Common-Mode Failure Mitigation: Increasing Diversity through High-Level Synthesis

Farah Naz Taher[1], Matthew Joslin[1], Anjana Balachandran[2], Zhiqi Zhu[1], and Benjamin Carrion Schafer [1]
Email: {farah.taher,matthew.joslin}@utdallas.edu,anjana.balachandran@polyu.hk,{zhiqi.zhu,schaferb}@utdallas.edu
[1] The University of Texas at Dallas , [2] The Hong Kong Polytechnic University

***Abstract*—Fault tolerance is vital in many domains. One popular way to increase fault-tolerance is through hardware redundancy. However, basic redundancy cannot cope with Common Mode Failures (CMFs). One way to address CMF is through the use of diversity in combination with traditional hardware redundancy. This work proposes an automatic design space exploration (DSE) method to generate optimized redundant hardware accelerators with maximum diversity to protect against CMFs given as a single behavioral description for High-Level Synthesis (HLS). For this purpose, this work exploits one of the main advantages of C-based VLSI design over the traditional RT-level design based on low-level Hardware Description Languages (HDLs): The ability to generate micro-architectures with unique characteristics from the same behavioral description. Experimental results show that the proposed method provides a significant diversity increment compared to using traditional RTL-based exploration to generate diverse designs.**

I. Introduction

Redundancy and diversity have been widely used throughout disciplines and mankind to tolerate design faults. E.g. airlines require a pilot and a co-pilot in each cabin and for complicated surgeries, two or more surgeons need to be present. In most of the safety critical systems which require mechanisms to tolerate faults and the typical way to address these is through redundant systems. These systems are built by replicating the same hardware N times. This is normally called N-Modular Redundancy (NMR). Although effective, this approach cannot protect against Common Mode failures (CMFs). CMFs can result from failures that affect more than one module at the same time, generally due to a common-cause in a redundant system. CMF can occur due to external (e.g. EMI, power-supply disturbances, and radiation), internal or design mistakes. Although redundancy can be effective for physical fault detection and system recovery; but simple redundant systems cannot protect against CMFs because the replicated hardware modules will produce the same erroneous output and hence the voter will interpret the output as correct [1].

Design diversity has been proposed to address CMF detection. While for larger electronic systems these channels have been built by sourcing parts from different vendors. The most rudimentary form of design diversity is to have multiple parallel teams building the same circuit using different tools. Nevertheless, the reliability of these redundant systems does not only depend on the reliability of each version, but also on the differences (dissimilarities) between them. This technique adds a considerable cost overhead to the overall design proportional to the level of redundancy. Thus, automated methods to obtain different implementations of the same circuit which output different values in the presence of CMFs are required.

Fig.1 shows a motivational example for this work. Fig.1(a) shows a traditional fault-tolerant flow which does not take into account hardware diversity. In this case, the exact same hardware channel is replicated twice and a voter checks if the outputs match or not, also called Duplication with compare (DWC). This configuration cannot detect CMF which lead to the same erroneous outputs. To address this, previous work achieved diversity by synthesizing the RTL description with different constraints, such as maximum fanout, target synthesis frequency, etc [2], [3]. As shown in Fig:1(b), this leads to a family of micro-architectures with different performance and area, which in some cases can detect CMFs.

Currently, VLSI design methodology is transitioning from the use of low level Hardware Description Languages (HDLs) to higher levels of abstraction. This work advocates a new approach to fault-tolerance for CMF avoidance, based on leveraging one of the main advantages of C-based VLSI design: the ability to generate micro-architecture with different characteristics from the same behavioral description. Fig.1(c) highlights the proposed flow. As seen, the design space is much larger than in Fig:1(b) because completely different micro-architectures can be obtained from the behavioral level by setting different synthesis options as opposed to modifying the RTL synthesis constraints. The main synthesis directives used in HLS include the specification of how to synthesize arrays, loops and functions. E.g. arrays can be synthesized as RAMs (with a different number of ports) or registers, loops unrolled, partially unrolled or pipelined and functions inlined or not. Setting different combinations of these synthesis attributes leads to a completely new micro-architecture with unique area, power, performance and reliability trade-offs. Thus, the main goal of this paper is to investigate if the unique ability of behavioral VLSI design of diverse synthesis can be used to create diverse redundant hardware systems.

As we observe in Fig: 1(d), for both instances, design space exploration in High level synthesis creates a larger design pool compared to the exploration result at the RT-level. In summary, the main contribution of this paper are:

- Present an HLS design space explorer to maximize the diversity of hardware accelerators to protect against CMFs.
- Study the increase in diversity and hence CMF protection of traditional RT-level design diversity compared behavioral-level diversity.

