Empirical Analysis of the Specialization of a Diversity Metric per Circuit Path

Sergi Alcaide[‡,†], Carles Hernandez[⋆,†], Jaume Abella[†]
[†]Barcelona Supercomputing Center (BSC)
[‡] Universitat Politècnica de Catalunya (UPC)
[⋆] Universitat Politècnica de Valencia (UPV)

***Abstract*—Diversity can be obtained by using different implementations of the same hardware functionality and can be applied at different abstraction levels (e.g., gate level, transistor level). However, although diversity is well-understood at a qualitative level, it is hard to quantify it since it relates to the specific target that creates the common cause fault and the abstraction level at which it is considered. In this paper, we analyze the behaviour of a diversity metric based on circuit path analysis in the context of systematic fault patterns.**

I. Introduction

Safety-related systems must undergo an exhaustive validation and verification process before being deployed to prove that their safety goals are met. This process needs to collect enough evidence to prove that either the risk of death or serious injury to people, the loss or severe damage to equipment/property and/or the environmental harm due to the system's malfunctioning behavior can be regarded as negligible. Therefore, a hazard analysis and risk assessment is needed for safety-related systems to determine the degree of prevention or mitigation needed in order to avoid unreasonable risk.

These safety goals are defined at the scope of the whole system, but also they are propagated to all the components, so their composition preserves the safety goals. This translates into attaching to each component a Safety Integrity Level, which determines the type and amount of evidence needed in the verification and validation process to prove that the component under analysis will meet its specifications.

When mapping safety integrity levels to hardware components such as microcontrollers, meeting the requirements for high integrity levels requires providing hardware components with specific safety mechanisms. For instance, in the context of the automotive domain, microcontrollers providing Automotive Safety Integrity Level (ASIL) C and D functionalities often require some form of hardware redundancy [1], [2]. Although redundancy is very useful against most faults, it is vulnerable to faults that can produce the same failure on each of the system's instances (e.g., due to voltage droops). This type of failures are often produced by *common cause faults* and need, not only redundancy but also *diversity*, so that the manifestations of the fault are different in each diverse instance and hence, the fault can be detected before becoming a failure.

Diversity can be obtained by using different implementations of the same hardware functionality and can be applied at different abstraction levels (e.g., gate level, transistor level). However, although diversity is well-understood at a qualitative level, it is hard to quantify it since it relates to the specific target that creates the common cause fault and the abstraction level at which it is considered. Pionnering efforts to quantify diversity systematically have only been proven successful for random faults [3], [4].

DIMP, a diversity metric based on circuit path analysis, was proposed in [5]. While this metric provides interesting properties in terms of computational complexity and simplicity, it has only been proven under very specific scenarios [5]. In [6] DIMP was compared against other diversity metrics. Authors in [7] used DIMP in their evaluation because of the low-cost compared with other diversity metrics. Also, in [8] authors create different VLSI designs from the same C source code and evaluate their diversity using DIMP.

In this paper, we thoroughly evaluate the behavior of DIMP in the context of systematic fault patterns, by modifying the original metric in order to specialize it to this specific fault model. For this, we considered a subset of gates to be more susceptible to faults and required a distinctive treat during the diversity calculation. Later on, after not achieving to pair the specialized DIMP with the results from a fault injection campaign, we analyzed our specialized DIMP metric to identify the elements that could interfere. With this, we identified possible aliasing within the metric, which we corrected. Unfortunately, the final metric still did not provide the expected results.

II. A Diversity Metric based on circuit Path analysis

Using space redundancy protects systems against transient and permanent faults. However, protection against the particular case of common cause faults is not provided. Common cause faults can either be permanent or transient, but the characteristic that defines them is that affect multiple instances inside our redundant system. Therefore, if instances inside the sphere of replication (SoR) are affected simultaneously and identically, redundant elements will produce the same erroneous outputs, and the output comparator will not detect the errors, leading to a system failure.

In order to protect systems against common cause faults, diversity must be used across the different instances inside the SoR. Recently, DIMP, a diversity metric based on circuit

path analysis, was proposed to cover this gap. DIMP offers a systematic mechanism to quantify diversity, thus improving system robustness against common cause faults.

