calculated here) indicates a highly diverse value.

Figure 7: Fault impact in the presence of random faults.

Figure 8: D-metric calculated in the presence of random faults.

Figure 9: Reliability of different modules and designs in the presence of random faults.

Figure 10 shows that the reliability of the TMR system is still higher than any individual design.

Experiment 3: Figure 11 shows an interesting result which is that even in a TMR system implemented using duplicated modules, the fault impact at the system level is comparable to that of a TMR system implemented using diverse designs. The D-metric was not calculated for this experiment. However, had it been calculated, its value would be relatively high among the three modules even though they are replicas of each other. These results confirm that the value of the D-metric does not bear much
Table 1: Design-for-Diversity Techniques Summary

<table>
<thead>
<tr>
<th>Design</th>
<th>Inverted-Output (1)</th>
<th>Template-Based (2)</th>
<th>NAND-Based (3)</th>
<th>TMR System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault impact (# of faults)</td>
<td>48</td>
<td>221</td>
<td>111</td>
<td>25</td>
</tr>
<tr>
<td>Reliability</td>
<td>0.9375</td>
<td>0.712239583</td>
<td>0.85546875</td>
<td>0.967447917</td>
</tr>
<tr>
<td>D-metric</td>
<td>$D_{12} = 0.98958$</td>
<td>$D_{13} = 0.9921875$</td>
<td>$D_{23} = 0.9986979$</td>
<td></td>
</tr>
</tbody>
</table>

meaning when calculated using random faults only. The reliability (Figure 10) of the TMR system is again comparable to the one implementing diverse designs.

![Figure 10: Fault impact on a TMR system with duplicated modules in the presence of random faults.](image)

Experiments 4 & 5: these experiments show the diversity among designs generated using different design for diversity techniques, namely, Inverted-Output, Template-based, and NAND-based design. Table 1 summarizes the key values obtained in the presence of CMFs. We see that the three design techniques produce modules that are highly diverse and suitable for use in a TMR system. Data obtained by simulating random faults has been omitted for the reasons explained previously.

VI. 6. CONCLUSIONS AND FUTURE WORK

In this paper, we have demonstrated the results of a set of experiments geared towards the characterization of the effectiveness of different design-for-diversity techniques on a TMR system implementing a 3-bit multiplier function. The D-metric was used as the primary factor in this determination, but other metrics such as the fault-masking capability and the reliability were also studied. Results show that design diversity is extremely beneficial in the presence of Common-Mode Failures in a redundant system. However, in the presence of random faults the advantages are less obvious. It also shows that the design techniques studied provide a high level of diversity and are suitable for use in redundant systems.

Future work may include a systematic way of fault injection in order to be able to exhaustively test all fault combinations which will result in a more comprehensive set of D-metric values. Also, other benchmark circuits should be tested to provide a variety of functional behaviors.

REFERENCES