978-3-9819263-2-3/DATE19/©2019 EDAA

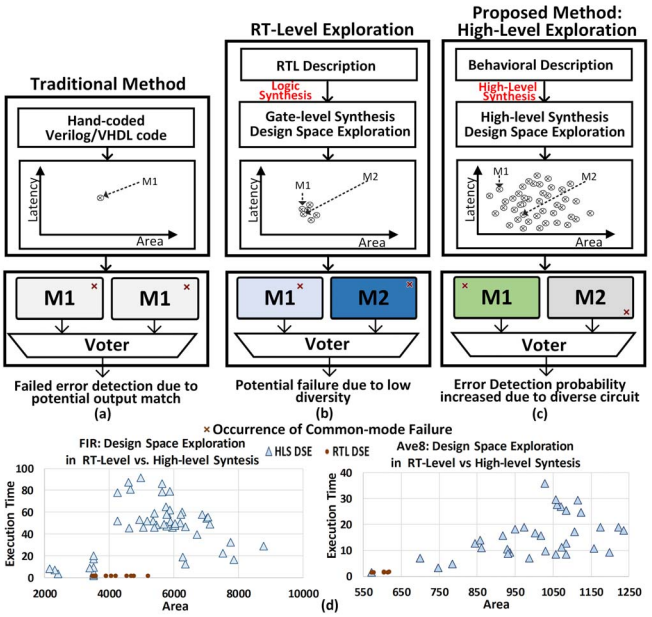

Fig. 1. Design flow for common-mode failure avoidance in traditional and proposed method

II. Related Work

In the software domain, researches have widely studied the effect of software diversity on fault tolerance. Previous works mainly focused on developing cost-effective ways to automatically add diversity to software. A popular technique has been diverse compiling, which makes use of the diversity of different compilers and compiler optimizations. This has been mainly achieved by setting different level of compiler optimizations (e.g. gcc -01, 02, -03) and by using different compilers (e.g. gcc, LLVM or icc). The purpose of diversity is to minimize the opportunities for causes of software faults in two or more versions [4].

Inspired by this, the hardware community has also adopted this idea [5]. The main idea is to have functional equivalent circuits with different logic netlists. Thus, if a transient common mode failure in redundant system happen, this will manifest itself differently at the outputs and thus, can be detected. Unfortunately, their method requires the fault injection at individual points of the netlist of the different netlists being evaluated, which is an NP-complete problem. Very recently, the authors in [3] proposed a new diversity metric based on the structural analysis of the gate netlist. The proposed methodology is orthogonal to this previous work as it allows to choose from a pool of micro-architectures with different levels of fault-tolerance to build the best redundant system with the highest diversity. In addition, it presents a method to generate this pool automatically.

III. Proposed Method

The proposed method is called High-Level Synthesis Design Space Exploration Method for Diverse Design Implementation (HDMI). Algorithm:1 summarizes the flow of our proposed method. The input to HDMI is a single behavioral description (C_{in}) and the output a set of gatenetlist pairs with highest

ALGORITHM 1: Proposed flow overview.

input : C: Input Behavioral Description,
M: Exploration mode; mr: mutation rate
output: $DMR = \{D_1, D_2\}$
DMR: Dual Modular Redundant System

1 /* **Genetic Algorithm** */
2 $population(D_1, D_2, ..., D_n) = gen\_population(C)$;
3 **do**
4 p_i,p_j= $gen\_parents(population)$;
5 $C_{pi,pj} = compute\_cost(p_i, p_j, M)$;
6 **do**
7 $o_{pi,pj} = gen\_offspring(p_i, p_j, mr)$;
8 $C_{pi,oij} = compute\_cost(p_i, o_{i,j}, M)$;
9 $C_{pj,oij} = compute\_cost(p_j, o_{i,j}, M)$;
10 $substitute\_parent(p_i, p_j, o_{ij}, C_{i,j}, C_{i,ij}, C_{j,ij})$;
11 **while** (N number of child doesn't improve result);
12 **while** (No smaller cost function obtained);
13 **return** (DMR with C_{min});

diversity. The proposed method is based on a traditional HLS design space exploration (DSE) with modified cost function to maximize diversity. Thus, before we explain in detail how HDMI works we define how to measure diversity.