A. Rationale behind DIMP

DIMP is built on the idea that lack of diversity occurs when starting a signal from an input pin (I_i) and traversing through the path until we reach an output pin (O_j), the devices that the signal goes through are *similar*. Therefore, in order to quantify diversity we must consider the traversed devices from a given input to a given output, also called a path. The comparison must be made on paths with the same input and output since they are carrying out the same functionality.

Once all irrelevant parts of the circuit have been filtered out, and the devices traversed to get from I_i to O_j kept, we can perform the comparison. Before, we must consider putting weights in each path to reflect the *importance* of each path in the manifestation of the given fault. For instance, if we consider the timing faults, longer paths can be more susceptible than shorter paths. Then, one may consider that the shorter paths are less likely to induce erroneous outputs and be less relevant than longer paths. In that case, we can put a small weight in paths that are shorter than a given threshold, for example.

B. A realization of DIMP

1 $DIMP = 0$
2 $MaxDIMP = 0$
3 **For** $i = 1$ **to** N
4 **For** $j = 1$ **to** M
5 $Paths^1_{i,j} = \bigcup \left(Paths\left(I^1_i, O^1_j\right)\right)$
6 $Paths^2_{i,j} = \bigcup \left(Paths\left(I^2_i, O^2_j\right)\right)$
7 **While** $Paths^1_{i,j} \neq \emptyset$ **and** $Paths^2_{i,j} \neq \emptyset$ **do**
8 Take $p^1_k \in Paths^1_{i,j}$ and $p^2_l \in Paths^2_{i,j}$ with highest $overlap(p^1_k, p^2_l)$
9 $DIMP = DIMP + weight(p^1_k, p^2_l) \cdot (1 - overlap(p^1_k, p^2_l))$
10 $MaxDIMP = MaxDIMP + weight(p^1_k, p^2_l)$
11 Remove p^1_k from $Paths^1_{i,j}$
12 Remove p^2_l from $Paths^2_{i,j}$
13 **Endwhile**
14 $RemainingPaths_{i,j} = Paths^1_{i,j} \cup Paths^2_{i,j}$
15 **While** $RemainingPaths_{i,j} \neq \emptyset$ **do**
16 Take any $p_q \in RemainingPaths_{i,j}$
17 $DIMP = DIMP + weight(p_q, \emptyset)$
18 $MaxDIMP = MaxDIMP + weight(p_q, \emptyset)$
19 Remove p_q from $RemainingPaths_{i,j}$
20 **Endwhile**
21 **Endfor**
22 **Endfor**
23 Return $\frac{DIMP}{MaxDIMP}$

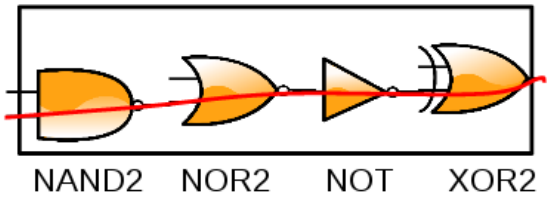
Fig. 1: Pseudocode of a realization of DIMP.

A particular realization of DIMP was proposed in [5]. This realization is intended for fault types like voltage droops, residual faults escaping diagnosis coverage, and permanent faults in the root of the power network of the circuit instances. Pseudocode of the realization is shown in Figure 1.

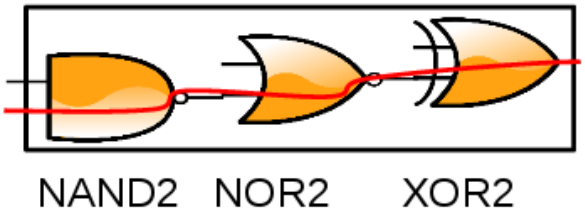
To implement this realization, we first initialize the two main values, $DIMP$ and $MaxDIMP$ to 0. As we will iterate over the designs, $DIMP$ will be the diversity value, while $MaxDIMP$ will be the maximum possible value of diversity. The loops at lines 3 and 4 stand for the number of inputs (N) and the number of outputs, (M). First, we will group all the paths for each pair $< I_i, O_j >$ separately, lines 5 and 6. Since we work at the gate level, the devices correspond to all gates that traverse from the input to the output. Then $Paths^1_{i,j}$, stands for all the paths from circuit 1 that traverse from input i to output j, and each path consists of an ordered list of gates.