A. Diversity Calculation

To our best knowledge, there are two existing method to quantify diversity. The authors of [6] proposed a diversity metric based on gate level fault injection. To speed up the diversity computation process, the authors in [3] proposed a fast method called DIversity Metric based on circuit Path analysis (DIMP) based on the static analysis of the different gate netlists. The idea is based on the concept that lack of diversity happens when a given input signal I_i propagates to a given output signal O_j through similar gates in both circuit instances. Thus, to quantify diversity, DIMP takes into account whether the same netlists are traversed in the same order from I_i to O_j in both circuit instances. To achieve this, all path of a given circuit are taken as inputs and circuit paths timing reports are obtained as output after logic synthesis listing all paths. The Diversity value is between 0 to 1.

$$DIMP = \sum weight_{(p^1_{i,j}, p^2_{i,j})} \cdot (1 - overlap_{(p^2_{i,j}, p^2_{i,j})})$$
$$MaxDIMP = \sum weight_{(p^1_{i,j}, p^2_{i,j})}, DV = \frac{DIMP}{maxDIMP} \quad (1)$$

where overlap is the number of gates repeated across paths in the same order, and weight is the maximum gate count of the paths. Two identical circuit instances will lead to diversity 0 since all the path are identical. The closer the diversity value is to 1, the more diverse the designs are and intuitively less susceptible to common-mode failure. Our proposed method uses this second diversity metric because of its simplicity.

B. Diversity-aware HLS Design Space Exploration:

The proposed HLS explorer is based on a Genetic Algorithm (GA) which has shown to lead to good results in multi-objective optimization problems [7]. The explorer has three objectives. Two objectives need to be minimized, area and execution time, and one maximized, the diversity. Because the final goal is to find design pairs to implement the DMR system shown in Fig. 1, a cost function that considers these three metrics for each new design pairs is needed.

The cost function nevertheless depends on the execution mode of the explorer, which can run in two modes: The first mode is the unconstraint case, in which the only objective is to find the design pairs with the highest diversity (highest DIMP value). In this case the cost function is defined as ($C_{di,dj} = 1/DV_{i,j}$), with $DV_{i,j}$ being the DIMP value between the two designs (D_i and D_j) being considered, with each design characterized by its area and latency, $D_i = \{A_i, L_i\}$ and $D_j = \{A_j, L_j\}$. In this case, a high diversity value is better as the objective is to maximize the diversity, which in turns reduces the cost function. This cost function is subject to the given total area (A_{max}) and execution time (L_{max}) constraint, and thus $A_{DMR} = \{A_i + A_j\} \leq A_{max}$ is the area of the complete DMR system composed of the two modules (A_i and $A_{j]}$)(the voter is excluded as it is the same in all of the solutions) and $L_{DMR} = max(L_i, L_j) \leq L_{max}$ the longest latency in clock cycles of the two designs, as these operate in parallel. In the unconstraint case, the explorer has to *sweep* the complete design space finding Pareto-optimal pairs of unique area vs. latency vs. diversity. Hence, the cost function used is defined as $C = \alpha A + \beta L - \gamma D$, where α, β and γ represents the weights representing the importance to minimize either the area, the latency or to maximize the diversity (thus, the negative sign in the diversity term). In this second unconstraint case, the explorer adaptively modifies these weights to sweep across the entire search space. The next subsections describe in detail how the explorer works.

HLS provides multiple *knobs* to generate different micro-architectures. They include global synthesis option that allows to *e.g.* control the FSM encoding and the target synthesis frequency, function unit constraints (limit of functional units to be instantiated) and local synthesis attributes in the form of pragmas or comments to control how to synthesize mainly arrays, loops and functions. These are basically comments inserted directly in the source code in before the explorable operation. Our proposed explorer explores these last two knobs as they are also the most important knobs as they lead to completely different micro-architectures. Algorithm:1 depicts an overview of the proposed GA-based diversity aware HLS DSE. The first step parses the behavioral description and extracts all the explorable operations in the code. These are operations that can be controlled through synthesis directives (in this work we consider arrays, loops and functions). It then continues by generating a pool of parents by randomly assigning each parent a unique list of pragmas. Each explorable operation is represented as a gene to which a synthesis attribute (pragma) is assigned. In this work, the pragmas we have used are: arrays={register, expand, logic, RAM, ROM}, loop={no, partial, all, fold}, and func={goto, inline}. With these pragmas, it is possible to synthesize arrays as memories or registers. Loops can be unrolled, partially unrolled or pipelined and functions inlined or synthesized as single hardware blocks (gotos). Two parents are in turn randomly selected (p_i and p_j) and their cost function computed depending on the type of exploration that is being run (unconstrained or constrained), thus leading to DMR system characterized by these three metrics $DMR_{pi,pj} = \{A_{DMR}, L_{DMR}, DV_{i,j}\}$. The list of attributes is then combined based on a randomly chosen cut-off point and some of the attributes mutated by choosing a different one from the attribute library. The mutation rate depends on the mutation rate mr specified by the user, which be default is set to $mr = 0.1$. The mutated offspring is then synthesized by calling the HLS tool and performing a logic synthesis on the generated RTL code (required to compute the DV) and the area and latency annotated. The diversity value is then computed using the DIMP value described previously, for the offspring and each of the parents leading to two new design pairs (DMR systems) $DMR_{pi,oij} = \{A_{DMR}, L_{DMR}, D_{i,ij}\}$ and $DMR_{pj,oij} = \{A_{DMR}, L_{DMR}, D_{j,ij}\}$. The offspring substitutes one of the parents if the cost of the new design pairs is lower than the cost of the two parents. Each new offspring is synthesized using the maximum number of functional units to fully parallelized the micro-architecture based on the pragma list and then again with a single functional unit of each time to maximize resource sharing. The algorithm will continue until N number of child designs do not improve any of the parents.