Once all the paths are grouped in sets, we will iterate over all the sets (line 7). Given a pair $< I_i, O_j >$, we will have paths from circuit 1 and paths from circuit 2; we will select the pair of paths, one for each circuit, that has the highest overlapping. We define $overlap(p^1_k, p^2_l)$ as the number of gates that repeat across paths in the same order, even if some other gates are interleaved. This is a problem known as LCS [9] (*Longest Common Subsequence problem*). We divide the value by the total number of gates in both paths; otherwise, the longer paths will easily have high values instead of shorter paths that are similar.

For instance if $p^1_k = \{NAND2, NOR2, NOT, XOR2\}$ and $p^2_l = \{NAND2, NOR2, XOR2\}$, see figure 2 then $\{NAND2, NOR2, XOR2\}$ will be the longest common subsequence and the $overlap(p^1_k, p^2_l) = 6/7$, since 6 gates repeat in both paths out of 7 total gates. Other definitions of $overlap(p^1_k, p^2_l)$ are possible.



(a) Example p^1_k



(b) Example p^2_l

Fig. 2: Small example: $\{NAND2,\ NOR2,\ XOR2\}$ is the solution of LCS

Because $DIMP$ variable is the diversity value, we increase $DIMP$ with the non-overlap (line 9), which is subtracting from 1 the value of the overlapping, and then we multiply it with the relative weight ($weight(p^1_k, p^2_l)$) of those paths. Since we target timing, we define $weight(p^1_k, p^2_l)$ as the maximum gate count of those paths. Other considerations

of weights can be employed according to the fault model being selected. $MaxDIMP$ is also increased with the relative weight (line 10), but as we explained, $MaxDIMP$ is the maximum potential value of diversity, so accordingly, we have 0 overlapping in all the cases. Then we remove the paths considered from the sets in lines 11 and 12. We continue evaluating and subtracting paths from the sets until one of the sets is empty. If the other set is still not empty, we consider the rest of the paths of the non-empty set as 0 overlapping (fully diverse) paths because we cannot compare against any other path from the other circuit. Once we iterate over all the sets, we will have the $DIMP$ and $MaxDIMP$ value computed. Finally, at line 23 we return the diversity value as the division of the circuits' diversity ($DIMP$) divided by the maximum potential diversity ($MaxDIMP$).

Notice that in the case of two identical circuits, the overlapping will be 1 in all the cases, leading to a value of $DIMP = 0$ and leading to 0 diversity. On the other hand, having two completely different circuits (e.g., one implemented with NOR gates and the other with NAND gates) will have a 0 overlapping in all the cases, which will lead to $DIMP = MaxDIMP$ and resulting in a diversity of 1.

III. DIMP Specialization

To understand how DIMP behaves when we tailor this metric to the faults caused by systematic defects, we define an appropriate fault model. For this model, we consider some gates are more vulnerable to certain conditions – such as magnetic fields or voltage droops – than others due to the impact of systematic process variations. Therefore, we assume that only some gates are affected by faults during the simulations, while all other design parts are fault-free. To model this, we will only inject faults to the selected gates. As a system model, we assume a hardware redundancy + diverisity design in which both instances are within the SoR. Therefore, system failure will occur when both instances generate the same erroneous outputs.

A. Tailoring DIMP to the specific fault model

We can easily adapt DIMP to the new specific fault model. To do so, we use the pseudo-code in Figure 1 but imposing a restriction when we create the sets of paths. Since we know that the probability of some gates to be faulty is zero for the specific fault model, we now consider only the paths that contain at least one gate of the gates regarded as more likely to fail (e.g., due systematic process variations). Thus, the specialization will ignore parts of the circuit and focus only on the paths containing the selected gate or set of gates. We expect this especialized DIMP metric to produce more accurate diversity results for our specific fault model.

B. Gate Selection

Our fault model requires a mechanism to decide which specific gates from each circuit we regard as faulty. The rationale behind this is that, due to process variability, some gates may be more vulnerable than others since they have different sizes and electrical properties, and this will also translate into a systematically higher fault probability. Note that a specific type of gate is not only determined by the logic function implemented but also by its strength. The strength of a gate is associated with its ability to switch a given capacitance and relates to the transistors' size used to build it. To model this, we create a tool that lists a design's gates and their number of occurrences. Using this tool, we noticed that most of the gates appear only once in the designs. Since we want to have a representative number of occurrences but still want to select just one gate type, we simply decided to choose the gate with more instances for each design. We also considered using the two most occurring gates for two circuits to observe how the metric will behave with more gate representatives. We performed three different experiments for these two circuits, one using both gates and two using one of the gates. Table I summarizes the gates selected for each circuit. Our working set is a subset of the ISCAS'89 circuits implemented using the umc65ll (UMC 65nm Low Leakage) technology library and for three different latency targets (1, 0.6, and 0.3 ns).

IV. Evaluation

A. Methodology

We have used the FALLES Fault Injector from [10] for the validation. However, first, we have modified some of his parameters. In particular, we have modified the fault injection to allow the injection to be either a stuck-at 1 or a stuck-at 0, and forcing the faults to be injected at the beginning of the simulation to have a more predictable behavior that we can use to validate DIMP. Table II shows the ISCAS circuits we have employed for this fault injection campaign.

FALLES Fault Injector is the tool in charge of performing the fault injection, launch the simulation, and analyze the results. The analysis phase consists of comparing the results of fault-injection with the Golden Run[1]. Finally, the FALLES Fault Injector will generate a huge file reporting all the erroneous outputs of the simulated design.

According to our system model, we must match the cases where a common cause failure appears. This is, when both designs created the same erroneous outputs with the same conditions (same input values). Following, we have extended the FALLES Fault Injector tool with a new phase, which performs the comparison between two experiments reports.

Due to the size of both input files, processing them is a heavy job. However, one of the restrictions of the matching is that errors must appear under the same input values for a given error. We only need to consider those errors that were produced with the same input values. Thus, one easy way to parallelize the work consists of splitting the errors based on the input values producing them.

In order to perform this task, we created a C++ tool that runs on a parallel cluster. This tool receives two analysis reports from the FALLES Fault Injector analysis phase and replicates

[1]Golden Run is an execution without errors. In this context, it is used to be compared with other executions to find errors.

Circuit		0.3		0.6		1.0	
	Gate Filtered	Num Gates	% of total gates	Num Gates	% of total gates	Num Gates	% of total gates
s1196	**ND2M2W**	28	4.90%	44	9.57%	38	12.62%
	INVM2W	28	4.90%	48	10.43%	22	7.31%
	both	56	9.81%	92	20.00%	60	19.93%
s1238	**ND2M2W**	39	6.29%	59	12.04%	34	10.97%
	INVM2W	30	4.84%	26	5.31%	34	10.97%
	both	69	11.13%	85	17.35%	68	21.94%
s1488	**ND2M2W**	56	10.67%	47	9.29%	30	8.98%
s1494	**ND2M2W**	51	8.89%	23	4.47%	28	7.98%
s386	**NR2M2W**	8	7.14%	7	6.25%	11	15.49%

TABLE I: Selected gate for each circuit and percentage respect to the total

the files for each worker thread. Then, using MPI (Message Passing Interface) as a parallel programming model, the master thread assigns to the workers the errors they must analyze (by giving them the input values they need to search for). Matching is then done in parallel by all the workers:

1) Read their private copy of the input files and select only the errors assigned.
2) Perform the matching between the two experiment reports.
3) Send back a message to the master with the information of how many matches they found and the total number of errors considered.
4) The master collects information from all the workers and calculates the final value.

ISCAS'89 Circuit	Inputs	Outputs
s1196	16	14
s1238	16	14
s1488	10	19
s1494	10	19
s386	9	7

TABLE II: ISCAS'89 working set

Because we model a hardware redundancy + diversity design, the evaluation consists of quantifying the number of cases that faults appearing in both designs could lead to a system failure. Following the fault model, the faults will be injected on both instances of a specified gate. Since we simulate each design separately, after the injection phase, we perform a matching that pairs the cases with the same input values leading to the same erroneous output. Finally, we calculate the fraction between the errors paired against all the errors produced.