C. Predictive Model Diversity Estimation

One of the problems with the previously described GA-based diversity explorer is that in order to compute the DIMP value, it has to perform a logic synthesis for each newly generated offspring. We have observed that the HLS process is relatively fast (in the order of seconds), while the logic synthesis process is much slower (in the order of minutes). To speedup the exploration we, therefore, investigated the use of predictive models that can be used to predict the diversity of a design pair with the information obtained right after the HLS process. For this, we used a well-known machine learning tool (WEKA [8]) which contains a library of predictive methods and run it on the results obtained from the GA-based explorer. The predictors were the area, segregated based on type (*i.e.* sequential vs. combinational, functional unit area, muxes area, decoder area, etc.), latency and delay values reported by the HLS tool and the predicted value the DIMP value calculated from the gatenetlists. It was found that the differences in sequential logic $\Delta S_{i,j} = S_i - S_j$ (basically registers) could be used as a simple predictor for diversity. We call this method predictive HDMI (pHDMI) The experimental section compares the results for the GA-based exploration with gatenetlist DIMP calculation and using $\Delta S_{i,j}$ as DIMP estimation.

D. RT-Level Design Space Exploration

Previous work, create diversity by generating functional equivalent gate netlists from a given RTL code. This can be achieved by setting the mapping effort to different levels, power effort, maximum fanout and target clock frequency. As shown in the motivational example, this leads to different netlists, but the actual micro-architecture does not change, thus, leading to a smaller search space and hence diversity range. To compare our proposed method, we have implemented an RTL explorer which takes as inputs an RTL description in Verilog or VHDL and synthesizes it (logic synthesis) using the following *knobs*: mapping effort={medium, high},

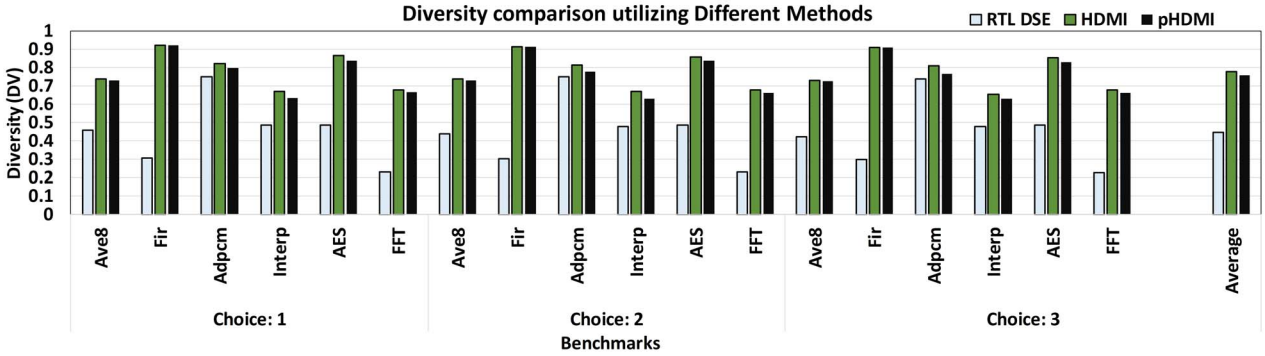

Fig. 2. Top 3 design pair choices using High-level synthesis and gate-level netlist design space exploration methods

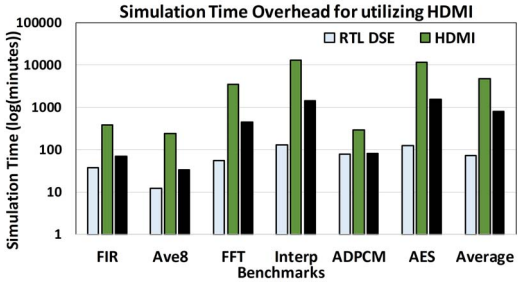

Fig. 3. Comparison between simulation times for unconstrained and search space reduction methods

area effort={medium, high}, and power effort={medium, high}, fanout={2,4,6}, clock = {n,n/2,n/4,2n}, where n is the original target synthesis frequency. Because an exhaustive enumeration of all the exploration knobs is not large, we perform a brute force search.

IV. Experimental Results

We have chosen six different benchmarks with different complexities from the open source synthesizable SystemC benchmark suite (S2CBench) [9] as hardware accelerators to test the effectiveness of our proposed method. The tool used for the HLS exploration is CyberWorkBench v.6.1 from NEC [10] and Synopsys Design Compiler (DC) [11] as logic synthesis tool. The target technology is Nangate Opensource 45nm. The experiments are conducted on an Intel i7-6700 @3.50GHZ CPU and 16 GB memory, running CentOS 7.

The 6 benchmarks were explored in an unconstrained mode to fully observed the benefits of raising the level of abstraction. Fig:2 shows the results of the three methods highlighting the three design pairs found by each of the methods with the highest diversity. For the traditional RTL-based method we synthesized the behavioral description using the default options from the HLS tools and fed the RTL code generated to the RTL-based explorer. HDMI is the full GA-based explorer performing a logic synthesis on each newly generated offspring and pHDMI is the predictive-based HDMI version that only uses the difference in sequential logic as diversity predictor and then fully computing the DIMP value for the selected design pairs in order to report the actual diversity values. For all the benchmarks we observe that raising the level of abstraction to C leads to more diverse designs. The average diversity value increased by 1.75× and 1.70× for the HDMI and pHDMI method respectively. This proves that using HLS leads to more robust designs against CMFs.

Fig: 3 compares the three methods in terms of the runtime to obtain these results. The full HDMI method takes on average 6× longer than the predictive HDMI (pHDMI) and on average 66× longer than the RTL-based exploration. The main reason for the long running time for the HDMI method is that it requires a full logic synthesis for every new offspring. Although the runtime is high we believe that it is still acceptable especially considering the extra fault tolerance achieved and that this exploration process only requires to be executed once. Based on these results, we can conclude that raising the level of abstraction can significantly increase the design diversity and hence protect against CMFs.

V. Conclusion

In this work, we have shown the raising the level of abstraction from RT-level to C-level has the additional advantage of being able to increase the design diversity to detect common mode failures. Experimental results have shown that the diversity achieved at the behavioral level of abstraction is on average 2× higher, thus, proving the benefit of using HLS in fault tolerance designs.

References

[1] S. Mitra, N. R. Saxena, and E. J. McCluskey, "Common-mode Failures in Redundant VLSI systems: A Survey," *IEEE Transactions on Reliability*, 2000.

[2] ——, "Efficient Design Diversity Estimation for Combinational Circuits," *IEEE Transactions on Computers*, 2004.

[3] S. Alcaide, C. Hernandez, A. Roca, and J. Abella, "DIMP: A Low-Cost Diversity Metric Based on Circuit Path Analysis," in *Design Automation Conference(DAC)*, 2017.

[4] A. Höller, N. Kajtazovic, K. Römer, and C. Kreiner, "Evaluation of Diverse Compiling for Software Fault Tolerance," in *Design Automation and Test in Europe (DATE)*, 2015.

[5] S. Mitra, "Globally Optimized Robust Systems to Overcome Scaled CMOS Reliability Challenges," in *Design Automation and Test in Europe (DATE)*, 2008.

[6] S. Mitra, N. R. Saxena, and E. J. McCluskey, "A Design Diversity Metric and Analysis of Redundant Systems," *IEEE Transactions on Computers*, 2002.

[7] A. E. Eiben, P.-E. Raue, and Z. Ruttkay, "Genetic Algorithms with Multi-parent Recombination," in *International Conference on Parallel Problem Solving from Nature*, 1994.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *ACM SIGKDD explorations newsletter*, 2009.

[9] B. Carrion Schafer and A. Mahapatra, "S2CBench: Synthesizable SystemC Benchmark Suite for High-level Synthesis," *IEEE Embedded Systems Letters*, 2014.

[10] NEC, "CyberWorkBench," Version 6.1, URL: https://www.cyberworkbench.com, 2018.

[11] Synopsys, "Design Compiler," Version 1.3.95, URL: https://www.synopsys.com, 2006.