The simulation starts with the injection performed at instance 0. Particularly, the injections performed are stuck-at 1 fault. Then the simulation runs during $20ns$ of simulated time to allow all the signals to be propagated. We repeat this process for all the possible inputs (2^n cases, n number of inputs). We cannot use all the circuits from ISCAS'89 because simulations increase exponentially with respect to the number of input signals a circuit has. Furthermore, for each possible input, we perform a fault injection simulation for all the design's possible injection points.

Once simulations are finished, the analyzer phase of the FALLES Fault Injector reports the errors found compared to the Golden Run. Since the file contains the injections done on all the possible injection points, we must filter all the simulations whose injection point was a different gate than the ones selected. With the filter process, we eliminate some of the unnecessary data reported by the FALLES Fault Injector to reduce the files' size. Note that due to the high amount of required injections, reducing the amount of tracked information is crucial to keep the problem tractable. In fact, we only save for each error: the timestamp of the error, the input values, and the erroneous output value.

After the previous process, we run our matching tool on a parallel cluster to perform the matching. Because our model is a hardware redundancy + diversity design, we will match those errors that have the same input and output values, which are the ones that can lead to a common cause failure, since are those not detectable by the output checking (or comparison) mechanism. In order to parallelize the work, we divide the inputs between the available workers, so each worker has to analyze the errors of $\frac{2^n}{w}$ inputs, where w is the number of workers and n the number of inputs of the design. Notice that there is no work balancing since we do not know a priori the number of errors reported for each input. Even with the parallelization, this final step can take up 2 days in our cluster setup, mostly due to the large file sizes.

After the matching phase, we are ready to compute the final value. The value consists of a rate between the number of errors that we matched divided by the total number of errors produced. This formula can be seen as an approximation of the Architectural vulnerability factor on our model, considering we divide the number of errors that can lead to the system failure by all the errors produced by the injections.

1) Results: A summary of the results obtained with the experiments is shown in Table III. We have divided the results for each circuit and gate filtered. For each comparison, we have the values for the generic DIMP, which are the same shown in Section II, plus the values for the specialized DIMP and the matching percentage from the fault injection. We expect to have the highest percentage of matching for the less diverse circuits. Additionally, we should expect the specialized DIMP results to be more precise than the regular DIMP metric and follow the mentioned trend more clearly.

In general, Specialized DIMP results still follow the pattern seen in the generic DIMP metric, which is that the lowest values are on the $0.3\ vs\ 0.6$ pairs and the highest on $0.3\ vs\ 1.0$

CIRCUIT + Gate/s selected		Generic DIMP	Specialized DIMP	Matching percentage	Matching percentage after filtering
s1196					
	0.3∼0.6	0.879962	0.699909	63.25%	1.36%
Gate: ND2M2W	0.3∼1.0	0.989662	0.987574	60.88%	0.23%
	0.6∼1.0	0.987646	0.972088	62.26%	1.61%
	0.3∼0.6	0.879962	0.596522	71.30%	1.04%
Gate: INVM2W	0.3∼1.0	0.989662	0.964201	60.74%	0.06%
	0.6∼1.0	0.987646	0.935969	63.31%	1.03%
	0.3∼0.6	0.879962	0.667136	70.13%	0.97%
Gate: INVM2W + ND2M2W	0.3∼1.0	0.989662	0.983005	68.19%	0.23%
	0.6∼1.0	0.987646	0.967317	68.60%	2.61%
s1238					
	0.3∼0.6	0.881975	0.529878	39.42%	0.66%
Gate: ND2M2W	0.3∼1.0	0.979199	0.956967	34.43%	7.26%
	0.6∼1.0	0.976734	0.936694	62.73%	3.67%
	0.3∼0.6	0.881975	0.628794	64.98%	0.15%
Gate: INVM2W	0.3∼1.0	0.979199	0.982548	52.54%	0.27%
	0.6∼1.0	0.976734	0.977272	55.70%	0.17%
	0.3∼0.6	0.881975	0.573767	61.13%	0.26%
Gate: INVM2W + ND2M2W	0.3∼1.0	0.979199	0.968309	54.56%	0.11%
	0.6∼1.0	0.976734	0.957104	65.82%	0.18%
s1488					
	0.3∼0.6	0.76609	0.673546	43.95%	3.37%
Gate: ND2M2W	0.3∼1.0	0.988728	0.901524	34.48%	3.08%
	0.6∼1.0	0.976887	0.794531	39.41%	5.43%
s1494					
	0.3∼0.6	0.838476	0.880282	32.33%	0.13%
Gate: ND2M2W	0.3∼1.0	0.952615	0.923454	35.39%	2.90%
	0.6∼1.0	0.936673	0.932814	39.81%	1.61%
s386					
	0.3∼0.6	0.821197	0.71831	58.31%	12.56%
Gate: NR2M2W	0.3∼1.0	0.968137	0.879121	55.43%	0.22%
	0.6∼1.0	0.935917	0.820628	62.78%	0.74%

TABLE III: We have from left to right from each pair of circuits, Generic DIMP, Specialized DIMP, and the last two columns show the matching percentage from the fault injection. First the initial one, and the rightmost after the aliasing removal.

pairs. We believe that both metrics have similar results due to the selection of gates: since we selected the most representative gates of the circuit, the results are similar. Notice that the values of the 0.3 vs 0.6 pairs are lower than on the generic DIMP, because now we consider paths that at least have one gate in common, the selected one. Therefore, overlapping will never be 0 on those paths, except when we run out of paths from one of the circuits and the other one still has unmatched paths.

As mentioned before, one should expect matching percentages to be higher when the DIMP values are close to 0, since we have lower DIMP values when gates across paths are "similar" in both designs. More precisely, in the case of specialized DIMP, paths containing the potentially erroneous gates are analyzed and the rest of the design is ignored. Considering we know in advance the potentially erroneous gates, we also know that the paths analyzed with specialized DIMP are the ones that will traverse the erroneous signal. Therefore, lower values of specialized DIMP will indicate that the mentioned paths are very similar and, thus, will potentially have similar erroneous outputs.

It is important to remark that matching percentages measured are very high for all circuits (lowest value is above 30%). Thus, this means that at least 1 out of every 3 injections can be matched and potentially lead to a system failure (in these particular designs). This is a very relevant result and can estimate how important it is to have diverse implementations to face common case faults since the percentage of matching in identical designs will be even higher.

In summary, we cannot generally see a clear correlation between DIMP, neither the generic nor the specialized, and the matching percentage. Only in three cases (s1196-INVM2W, s1488-ND2M2W, and s1238-INVM2W) the behavior is the one expected. In that respect, in the next section, we try to understand the reason for this and propose different approaches to improve the results.

B. Removing the aliasing

As shown before, our metric does not fit the needs of the specified fault model since we cannot correlate the behaviour seen in the matching percentages with the values obtained neither from DIMP nor the specialized DIMP. Therefore, we need to reconsider all the steps taken and try to identify the reason why we are obtaining such results.

Our hypothesis are that either (1) specialized DIMP is not well suited for the particular fault model or that (2) experiments are not representative enough, or (3) both of them.

If we start with 1, considering that the specialized DIMP algorithm is wrong, still cannot explain per se why the generic DIMP, which has proved to behave correctly, cannot correlate with the results. Although the regular DIMP implementation is more generic, we expect at least some degree of correlation, but this is not the case. Furthermore, the results of the matching seem to have different patterns for each circuit. Thus, we consider that we are having some form of noise or aliasing in our results that makes results not to be consistent across circuits, thus considering hypothesis number 2.

Surprisingly, matching percentages are enormous, being 32% the lowest value. Meaning, that almost 1 out of 3 errors can potentially cause a system failure. We believe that this is again, because of the aliasing since we are comparing faults that are injected at gates belonging to different paths and, therefore, DIMP quantification is not considering that effect. Note that DIMP metric is computed on a per-path basis, and therefore, matching errors originated at different paths can only contribute to introducing noise to the obtained results.

In order to remove this aliasing, we introduce a new filter or constraint in the matching process. In fact, we impose to the matching of errors that injections belong to gates that are in the same path. Now, when applying the filter after the FALLES Fault Injector tool, we must save the path in which the injection was made to filter out this result. With this new condition, we expect the noise within the matching percentages to be reduced and to have a better correlation with the specialized DIMP.

1) Results: Looking at the results shown at the last column of the Table III, the first thing we see is that, as expected, the new restriction has reduced the matching percentages significantly, being now 12.56% the highest value among all the circuits. However, despite that, results are still inconsistent across different circuits. The expected behavior can only be observed in 3 out of 9 experiments (s1196-INVM2W, s1238-INVM2W ND2M2W and s386-NR2M2W), which means that, although we have removed the inter-paths aliasing, we cannot match the specialized DIMP with the results yet.

We have also evaluated some new experiments where we used two gates at the same time. Unfortunately, these results are inconsistent since only one of them matches the expected behavior.

V. Conclusions

Hardware redunancy is widely used in safety-related microcontrollers to meet the standard requirements of the highest integrity levels. However, hardware redundancy per se is susceptible to common cause faults, which can produce failures. In order to avoid these failures, diversity is required. Unfortunately, diversity is a property difficult to evaluate. In this work, we have used a previously proposed diversity metric called DIMP, and try to specialize it to a particular fault model (systematic defects due process variation). Systematic faults are becoming more relevant as the transistor's size keeps shrinking and more transistors are added to a single chip. After analyzing the fault model, we have modified the original DIMP metric to weigth the importance of certain gates that are more susceptible to systematic faults.

Despite our efforts in finding a correlation between the specialized DIMP and the percentage of matching reported by our evaluation method, we have failed to show the expected correlation. We have deeply investigated the reasons for that, but we think they are related to the following causes. For the generic DIMP, diversity is computed on a per path basis, and the particular fault-model we have employed in this chapter is not suitable for this restriction. In that respect, we have specialized DIMP to filter on a per-path basis and failed to show the expected behavior. Our intuition here is that after filtering out so many gates – the ones not belonging to the path and the ones not having a specified gate – and also having in mind that the circuits that we were able to evaluate are the smallest ones, the remaining parts of the circuits are too small to provide any meaningful result. We plan to perform modifications to our evaluation method to solve that limitation. However, we leave this as future work.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 871467. This work has also been partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under grant PID2019-107255GB.

References

[1] Freescale Semiconductor, "Qorivva MPC5643L microcontroller data sheet. rev. 9," 2013.

[2] Infineon, *Tricore 1. 32-bit Unified Processor Core v1.3*, October 2005.

[3] S. Mitra, N. Saxena, and E. McCluskey, "A design diversity metric and analysis of redundant systems," *IEEE Transactions on Computers*, vol. 51, no. 5, 2002.

[4] S. Mitra, N. Saxena, and E. McCluskey, "Techniques for estimation of design diversity for combinational logic circuits," in *DSN*, 2001.

[5] S. Alcaide, C. Hernandez, A. Roca, and J. Abella, "DIMP: A low-Cost Diversity Metric based on circuit Path analysis," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ser. DAC '17. New York, NY, USA: ACM, 2017, pp. 45:1–45:6. [Online]. Available: http://doi.acm.org/10.1145/3061639.3062231

[6] F. N. Taher, A. Balachandran, and B. Carrion Schafer, "Learning-based diversity estimation: Leveraging the power of high-level synthesis to mitigate common-mode failure," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, 2019, pp. 460–467.

[7] F. N. Taher, M. Joslin, A. Balachandran, Z. Zhu, and B. C. Schafer, "Common-mode failure mitigation: Increasing diversity through high-level synthesis," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1563–1566.

[8] M. R. Babu, F. N. Taher, A. Balachandran, and B. Carrion Schafer, "Efficient hardware acceleration for design diversity calculation to mitigate common mode failures," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019, pp. 267–270.

[9] Wikipedia, "Longest common subsequence problem — wikipedia, the free encyclopedia," 2017, [Online; accessed 13-Jan-2021]. [Online]. Available: https://en.wikipedia.org/wiki/Longest_common_subsequence_problem

[10] J. Espinosa, C. Hernandez, J. Abella, D. de Andres, and J. C. Ruiz, "Analysis and rtl correlation of instruction set simulators for automotive microcontroller robustness verification," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.